Ply and Bar Visibility - Some Advanced Concepts in Graph Drawing

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Niklas Heinsohn
aus Flensburg

Tübingen
2019

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation: 04.06.2019

Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Michael Kaufmann
2. Berichterstatter:	Prof. Dr. Klaus-Jörn Lange

Acknowledgements

First and foremost I want to thank my supervisor Michael Kaufmann for his support, for his patience with me and my thesis, the countless rides to mensa, and nonetheless for the overall warm and welcoming atmosphere in his working group.

This of course, also applies to my colleagues, namely Michael Bekos, Patrizio Angelini, Henry Förster, Thomas Schneck, our current visitor Alessandra Tappini and former colleagues Christian Zielke, Andreas Gerasch, Robert Krug and Till Bruckdorfer.

A very special "Thank you" has to go to our secretary Renate Hallmayer. Without her we all would be lost in paperwork. She helped us in any situations.

Furthermore, I want to thank my employers for the opportunity to work with them. Here I want to mention Michael Kaufmann and Franz Josef Brandenburg from the University of Passau. Throughout my time at the university I spent a lot of time teaching math. I want to thank all my employers at the University of Tübingen, namely Peter Hauck, Ulrike von Luxburg and Rüdiger Zeller. I think we did a great job teaching the next generation of computer scientists.

I also would like to thank all my co-authors throughout the years, especially, I want to thank Felice De Luca here, as we spent a lot of time at Skype together.

Once again, I want to thank Felice De Luca and Lukas Bachus for their work on the *ply-number*. Their work gave me a good foundation to continue.

Thanks to all the proof-readers for their time and effort as well as encouraging words. The award for most found spelling mistakes goes to Anja Dollinger.

I want to give a great thanks to my friends who kept me sane thought the harder times. To highlight a few: Vladimir Piven, Alexander Seitz, my (ex-)smoking companions for nice discussions and my motorbikers for distraction.

Here, I want to thank the one person, who stood by my side during all the time. Dear Marie, thank you for your patience, your support and all the good meals, I would have missed without you.

Last but not least I want to thank my family for their overwhelming, unconditional and never hesitating support.

Finally, I want to thank all the others, I have forgotten to mention, for their indulgence.

Zusammenfassung

Heutzutage sind Graphen als fundamentales mathematisches Modell zur Veranschaulichung von Relationen zwischen Objekten aus der Informatik nicht mehr wegzudenken. Ihre Anwendungen reichen von der Analyse sozialer Netzwerke über Routenplanung bis hin zu Interaktionsmodellen in der Biologie oder Chemie. Üblicherweise werden die Beziehungen zwischen den Objekten durch eine Zeichnung des Graphen veranschaulicht.

In dieser Arbeit wird die Ply-Zahl einer Zeichnung, welche als ästhetisches Kriterium vorgeschlagen wurde [30, 36, 42], untersucht. Gegeben sei eine gradlinige Zeichnung Γ eines Graphen $G = (V, E)$ in der Ebene. Für jeden Knoten v wird die Ply-Scheibe D_v als Scheibe definiert, wobei das Zentrum der Scheibe auf dem Knoten v liegt und der Radius r von D_v halb so lang ist, wie die längste inzidente Kante zu v. Die Ply-Zahl einer Zeichnung entspricht der maximalen Anzahl der sich überlappenden Ply-Scheiben in der Ebene. Zusätzlich beschreibt die maximale Anzahl an Scheiben, in denen sich ein Knoten befindet, die Knoten-Ply-Zahl.

Diese Arbeit beschäftigt sich mit theoretischen und praktischen Aspekten der Ply-Zahl von Zeichnungen. Es werden Eigenschaften von Zeichnungen präsentiert, die eine konstante Ply-Zahl haben. Zudem wird der Zusammenhang zwischen der Ply- und Knoten-Ply-Zahl einer Zeichnung untersucht.

Ein weiteres Kapitel widmet sich Graphen, welche Zeichnungen mit Knoten-Ply-Zahl 1 haben, sowie Graphen die keine Zeichnung mit Knoten-Ply-Zahl 1 haben. Insbesondere wird durch eine ausführliche Fallunterscheidung gezeigt, dass der vollständige Graph K_8, sowie der vollständig bipartite Graph $K_{3,15}$ nicht mit Knoten-Ply-Zahl 1 gezeichnet werden können.

Wir stellen ein Programm vor, dass es dem Nutzer ermöglicht ein intuitives Verständnis der Ply-Zahl als Parameter von geradlinigen Zeichnungen zu erlangen. Eine Zeichnung kann interaktiv modifiziert werden, was unter anderem auch für die theoretische Untersuchungen nutzbar ist.

Dazu präsentieren wir einen schnellen Algorithmus, der es ermöglicht zur Ply-Zahl sofortige Rückmeldung bei etwaiger Konfiguration der Zeichnung zu erhal-

ten. Das war mit bisherigen Implementationen nicht möglich [7, 30]. Die Zeit zur Berechnung der Ply-Zahl für eine Zeichnung konnte von Sekunden auf Millisekunden verbessert werden. Das Programm beinhaltet zusätzlich verschiedene Layout-Mechanismen und die Möglichkeit die Ply-Zahl automatisiert zu optimieren. Mit verschiedenen Experimenten untersuchen wir die Layoutmethoden bezüglich der Ply-Zahl.

Neben der Ply-Zahl von geradlinigen Zeichnungen betrachten wir auch Bar Sichtbarkeitsrepräsentationen. In einer Bar (k, j)-Sichtbarkeitrepräsentation werden Knoten als horizontale Liniensegmente, genannt Bars, dargestellt und Kanten sind gegeben durch vertikale Segmente zwischen ihren inzidenten Knoten, sodass jede Kante höchstens k Bars kreuzt und jeder Bar von höchstens j Kanten gekreuzt wird.

In vorangehenden Publikationen wird der Begriff Bar k-Sichtbarkeit unterschiedlich verwendet [15, 44, 83]. Wir geben die Möglichkeit diese Definitionen zu unterscheiden, und zwar in Bar $(1, \infty)$- und Bar $(1, 1)$-Sichtbarkeitsrepräsentationen.

Zudem beschäftight sich diese Arbeit mit maximalen Bar $(1, j)$-Sichtbarkeits Graphen. Es kann eine Hierarchie dieser Graphklassen angegeben werden. D.h. für alle j gibt es Graphen mit Bar $(1, j + 1)$-Sichtbarkeitsrepräsentation, der keine Bar $(1, j)$-Sichtbarkeitsrepräsentation hat.

Des weiteren werden Bar $(k, 1)$-Sichtbarkeitsrepräsentationen untersucht und wir können zeigen, dass es eingebettete Graphen mit Bar $(k, 1)$-Sichtbarkeitsrepräsentation für ein $k > 1$ gibt, welche aber keine Bar $(1, 1)$-Sichtbarkeitsrepräsentation haben. Unsere Ergebnisse sind die ersten zu dieser Graph Klasse, welche eine Klasse von nicht-planaren Graphen mit wenigen Kanten ist.

Abstract

Graphs are a fundamental mathematical model to represent relationships between objects and are used throughout a huge variety of disciplines in theory and practise. The use of graphs ranges in computer science from social network analysis to molecular interaction modelling in biology or chemistry. Accessing data from a graph very often involves a drawing of the graph. The *ply-number* has been defined as an parameter to evaluate the readability of a drawing [30, 36, 42].

Given a straight-line drawing Γ of a graph $G = (V, E)$ in the plane, for every vertex v the *ply-disk* D_v is defined as a disk, centered in v, where the radius r_v of D_v is half the length of the longest edge incident to v. The maximum number of overlapping disks at any point in the plane defines the *ply-number* of the drawing and the maximum number of overlapping *ply-disks* at any vertex defines *vertex-ply-number*. In this thesis, we present theoretical results on the *ply-number* and *vertex-ply-number* of drawings. We identify graphs, which have drawings with constant *ply-number* and evaluate the relationship between the *ply-number* and the *vertex-ply-number* in drawings.

To evaluate graphs that do or do not have *empty-ply* drawings, that is a drawing with *vertex-ply-number* 1, we present a comprehensive case distinction to show that the complete graph K_8 and the complete bipartite graph $K_{2,15}$ do not have *empty-ply* drawings.

We develop a supportive tool to give the user an intuitive understanding of the *ply-number* of straight-line drawings. In particular, we present a fast algorithm to compute the *ply-number* of a given drawing to enable instant feedback to the user, which was not possible with previous implementations [7, 30]. Furthermore, our tool is equipped with different layout methods and a workflow to optimize the *ply-number* for a given graph.

We evaluate our algorithm to compute the *ply-number* and the optimization of drawings regarding this value with an extensive set of experiments. In fact, we were able to reduce the time to compute the *ply-number* from seconds to milliseconds.

Beside the *ply-number* of straight-line drawings, we investigate *bar-visibility* rep-

resentations of graphs. A bar (k, j)-visibility drawing of a graph G is a drawing of G, where each vertex is drawn as a horizontal line segment, called bar, and each edge is drawn as a vertical line segment between its incident vertices such that each edge crosses at most k bars and each bar is crossed at most j times.

We clarify the differences between previous definitions on bar k-visibility representations [15, 44, 83], as they relate to bar $(1, \infty)$- or bar $(1, 1)$-visibility representations.

In this thesis, we especially look at maximal bar $(1, j)$-visibility graphs and conclude a hierarchy of these graph classes. That is, there exist graphs, which have a bar $(1, j + 1)$-visibility representation, but no bar $(1, j)$-visibility representation for every j. We investigate bar $(k, 1)$-visibility representations and show that there exist embedded graphs that are bar $(k, 1)$-visible but not bar $(1, 1)$-visible for $k > 1$. These are the first results on bar $(k, 1)$-visibility drawings, which is a class of sparse graphs beyond planarity.

Contents

Chapter 1

Introduction

Throughout a huge variety of applications and disciplines computer science became an important factor in the last century. Due to the fast development of technology, nowadays computers are used everywhere to compute, analyze, predict, and optimize processes and data. To represent and access data we need efficient models and data structures. Graphs are the common mathematical model to represent relationships between objects and occur throughout many applications and disciplines. In general, a graph is defined consisting of a set of objects V and a set of binary relations from $V \times V$. In graph theory the objects will be called vertices and the relations will be called edges. An example is presented in Figure 1.1. Next to the structure of a graph there can be further information stored in the vertices or at the edges. A common example is a cost function which assigns costs to edges. On these graphs we often look for a low cost connection between vertices.

The use of graphs range from social networks, where vertices represent individuals and edges represent a social connection i.e. friendship or kinship [12],via electronic circuits [93], to interaction graphs between molecules in biology or chemistry [82].

A famous example is the Facebook-Graph (see Figure 1.2) presented by Paul Butler [24]. The friendship relations across the world can be observed due to the mapping of persons to geographic locations. The friendship relations are presented as arcs according to the geographical distance between the friends. The information

```
V = {1,2,3,4,5,6,7}
E = {(1,2), (1,7), (2,3), (2,5),
     (3.4), (5,6), (5,7), (6,7),
     (4,7)}
```

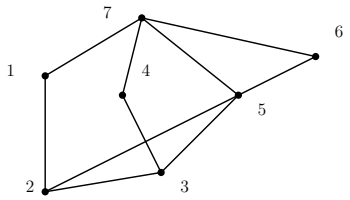


Figure 1.1: A graph $G = (V, E)$ represented as list of vertices and list of edges on the left. On the right side is a drawing in the plane of this graph.

Figure 1.2: The members of Facebook (2010) were mapped to their geographic locations and the friendship relations are presented by arcs dependent on the geographic distance between the persons. This Figure is taken from [24].

we can get from this drawing is an overview of the regions where Facebook is commonly used, whereas we cannot get information on individuals. A further interesting note is that it is possible to recognize the shape of the continents. Other well known examples where the geographical positioning is important are road maps [42] or metro maps [23]. An example for relative positioning according to the geographical data are maps of bus stops as shown in Figure 1.3. Even though these maps do not reflect real distances, the reader can maintain his orientation and access the information he needs.

In biology the interaction between molecules can be represented as graph, with molecules as vertices and interactions as edges [66]. Figure 1.4 presents the combination of 17 metabolic pathways which where altered during an infection [82]. The metabolic pathways include amino acid metabolism, energetic metabolism, fatty acid metabolism and carbon metabolism.

In chemistry we can model molecules as a set of connected atoms and the edges correspond to chemical bonds [8, 11, 91, 92]. These graph representations are used to predict the protein's structure and interaction potentials.

To make the data stored in a graph accessible for humans, we need a graphical representation which usually involves a drawing of the underlying graph. A drawing of a graph is a mapping of vertices to the plane and connecting curves between vertices represent the edges. There exist several methods to draw graphs [34, 86] and several aesthetic criteria have been defined [72, 73, 74, 75, 94].

Aesthetic criteria specify properties of the drawing to achieve readability. Besides

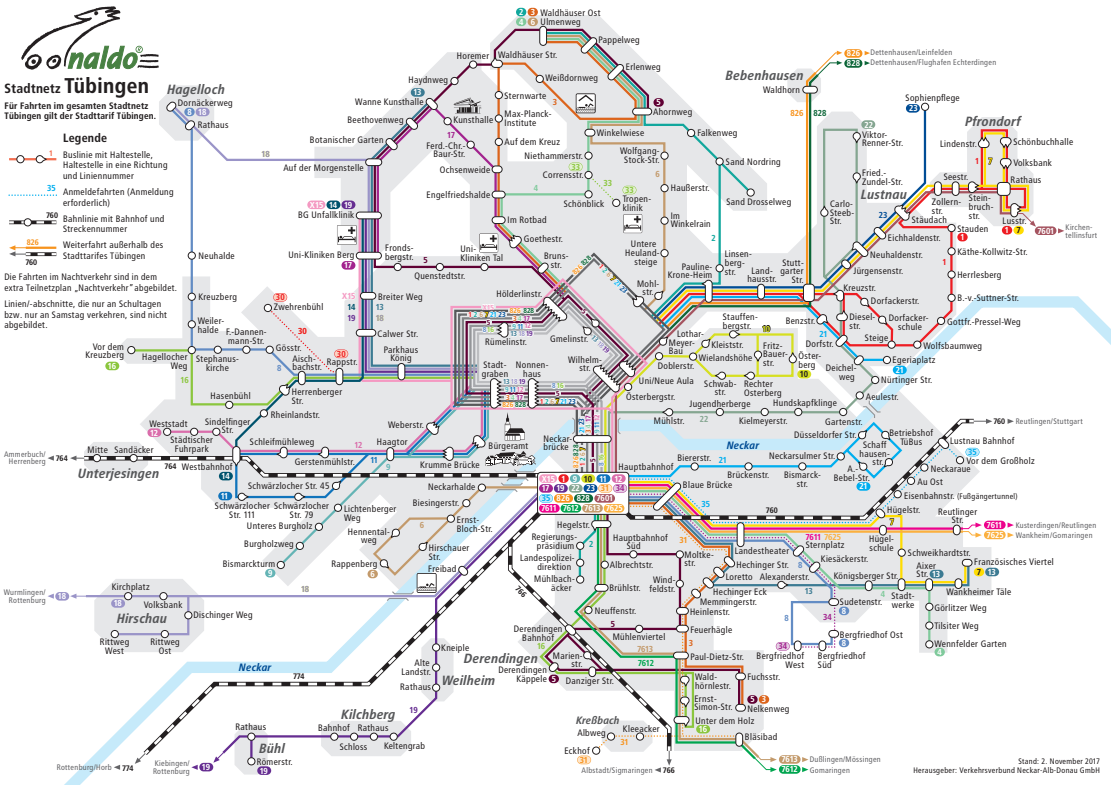


Figure 1.3: Map of public transportation in Tübingen, published by Verkehrsverbund Neckar-Alb-Donau [38]. Note that this map does not reflect the real geography of Tübingen but maintains the relative positioning between the bus stops.

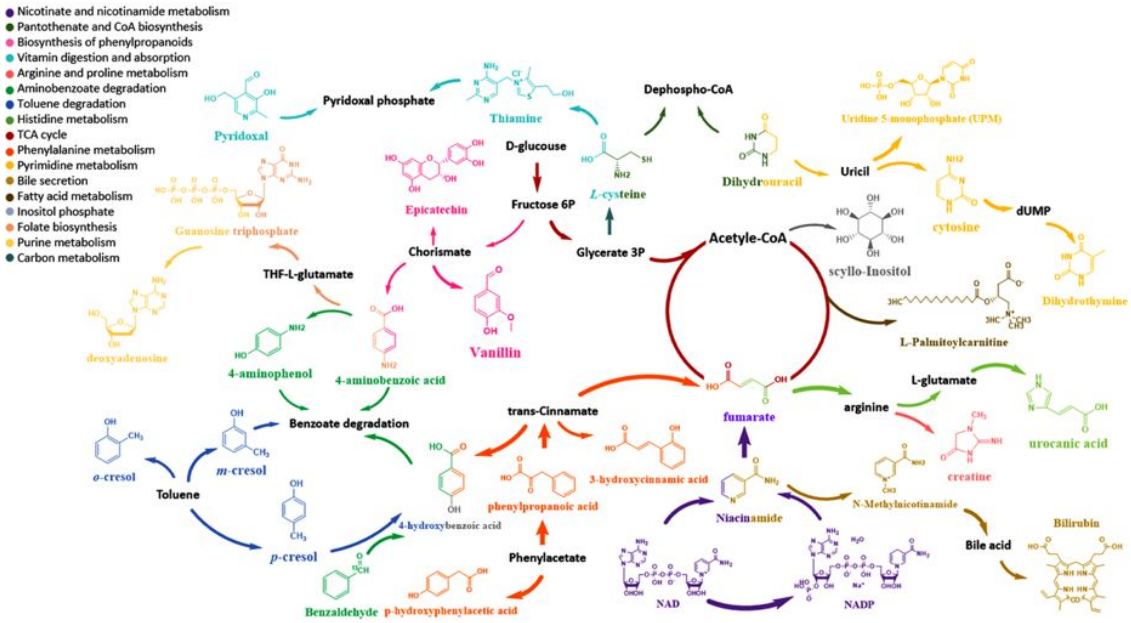


Figure 1.4: 17 interacting metabolic pathways are shown. These pathways were altered during a treatment [82].

drawing vertices at different coordinates to assure that they are distinguishable, commonly used criteria are area requirements, number of crossings, maximum edge length, uniformity of edge lengths or angular resolution [6, 33, 34]. In the following, we mention only a few aspects of aesthetic criteria, which we will refer to in this thesis. For a complete overview we would like to refer to the textbook by Di Battista et al. [34].

The area of a drawing is one important factor. On the one hand, the drawing should fit on the screen, on the other hand it should be readable. Connected vertices should be close enough to each other such that the reader can observe the connection whereas the vertices should be far enough apart so connections can be distinguished. Closely related to area requirements of a drawing are edge lengths constraints. Edge lengths can range from uniformity to exponential edge ratios.

The other most considered aesthetic criterion is the number of crossings per edge in graph drawings [26, 59, 75, 94]. The graphs that can be drawn without crossings are called planar graphs. This set of graphs is classified by K_5 and $K_{3,3}$ minor free. That is they do not contain one of these graphs as a minor subgraph. Testing whether a graph can be drawn planar is possible in linear time. Allowing at most one crossing per edge results in the class of 1-planar graphs [76, 94] where K_5 and $K_{3,3}$ are minimal examples of 1-planar graphs which are not planar. The number of edges for 1-planar graphs is bounded by $4n - 8$ where n is the number of vertices. The recognition of k-planar graphs is known to be NP-hard for any $k \geq 1$ [25, 52, 64].

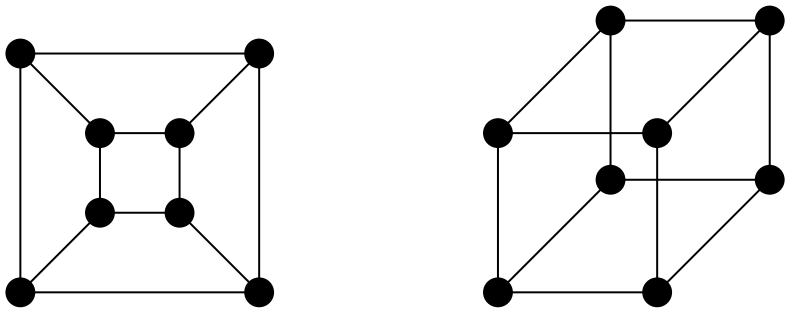


Figure 1.5: The cube graph is drawn planar on the left side. On the right side we have 2 right angle crossings and all edges have the same length. The angular resolution is 45° in both drawings.

The angular resolution of a planar straight-line drawing is the minimum angle formed by two edges incident to the same vertex. The angular resolution is bounded by the maximum degree in the graph. Closely related to the angular resolution is the crossing resolution. This describes the minimal angle formed by two edges crossing each other. A special class of graphs requires the crossing resolution to be 90°. These graphs are called *right angle crossing* (RAC) graphs, an example is presented in Figure 1.5. Note that in the previous Figure 1.3, showing the map of public transportation in Tübingen, the crossing angles are large. Besides time complexity limitations, the above aesthetics are also competitive in that the optimality of one often prevents the optimality of others [33].

A commonly used approach to draw graphs planar and beyond are force-directed algorithms. A subclass of these are spring embedders, mimicking the edges as physical springs with a natural length. For a physical model we can compute the entropy, called *stress*. Minimizing stress in a drawing often results in an aesthetically pleasing drawing [49]. Physically large graphs have higher entropy than smaller graphs and symmetric structures have less entropy than irregular structures.

Proximity graphs have numerous applications to describe a underlying structure of a set of points in computer graphics, computational geometry, pattern recognition, computational morphology, numerical analysis, computational biology, and geographical information systems (see, e.g., [51, 65, 69]).

The class of *proximity graphs* are known to be aesthetically pleasing in terms of planarity, edge length uniformity and vertex distribution [86]. In *proximity drawings* vertices are mapped to a metric space and edges are drawn as a straight-line segment. In a metric space distances between any two points are well defined and the triangle inequality holds. In proximity graphs vertices are connected if they are "sufficiently" close to each other. The closeness can be defined in various ways, i.e. *globally*, e.g. Euclidean minimum spanning trees [13] or *locally*, e.g. relative neighbourhood graphs [61, 90], Gabriel graphs [48], and Delaunay triangulations [32]. In any definition of *proximity* vertices are connected if and only if a defined region,

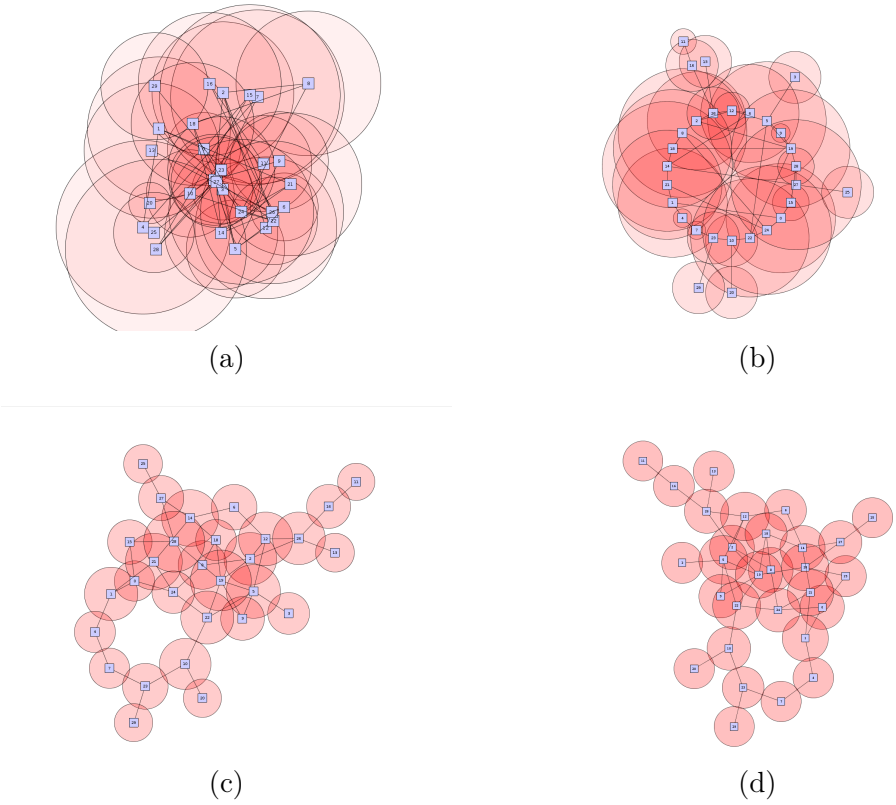


Figure 1.6: A graph with 30 vertices and 40 edges is drawn by our tool in different layouts: (a) randomly placed vertices with ply 15, (b) circular layout with ply 7, (c) organic layout with ply 4, (d) the lowest known ply number of 3 for this graph. These figures are created by our tool presented in Chapter 3.

called *proximity region*, is empty in terms of there is no "closer" vertex. Geometrical graphs such as road maps or metro maps are closely related to proximity drawings since usually cities in close proximity are connected by roads.

Recently, a new parameter called *ply-number* was introduced as a quality metric for graph layouts [36]. Given a straight-line drawing Γ of a graph $G = (V, E)$, for every vertex v the ply disk D_v is defined as a disk centered at v, where the radius of the disk is half the length of the longest edge incident to v. The *ply-number* of Γ is the maximal number of overlapping disks at any point in $\mathbb{R}^2$. Clearly, the *ply-number* depends on edge-length uniformity and vertex distribution in the plane.

The correlation between the *ply-number* of drawings and other known metrics to evaluate the aesthetics of a drawing has been shown in a recent study by De Luca et al. [30]. This study also confirmed that force-directed algorithms tend to produce visually pleasing drawings as suggested in [62]. Some drawings of the same graph with different *ply-number* are shown in Figure 1.6.

In general, road maps can have crossings and proximity drawings cannot. The key observation by [36] is that the *ply-number* of road maps is low. We can define the *ply-disk* to be a *proximity region* where no vertex is allowed to be inside another vertex's *ply-disk* . These graphs are called *empty-ply* graphs. As relaxation of *empty-ply* drawings we define the *vertex-ply-number* to be the maximal number of overlapping *ply-disks* at any vertex.

Theoretical results have been obtained on the *ply-number* of graphs [3, 36] and there exist many geometrical graphs admitting natural drawings with low *ply-number* [42].

To compute the *ply-number* for a given drawing, a first prototype of the basic plane-sweep algorithm has been implemented in [7]. We reimplemented the algorithm, improved it and added new features. The experimental study of De Luca et al. [30] set a benchmark on the computation of the ply number for a given drawing. The authors evaluated several layouts by force directed algorithms according to the ply number. To make our comparison possible, the authors kindly provided some of their data. Both previous implementations of the ply computations are based on the Apfloat library [89] which allows calculations on high precision levels at the cost of time.

After motivating the *ply-number* by geometrical drawings we will now sketch the important concept of *bar-visibility* drawings. Note that in these drawings the relative positioning of the vertices is not predetermined. In a visibility drawing vertices are represented as objects in the plane and two vertices are connected if they can "see" each other. Visibility representations and especially *bar-visibility* representations are well studied for planar graphs.

In *bar-visibility* drawings vertices are represented as disjoint horizontal line segments and edges are vertical line of sights of non zero width [50, 86]. This model can be extended to beyond planarity by allowing the vertical edges to cross bars. We define two parameters, namely bar (k, j)-visibility, where any edge can cross at most k bars and any bar can be crossed by at most j edges. An example for a bar $(1, 2)$-visibility drawing is presented in Figure 1.7.

In *bar-visibility* we distinguish between the weak and the strong model. In the strong version of *bar-visibility* an edge is present if and only if there exist a line-of-sight whereas in the weak model an edge might exist if there is a line-of-sight. *Bar-visibility* is part of a very basic paradigm in graph drawing approaches for orthogonal or poly-line drawings. The so called visibility approach consisting of a planarization step, where crossings are replaced by dummy vertices, then a computation of a *bar-visibility* representation of the new graph. As a last step the horizontal and vertical line segments are replaced [34].

This technique is heavily used in VLSI and chip design [50], because many wires are routed similarly. There are many different approaches to drawing orthogonal graphs. Early results draw the graph using few bends but sacrificed size or

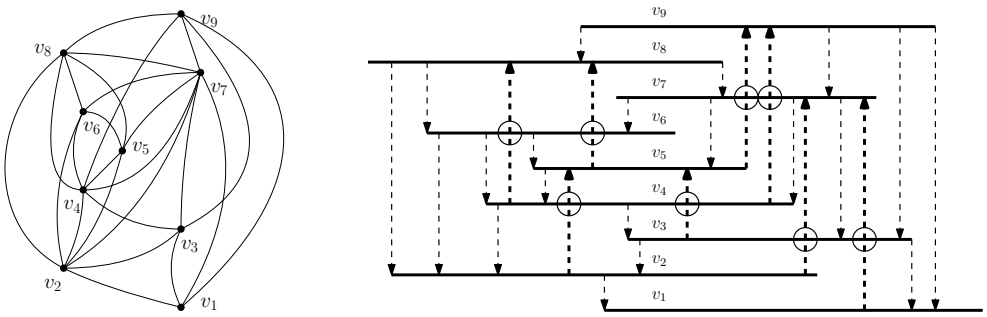


Figure 1.7: (a) A graph with 9 vertices and several crossing edges is drawn in the plane. The same graph is drawn as a *bar-visibility* representation in (b). For clarification planar edges are indicated by a downward pointing arrow and edges which are crossing bars by upward pointing arrows. We mark the crossings by small circles. Note that some bars are crossed twice. This is a bar $(1, 2)$-visibility representation.

running-time efficiency. Improved techniques, involving computing a visibility representation, yielded orthogonal drawings in linear time with few bends and small size [39, 43, 70, 78, 87].

In the previous applications *bar-visibility* representations were computed on planar graphs. The extension to bar k-visibility by Brandenburg [15], Evans et al. [44] and Sultana et al. [83] introduced *bar-visibility* representations beyond planarity. Note that these works have a slightly different definition of bar k visibility.

1.1 Outline of this Thesis

We start by presenting theoretical results on the *ply-number* and *vertex-ply-number* of graphs in Chapter 2. We give an introduction on previous results and focus on graph classes with known bounds, in particular for drawings with constant *ply-number* 1 and 2. Clearly the *ply-number* for any drawing is linearly upper bounded, since every vertex has exactly 1 *ply-disk*. We present a technique to guarantee a drawing with *ply-number* of at most $\frac{n}{2}$ for an graph with n vertices. Later we point out the relation between the *ply-number* and the *vertex-ply-number* in detail, followed by an introduction to *empty-ply* drawings, that is drawings with *vertex-ply-number* 1. We investigate *empty-ply* drawings regarding their properties. We deduce edge-length ratios, maximal vertex-degree and area requirements for these drawings. We conclude this chapter with a section on graphs with or without *empty-ply* drawings.

We continue in Chapter 3 by presenting our algorithm to compute the *ply-number* for given drawings. We investigate different layout techniques and compare

the *ply-numbers* of the drawings. There exist many tools and layout algorithms for graph drawing provided e.g. by OGDF [27] or the yFiles library [97].

Our main goal was to present a supportive tool to give the user an intuitive understanding of the *ply-number* and how it is affected by manual changes of the drawing. The identification of graphs with low *ply-number* as well as the development of strategies to optimize parameters of drawings involve frequent examination of graph drawings. Therefore we use our fast and accurate algorithm to compute the *ply-number* and present an experimental setting to evaluate optimization strategies.

Our fast algorithm to compute the *ply-number* for a given drawing based on a plane-sweep algorithm which is known to be a powerful technique in computational geometry. Furthermore we provide methods to modify the drawing to reduce the *ply-number* interactively as well as automatically.

In Chapter 4 we clarify the differences in the definitions of bar k-visibility by Brandenburg [15], Sultana [83], and Evans [44], by introducing bar (k, j)-visibility. Recall that in a bar (k, j)-visibility representation any edge is allowed to "see through" k bars and each bar can be crossed by at most j edges.

We present maximal bar $(1, j)$- visibility graphs for different j and conclude that there exists an infinite hierarchy. That is, for any $j \geq 1$ there exists a graph which is bar $(1, j)$, but not bar $(1, j - 1)$-visible. Furthermore, we investigate maximal graphs with low density. These graphs have few edges per vertex-segment but adding any edge would contradict the bar-visibility constraints. Parts of this thesis have been published in [4, 16], and [56].

We conclude with a short summary on our results and give an outlook to open problems in Chapter 5.

Chapter 2

Theory of Ply

In this chapter we formally introduce *ply* concepts and theoretical results on the *ply-number* of straight-line drawings in general. Particularly, we summarize and extend our results on *empty-ply* drawings as published in [4]. Given a straight-line drawing of a graph $G = (V, E)$ in the plane, we define the *ply-disk* D_v for every vertex $v \in V$. The disk is centered at v and has a radius r equal to half of the length of the longest incident edge of v. The *ply-number* of a drawing is defined by the maximum number of overlapping *ply-disks* for any point in $\mathbb{R}^2$. Furthermore, the *vertex-ply-number* denotes the maximum number of overlapping disks for any vertex $v \in V$. Both concepts are indicated in Figure 2.1. Note that the *vertex-ply-number* is at least one, if the graph has at least one edge, since every vertex is contained in its own *ply-disk*. We require all vertices to have distinct coordinates in the plane, since otherwise the length of an edge and thereby the radius of a *ply-disk* might be 0. This is a common understanding in graph drawing. The drawings of graphs where each vertex v is contained in exactly one *ply-disk*, namely its own *ply-disk* D_v are called *empty-ply* drawings.

In this chapter we will focus on our main contribution to [4], namely the proof that the graph K_8 does not admit an empty-ply drawing, whereas K_7 does. As our second contribution, we will present the proof that any complete bipartite graph $K_{2,n}$ where $n \geq 15$ does not admit an *empty-ply* drawing, while we know an *empty-ply* drawing of $K_{2,12}$.

2.1 Introduction

The *ply-number* for a given straight-line drawing has been recently introduced as a parameter to evaluate the aesthetic appearance of a drawing [3] [30] [36].

The foundation for the definition of the *ply-number* as aesthetic criterion was aroused by the study of road networks. Road networks can be described as embed-

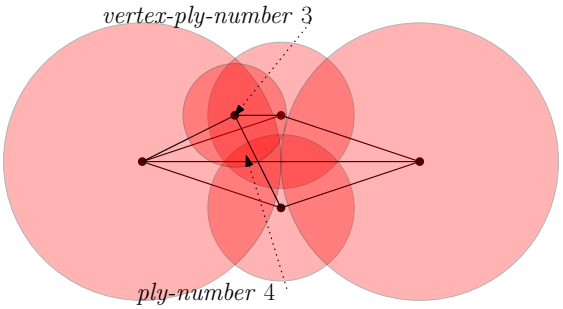


Figure 2.1: A drawing with *ply-number* 4 and *vertex-ply-number* 3.

ded graphs on the surface of a sphere and are not necessarily planar.

In the work by Eppstein and Goodrich [42], the authors propose to characterize road networks as subgraphs of disk-intersection graphs. During the paper, the authors had a close look to the road network of the state California including more than 6,000 edge crossings. They observed that the maximum number of overlapping *ply-disks* was surprisingly low. Thereby, although road networks are not planar, they have low *ply-number*.

To support the value of the *ply-number* as an aesthetic criterion we can state the relation between the maximum number of overlapping *ply-disks* and the distribution of the vertices in the plane as well as the distribution of different edge lengths throughout the drawing. The *ply-number* turns out to be low if the vertices are distributed evenly in the plane and the factor of different edge-length is close to 1. In the following sections we will summarize results regarding graph classes that admit drawings with constant *ply-number*, logarithmic *ply-number* and present an upper bound on the *ply-number* which is $\lfloor \frac{|V|}{2} \rfloor$ for any graph $G = (V, E)$.

Reading about visual criteria in graph drawing minimizing edge crossings is the most cited and the most commonly used aesthetic [59, 72, 94]. In conformity with force-directed and stress-based algorithms, crossing-free layouts for planar graphs are not necessarily required. In fact, since edge crossings are not explicitly considered, many planar graphs (e.g. trees, nested triangles, nested squares and skeletons of 3D polyhedra, such as the cube, the octahedron, the dodecahedron) invariably have crossings in force-directed layouts. At the same time, it can be observed that such layouts have low *ply-number*.

Since concepts observed in road networks are closely related to proximity drawings, the idea of *empty-ply* drawings was suggested. An *empty-ply* drawing is a drawing where the *ply-disk* for every vertex is not allowed to contain any other vertex than its center itself. A proximity drawing of a graph is a straight-line drawing in which for any two vertices u and v connected by an edge (u, v) there exists a region, called *proximity region*, which does not contain other vertices in its interior. Proximity drawings are highly restricted in edge and vertex distributions.

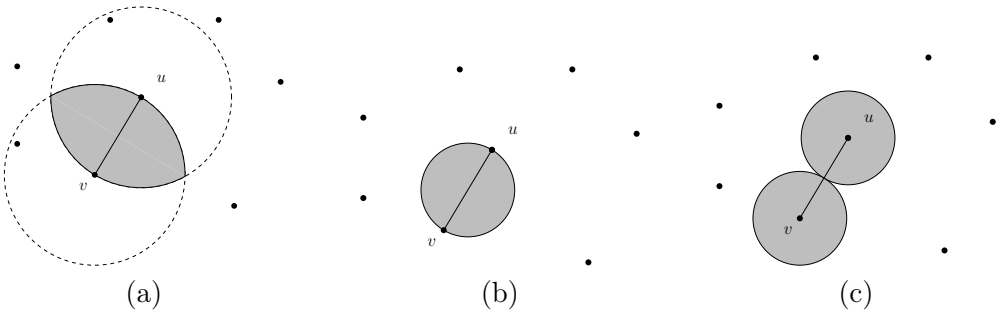


Figure 2.2: The grey area marks the *proximity region* in a relative neighbourhood graph (a). There exists an edge between the vertices u and v if there does not exist a vertex w which is closer to u than to v and closer to v than to u. The *proximity region* can be described as intersection of two disks. (b) In a Gabriel graph the *proximity region* for any edge is the smallest circle including the vertices u and v. (c) The *ply-disks* defined by the edges can be seen as *proxmitity regions*. Note that in contrast to the other examples here each *proximity region* contains one vertex.

Furthermore, if a graph $G = (V, E)$ admits a proximity drawing, G is called *proximity graph*. In general a proximity region defines an area in the plane, where any point is closer to the vertices u and v than to any other vertex of the graph. This property can vary due to different metrics and thereby different definitions of proximity regions induce different sets of proximity drawings [65].

A proximity region can be defined *global*, e.g. Euclidean minimum spanning trees [13] or *local*, e.g. relative neighbourhood graphs [90, 61], Gabriel graphs [48], and Delaunay triangulations [32]. On a set V of vertices in the plane, a Euclidean minimum spanning tree connects the vertices, such that the sum of all lengths of the connections is minimal. Thereby the proximity regions are defined globally in a sense that there cannot exist a vertex such that using a different connection results in a lower global cost. In a *relative neighbourhood* drawing there exists an edge between two vertices u and v if there does not exist a third vertex w which is closer to u than v and closer to v than u. The *proximity region* for this model is shown in Figure 2.2a.

In a Gabriel graph the *proximity region* of one edge is defined to be the smallest circle including the connected vertices as shown in Figure 2.2b. Gabriel graphs have been a relevant topic in geographic variation research [67]. It is known that not all degree 4 graphs have Gabriel drawings [14] and in correlation to this result we will sketch the proof that there exist 4-ary trees which do not admit an *empty-ply* drawing. For the interested readers the summary by Giuseppe Liotta [65] gives a nice introduction on the general topic of *proximity drawings*.

To study proximity drawings, there exist two general models. In the *strong* model, an edge between two vertices u and v exists if and only if the corresponding *proximity region* is empty. In the *weak* version of this model, an edge might exist,

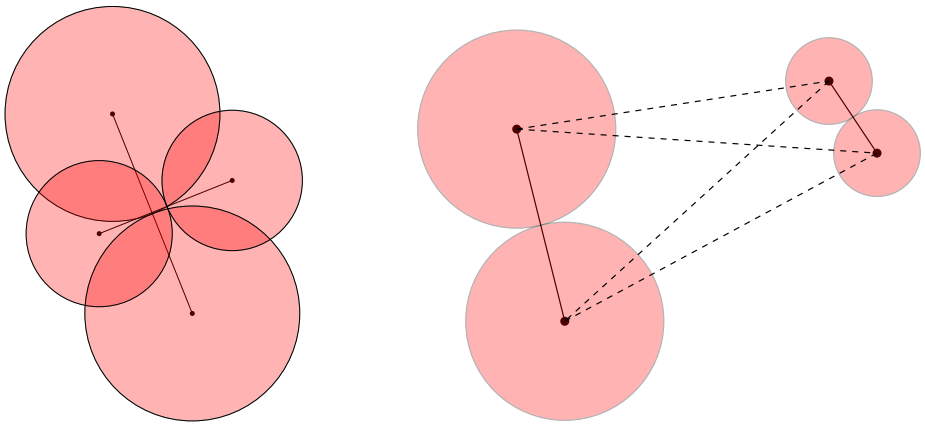
if the *proximity region* is empty. Thereby the *weak* model is the more general version [35].

Recall that the *ply-number* of a graph is defined by the maximal number overlapping *ply-disks* for any point in the plane. To look at *ply-drawings* from the proximity perspective, we define the *proximity region* to be equal to the *ply-disks* and more general introduce the parameter *vertex-ply*. Formally the *vertex-ply-number* is the maximal number of overlapping *ply-disks* for any vertex in the plane. The set of considerable coordinates is reduced from $\mathbb{R}^2$ to the coordinates of the vertices. By definition of the *ply-disk* there is always at least one vertex inside and thereby the *vertex-ply* of any drawing is at least 1 if the graph has at least one edge. This is one major difference towards the initial definition of *proximity* where the region has to be empty. The *ply-disks* as *proximity regions* are presented in Figure 2.2c. So we call a drawing where each vertex is contained only in its own *ply-disk* (*vertex-ply* $= 1$) an *empty-ply* drawing, since the *ply-disk* is empty in terms of other vertices. This definition is in a sense a weak model for *proximity drawings*. Note that, given an *empty-ply* drawing, the *ply-number* of the graph can be larger than 1. It is known, that in the most studied proximity definitions as the ones mentioned above, the drawings are planar. Crossing edges are not forbidden in *empty-ply* drawings as we can see in Figure 2.3a. Another difference is the connectivity of the graph. In *proximity* drawings the graphs are usually connected and in Figure 2.3b we present an unconnected graph where adding any edge would violate the *empty-ply* property. These two observations suggest, that empty-ply drawings are not directly comparable to previous defined types of proximity drawings.

There exist several results on the comparability of different proximity models. For example, Delaunay triangulations contain a Gabriel graph as spanning subgraph, which again contains a relative-neighbourhood graph as spanning subgraph, which again contains a minimum spanning tree as subgraph [65]. One natural question is to ask, whether *empty-ply* drawings can be included into this comparison. At this point we have to note, that *empty-ply* drawings might be non-planar drawings, which is not the case for any of the other proximity models. Furthermore, there exist *empty-ply* drawings which are not connected and cannot be extended by adding edges without violating the *empty-ply* property.

Furthermore, there exists a close relation between *empty-ply* drawings and *partial edge drawings* (PEDs) [20]. A PED is a straight-line drawing of a graph, where the edges are drawn partially in a way that the middle part of the edge is neglected and two edge segments incident to vertices, called *stubs*, remain in the drawing. The *stubs* are not allowed to cross. We present a nontrivial relation regarding the length of the stubs and *empty-ply* drawings [4].

In the following, we will give an overview on the closely related previous work regarding the *ply-number* of drawing. Then we will present some details on known



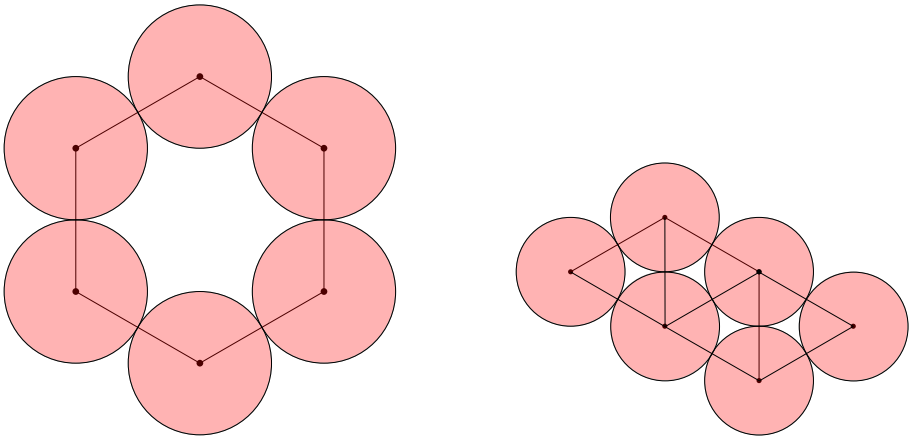
(a) (b)

Figure 2.3: In general *proximity graphs* are connected and planar. In terms of *empty-ply* these conditions are not necessarily true since in (a) we have a non-planar *empty-ply* drawing with *ply-number* 2. (b) An *empty-ply* drawing where the addition of any edge would violate the *empty-ply* property and thereby it is a maximal drawing which is not connected.

graph classes with constant *ply-number* and give an intuition on the relationship between the *ply-number* and the *vertex-ply-number*. We will present our main contribution, namely the proof that the complete graph K_7 does admit an *empty-ply* drawing whereas the complete graph K_8 does not. Thereby we can conclude that the complete graph K_n where n is the number of vertices does not admit an *empty-ply* drawing for any $n \geq 8$. In [4] we presented a proof for $n \geq 9$ and here we present a refined version for $n \geq 8$. Furthermore we go into the proof in detail, that the complete bipartite graph $K_{2,15}$ is not *empty-ply* drawable whereas $K_{2,12}$ can be drawn (cf. Theorem 2.15).

2.2 Previous Work

The *ply-number* of straight-line drawings was initially suggested by Eppstein and Goodrich during their study of road networks [42]. Since then there has been some activity on this topic. In the work Low Ply Drawings by Di Giacomo et al. [36], the authors show that the general recognition problem, namely whether a given graph admits a drawing with *ply-number* 1 is NP-hard since a graph can be drawn with *ply-number* 1, if and only if the graph has a *unit-disk* representation. Meanwhile there exists an $O(n \log n)$-time algorithm to decide whether an internally triangulated biconnected planar graph can be drawn with *ply-number* 1. The authors state some graph classes that admit drawings with *ply-number* 1. These are simple cycles (see Figure 2.4a) and internally triangulated biconnected planar graphs with maximum



(a) (b)

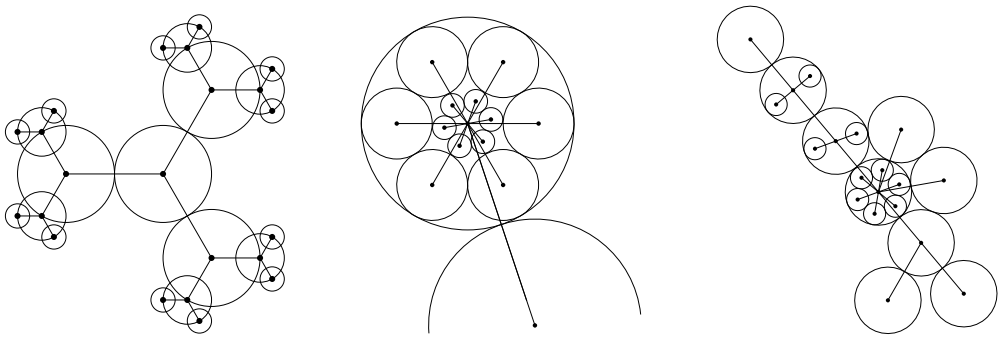
Figure 2.4: (a) a simple cycle can be drawn with *ply-number* 1, (b) an example for an internally triangulated biconnected planar graphs with maximum vertex degree 4 drawn with *ply-number* 1.

vertex degree 4. An example for such a graph is shown in Figure 2.4b. Furthermore, the authors present some graph classes that can be drawn with constant *ply-number*, namely *ply-number* 2, like binary trees (see Figure 2.5a), stars (see Figure 2.5b) and caterpillars (see Figure 2.5c). The stars can be constructed such that there is one layer of ≤ 6 vertices whose *ply-disks* do not overlap and all other vertices can be placed in a non overlapping fashion inside the star's center's *ply-disk*. The caterpillars can be reduced to a path with non overlapping *ply-disks* and for the legs the same rules as for the star can be applied.

Arising from this work are the questions whether it is possible to draw these graphs in polynomial area, if ternary trees can be drawn with constant *ply-number* 2 and if *empty-ply* drawings have a constant *ply-number*.

The first questions were tackled by the follow-up work 'Low Ply Drawings of Trees' by Angelini et. al. [3] and our results on the question regarding the *empty-ply* setting presented in [4] will be summarized in the next sections.

In [3], the authors proved an exponential lower bound on the area requirements of drawings with constant *ply-number* for stars and therefore for general caterpillars too. This result answers the first question. Furthermore the authors present trees with maximum degree 11 that are not drawable with constant *ply-number*. A natural consequence of these results is to check whether these graphs can be drawn with logarithmic *ply-number* and one of the results of [3] is an algorithm to construct a drawing of every tree with maximum degree 6 in polynomial area.



(a) (b) (c)

Figure 2.5: Graph classes that admit drawings with constant *ply-number* 2 e.g. (a) binary trees, (b) stars and (c) caterpillars.

2.3 Graph classes with bounded ply-number

In this section we start with a categorization of graphs and graph classes with a bounded *ply-number*. We explore properties of graphs that admit drawings with *ply-number* 1 and recall why the recognition of these graphs is NP-hard. Then we will shortly summarize the known results on stars, binary trees and caterpillars and give the rules to construct drawings of these graphs with *ply-number* 2. The construction of binary trees in this case is of special interest, since we construct *empty-ply* drawings of binary trees. Finally we give an upper bound on the *ply-number* for any graph.

2.3.1 *Ply-number* 1

A drawing Γ of a graph $G = (V, E)$ is a drawing with *ply-number* 1 if and only if two conditions hold:

1. every edge $e \in E$ has the length ℓ
2. the distance between any pair of vertices $u, v \in V$ is at least ℓ.

Let us assume for simplicity that the graph G is connected. Otherwise we can adjust the arguments for each connected component individually and ensure the correct distances between any two components.

We can prove the first condition by contradiction. Assume there exist edges with different lengths in the connected graph. Let the lengths be w.l.o.g. ℓ and $\ell - \epsilon$ for any $\epsilon > 0$. Using the connectivity of the graph there exists a vertex $v \in V$ whose incident edges have different lengths. Let the edge with length $\ell - \epsilon$ be (u, v) and the edge with length ℓ be (v, w) as illustrated in Figure 2.6. The radii are $\frac{\ell}{2}$ and

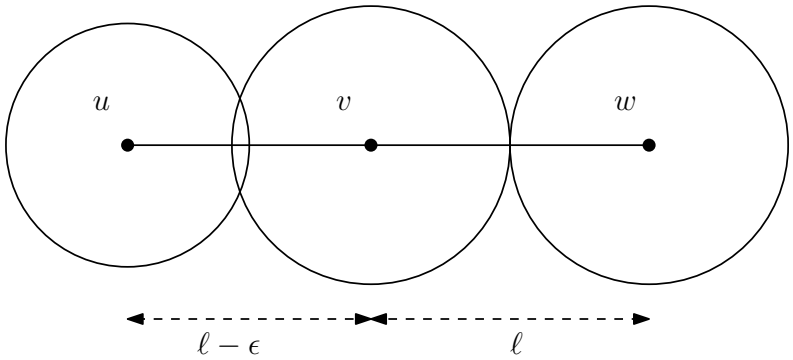


Figure 2.6: The vertex v has incident edges with the lengths ℓ and $\ell - \epsilon$. The radius of the *ply-disk* D_v is $\frac{\ell}{2}$ whereas the radius of $D_u \geq \frac{\ell-\epsilon}{2}$. The *ply-disks* D_v and D_u do overlap and thereby the *ply-number* of the drawing is at least 2.

$\frac{\ell-\epsilon}{2}$ and the disks of u and v overlap since

$$\frac{\ell}{2} + \frac{\ell - \epsilon}{2} = \frac{2\ell - \epsilon}{2} > \frac{2\ell - 2\epsilon}{2} = \ell - \epsilon.$$

We can conclude that for any vertex v all edges have to be the same length, namely ℓ and since the graph is connected this has to apply for all edges $e \in E$.

The second condition is rather obvious. Since the graph is connected, every *ply-disk* has the radius $\frac{\ell}{2}$ and thereby the *ply-disks* D_u and D_v for any pair of vertices $u, v \in V$ do overlap, if and only if the distance between u and v is less than $2 \cdot \frac{\ell}{2} = \ell$. Hence the distance between any pair of vertices has to be at least ℓ.

From the previous properties we can deduce some further properties of graphs that admit a drawing with *ply-number* 1, namely G has to be planar and the maximum degree for any vertex in G is 6 since the angular resolution between any two edges has to be greater or equal to 60°. The authors of [36] additionally exploit the relation between drawings with *ply-number* 1 and *unit-disk* contact representations. In a *unit-disk* contact representation of a graph $G = (V, E)$ every vertex is represented as a disk with unit radius. Recall that, disks are not allowed to overlap and an edge between u and v exists, if the corresponding disks touch. It is known to be NP-hard to recognize whether a given graph has a *unit-disk* contact representation [18, 41, 45].

In any drawing with *ply-number* 1 the *ply-disks* have equal (unit) radius and if there exist an edge (u, v) the *ply-disks* D_u and D_v touch in the center of the edge. Note that for *unit disk* graphs there exist a *strong* and a *weak* model. In the *strong* model there exists an edge if and only if the disks touch each other. Whereas in the *weak* model there might exist an edge if the disks touch. Thereby we consider the *weak* version of the *unit-disk* contact representation since we do not necessarily require the existence of an edge if the *ply-disks* touch. An example of the wheel

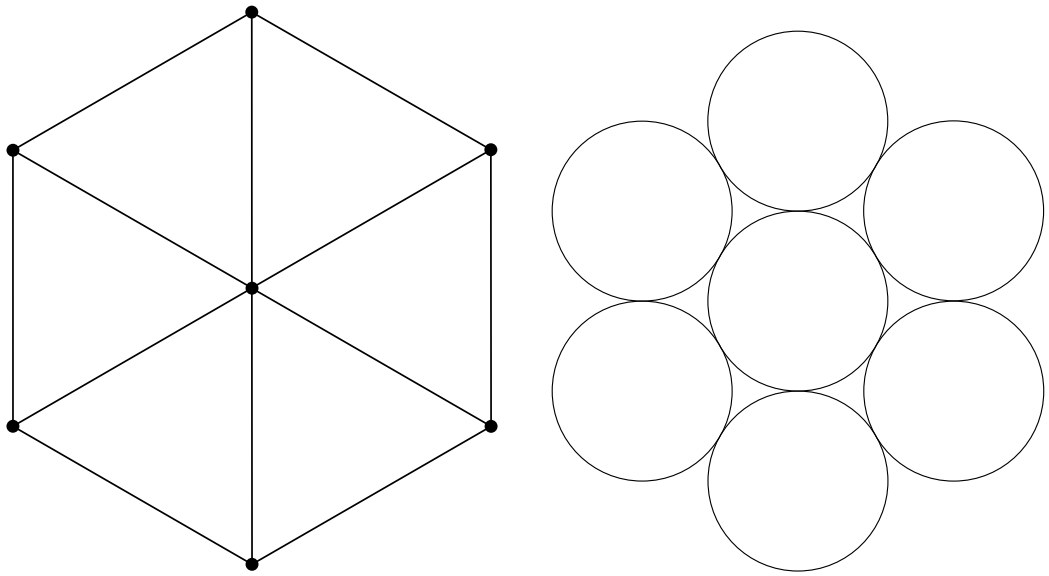


Figure 2.7: The wheel graph with $n = 7$ vertices (left) and its *unit-disk* contact representation (right).

graph with $n = 7$ vertices and its *unit-disk* contact representation is shown in Figure 2.7. Note that in the *strong* model of the *unit-disk* contact representation the wheel graph with $n = 7$ vertices minus one edge is not representable as *unit-disk* contact representation. whereas it would still be drawable with *ply-number* 1.

Known graph classes with these properties are paths, internally triangulated biconnected planar graphs with maximum vertex degree 4 [36], the wheel graph with $n = 7$ vertices.

2.3.2 *Ply-number* 2

Recall that for any drawing Γ of a graph $G = (V, E)$ with *ply-number* 2 at most 2 *ply-disks* overlap in any point of the plane. Let D be the set of all *ply-disks* of the drawing, then we can decompose D into two disjoint subsets D_1 and D_2 such that $D_1 \cup D_2 = D$ and $D_1 \cap D_2 = \emptyset$ and the *ply-disks* in D_1 (D_2 respectively) are independent of each other, meaning they do not overlap in any point in the plane. Note that *ply-disks* are still allowed to touch each other. By the work of Angelini et al. [3] we know that stars and binary trees admit drawings with *ply-number* 2.

Lemma 2.1. Any star can be drawn with *ply-number* 2 in exponential area.

Proof. Let the graph $G = (V, E)$ be a star. A star has a center $c \in V$ and $n - 1$ vertices where every vertex is connected to c by an edge. Formally the set of edges can be described as $E = \{(c, v)\ \forall v \in V \setminus \{c\}\}$. We can exploit the property that for any two edges $(c, v), (c, u) \in E$ the *ply-disks* D_v and D_u do not overlap if the

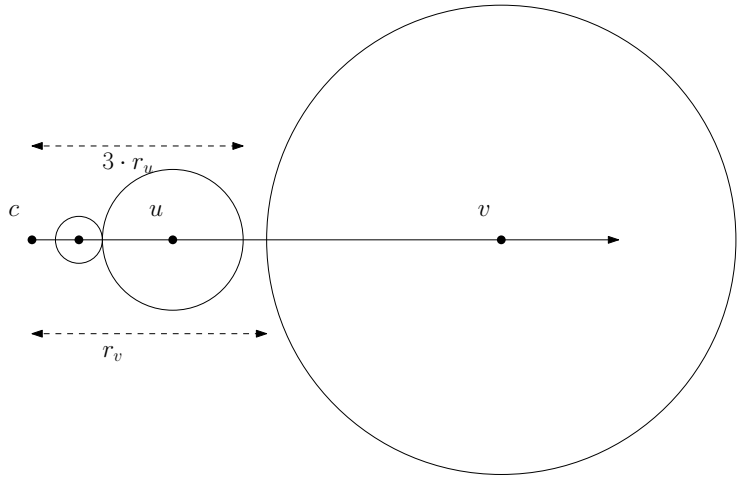


Figure 2.8: The *ply-disks* of two vertices which are not the center of a star do not overlap, if the length of the edges differ by the factor of at least 3.

lengths of the edges differ by a factor of at least 3. Note that in a star, the edges (c, u) and (c, v) define the radii r_u, r_v of the *ply-disks* D_u and D_v. Assume the vertices u and v to be on a line (ray) from the center vertex c of G as indicated in Figure 2.8. Let w.l.o.g. the vertex u be closer to c than v and assume for the contrary that the *ply-disks* D_u and D_v overlap. That is

$$\frac{3}{2}\,\mathrm{dist}(c,u) > \frac{1}{2}\,\mathrm{dist}(c,v) \Leftrightarrow 3r_u > r_v.$$

Thereby the *ply-disks* of the leaves do not overlap, if the distance for any two vertices differs by a factor of at least 3, namely

$$3r_u \leq r_v \; \forall\{u, v \in V |\; \mathrm{dist}(c,u) <\; \mathrm{dist}(c,v)\}.$$

Obviously the longest edge in this drawing will have the length in $\Omega(3^n)$. To improve the area requirements for such a drawing we can introduce the notation of layers. From the previous section we know that we can place up to 6 leaves at the same distance from c as indicated in Figure 2.5b. This does improve the area requirements but it is still exponential in the number of leaves [3]. By now we have shown that the *ply-disks* of the leaves can be drawn independently, meaning without any overlaps. The *ply-disk* of c is defined by the furthest layer and will completely include the *ply-disks* of the vertices in the inner layers.The ratio between the shortest and the longest edge in the drawing is exponential in the number of vertices and thus the area requirement of this drawing is exponential in the number of vertices. By the construction we can produce a drawing for any star with *ply-number* 2. □

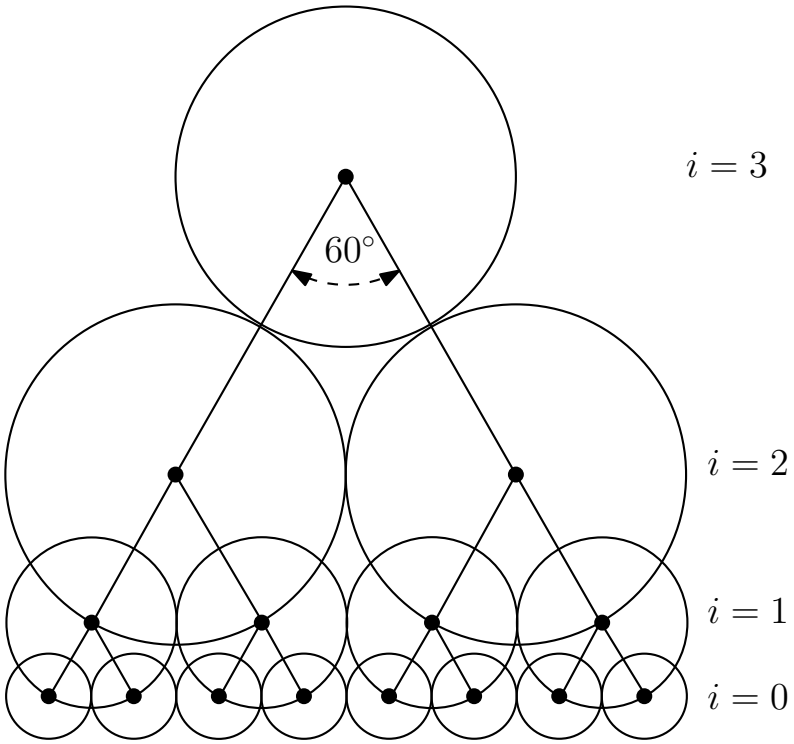


Figure 2.9: A binary tree is drawn in a level-by-level fashion. The angle of 60° between any two children is maintained.

Lemma 2.2. Binary trees can be drawn with *ply-number* 2 in exponential area.

Proof. Let $G = (V, E)$ be a complete binary tree. We give a construction and show that the drawing has *ply-number* 2 by showing that the *ply-disks* of independent vertices do not overlap. We draw the tree in a level-by-level fashion and reduce the lengths of the edges by the factor $\frac{1}{2}$ in each level. Meanwhile we maintain an angle of 60° between the two children of the parent as illustrated in Figure 2.9. Let w.l.o.g. the leaves at level $i = 0$ be connected to their parents by edges with length 1, then the radii of *ply-disks* at level i are $2^{(i-1)}$ except for the root, which has the same radius as its children. The total distance along the edges from the leaves to the vertices at level i can be written as $2^i - 1$. Note that all vertices at the same level are drawn at the same height and their *ply-disks* have the same radii. Furthermore the children and the parent form an equilateral triangle and thereby the *ply-disks* in the same level do touch but have no overlap. It is left to show that overlaps do not occur between levels that are not adjacent.

For this purpose we argue on the area requirement of the levels. Observe that the rightmost and the leftmost children at level i of a common ancestor at level k

form an equilateral triangle with lengths $2^k - 2^i$. The height h of this triangle is

$$h = \frac{\sqrt{3}}{2}\left(2^k - 2^i\right)$$

Thus disks at level i and level k do not overlap, if the sum of the radii is smaller than the height of the triangle.

$$r_i + r_k = 2^{i-1} + 2^{k-1} < \frac{\sqrt{3}}{2}\left(2^k - 2^i\right), \text{ if } 2^i \leq \frac{2^k}{4},$$

meaning $k \geq i + 2$. Thereby the *ply-disks* do not overlap, if the level differs by at least 2. We can now simply assign the *ply-disks* of the even levels to D_1, since they do not overlap each other. The *ply-disks* of the odd levels are assigned to D_2 and $D_1 \cap D_2 = \emptyset$ as well as $D_1 \cup D_2 = D$ follows. The ratio between the shortest and the longest edge in the drawing and thereby the area requirement is exponential in the number of vertices in the tree. □

Lemma 2.3. Caterpillars can be drawn with *ply-number* 2 in exponential area.

Proof. From the construction of stars, we can deduce the *ply-number* 2 drawability for caterpillars arguing about the construction rules for these. Recall that a caterpillar consists of a path of vertices, which is always drawable with *ply-number* 1 [36] and for the legs we can use the argument for stars, where the vertices of the path relate to the outermost layer of the star. □

2.4 Upper bound on the *ply-number* for any graph

In the following we present an argumentation that the *ply-number* for general graphs has a linear upper bound of $\frac{n}{2}$ where n is the number of vertices of the graph G.

Theorem 2.4. *Any graph G with n vertices can be drawn with a* ply-number *less or equal to $\frac{n}{2}$ in polynomial area.*

Proof. Let us assume that $G = (V, E)$ is the complete graph K_n with n vertices. We will show that G can be drawn with *ply-number* $\frac{n}{2}$ and conclude that any graph with n vertices can be drawn with *ply-number* $\leq \frac{n}{2}$ since removing edges cannot increase the *ply-number* in a given drawing. Any removed edge can either be not related to any *ply-disk* or can define the radius of a *ply-disk*. In the second case this edge was the longest edge adjacent to a vertex, meaning that all other edges incident to this vertex are at most as long as the removed one. Thereby the radius of the vertex's *ply-disk* might decrease but cannot increase. Decreasing the radius of any *ply-disk* might reduce the number of overlapping disks.

Let us w.l.o.g. assume that the number of vertices n is even. We place the vertices regularly on a circle, that is at forming angles of $\frac{360}{n}^\circ$ at the center as shown

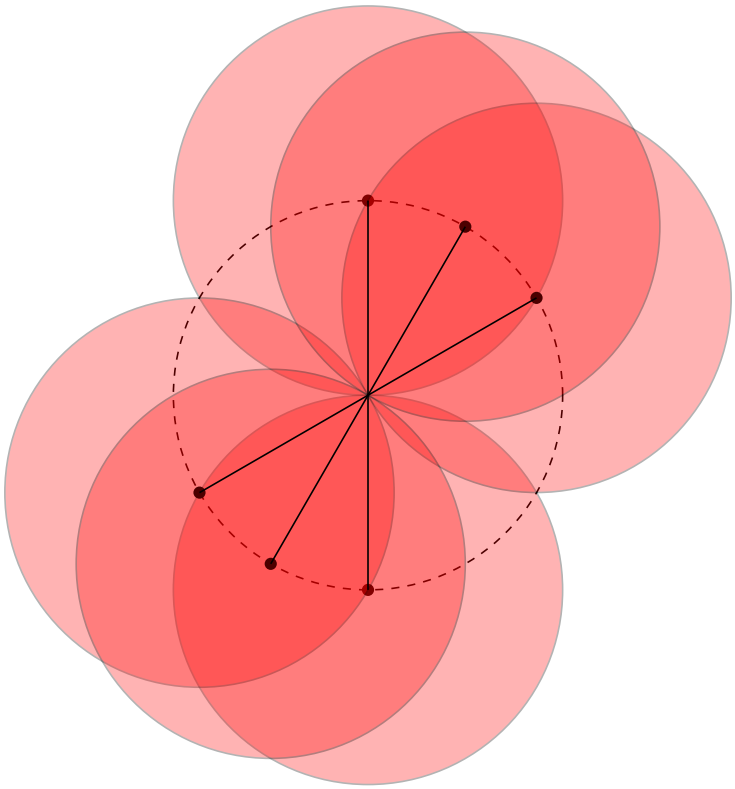


Figure 2.10: For clarification only 6 vertices at opposite positions and their longest edges and *ply-disks* are drawn of the complete graph K_n where the vertices are placed regularly on a circle. Note that all *ply-disks* touch in the center of the circle. For any coordinate in the plane with maximal overlapping *ply-disks* we observe that for every *ply-disk* which contributes to the *ply-number* there exists one *ply-disk* on the opposite site of the center such that this *ply-disk* cannot be involved in that region. Let the distance between any two neighboured vertices be equal to 1, the circle fits in a $n \times n$ square.

in Figure 2.10. Since we look at the complete graph K_n the longest edge passes exactly through the center of the circle. Furthermore, any longest edge defines the radius of exactly two *ply-disks*, namely of opposing vertices. The *ply-disks* touch in the center of the circle but do not overlap each other.

Now we have a look at one coordinate in the plane where the maximal number of overlapping disks occurs. Note that for any *ply-disk* involved in this maximal overlap, there exists the *ply-disk* of the opposing vertex in the circle defined by the same longest edge. Hence, for any *ply-disk* contributing there is one not contributing to the maximal overlap. This holds true for every *ply-disk* and thereby the *ply-number* of this drawing is $\leq \frac{n}{2}$. Although the vertices might not be on integer coordinates, we can scale the shortest edges, namely the edges that connect neighboured vertices around the circle to have length 1. In that case the longest edges have length shorter than n. The circumference of the circle is between n and

$2n$. Estimating the circumference of the circle to be $2n$ we obtain the diameter and thereby the length of the longest edge to be $\leq n$.

$$2\pi r \leq 2n \Leftrightarrow 2r \leq \frac{2}{\pi}n < n$$

Thereby, the circle fits in a square of $n \times n$ and the area requirement for this drawing is $\in \Theta(n^2)$.

Note that if the number of vertices n was odd, every vertex would have two opposing vertices sharing the same longest edge. We can still state, that for every *ply-disk* involved in the maximum overlap there is at least one other *ply-disk* namely one of the opposing vertices such that this *ply-disk* cannot be involved in the same maximal overlap. In this case the center of the circle is not covered by any *ply-disk*. Note that the argumentation holds for any drawing where each longest edge, meaning the defining edges for the *ply-disk* radii, defines the radii for both incident vertices. If n is even, that is if the longest edges form a perfect matching on G. Since we argue about the complete graph K_n, where all possible edges are present, the vertices may vary by an ϵ from the circle in other drawings. □

2.5 Relationship between Ply and Vertex-Ply

Initially introduced in [4] by Angelini et al. the concept of *vertex-ply-number* describes the maximal number of *ply-disks* any vertex is contained in. Formally, in contrast to the *ply-number*, the coordinates for the maximal number of overlapping *ply-disks* is reduced from $\mathbb{R}^2$ to the set of coordinates of the vertices in the drawing. Note that the *vertex-ply-number* of any drawing is at least 1 by definition of *ply-disks*. We call a drawing Γ of a graph $G = (V, E)$ an *empty-ply* drawing if the *vertex-ply-number* is 1. The *ply-disks* are empty in the sense that they do not contain any vertex but the defining one. This definition is close to *proximity drawings* where the *proximity region* has to be empty for an edge to exist but is not directly comparable, since the previously mentioned Gabriel graphs are subgraphs of relative neighbourhood graphs, which are again subgraphs of minimal spanning trees. In the following we will present some results in the relationship between *ply-number* and *vertex-ply-number* of drawings.

Theorem 2.5. *Any drawing with* vertex-ply h *has a* ply-number $\leq 5h$.

Proof. Let Γ be the drawing of a graph $G = (V, E)$ with *vertex-ply* h and assume for a contradiction that the *ply-number* of Γ is at least $5h + 1$. So there exists a coordinate x in the plane in the interior of at least k *ply-disks*, where $k \geq 5h + 1$. Let the k *ply-disks* of the k vertices $v_1, \ldots, v_k$ be ordered in a radial fashion around x. Let w.l.o.g. v_1 be the closest vertex to x, namely the vertex such that the condition $d(v_1, x) \leq d(v_j, x)$ for all $2 \leq j \leq k$ holds.

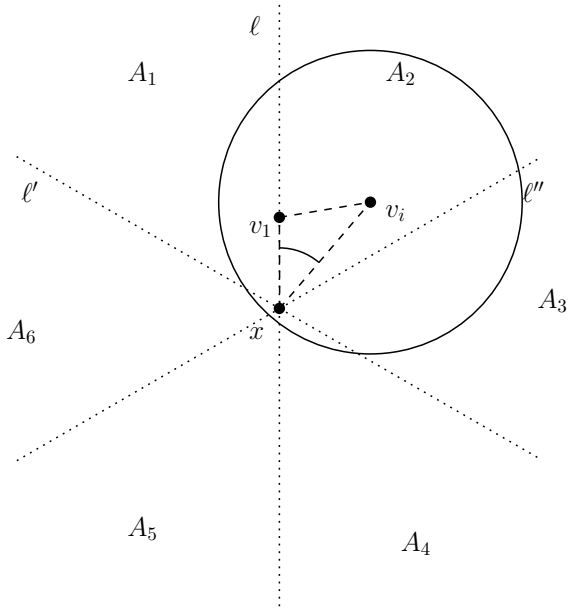


Figure 2.11: An illustration for the proof for Theorem 2.5.

We now consider the line ℓ passing through v_1 and x as well as the two lines ℓ' and ℓ'' rotated by $\frac{\pi}{3}$ and $-\frac{\pi}{3}$ in x. These lines define six wedges, namely $A_1, \ldots, A_6$ with an internal angle of $\frac{\pi}{3}$ each. An illustration of this setting is given in Figure 2.11.

Let the wedges adjacent to v_1 be A_1 and A_2. For any vertex $v_i \in A_1 \cup A_2$ the angle $\angle v_1 x v_i$ is less than $\frac{\pi}{3}$. This implies by the triangle inequality that the distance between v_1 and v_i is less than the distance between x and v_i, and thereby v_1 belongs to the *ply-disk* D_{v_i} of v_i. Note that the *ply-disk* of any other vertex v_i in either A_1 or A_2 also overlaps v_1 and thereby there can be at most h vertices in A_1 and A_2, including v_1. Hence, if there exist h vertices different from v_1 in $A_1 \cup A_2$, the drawing has a vertex-ply of $h+1$, which is a contradiction.

Since the wedge $A_1 \cup A_2$ contains at most h vertices including v_1, there are at least $4h+1$ vertices in the wedges $A_3, \ldots, A_6$ whereas there exists one of these wedges containing at least $h+1$ vertices by the pigeon-hole principle. For each of these wedges we can argue about the vertex which is closest to x like above. We can conclude that none of the wedges can have $h+1$ vertices in its interior and the *ply-number* of any drawing with *vertex-ply* h has to be $\leq 5h$. □

Corollary 2.6. The *ply-number* of an *empty-ply* drawing of a graph is at most 5.

This Corollary is a direct consequence of Theorem 2.5 since in any *empty-ply* drawing every vertex is in the interior of its own *ply-disk*. Thus, the *vertex-ply-number* of the drawing is 1 and by Theorem 2.5 the *ply-number* is ≤ 5. Note that the converse of Corollary 2.6 does not hold. If a graph does not admit an *empty-ply* drawing the implication that the *ply-number* is larger than 5 is not true.

For example, let the graph G be a star whose center has a degree larger than 24. This star does not admit an *empty-ply* drawing by Theorem 2.12 but can be drawn with constant *ply-number* 2 (see Lemma 2.1).

2.6 Properties of *empty-ply* drawings

We will continue to summarize the results presented in our paper [4] regarding the properties of *empty-ply* drawings. Beside Corollary 2.6 we state requirements for edge-length ratio throughout the drawing as well as space requirements. Furthermore we give an upper bound on the vertex-degree and point out the relation between *empty-ply* drawings and $\frac{1}{4}$-SHPED drawings [21].

Let Γ be a straight-line drawing of a graph $G = (V, E)$ which is *empty-ply*. That is every *ply-disk* D_v for every vertex v has the radius half the longest incident edge to v and does not contain any other vertex beside v in its interior.

Lemma 2.7. In any *empty-ply* drawing, for any two edges (u, v) and (u, w) incident to the same vertex u, we have $\frac{1}{2} \leq \frac{|uv|}{|uw|} \leq 2$.

Proof. Given an *empty-ply* drawing Γ of a graph $G = (V, E)$, for each vertex v and for each edge (v, w) the radius r_v of the *ply-disk* D_v is $\geq \frac{|vw|}{2}$. By definition no other vertex is allowed to be in the interior of D_v, formally $r_v \leq |vu|$ for any vertex u. We can summarize this as $\frac{|vw|}{2} \leq r_v \leq |vu|$ and thereby

$$\frac{1}{2} \leq \frac{|vu|}{|vw|} \text{ and } \frac{|vw|}{|vu|} \leq 2$$

for any edge (v, w) and any vertex u. Hence for any two edges (u, v) and (u, w) incident to u we can summarize the formula to

$$\frac{1}{2} \leq \frac{|uv|}{|uw|} \leq 2.$$

□

Lemma 2.8. The radii r_u and r_v of the *ply-disks* of two adjacent vertices u and v can differ by a factor of at most $\frac{1}{2} \leq \frac{r_u}{r_v} \leq 2$.

Proof. Since u and v are adjacent vertices the radii are at least $\frac{|uv|}{2}$. Let w.l.o.g u be the vertex with the larger radius, meaning $r_v < r_u$ and $r_v = \frac{|uv|}{2}$. By Lemma 2.7 the longest incident edge to u can have the length $2 \cdot |uv|$ and thus $r_u \leq |uv|$. Thereby $\frac{r_v}{r_u} \geq \frac{r_v}{|uv|} = \frac{1}{2}$ and vice versa $\frac{r_u}{r_v} \leq \frac{|uv|}{r_v} = 2$. □

Figure 2.12: A path of length 5 is drawn *empty-ply* with the edge lengths differing by the factor of 2. Note that the length of two adjacent edges can differ by a factor of at most 2 by Lemma 2.7.

Lemma 2.9. In any *empty-ply* drawing of a connected graph G, the length of the longest edge is at most $2^{\text{diam}(G)}$ times the length of the shortest edge.

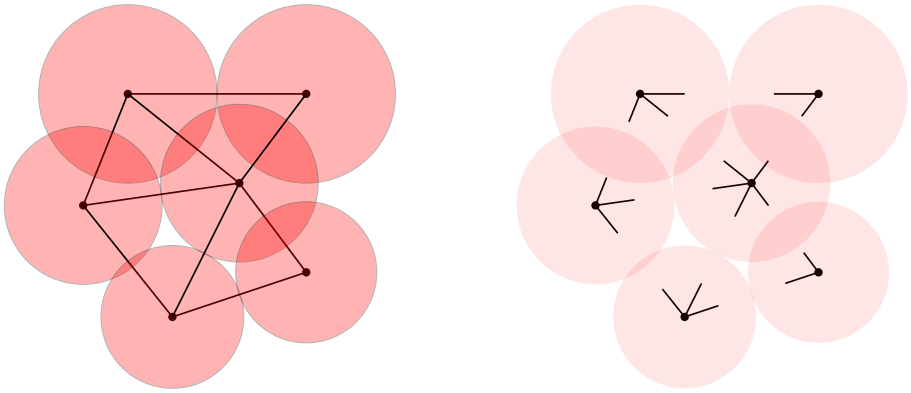
Proof. Let Γ be an *empty-ply* drawing of a connected graph G and let $\text{diam}(G)$ be the diameter of G. The diameter is defined to be the length of the longest shortest path between any two vertices in G. Let p be a longest shortest path. Note that this path is simple per definition, meaning each vertex might be contained at most once, since otherwise there exists a cycle and p could not be a shortest path at all. Recall that by Lemma 2.7 any two incident edges can differ in length by at most a factor of 2. Assume the shortest edge e_s in Γ has length 1. We can now choose one of the endpoints of e_s and proceed in a breadth-first search manner from this vertex where the edge length between the steps can increase by at most the factor of 2. Note that after at most $\text{diam}(G)$ steps we visited every edge in G and thereby the longest edge can have at most the length $2^{\text{diam}(G)}$. □

The relations throughout the edge length ratios can be observed in Figure 2.12 as well as the consequences for the radii of adjacent *ply-disks*. Another relation on the area distribution of *empty-ply* drawings is stated in the following.

Lemma 2.10. If Γ is an *empty-ply* drawing, the sum of the areas of all *ply-disks* does not exceed 4 times the area of their union.

Proof. Let Γ be a straight-line drawing of $G = (V, E)$ and consider the set of disks D'_v for every vertex v centered in v with the radius $\frac{r_v}{2}$ shrinking the *ply-disks* D_v by the factor of 2. Note that the disks D'_v are pairwise disjoint if and only if Γ is an *empty-ply* drawing. Every disk D'_v has an area four times smaller than D_v and is drawn inside the union of all *ply-disks*. □

We want to state the close relation between *empty-ply* drawings and *partial edge* drawings, in particular $\frac{1}{4}$-SHPED drawings. Partial edge drawings have been in-



(a) (b)

Figure 2.13: (a) an *empty-ply* drawing and the corresponding $\frac{1}{4}$-SHPED drawing (b). In $\frac{1}{4}$-SHPED only the first and the last $\frac{1}{4}$ of each edge is drawn.

troduced by the work by Bruckdorfer et al. [19, 21]. Partial edge drawings use the "principle of closure" [96] to neglect unnecessary information in straight-line drawings [22]. Some parts of straight-line edges can be neglected while the connection is still observable in the drawing. This is used to draw edges partially to avoid edge crossings. To maintain the association of edges to vertices the remaining part of the edge, called *stub*, is drawn at the vertex. $\frac{1}{4}$-SHPED drawings follow this drawing model but require the stubs at each vertex to have the length equal to $\frac{1}{4}$ of length of the edge. Furthermore, stubs are not allowed to cross. Since both vertices have stubs, half of every edge is drawn in Γ. Figure 2.13b shows a $\frac{1}{4}$-SHPED drawing of the *empty-ply* drawing in 2.13a.

Lemma 2.11. Any *empty-ply* drawing determines a $\frac{1}{4}$-SHPED drawing.

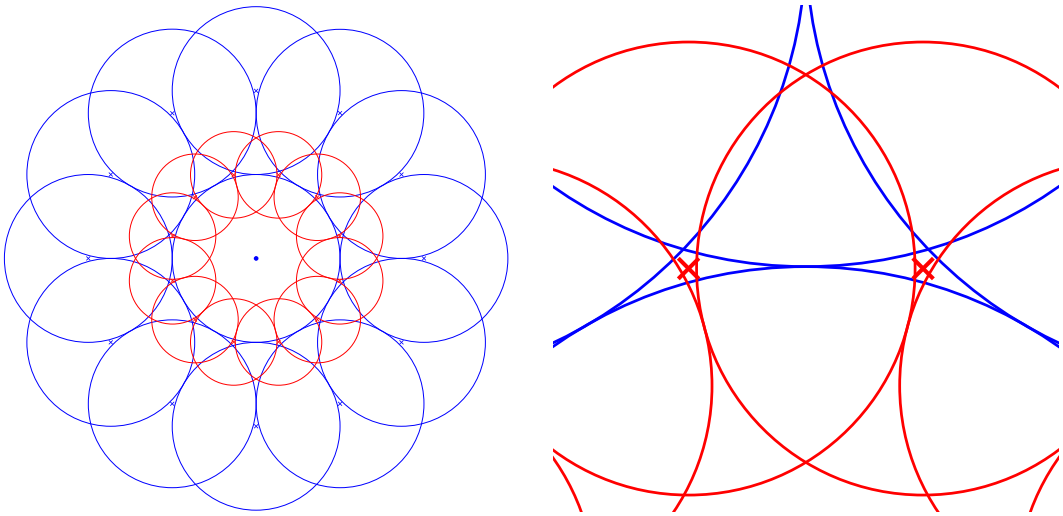
Proof. Consider the disks D'_v as defined previously with half the radius of the *ply-disks* D_v centered at v. In any *empty-ply* drawing these disks do not overlap and have the radius $\frac{1}{4}$ of the longest edge incident to v. Thereby the stubs are drawn inside the disks D'_v and cannot cross any other stubs. □

Note that the converse of Lemma 2.11 is not true. Since any planar straight-line drawing admits a $\frac{1}{4}$-SHPED drawing, whereas in the drawing presented in Figure 2.3b some planar edges cannot be drawn without violating the *empty-ply* property.

As a final theorem for this section we use geometric arguments to prove that

Theorem 2.12. *The maximum degree of any empty-ply drawing is 24.*

Proof. Let Γ be an *empty-ply* drawing of a graph $G = (V, E)$ and let's assume for the sake of a contradiction that there exists a vertex v with degree larger than 24.



(a) (b)

Figure 2.14: (a) a star with 24 leaves is drawn *empty-ply* with exactly two different distances. For clarification the edges are omitted. (b) the close neighbourhood of the topmost inner vertices regarding the surrounding *ply-disks*.

By Lemma 2.7 the lengths of all edges of v are in the closed interval $[m, 2m]$ where m is the length of the shortest edge incident to v. We can partition the vertices to be contained either in the interval $[m, \sqrt{2}m]$ or in $[\sqrt{2}m, 2m]$. Note that one of the partitions contains at least 13 vertices. In one of the intervals there exist two neighbours u, w of v such that $|vu| \leq |vw| \leq \sqrt{2}|vu|$ and the angle $\alpha = \angle uvw \leq \frac{2\pi}{13}$.

We may assume w.l.o.g. that $|uv| = 1$ by scaling Γ by the factor $|uv|^{-1}$ and thereby $|vw| = q \in [1, \sqrt{2}]$. As Γ is an *empty-ply* drawing, we know that $|uw| \geq \frac{q}{2}$ since u is not in the interior of the *ply-disk* D_w and by the law of cosine $|uw|^2 = 1 + q^2 - 2q\cos(\alpha)$. Combining these equations yields a quadratic inequality

$$\frac{q^2}{4} \leq 1 + q^2 - 2q\cos(\alpha) \Leftrightarrow 0 \leq q^2 - \frac{8}{3}\cos(\alpha)q + \frac{4}{3}$$

and thereby either

$$q \leq \frac{4}{3}\cos(\alpha) - \sqrt{\left(\frac{4}{3}\cos(\alpha)\right)^2 - \frac{4}{3}}$$

or

$$q \geq \frac{4}{3}\cos(\alpha) + \sqrt{\left(\frac{4}{3}\cos(\alpha)\right)^2 - \frac{4}{3}}.$$

Inserting $\alpha = \frac{2\pi}{13}$ in these equation yields

$$q \leq \frac{4}{3}\cos\left(\frac{2\pi}{13}\right) - \sqrt{\left(\frac{4}{3}\cos\left(\frac{2\pi}{13}\right)\right)^2 - \frac{4}{3}} \approx 0.934 < 1$$

or

$$q \geq \frac{4}{3}\cos\left(\frac{2\pi}{13}\right) + \sqrt{\left(\frac{4}{3}\cos\left(\frac{2\pi}{13}\right)\right)^2 - \frac{4}{3}} \approx 1.426 > \sqrt{2}$$

which contradicts $q \in [1, \sqrt{2}]$. Note that the bound of the degree for any vertex to be 24 is tight, see the *empty-ply* drawing of a star with 24 leaves in Figure 2.14a. In Figure 2.14b we present the close relation between a vertex on the inner distance to the *ply-disks* of the vertices placed at the further distance. Furthermore, note that the drawing uses exactly two different lengths of edges. □

To conclude this section on *empty-ply* drawings, we will state a theorem to present a relation between *squares* of graphs with *ply-number* 1 and *empty-ply* drawings. The *square* of a graph G, denoted by G^2, is obtained from G by adding edges between each vertex and the neighbours of its neighbours.

Theorem 2.13. *Let G^2 be the square of a graph G. If G admits a drawing with* ply-number *1, then G^2 admits an* empty-ply *drawing.*

Proof. Let Γ be a straight-line drawing of the graph $G = (V, E)$ with *ply-number* 1. By Section 2.3.1 we know that all edges in Γ have the same length, say 1, and any two non-adjacent vertices are at distance of at least 1 of each other. Adding the edges of $G^2 \setminus G$ produces a drawing Γ^2 of G^2. The newly added edges in Γ^2 have the lengths $1 \leq \ell \leq 2$, which implies that the maximal radius for any *ply-disk* in Γ^2 is at most 1 and thereby Γ^2 is an *empty-ply* drawing of G^2. □

Note that Theorem 2.13 cannot be extended to graphs with larger *ply-number*. A simple example is a star with n vertices which can be drawn with constant *ply-number* 2 according to Lemma 2.1 on page 19. The square of a star with n vertices is the complete graph K_n which does not admit an *empty-ply* drawing for $n > 7$. This result is presented in the following Section 2.7 in Theorem 2.16.

Corollary 2.14. The complete graph K_7 admits an *empty-ply* drawing.

Proof. As a direct consequence of Theorem 2.13 the complete graph K_7 admits an *empty-ply* drawing since it is the square of the wheel graph W_7 which admits a *ply-number* 1 drawing, see Figure 2.7. An *empty-ply* drawing of K_7 is presented in Figure 2.15. □

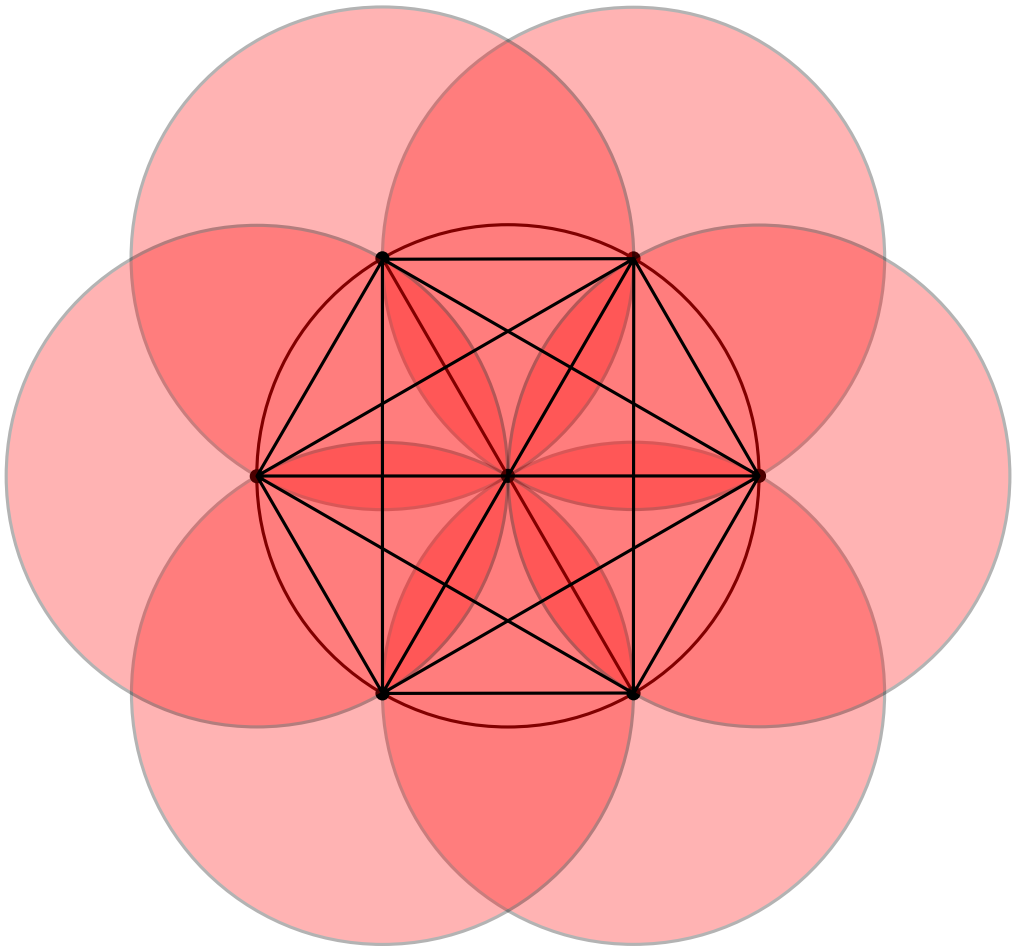


Figure 2.15: An *empty-ply* drawing of the complete graph K_7. Note that the edges are drawn on top of each other.

2.7 Graphs with or without *empty-ply* drawings

In this section we will summarize results on graph classes which do or do not admit *empty-ply* drawings. We will present *empty-ply* drawings of the complete graph K_7 and thereby any graph with $n \leq 7$ vertices admit *empty-ply* drawings. Furthermore, we present *empty-ply* drawings for some complete bipartite graphs. We will present more detailed proofs for two Theorems, namely Theorem 2.16 saying that the complete graph K_n where $n \geq 8$ does not admit an empty-ply drawing and Theorem 2.21 saying that the complete bipartite graph $K_{2,n}$ does not admit an empty-ply drawing if $n \geq 15$. Since we give an *empty-ply* drawing of K_7, the first bound is tight, whereas we can construct an *empty-ply* drawing for $K_{2,12}$ there is still a small gap.

Theorem 2.15. *The complete graphs $K_{1,24}, K_{2,12}, K_{3,9}, K_{4,6}$ and K_7 admit empty ply drawings.*

Proof. The star $K_{1,24}$ is presented in the previous section in Figure 2.14a and the graphs $K_{2,12}, K_{3,9}, K_{4,6}$ in Figures 2.16a 2.16b and 2.16c. Note that in the drawing of K_7 (Figure 2.15) and $K_{4,6}$ the edges overlap. □

In the following we will present the extensive proof which is based on a case distinction according to the regions where vertices might be placed after fixing the longest edge $e = (x_1, x_2)$ of K_n.

Theorem 2.16. *The complete graph K_n where $n \geq 8$ does not admit an empty-ply drawing.*

For this proof it suffices to show that K_8 does not admit an *empty-ply* drawing since K_8 is a subgraph for any larger $K_n, n \geq 8$. For the sake of contradiction let Γ be an *empty-ply* drawing of K_8 where $e = (x_1, x_2)$ is the longest edge of Γ with length 2. Thus, the remaining six vertices lie in the intersection of the annuli centered at x_1, x_2 with radii 1 and 2. Without loss of generality we draw this edge as horizontal line segment as in Figure 2.17. We partition the intersection in four major areas as follows:

$$
\begin{aligned}
A &= \{x \in \mathbb{R}^2 : \sqrt{2} < |x_1x| \leq 2, \sqrt{2} < |x_2x| \leq 2\},\\
B &= \{x \in \mathbb{R}^2 : 1 \leq |x_1x| \leq \sqrt{2}, \sqrt{2} \leq |x_2x| \leq 2\},\\
C &= \{x \in \mathbb{R}^2 : \sqrt{2} \leq |x_1x| \leq 2, 1 \leq |x_2x| \leq \sqrt{2}\},\\
D &= \{x \in \mathbb{R}^2 : 1 \leq |x_1x| \leq \sqrt{2}, 1 \leq |x_2x| \leq \sqrt{2}\}.
\end{aligned}
$$

Note that A^+ denotes the area above the longest edge e and A^- the area below as well as for B, C and D. In the following we will state 9 preliminary observations

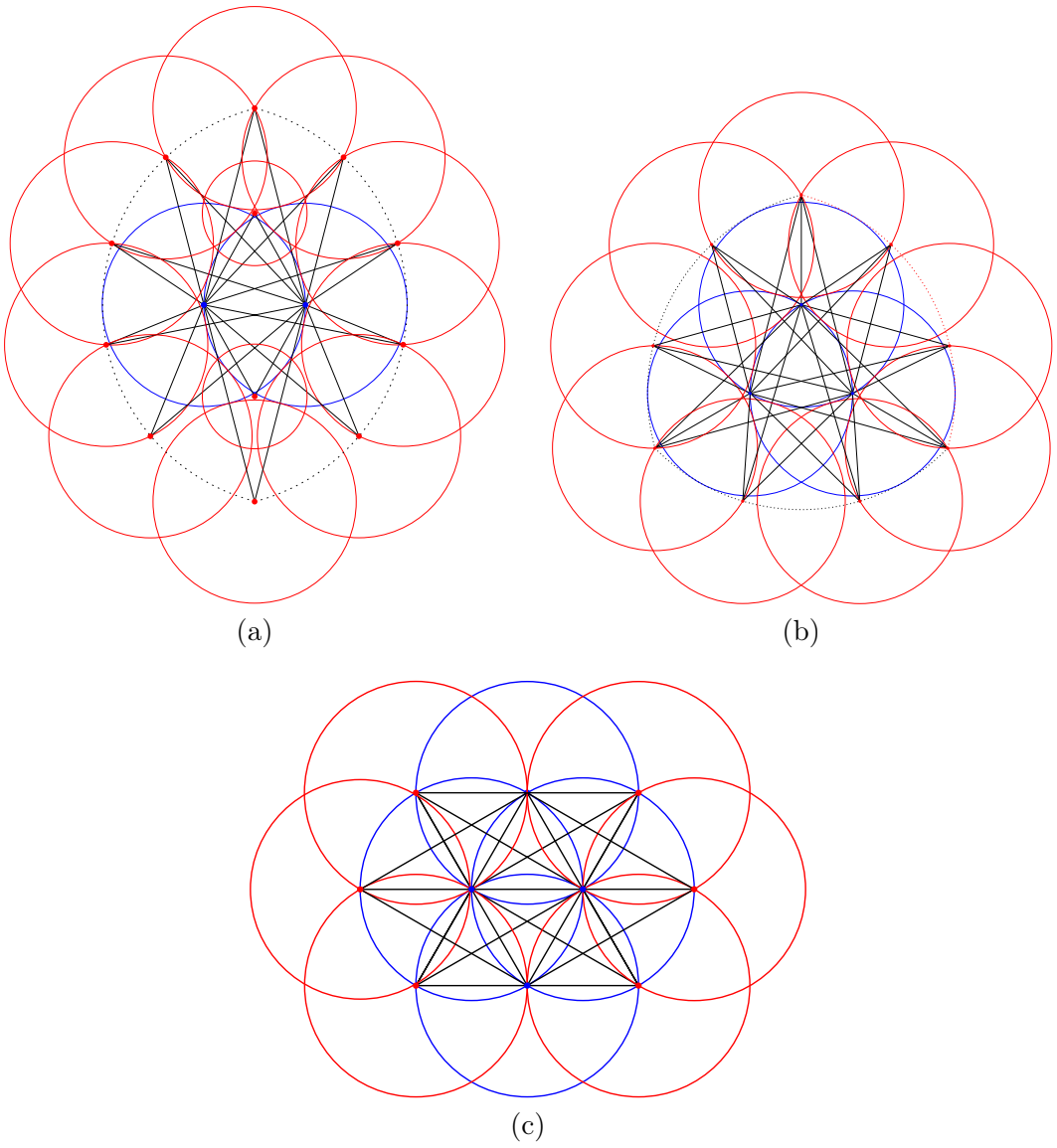


Figure 2.16: The complete bipartite graphs $K_{2,12}$ (a) , $K_{3,9}$ (b) and $K_{4,6}$ (c) can be drawn *empty-ply*. Note that in the *empty-ply* drawing of $K_{4,6}$ the edges overlap.

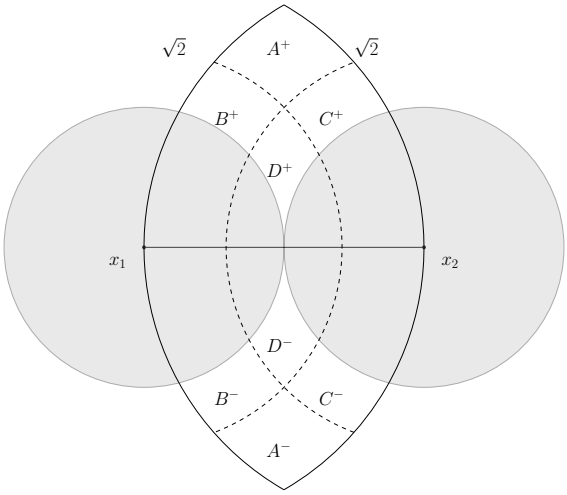


Figure 2.17: The four major areas A, B, C and D in the anulli centered in x_1 and x_2 are presented where $+$ denotes the area above the longest edge (x_1, x_2) and $-$ below respectively.

derived by placing vertices in specified areas. Afterwards we conclude the theorem by a case distinction on the number of vertices above or below the longest edge e in Lemma 2.17 on page 43 to Lemma 2.20 on page 46 using the observations.

Observation 1. There is at most one vertex in B^+ and by symmetry in B^- as well as in C^+ and C^-.

Proof. Note that the diameter of B^+, meaning the diameter of the smallest circle containing the complete area of B^+, is ≤ 0.75 (see Figure 2.18). For further argumentation we split the area of B^+ again into two parts like indicated in Figure 2.19:

$$B_1 = \{x \in B : \sqrt{2} \leq |x_2 x| \leq \sqrt{3}\},$$
$$B_2 = \{x \in B : \sqrt{3} \leq |x_2 x| \leq 2\}.$$

Any vertex located in the outer area B_2^+ has a ply disk radius of $\geq \frac{\sqrt{3}}{2} \approx 0.85$. Thereby we can conclude that whenever there exists a vertex in B_2^+ there cannot be a vertex in B_1^+, since its *ply-disk* completely covers B^+. The diameter of the inner area B_1^+ is 0.54 and thereby there can be at most one vertex in B_1^+, since its *ply-disk* has a radius of at least $\frac{\sqrt{2}}{2} \geq 0.7$. Hence, there can exist at most one vertex in B^+ which is either located in B_1^+ or B_2^+. □

Observation 2. There is at most one vertex in A^+ and by symmetry in A^-.

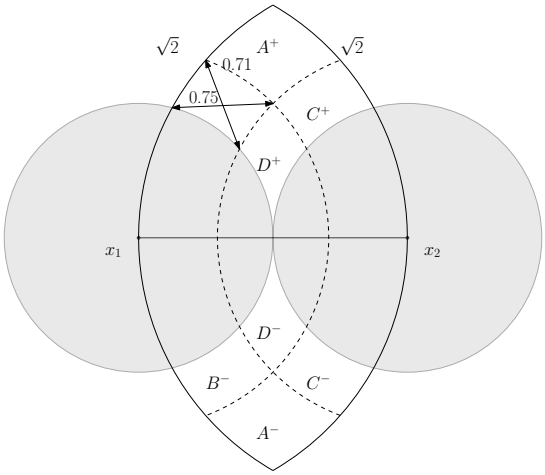


Figure 2.18: The diameter of B^+ is 0.75.

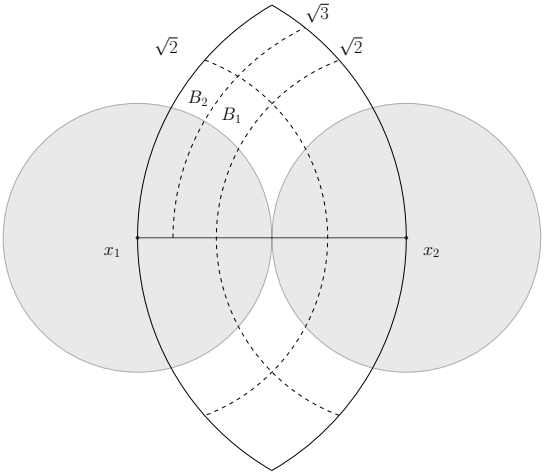


Figure 2.19: Split into B_1^+ and B_2^+.

Proof. For this argumentation we split the area of A above e, namely

$$A^+ = \{x \in \mathbb{R}^2 : \sqrt{2} < |x_1x| \leq 2, \sqrt{2} < |x_2x| \leq 2\}$$

into four parts as in Figure 2.20:

$$\begin{aligned}
A_1^+ &= \{x \in \mathbb{R}^2 : \sqrt{3} < |x_1x| \leq 2, \sqrt{3} < |x_2x| \leq 2\},\\
A_2^+ &= \{x \in \mathbb{R}^2 : \sqrt{2} < |x_1x| \leq \sqrt{3}, \sqrt{3} < |x_2x| \leq 2\},\\
A_3^+ &= \{x \in \mathbb{R}^2 : \sqrt{3} < |x_1x| \leq 2, \sqrt{2} < |x_2x| \leq \sqrt{3}\},\\
A_4^+ &= \{x \in \mathbb{R}^2 : \sqrt{2} < |x_1x| \leq \sqrt{3}, \sqrt{2} < |x_2x| \leq \sqrt{3}\}.
\end{aligned}$$

In the following we will consider the possible placement from top to bottom in a case distinction showing that any placement excludes the placement of any other vertex.

Case 1. Assume there exists a vertex x in A_1^+.

The *ply-disk* D_x has a radius of at least $\frac{\sqrt{3}}{2} > 0.85$ but the maximal distance of any other vertex in A^+ to x is ≤ 0.79 (cf. Figure 2.21). Thus, whenever there exists a vertex in A_1^+ there cannot be another vertex in A^+.

Case 2. Assume there exists a vertex x in A_2^+.

Note that at this point it suffices to argue about A_2^+, A_3^+ and A_4^+ since the placement of any vertex in A_1^+ already contradicts the assumption. We can treat the area A_4^+ the same way as in Case 1 since the maximal distance between any two vertices in A_2^+ and A_4^+ is less than 0.76 and the *ply-disk* D_x has a radius of at least $\frac{\sqrt{3}}{2} \approx 0.85$ (cf. Figure 2.22) and thereby A_4^+ has to be empty.

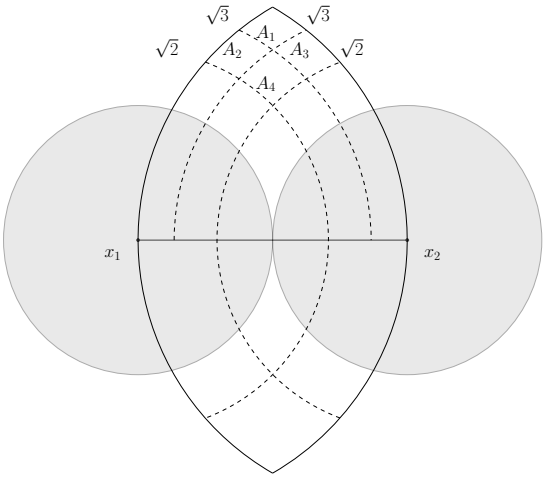


Figure 2.20: Split A^+ into four regions.

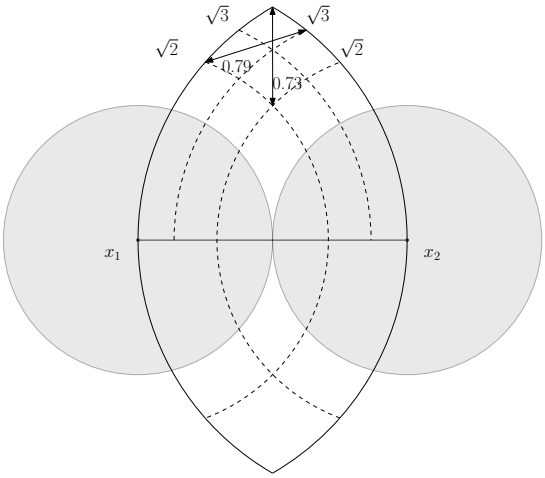


Figure 2.21: Maximal distance of any vertex in A to any vertex in A_1.

Assume there exists a second vertex in A_3^+. The *ply-disk* D_x of the vertex x in A_2^+ has a radius of at least $\frac{\sqrt{3}}{2}$ and thereby, if there exists a vertex in A_3^+, it has to be placed at a distance of at least $\frac{\sqrt{3}}{2}$ from the rightmost point of A_3^+. From this placement we can deduce that the radius of the *ply-disk* D_x has to be at least ≈ 0.92 since the distance to x_2 has increased, whereas in the next iteration the distance to the rightmost coordinate of A_3^+ has to increase, too. In Figure 2.23 we illustrate the increasing distance to the rightmost coordinate of A_3^+.

We can express the increasing distance between x and x_2 as a series $f(n)$, where $n \in \mathbb{N}$ is the number of iterations. The distance $f(n)$ between x and x_2 is defined as the length of the side of the triangle defined by the length $\sqrt{2}$, the length of 2 between x_1 and x_2, and the angle of $41.41° + \alpha$ as illustrated in Figure 2.24. Note that the angle α depends on the radius of the *ply-disk* D_x in the previous iteration. $\cos(\alpha)$ is defined by the law of cosine and the starting value for the series $f(1) = \sqrt{3}$ as we start with the minimal size for the *ply-disk* D_x.

$$
\begin{aligned}
f(1) &= \sqrt{3} \\
f(n) &= \sqrt{2^2 + \sqrt{2}^2 - 2 \cdot 2 \cdot \sqrt{2} \left(\frac{3}{4} \cos(\alpha) - \frac{\sqrt{7}}{4} \cdot \sin(\alpha) \right)} \\
\cos(\alpha) &= \left(\frac{2^2 + \sqrt{2}^2 - \left(\frac{f(n-1)}{2} \right)^2}{2 \cdot 2 \cdot \sqrt{2}} \right)
\end{aligned}
$$

We now observe that $\lim_{n \to \infty} f(n) = 2$. Thereby, there exists exactly one valid con-

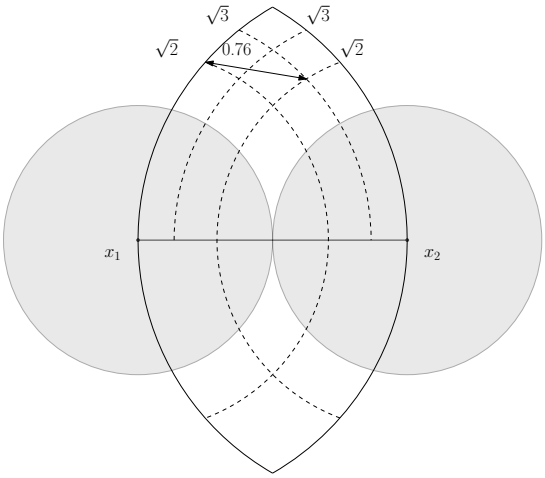


Figure 2.22: The maximal distance between any two vertices in A_2^+ and A_4^+

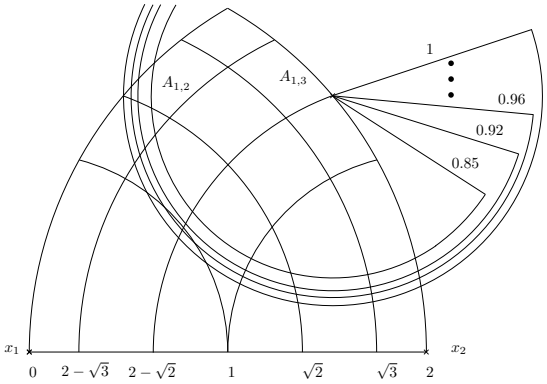


Figure 2.23: A schematic description of the increasing ply-radius of the vertex in A_2^+.

figuration which corresponds to actually placing $x \in A_2^+$ at the highest coordinate of B^+ and the vertex $y \in A_3^+$ at the highest coordinate of C^+ with distance 1 of each other. This contradicts the assumption. We conclude that whenever there exists one vertex in A_2^+ then A_3^+ has to be empty.

Case 3. Assume the vertex x is placed in A_4^+.

We show that there cannot be another vertex in A_4^+ since placing a vertex anywhere else in A^+ already contradicts the assumption. Observe that the diameter of the region A_4^+ is 0.5 and the *ply-disk* D_x has a radius of at least $\frac{\sqrt{2}}{2} \approx 0.7$. Thus there is at most one vertex in A_4^+.

The three cases conclude our observation, that there exists at most one vertex in A^+, which can be either in A_1^+, A_2^+, A_3^+ or A_4^+. □

Observation 3. There are at most two vertices in D^+.

Proof. Let $x \in D^+$ be the vertex at the coordinate in D^+ with the largest distance from e. Note that this point is unique. The diameter of the set $D^+ \setminus D_x$ is less than 0.3 and hence there cannot be placed two more vertices whose *ply-disks* have a radius of at least 0.5 (cf. Figure 2.25).

□

Observation 4. There are at most three vertices in the union of D^+ and D^-.

Proof. For the sake of contradiction we assume that there exist two vertices in D^+ as well as two vertices in D^-. We will partition each of D^+ and D^- into three parts. Let d_t be the topmost vertex in the union of D^+. Assume the second vertex d_b in D^+ below d_t to be placed close to the center c of the longest edge e. Thereby

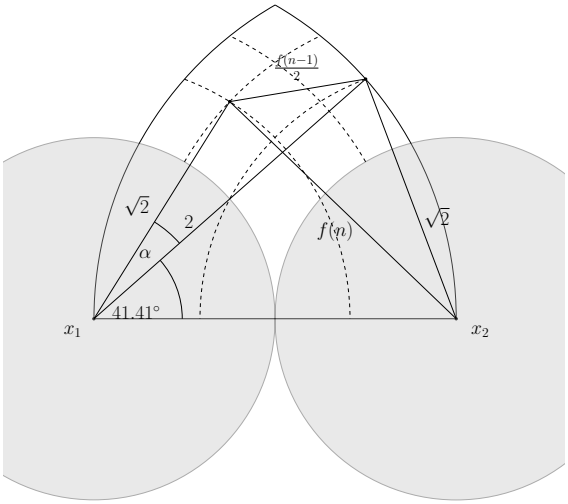


Figure 2.24: $f(n)$ describes the minimal valid distance of any vertex in A_2^+ to the vertex x_2. $f(n)$ is the length of the side of the triangle defined by the edge length of $\sqrt{2}$, 2 and the angle $41.41° + \alpha$. The angle α depends on the ply radius of the vertex in A_2^+ of the previous iteration.

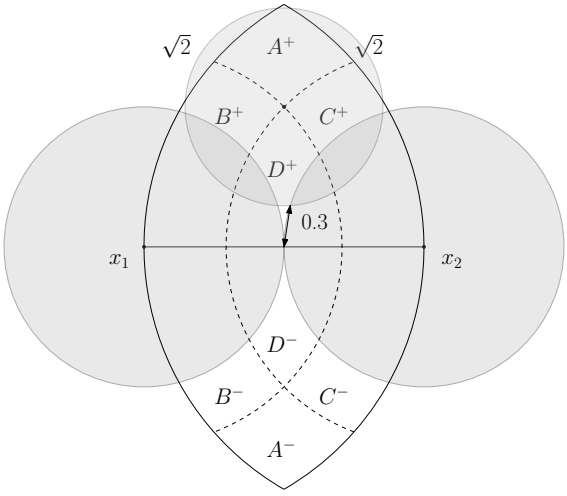


Figure 2.25: One vertex is placed at the highest coordinate in D^+. Its ply disk indicates a small remaining region for any other vertex in D^+.

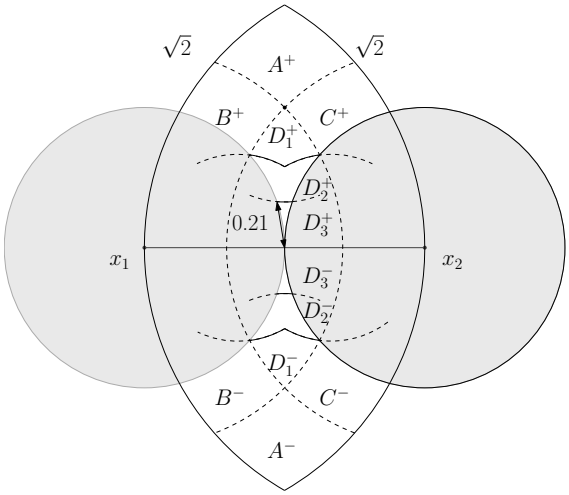


Figure 2.26: Partition of D^+ and D^-. Whenever there is a vertex in D_2^+ then D_1^+ and D_3^+ are empty. Placing vertices in D_2^+ excludes the placement in either D_1^1 or D_3^+.

the distance between d_t and either x_1 or x_2 has to fulfil the following equation since otherwise it contradicts the placement of d_b.

$$\frac{\max\{\text{dist}(d_t, x_1), \text{dist}(d_t, x_2)\}}{2} \geq \text{dist}(d_t, c)$$

The area which fulfils this equation is called D_1^+ and by symmetry D_1^- below e, and is illustrated in Figure 2.26.

We revise the argumentation from Observation 3 by assuming that there exists one vertex d_t in D_1^+ and one vertex d_b in D_1^-. The third vertex d' has to be placed in D^+ and thereby, has a distance of at least $\frac{1.58}{2}$ of the topmost coordinate of D^+, since 1.58 is the minimal distance between the area of D_1^- and the topmost coordinate of D^+. Previously the longest edge of d_t was either (d_t, x_1) or $(d_t x_2)$. The area to place the second vertex in D^+ above the center of the longest edge is named D_3^+ (D_3^- below respectively) and the region in between is called D_2^+ (D_2^-). The region D_3 is defined by the minimal *ply-disk* radius assuming one vertex in D_1^+ and one vertex in D_1^-. Placing any vertex in D_2^+ or D_2^- contradicts the assumption, since no vertex can be placed in either D_1^+ or D_3^+. The only remaining possibility to place 4 vertices in the union of $D^+ \cup D^-$ is in D_1^+, D_3^+, D_1^- and D_3^-.

Placing two vertices in D_1^- and D_1^+ the remaining two vertices have to be in D_3^+ and D_3^-. Furthermore their *ply-disks* have a radius of at least $\frac{1}{2}$ and thus need a distance ≥ 0.5 of each other. According to the diameter of $D_3^+ \cup D_3^- \leq 0.42$, there cannot be two vertices with *ply-disk* radii of at least 0.5, which is a contradiction. □

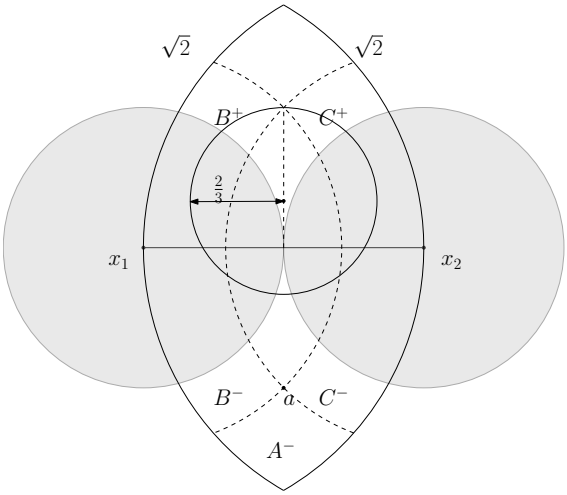


Figure 2.27: A vertex a in A^- causes the ply disk of any vertex in D^+ to cover the center of (x_1, x_2) and thus there can not be a second vertex in D^+.

Observation 5. Whenever there exists a vertex in A^+ there cannot be a vertex in A^-.

Proof. Assume there exist one vertex in A^+ and one vertex in A^-. The minimal distance between these vertices is > 2, which contradicts the maximality of the longest edge (x_1, x_2). □

Observation 6. Whenever one vertex exists in A^- there is at most one vertex in D^+.

Proof. Assume the vertex $a \in A^-$ is fixed at the coordinate with the closest distance to e, as this position is unique. Furthermore, the distance of any vertex in A^- to the center c of (x_1, x_2) is > 1, but the distance of any point $d \in D^+$ to c is < 1 due to the maximality of (x_1, x_2). Thus, if d is at the furthest possible coordinate of e, the center c of e is covered by its *ply-disk*. For any vertex $x \in D^+$ to cover the central coordinate c can be characterized by solving the equation

$$\text{dist}(a, c) \leq 2 \cdot \text{dist}(x, c).$$

This equation results in a disk which completely covers the region D^+ (shown in Figure 2.27). The circle is centered $\frac{1}{3}$ above c and has a radius of $\frac{2}{3}$. This region denotes the area where placing any vertex, the vertex's *ply-disk* covers c. This relates to the fact that after placing the further vertex there cannot be a second closer vertex in D^+ to the center c of e. □

Observation 7. Whenever there exists one vertex in A^+, then there are at most two vertices in $D^+ \cup D^-$.

Proof. Note that this observation is an extension to Observation 6 arguing about the complete region D rather than one side.

Assume there exist four vertices in A^+, D^+ and D^-. By Observation 2 there is at most one vertex in A^+ and by Observation 6 there is at most one vertex in D^-, thus there have to be exactly two vertices in D^+.

The top vertex $d_t \in D^+$ has at least the distance $\frac{1}{\sqrt{3}}$ from c by Observation 4. Assume that the vertex d_t is placed exactly at this coordinate. The minimal distance d from the vertex $a \in A^+$ to either x_1 or x_2 can be computed by solving the following equation:

$$\begin{aligned} d^2 &= 1^2 + \left(\frac{d}{2} + \frac{1}{\sqrt{3}}\right)^2 \\ d^2 &= 1 + \left(\frac{d}{2}\right)^2 + 2 \cdot \frac{d}{2} \cdot \frac{1}{\sqrt{3}} + \frac{1}{3} \\ d^2(1 - \frac{1}{4}) &= 1 + \frac{1}{3} + \frac{d}{\sqrt{3}} \\ \frac{3}{4}d^2 - \frac{1}{\sqrt{3}}d - \frac{4}{3} &= 0 \end{aligned}$$

The positive solution of the equation is $d = \frac{2}{9}\left(\sqrt{3} + \sqrt{39}\right) \approx 1.77$ and thus the minimal distance of any vertex $a \in A^+$ to the center c of (x_1, x_3) is $\frac{d}{2} + \frac{1}{\sqrt{3}} \approx 1.47$ (cf. Figure 2.28). By the maximality of e the distance between a and the vertex in D^- has to be ≤ 2. Thereby there can be at most one additional vertex in either D^+ or D^- since the *ply-disk* of the second vertex in D^+ with a radius of at least $\frac{1.47}{2}$ obstructs the remaining region to place any vertex below the edge $e = (x_1, x_2)$. This contradicts our assumption that there exist four vertices in the union $A^+ \cup D^+ \cup D^-$. □

Observation 8. Whenever there exists exactly one vertex in B^- and exactly one vertex in C^+, then there exists at most one vertex in $D^+ \cup D^-$.

Proof. We will use the same partition of B^- and C^+ as in Observation 1 and distinguish three cases about the placement within B^- and C^+.

Case 1. There exist one vertex in B_2^- and one vertex in C_2^+.

Note that the placement of the vertices is unique in this case. The distance of the two vertices is 2. Their *ply-disks* meet at the center of (x_1, x_2) and the remaining part of D^+ and D^- is covered. This implies a unique coordinate to place exactly one vertex in D^+.

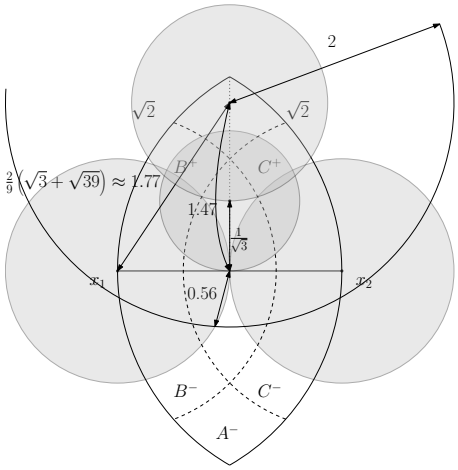


Figure 2.28: Whenever there exist exactly two vertices in D^+ and one in A^+ there cannot exist a vertex in D^-.

Case 2. There exist one vertex in B_1^- and one vertex in C_2^+.

By placing the vertex in C_2^+ at the rightmost coordinate and the vertex in B_1^- at the lowest possible coordinate, we can obtain the maximal possible diameter for the area to place vertices in D^+, which is ≤ 0.59. Note that D^- is completely covered by *ply-disks*. Furthermore, observe that moving any of the vertices in C_2^+ or B_1^- reduces the diameter of the free area in D^+, since either the topmost coordinate of the region moves down or the bottommost coordinate of the region moves up. An illustration is given in Figure 2.29.

We will argue on the regions above the conceptual line between the vertex in C_2^+ and x_1. Placing a vertex d_t above the conceptual line the radius of its *ply-disk* D_{d_t} is $\geq \sqrt{\frac{3}{8}} \approx 0.61$, since the distance to the vertex x_2 is at least $\sqrt{\frac{3}{2}}$. Thereby any vertex above the conceptual line obstructs the complete remaining area in D^+. The region below has a diameter of ≤ 0.34 and therefore there exists at most one vertex with ply radius ≥ 0.5.

Case 3. There exist one vertex in B_1^- and one vertex in C_1^+.

In any configuration the rightmost coordinate of D^-, as well as the leftmost coordinate of D^+, is covered by the *ply-disks* of the vertices in B_1^- and the vertex in C_1^+. Thereby we cannot place vertices in D_3^+ or D_3^- as defined in Observation 4. The remaining possible configurations result in one of the following cases:

a) D_2^+ and D_2^- are covered.
Similar to Observation+ 4, there cannot be two vertices in $D_3^+ \cup D_3^-$.

b) Exactly one of the D_2 is not covered completely and the center of (x_1, x_2) is covered.

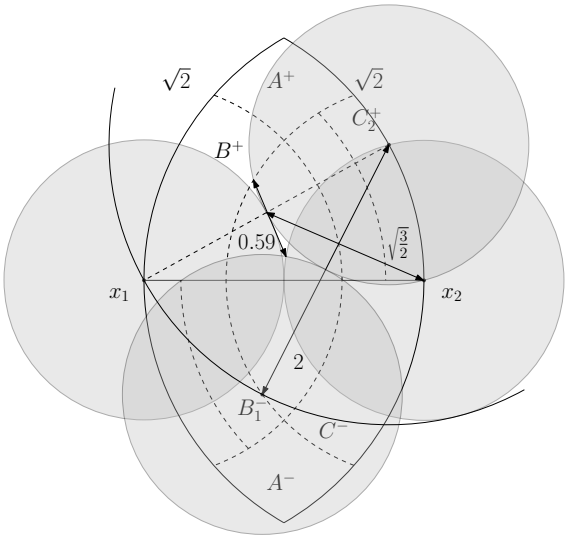


Figure 2.29: The region with the maximal possible diameter in D^+ whenever there exist vertices in B_1^- and C_2^+

Thus there can be at most one vertex in either D^+ or D^-.

□

These three cases conclude our Observation 8 that whenever there exist vertices in B^- and C^+ there exists at most one vertex in $D^+ \cup D^-$.

Observation 9. Whenever there exist one vertex in B^- and one vertex in A^+ either B^+ or C^+ is empty.

Proof. Any vertex in A^+ has a ply-radius of at least $\frac{\text{dist}(B^-,A^+)}{2} \approx \frac{1.68}{2}$. Thus, this vertex covers either B^+ or C^+ completely depending whether the vertex is closer to B^+ or C^+ as presented in Figure 2.30. □

In the next four lemmas we distinguish the different cases about the number of vertices placed below the edge $e = (x_1, x_2)$. Note that we fixed the first two vertices namely x_1 and x_2 in the beginning to argue about the longest edge e.

Lemma 2.17. Whenever no vertex is below the edge (x_1, x_2), then $n = 2 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$.

Proof. Assume there exists an *empty-ply* drawing of K_8 without any vertex below the longest edge $e = (x_1, x_2)$. We can conclude from Observation 1 that the number of vertices in $|B^+| \leq 1$ and in $|C^+| \leq 1$, additionally by Observation 2 the number of vertices in $|A^+| \leq 1$ and finally by Observation 3 the number of vertices in $|D^+| \leq 2$. Thereby $|A^+| + |B^+| + |C^+| + |D^+| \leq 5$ and the lemma follows. □

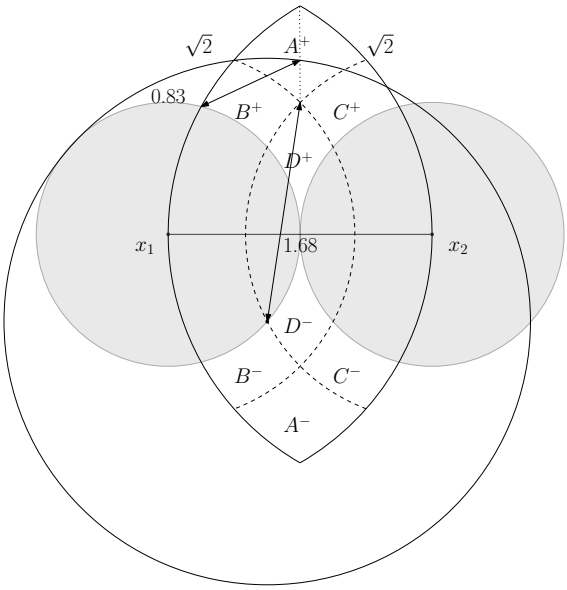


Figure 2.30: Any vertex in A^+ has a ply radius of at least $\frac{\text{dist}(B^-,A^+)}{2} \approx \frac{1.68}{2}$. Thus this vertex covers either B^+ or C^+ completely whether the vertex is closer to B^+ or C^+.

Lemma 2.18. Whenever exactly one vertex is below the edge (x_1, x_2), then $n = 3 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$.

Proof. Assume there exists an *empty-ply* drawing of K_8 with exactly one vertex below the longest edge $e = (x_1, x_2)$. By the same argumentation as in Lemma 2.17, there have to be exactly five vertices above (x_1, x_2), namely exactly one in A^+, one in B^+, one in C^+ and two in D^+.

By Observation 5 the vertex below e cannot be placed in A^-, which would contradict the placement in A^+. Furthermore, by Observation 8 the vertex below cannot be placed in B^- or C^-, since that would contradict the placement of vertices in B^+ or C^+, and finally by Observation 7 the vertex below cannot be placed in D^-, since that contradicts the placement of two vertices in D^+. This contradicts our assumption that K_n with $n \geq 8$ can be drawn with exactly one vertex below (x_1, x_2). □

Lemma 2.19. Whenever exactly two vertices are below (x_1, x_2), then $n = 4 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$.

Proof. Assume there exists an *empty-ply* drawing of K_n with $n \geq 8$ with exactly two vertices below the longest edge e, then there are at least four vertices above (x_1, x_2). We will distinguish four cases about the placement of the two vertices below.

Case 1. There exists one vertex in A^-.

By Observation 5 there cannot be any vertex in A^+ and by Observation 6 there can be at most one vertex in D^+. Since there can be at most one vertex in B^+ and one vertex in C^+ by Observation 1 the statement $n = 4+|A^+|+|B^+|+|C^+|+|D^+| \leq 7$ holds. This is independent of the placement of the second vertex below e.

Case 2. A^- is empty and there are exactly two vertices in D^-.

By the negation of Observation 6 the placement of any vertex in A^+ contradicts the assumption that there exist two vertices in D^-. Furthermore, by Observation 4 there is at most one vertex in D^+. All in all, since $|B^+| \leq 1$ and $|C^+| \leq 1$ (Observation 1), the statement $n = 4 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$ holds.

Case 3. A^- is empty and there is exactly one vertex in D^-.

The second vertex below e exists in either B^- or in C^-. Since both cases are symmetric, we assume without loss of generality that there exists exactly one vertex in B^-. We will argue in two cases on the arrangement of vertices by the existence of any vertex in A^+.

a) Assume there exists a vertex in A^+:
   By Observation 9 at least one of either B^+ or C^+ does not host any vertex. Using the vertex in D^- we can argue by Observation 7 that there exists at most one vertex in D^+. Thereby $n = 4+|A^+|+|B^+|+|C^+|+|D^+| \leq 7$ holds.

b) Assume A^+ to be empty:
   Assuming K_8 is *empty-ply* drawable and there exist exactly four vertices above e, namely exactly one in B^+, one in C^+ and two in D^+. But by Observation 8 there can be at most one vertex in $D^+ \cup D^-$ which is already placed in D^-.

This concludes the case where exactly one vertex is placed in D^-, since $n = 4 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$ holds in both subcases. The remaining case is the following:

Case 4. There exist exactly one vertex in B^- and exactly one vertex in C^-.

Assuming K_8 is *empty-ply* drawable and there exist four vertices above e. Since we can place at most one vertex in A^+ and at most two vertices in D^+ (Observation 2 and 3), we can conclude that there exists at least one vertex in either B^+ or C^+. By Observation 8 we can deduce that there can be at most one vertex in $D^+ \cup D^-$, hence A^+ must also contain one vertex. Since there is a vertex in A^+ and one in B^- there can be at most one vertex in either B^+ or C^+ by Observation 9. We can conclude that we cannot place four vertices above e and again $n = 4 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$ holds. □

Lemma 2.20. Whenever exactly three vertices are below the edge $e = (x_1, x_2)$ then $n = 5 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$.

Proof. Assume that K_8 is *empty-ply* drawable and there exist three vertices above e. Note that in any case there exists at least one vertex in either B^- or C^- since the placement of two vertices in D^- and one vertex in A^- implies by Observation 7 that D^+ is empty and thereby contradicts the placement of more than two vertices above e. We distinguish the following cases by the placement of the vertices below (x_1, x_2).

Case 1. There exist three vertices below, namely in A^-, in B^- and in C^-.

The number of vertices in $|A^+| = 0$ directly by Observation 5. Furthermore, B^+ and C^+ are empty by Observation 9, since there exist exactly two vertices in B^- and C^-. Additionally, there is at most one vertex in D^+ by Observation 6 and thereby $n = 5 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$ holds.

Case 2. There exist three vertices below e, namely in A^-, in D^- and without loss of generality in B^-.

Again, by Observation 5, there cannot be vertices in A^+, i.e. $|A^+| = 0$ and by Observation 6 there exists at most one vertex in D^+. Assuming that K_8 is *empty-ply* drawable, there have to be three vertices above e, namely exactly one in B^+, exactly one in C^+ and exactly one in D^+. By Observation 8 we can conclude, due to the existence of vertices in C^+ and B^-, that $|D^+ \cup D^-| \leq 1$, which contradicts the case since there already exists one vertex in D^-.

Case 3. There exist exactly three vertices below, namely in B^-, in C^- and in D^-.

If there exists a vertex in either B^+ or C^+, then by Observation 8 D^+ is empty, since there already exists a vertex in D^-. Furthermore, by Observation 9 whenever there exists a vertex in A^+, either C^+ or B^+ is empty. The remaining possible placements are:

1. two vertices in D^+
2. one vertex in B^+ and
   a) one vertex in A^+
   b) one vertex in C^+
3. one vertex in A^+ and exactly one vertex in D^+

In any case we cannot place a third vertex above e. Thereby, there exist at most two vertices above (x_1, x_2) and $n = 5 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$.

Case 4. There exist three vertices below, namely two vertices in D^- and without loss of generality one vertex in B^-.

We can deduce that there cannot be any vertices in A^+ by Observation 6 since there already exist two vertices in D^-. Furthermore, there cannot be a vertex in C^+ by Observation 8 due to the existence of two vertices in D^-. There can be at most one vertex in D^+ by Observation 4, and by Observation 1 there is at most one more vertex in B^+. We can conclude that there exist at most two vertices above the longest edge (x_1, x_2) and $n = 5 + |A^+| + |B^+| + |C^+| + |D^+| \leq 7$. □

The four lemmas conclude the proof of Theorem 2.16. K_8 cannot be drawn *empty-ply*, since the placement of more than 3 vertices below will be symmetric to the placement of the corresponding number of vertices above.

□

Theorem 2.21. *The complete bipartite graph $K_{2,n}$ where $n \geq 15$ does not admit an empty-ply drawing.*

Before actually stating the proof, we give some preliminary conditions on the vertices of an *empty-ply* drawing of a complete bipartite graph $K_{1,X}$ and then we extend these requirements to $K_{2,X}$, where $X \geq 1$.

Given an *empty-ply* drawing Γ of a complete bipartite graph $G \in K_{1,X}$ where $X \geq 2$ and let the vertex u be the central vertex in the first set, namely V_1 and $2m$ be the distance to the farthest adjacent vertex to u. Thereby the radius of the *ply-disk* D_u is m and all adjacent vertices to u have to lie at distances in the range $R = [m, 2m]$ from the center u. Similar to the proof of Theorem 2.12 we can split the range R into $R_1 = [m, \sqrt{2}m]$ and $R_2 = [\sqrt{2}m, 2m]$ as indicated in Figure 2.31 where the drawing is scaled such that $m = 1$.

Lemma 2.22. In any *empty-ply* drawing of $K_{1,X}$ with $X > 1$ the angular distance of any two vertices $x_1, x_2 \in V_2$ drawn both in R_1 or R_2, is $\geq 27.89°$.

Proof. We prove the statement for R_2, the larger range. The proof for R_1 is analogous by scaling arguments.

Given two vertices $x_1, x_2 \in V_2$ drawn in R_2 at distances $2m$ and $\sqrt{2}m$ from u (i.e. the maximum and the minimum distances from u). We can formulate the triangle formed by u, x_1 and x_2 and specify the sides with length $2m$, m and $\sqrt{2}m$. Hence the angle α formed by the two sides incident to u is given by the following equation:

$$1^2 \cdot m^2 = (2^2 + \sqrt{2}^2 - 2 \cdot 2 \cdot \sqrt{2} \cdot \cos(\alpha)) \cdot m^2$$

$$\cos(\alpha) = \frac{2^2 + \sqrt{2}^2 - 1^2}{2 \cdot 2 \cdot \sqrt{2}}$$

$$\alpha = 27.89°$$

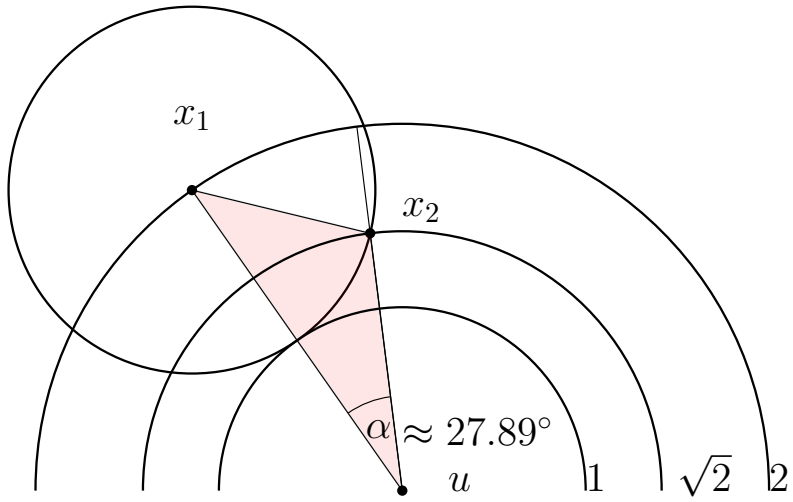


Figure 2.31: The minimum angular distance with respect to u of two vertices, both with distance in the same range $[\sqrt{2}m, 2m]$ or $[m, \sqrt{2}m]$ from u is 27.89°. The triangle with side lengths $m, \sqrt{2}m$, and $2m$ is indicated in red. Thereby in any circular sector with angle $< 27.89°$ there can be at most 2 vertices if and only if exactly one is in the range $[m, \sqrt{2}m]$ and exactly one is in $[\sqrt{2}m, 2m]$.

Thereby, we can conclude that any two vertices drawn both in R_2 (resp. R_1) with an angular distance of exactly $\alpha = 27.89°$, then the distances with respect to u are $\sqrt{2}m$ and $2m$ (resp. m and $\sqrt{2}m$) from u.

Furthermore, note that the extension of the ray passing through the closer vertex to u, namely x_2, is completely covered by the potential *ply-disk* of the vertex at distance $2m$, namely x_1 in Figure 2.31. Thereby there cannot exist any other vertex within the angular region of α. By a simple scaling argument the same holds true for the closer range $R_1 = [m, \sqrt{2}m]$. On the opposite this implies that any angular region with an angle $< \alpha$ can contain at most one vertex in R_1 and at most one vertex in R_2 and thereby at most 2 vertices in total. □

Note that this proof yields a version of Theorem 2.12 since there cannot be 13 disjoint angular regions of 27.89° in a total of 360° in either range and thereby at most 12 vertices per range in a star and thus $K_{1,25}$ is not *empty-ply* drawable.

Definition 2.23. The ***cover-disk*** $\mathrm{Cov}_v(u)$ of u with respect to v is defined for two adjacent vertices u and v, and describes the circular area such that the *ply-disk* of any vertex adjacent to v placed in $\mathrm{Cov}_v(u)$ contains u in its interior. Formally, the region is circular fulfilling the inequality $\mathrm{dist}(x, u) < \frac{\mathrm{dist}(x,v)}{2}$.

Considering *empty-ply* drawings the placement of any vertex prohibits the union of its ply-disk and its *cover-disk*. Note that this concept is only valid, if $(u, v) \in E$. An illustration is presented in Figure 2.32

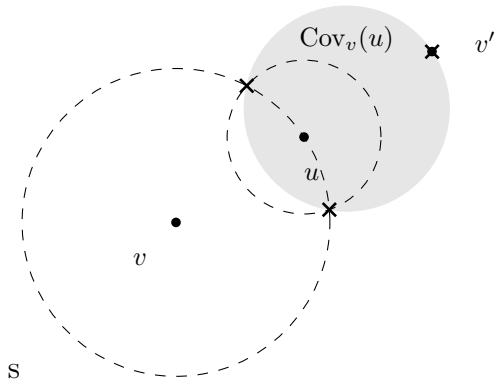


Figure 2.32: The area in which any vertex x adjacent to v the *ply-disk* D_x will have u in its interior. This region is called *cover-disk* of u regarding v $\mathrm{Cov}_v(u)$. This region can be geometrically constructed, by taking a circle with radius $\frac{\mathrm{dist}(u,v)}{2}$ centered at u and a circle width radius $\mathrm{dist}(u,v)$ centered at v. The intersections of these two circles together with v', namely the point v mirrored at u define a circle, which coincides with $\mathrm{Cov}_v(u)$.

We can extend the argument given in the proof of Lemma 2.22 to two vertices in the complete bipartite graph $K_{2,X}$. Let, w.l.o.g. the vertices in the first set, namely $u, v \in V_1$ lie on a horizontal line, l_h, at distance 1 from each other. Furthermore let $2m_u$ be the largest distance for any vertex from the second set to u and $2m_v$ the largest distance to v. We can now define the regions regarding u and v namely $R_1(u) = [m_u, \sqrt{2}m_u]$ the closer region to u, and $R_2(u) = [\sqrt{2}m_u, 2m_u]$ the further region regarding the vertex u. The regions $R_1(v)$ and $R_2(v)$ are defined respectively. Note that the radius of the *ply-disk* D_u is m_u, and the radius of D_v is m_v and $m_u \leq 1$, as well as $m_v \leq 1$ since their *ply-disks* cannot exceed the distance of 1 between u and v.

Let l_c be the perpendicular bisector of (u, v) splitting the plane in the left half-plane containing u and the right half-plane containing v. Note that for any vertex in the second set which is placed in the left half-plane the following condition $\mathrm{dist}(u, x_i) \leq \mathrm{dist}(v, x_i)$ holds and thereby, the distance to v defines the radius of its *ply-disk*. Furthermore, any vertex placed on the right side of l_c, the distance to u defines the radius of its *ply-disk*. Thereby the *ply-disks* of the vertices in the same half-plane depend only on either u or v. We can define the *cover-disk* $\mathrm{Cov}_u(v)$ and $\mathrm{Cov}_v(u)$ as shown in Figure 2.33.

In any *empty-ply* drawing the X vertices of the second set in a complete bipartite graph $K_{2,X}$ have to be placed in the ranges $R_1 = R_1(u) \cup R_1(v)$ or $R_2 = R_2(u) \cup R_2(v) \setminus R_1$ but not inside any *cover-disk*.

We can conclude the following lemmas about the maximal number of vertices in the areas R_1 and R_2 dependent on u and v.

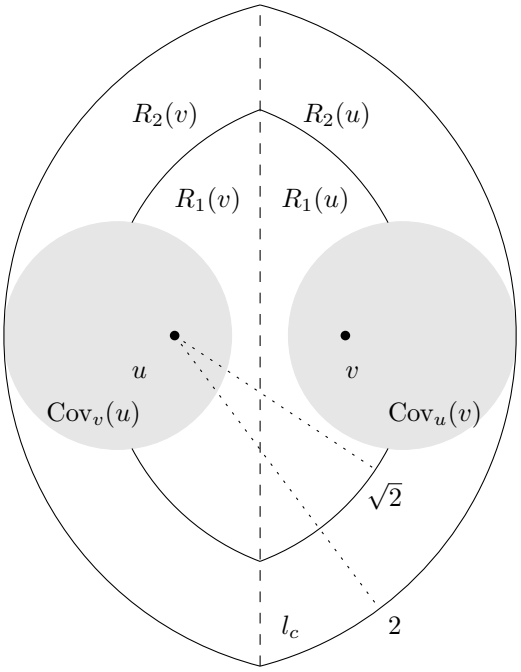


Figure 2.33: The two vertices of the first set, namely u and v are drawn horizontally with distance 1 of each other. The perpendicular line l_c splitting the plane into two half-planes where in the left half-plane the distance to the vertex v defines the *ply-disk* for any vertex and on the right side respectively the distance to the vertex u.

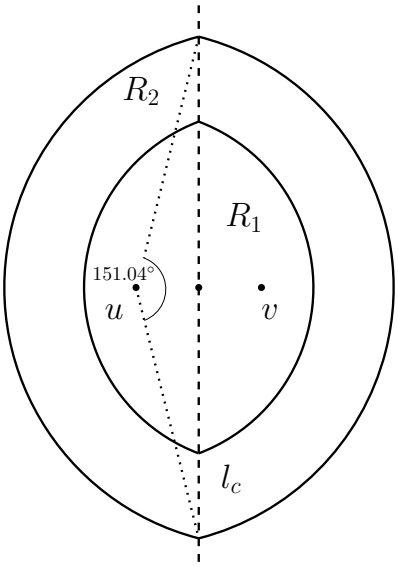


Figure 2.34: The angle to place sectors in $R_2(u)$ is $\beta_2 = 151.04°$ which is maximized if the radius of the *ply-disk* D_u of u and v is maximal, i.e. $m_u = 1$.

Lemma 2.24. There are at most 10 vertices in $R_2 = R_2(u) \cup R_2(v) \setminus R_1$.

Proof. Let Γ be an *empty-ply* drawing of $K_{2,X}$ and let w.l.o.g. $\text{dist}(u, v) = 1$. By the properties of *empty-ply* drawings all vertices must be drawn inside the intersection of the annuli centered at u and v with radii 2. By construction, the radii of *ply-disks* of vertices placed left (or right) of the central bisector l_c are purely dependent on v (u respectively). Thereby the number of vertices placed in R_2 can be estimated using Lemma 2.22 by the number of non-overlapping sectors of angular size 27.89°.

By the placement left or right of l_c the *ply-disks* are dependent on either u or v. We can now define the maximal available angle regarding u and v to place vertices in the further range R_2. Note that this is a slight overestimation in total available area to place vertices. Observe that the maximum angle to place the sectors is available if *ply-disks* of u and v have a radius 1, i.e. there is at least one vertex at distance 2 from u and v. In this assumption we yield $m = 1$ and R_2 equals the union of the ranges $[\sqrt{2}, 2]$ from both vertices u and v. The available angle to place the sectors for each vertex can be calculated using the coordinates with distance 2 from u and v on l_c, that is $\beta_2 = 151.04°$ as illustrated in Figure 2.34. We can give an upper bound on the number of non-overlapping angular sectors of $\alpha = 27.89°$ each by $\lfloor \frac{\lceil 2\beta_2 \rceil}{\alpha} \rfloor = 10$. Since we cannot place two vertices in R_2 with an angular distance from either u or v less than 27.89°, we cannot place 11 vertices in the range $R_2 = R_2(u) \cup R_2(v)$ and the lemma follows. □

Note that we cannot place 11 vertices whereas we observe $\gamma = (2\beta_2) - (10\alpha) = 302,08° - 278.9° = 23,18°$ of freedom, meaning the placement of the vertices is

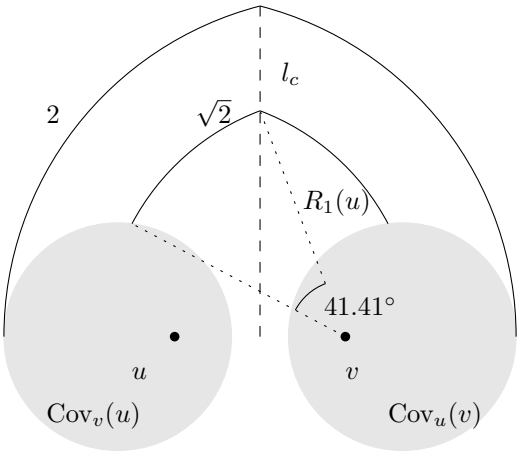


Figure 2.35: The maximal possible angular area regarding v on the left side of l_c. Decreasing the distance of $\sqrt{2}$ for the range $R_1(v)$ decreases the angular area.

not unique whereas the 11th vertex would require at least 27.89°. For the closer region we take the *cover-disks* $\mathrm{Cov}_u(v)$ and $\mathrm{Cov}_v(u)$ into account and additionally the available cone is smaller. Thus, we state the following Lemma.

Lemma 2.25. In the area $R_1 = R_1(u) \cup R_1(v)$ there are at most 6 vertices.

Proof. Similar to the previous proof we can denote the maximal available cone with respect to u and v. We assume the maximal distance for vertices in R_1 to be $\sqrt{2}$. Note that in the close range R_1 we are limited by the *cover-disks* and the *ply-disks* of u and v. Assuming there exists a vertex at distance $\sqrt{2}$ from u, there has to be another vertex at distance 2 from u. Since otherwise the closer vertex cannot be in the closer range $R_1(u)$.

Computing the available angle, again with respect to l_c above u and v, the result is $\beta_1 \leq 41.41°$ per vertex u and v. Note that this cone to place vertices is again an overestimation since we only regard the *cover-disks* and neglect the *ply-disks* for simplification. An illustration of the cone is given in Figure 2.35.

We can estimate the total available angular area above u and v by $2 \cdot \beta_1$ and below u and v respectively. The maximum number of vertices we can place either above or below is estimated by $\lceil \frac{2\beta_1}{\alpha} \rceil = \lceil 2.97 \rceil = 3$ and thereby we cannot place 4 vertices above or below u and v. The number of vertices in R_1 is limited to 6. □

Note that at this point Lemma 2.24 and 2.25 imply that the complete bipartite graph with 19 vertices in total, namely $K_{2,17}$ does not admit an *empty-ply* drawing. However, refining the argumentation involving the *ply-disks* of u and v, we can further reduce the number of potential vertices placed in R_2 and R_1. Essentially in the following lemmas we reverse the argumentation assuming a certain number of vertices and argue on the radii of the *ply-disks* by enforcing the necessary angular area around u and v.

Lemma 2.26. If there exist exactly 10 vertices in R_2, then the number of vertices in R_1 is at most 4.

Proof. Assume there exist 10 vertices in the further range R_2, then we need an angular area of at least $10 \cdot 27.89°$ around u and v. This implies that $m \in [\frac{1.88}{2}, 1]$ since otherwise there cannot be 10 vertices in R_2. Thereby the radii of the *ply-disks* D_u and D_v are at least $\frac{1.88}{2}$ and the area to place vertices in R_1 shrinks, since we argue on *empty-ply* drawings. The vertices in R_1 have to be placed in the intersections of the disks with radius $\sqrt{2}m$, where we assume $m = 1$, centered in u and v and not in the disks with radius $\frac{1.88}{2}$, namely the *ply-disks*.

We will argue on the area above and below u and v separately, since the *ply-disks* D_u and D_v overlap. Furthermore, any vertex's *ply-disk* drawn above u and v does not interfere with the area to place vertices below u and v.

We can observe that any vertex's *ply-disk* drawn in the upper part left of l_c entirely covers $R_1(v)$ above u and v. Thereby there can be at most 2 vertices in the upper part of R_1 and respectively at most 2 vertices in the bottom part of R_1. We conclude that there can be at most 4 vertices in R_1 in total, if there are exactly 10 vertices in R_2. □

Lemma 2.27. If there exist exactly 9 vertices in R_2 then the number of vertices in R_1 is at most 5.

Proof. Assume there exist exactly 9 vertices in R_2, then we can again conclude that $m \in [\frac{1.78}{2}, 1]$ by the same argument as in the previous proof. Thereby the *ply-disk* D_u or D_v has a radius of at least $\frac{1.78}{2}$. Assume there exist three vertices w.l.o.g in R_1 in the range $[\frac{1.78}{2}, \sqrt{2}]$ above u and v, then at least one of the vertices has a minimal distance to either u or $v \leq 1.33$. We can observe that any vertex with distance ≥ 1.33 implies a *cover-disk* regarding u or v such that there exists an angular region in R_2 with at least $45.44°$ in which no vertex in R_2 can be placed. Thereby, we can conclude that there might exist 3 vertices in R_1 either in the top or in the bottom part whereas assuming there exist 3 vertices in the top and in the bottom part contradicts the placement of 9 vertices in R_2 in the first place since the angular area is not sufficient to place 9 vertices. We can conclude that there cannot be 9 vertices in R_2 and 6 vertices in R_1 at the same time. □

Note that Lemma 2.26 and 2.27 imply Theorem 2.21, since by Lemma 2.25 there are at most 6 vertices in the inner range R_1 in any drawing of a complete bipartite graph $K_{2,X}$. Thus the complete bipartite graph $K_{2,n}$ with $n \geq 15$ does not admit an *empty-ply* drawing.

2.8 Conclusion

To conclude the results of this Chapter, we want to shortly summarize our results mainly published in [4] where the main contribution are the extensive proofs in Section 2.7, namely Theorem 2.16 and Theorem 2.21.

We gave results on structural properties of drawings with constant *ply-number*, i.e. *ply-number* 1 and drawings which admit *ply-number* 2. We presented strict structural properties on the graphs that admit drawings with *ply-number* 1 and showed that any star, caterpillar or binary tree can be drawn with *ply-number* 2 in exponential area. Furthermore we gave an upper bound on the *ply-number* for any graph to be $\frac{n}{2}$, where n is the number of vertices in the graph.

Then we introduced the concept of *vertex-ply*, which is the maximum number of *ply-disks* a vertex is contained in over all vertices. We stated relations between the *ply-number* of graphs and the *vertex-ply-number* of graphs and proved that the factor of difference is at most 5. We examined the set of graphs admitting drawings with *vertex-ply*, 1 namely *empty-ply* drawings, in the sense that every *ply-disk* is empty apart from its defining vertex. Note that the set of graphs admitting drawings with *ply-number* 1 is a subset of the *empty-ply* graphs whereas this is not true for graphs with *ply-number* 2. An example is that any star can be drawn with *ply-number* 2 whereas for *empty-ply* drawings the maximum degree for any vertex is bounded by 24. Where on the contrary we were able to draw binary trees with constant *ply-number* and *empty-ply*.

As major part of this chapter we identified the complete graphs $K_{1,24}$, $K_{2,12}$, $K_{3,9}$, $K_{4,6}$ and K_7 to admit *empty-ply* drawings. We could prove that the bound on complete bipartite graphs $K_{1,24}$ is tight, meaning that $K_{1,25}$ does not admit an *empty-ply* drawing. The same holds true for the complete graph K_7, namely K_8 does not admit an *empty-ply* drawing. This proof was done by an extensive case analysis using the previously stated properties of *empty-ply* drawings and considering any possible placement of 5 vertices after fixing the longest edge of K_7. For the next graph class $K_{2,X}$ we fixed the coordinates of the 2 vertices from the first set rather than fixing the longest edge. Within the graph class $K_{2,X}$ we know that $K_{2,12}$ is *empty-ply* drawable and $K_{2,15}$ is not. Here is a small gap $X = 13$ and $X = 14$, which is not proven by now. We conjecture that $K_{2,13}$ does not admit an *empty-ply* drawing and it might be proven by refining the argumentation in Theorem 2.21 together with strict argumentation on the *ply-disks*. During our argumentation we rather extensively used the *cover-disks* and there the largest overestimation in terms of accuracy took place.

Although not proven, it is reasonable to assume that the *empty-ply* drawings of K_7 and $K_{4,6}$ are unique with respect to scaling and rotating, since in contrast to the other known complete graphs these drawings do not allow any movement of vertices without violating the *empty-ply* properties.

We conclude with some open problems arising by our work. Recall that we have

proven a tight bound for complete graphs K_7 and for complete bipartite graphs $K_{1,24}$. There is still a small gap whether $K_{2,13}$ or $K_{2,14}$ admit *empty-ply* drawings. How about larger values of n in complete bipartite graphs $K_{n,m}$?

Looking at *empty-ply* drawings from the proximity perspective, can we generalize to drawings where the *ply-disk* does not need to be empty but can contain at most k vertices? This corresponds to the generalization of k-*proximity* drawings where there are at most k vertices allowed to be in a *proximity* region for any edge. This would be an extension in the same fashion as the definition of k-Gabriel and k-relative- neighbourhood drawings [65].

Chapter 3

Applicational aspects of Ply

In the following we will introduce our implementation to compute the *ply-number* for a given drawing. We will start with a short summary on the program and its features and will go in detail for the computation algorithm. We will continue with a short section on the different layout algorithms. In the end, we will present our experimental results comparing our implementation to existing ply-computation by De Luca et al. presented at WALCOM'17 [30]. We are able to reduce the computation time from seconds to milliseconds for given drawings and thereby contribute to further research on the ply topic, by providing an efficient tool to examine graphs extensively by user interaction, as well as some automatic features to reduce the *ply-number*.

Our tool allows the investigation of a graph regarding its *ply-number*. As basic functionality, the tool is equipped with some graph layout algorithms, namely organic, circular and randomized (provided by yFiles library [97]), as presented in Figure 1.6, and allows for interactive manipulation of the drawing, such as moving vertices. Basic file formats are graphml [17] and gml [57], where both formats provide structural information on the graph and a drawing.

Furthermore, we provide a test if a given drawing is empty-ply, where no vertex is contained in any other vertex's disk. With our tool we identified that the complete graph $K_{4,6}$ admits an *empty-ply* drawing whereas this was previously known only for $K_{4,4}$ [4]. Our implementation is able to compute the *ply-number* for the drawing during runtime, meaning while the drawing is modified, for example by layout algorithms or user interaction.

Live feedback of the *ply-number* on interactive graph manipulations by users is another feature of our tool. We mark regions where the maximal *ply-number* occurred. The user can choose between different layouts as start configurations for the graph and is able to improve the *ply-number* automatically by a spring embedder or by manually manipulating the positions of the vertices accordingly.

3.1 Previous Work

The *ply-number* of drawings has been considered previously. There exist several theoretical results on the *ply-number* of drawings. For a short overview please refer to the previous Section 2.2.

From the applicational point of view, the importance of the *ply-number* has been evaluated in an experimental study by Felice de Luca et al. [30]. The experiments were designed to confirm that the *ply-number* of a drawing is closely related to other known metrics to evaluate aesthetics, like *stress* [40], *crossing number* [63], and *edge-length uniformity*. The edge-length uniformity is especially motivated by the property of drawings with *ply-number* one, since all edges have to be the same length [36].

Furthermore, the study investigates the *ply-number* for different layout algorithms. The evaluation has considered six different force-directed algorithms out of the most popular ones [62], namely the algorithm by Fruchterman-Reingold [47], a variant proposed by Frick et al. [46], the algorithm proposed by Kamada-Kawai [60], a stress minimization algorithm by Gansner et al. [49], the fast multipole multilevel method by Hachul and Jünger [53], and a model developed by Noack [68] which is based on a minimum energy layout to emphasize clusters in a graph.

The layout algorithms designed to reduce stress in the drawing compute drawings with low ply-values. As an intuitive understanding the authors indicate the relation between stress and edge uniformity. Furthermore, the fast multipole multilevel method performs very good even on large dense graphs regarding the *ply-number* of the drawing. There exist graph classes for which these force-directed layout algorithms produce drawings with a *ply-number* close to the optimum, like paths, circles, caterpillars, and binary trees. The authors point out that there exists a strong correlation between the *ply-number* of the drawing and stress, as well as a moderate correlation between the *ply-number* and edge-length uniformity. Even though they give an example for a drawing with low ply and high stress and vice versa.

In the bachelors thesis of Lukas Bachus [7] a prototype for the ply-computation algorithm for a given graph has been implemented. The prototype supports a graphical user interface and emphasizes the user to modify the drawing manually. It supports one basic layout type. The manual modification does not give instant feedback, but after the modification is finished, the *ply-number* will be computed and the *ply-disks* will be updated.

We reimplemented the ply-computation algorithm and improved its speed by a large factor, such that instant feedback on modifications is applicable in our tool. Furthermore, we added more basic layouts to choose from and we developed an optimization workflow to minimize the *ply-number* for a given graph.

Both previous algorithmic approaches are based on the plane-sweep technique motivated by computational geometry to compute the maximum number of overlapping *ply-disks*. When computing intersections between disks, precision errors are

one of the first challenges to deal with. To give an intuitive example, look at the complete graph with three vertices K_3. Every straight-line drawing of this graph is a triangle in the plane. Drawing an equilateral triangle results in a drawing with *ply-number* one, since every edge has equal length and thereby the *ply-disks* touch but do not overlap. As the mathematical affine reader knows, or simply can prove, an equilateral triangle in the plane requires irrational coordinates. Rounding these numbers to store the coordinates leads to precision errors. In this case the precision errors lead to the computation of a minimal overlap. In other cases they might result in missed overlaps. In many cases these precision errors do not affect the ply-computation significantly, but especially if there are many intersections at one coordinate, it shows its effect. To name one worst case example, take a complete graph with $n \geq 60$ vertices and draw the vertices regularly on a circle. In this scenario all *ply-disks* touch in the center of the circle and the rounding errors heavily affect the computation.

To come up with a solution to these precision errors, the approaches of Lukas Bachus [7] and De Luca et al. [30] use the `ApFloat` library which is available for Java. This library allows the computation with an arbitrary pre-chosen level of accuracy at the cost of computational effort [89]. The implementations are equipped with a computation inconsistency detector, such that they are able to increase the level of precision in their calculations, if they encounter such a problem. In case of several increments of the precision level, the computational effort increases immensely. In the following (Section 3.5.4) we will discuss the issue of inaccuracies in detail. We took a different approach to deal with inaccuracies, with respect to the computational effort.

Thereby, the implementation by De Luca et al. requires a drawing of a graph as input, which is then non editable. Even though, the implementation of [7] provides a graphical user interface, the *ply-number* is calculated if and only if there is no user interaction at this time. The response time of this implementation is reasonable on small graphs, whereas it might get stuck after several incrementations of the precision level, if the graph is large, or there are too many intersections at one coordinate.

One of our main goals was to speed up the computation to provide real-time evaluation with two additional features in mind, first live feedback on modification of the drawing and second optimization of the *ply-number*. An optimization process involves a large number of *ply-number* computations.

The previous implementations do not support optimization, where the tool by Lukas Bachus supports manual modifications. The experimental study by De Luca et al. compares the *ply-number* dependent on different layout algorithms. We introduce an optimization process to reduce the *ply-number* for a given drawing.

We evaluate our tool in comparison to the implementation of De Luca et al. especially we compare the results on a large set of graphs in Section 3.6. Since both approaches are based on the plane-sweep technique, we count the number of

executed events in both algorithms and log the computation time.

In a second set of experiments we compare different layout algorithms regarding the *ply-number* of the produced drawings. A special focus is the comparison between the fast multipole multilevel method [53], which has been evaluated to produce drawings with low *ply-number* consistently, and our optimization approach.

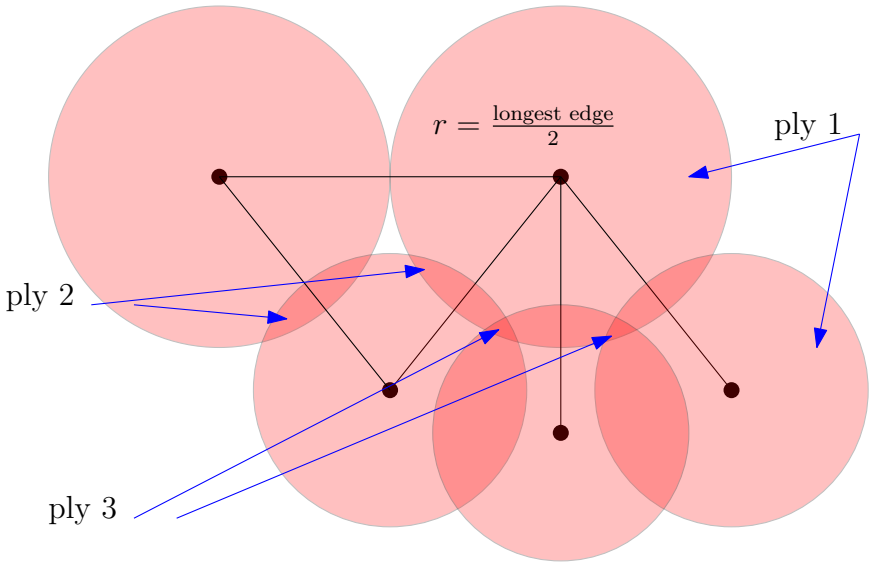


Figure 3.1: A graph with 5 vertices and 5 edges is drawn. For every vertex its *ply-disk* is drawn. Some regions with ply one, two, and three are indicated by arrows. The *ply-number* of the drawing is three, whereas the *ply-number* of the graph is one (cf. Figure 3.3)

3.2 Introduction to the Problem

We formally state the problem and recall the necessary definitions. Let Γ be a straight-line drawing of a graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges. We consider the graph to be simple and undirected, that is if there exists an edge connecting the vertices u and v, either $(u, v) \in E$ or $(v, u) \in E$, meaning both representations are equivalent. For simplicity we will write for a vertex u, the edge $(u, v) \in E$ if either (u, v) or (v, u) is $\in E$. For any vertex u there exists a coordinate $(u_x, u_y) \in \mathbb{R}^2$. We do not allow vertices to share the same coordinate in $\mathbb{R}^2$, that is $\forall u, v \in V, u \neq v : (u_x \neq v_x \vee u_y \neq u_y)$.

Ply-disk

For every vertex u the *ply-disk* D_u is defined as a disk centered at u where the radius r of the disk is half the length of the longest edge incident to u. In Figure 3.1 the *ply-disk* for every vertex is drawn. For consistency we will always draw the *ply-disks* in all figures, if not stated otherwise explicitly. Recall, that we do not allow two vertices to share the same coordinate in $\mathbb{R}^2$, in that way the radius of the *ply-disk* for every vertex is > 0, if the vertex has at least one incident edge.

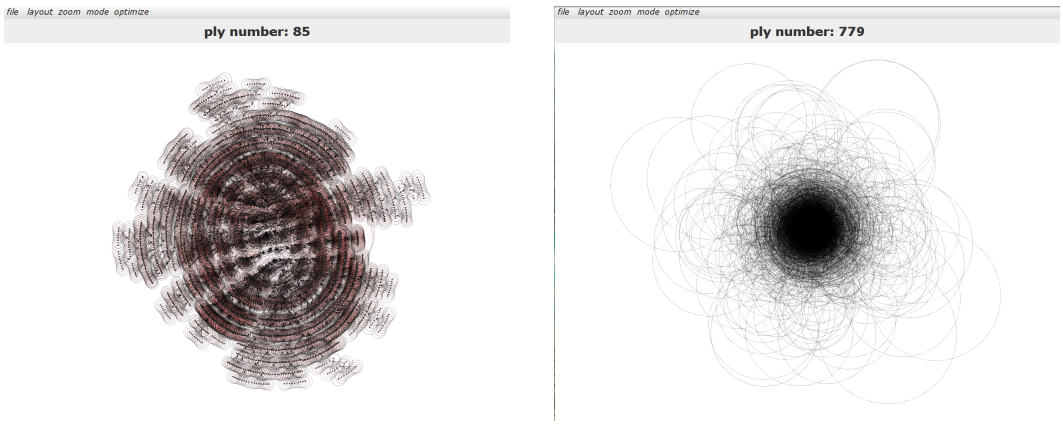


Figure 3.2: On the left side, a 9-ary tree with 3000 vertices is drawn with the minimum stress function by Gansner et al. [49]. This drawing admits a *ply-number* of 85. The same graph drawn by placing the vertices at random locations, admits a drawing with *ply-number* 779.

Ply-number of a drawing

The *ply-number* of the drawing Γ is defined as the maximum number of overlapping *ply-disks* at any point in $\mathbb{R}^2$. The main algorithm is defined to solve the problem of computing this value. The *ply-number* of the drawing in Figure 3.1 is three. Note that for any graph with at least one edge, the *ply-number* of the drawing Γ is at least one.

Ply-number of a graph

The *ply-number* for a graph is the minimal *ply-number* over all possible straight-line drawings. Note that the *ply-number* of a graph without any edges is 0. The graph presented in Figure 3.1 admits a drawing with *ply-number* one. A drawing is given in Figure 3.3, where the left component is an equilateral triangle and all edges have equal length. To weaken this definition, whenever we talk of the *ply-number* of a graph, we mean the smallest known number. On complete graphs, we were not able to construct drawings with a lower *ply-number* than $\frac{n}{2}$. In Figure 3.4 the complete graph K_{60} is drawn on a circle and this drawing has *ply-number* 30.

3.3 Program and Features

We present a tool to get an intuitive understanding of the *ply-number*, as well as explore and examine given drawings. The graphical user interface of our tool is shown in Figure 3.5. The graph is drawn and every edge is represented as a straight line. For every vertex the *ply-disk* is drawn. The disks are presented as red

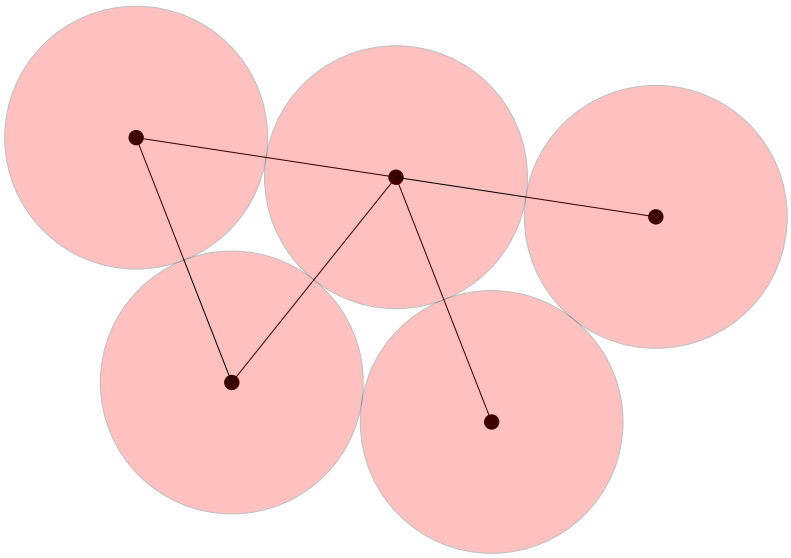


Figure 3.3: The same graph as in Figure 3.1 with 5 vertices and 5 edges is drawn with *ply-number* one. All edges have the same length, as the left component is an equilateral triangle, and the distance between any two nodes $\geq$ length of any edge. Note that no two *ply-disks* overlap each other.

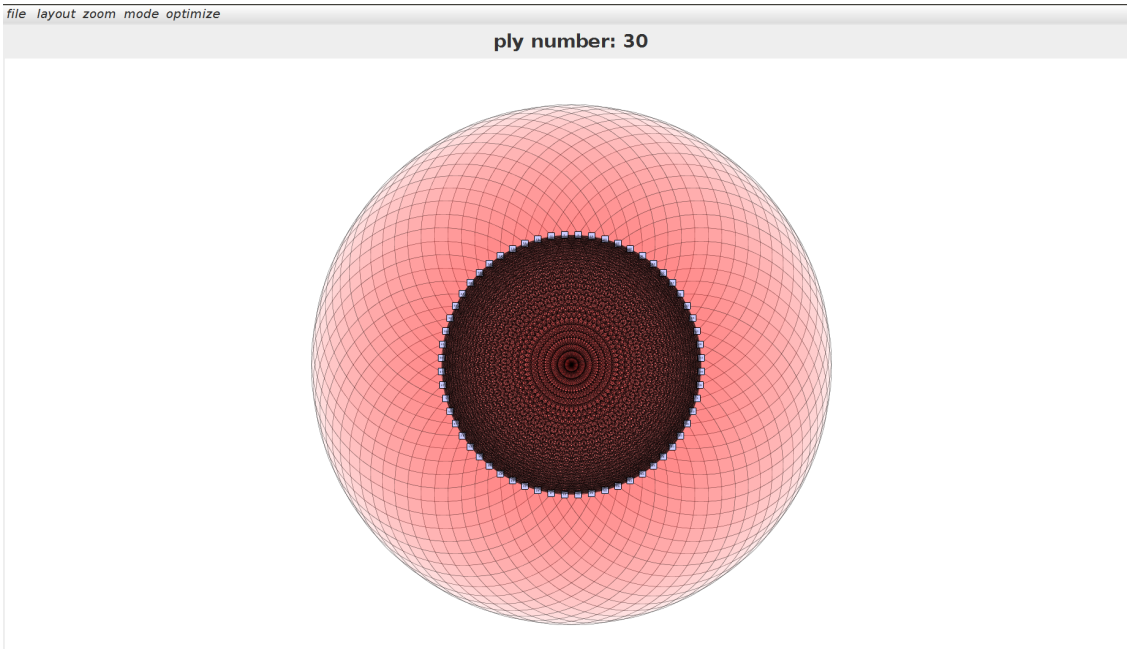


Figure 3.4: The complete graph K_{60} is drawn on a circle. The *ply-number* meets the theoretical upper bound of $\frac{n}{2} = 30$. Note that we conjecture, that this is the best we can do.

circles with some opacity level, such that the user gets some visual feedback, since overlaps are presented in a darker color. Additionally, we indicate the regions with the maximum *ply-number*, which we call *maximum ply-regions*, by drawing blue dots at the intersection points of the *ply-disks*. If the user selects a vertex to move, the edges are colored in a way, that the user can quickly identify the longest edge incident to the selected vertex. In addition to the instant feedback the user will be informed by a color-switch that the longest edge regarding the selected vertex changed.

The user can load and save graphs, choose different built-in layout styles, and modify and inspect the drawing according to its *ply-number*. Modification of the drawing includes moving vertices, where the *ply-disks* are updated instantaneously. In that way the user can get instant feedback on the changes. The indicators for the maximum ply-regions are updated as well. On small to medium graphs the *ply-number* is updated on the fly. For large or highly symmetric graphs we provide the option to switch off the automatic update function. If the automatic update function is toggled off, the *ply-disks* are still updated but the ply-computation is stalled until the modification is finished. In this setting the indicators of the maximum ply-region will not update appropriately.

Since the maximum ply-regions are a local phenomenon, we allow the user to zoom into the view, and shift the view to focus on the desired region and observe the local changes. Note that local changes might influence the *ply-disks* of many adjacent vertices.

LOAD & SAVE

We allow to load any graph from two main file formats, the first file format is `gml` [57], which provides information on the coordinates for each node, as well as information on the representation of the vertices and edges. In our tool we do not allow the modification of these meta-data beside the coordinates in the plane. Edges might contain meta-data regarding the edge routing. Since the *ply-number* is strictly defined on straight-line drawings we discard this information. In the `gml` format, data is stored as a list of node objects followed by a list of edge objects, as indicated in Figure 3.6 and 3.7. We save our graphs in the `gml` format.

As a second supported file format we chose `graphml` [17]. This file format is based on the `HTML` syntax and contains similar types of meta-data like coordinates in the plane, information on the representation of vertices and edges. Even though in these examples the amount of information in the `graphml` format seems to be larger, both formats are able to store the same information. An example of a vertex and an edge is presented in Figure 3.8.

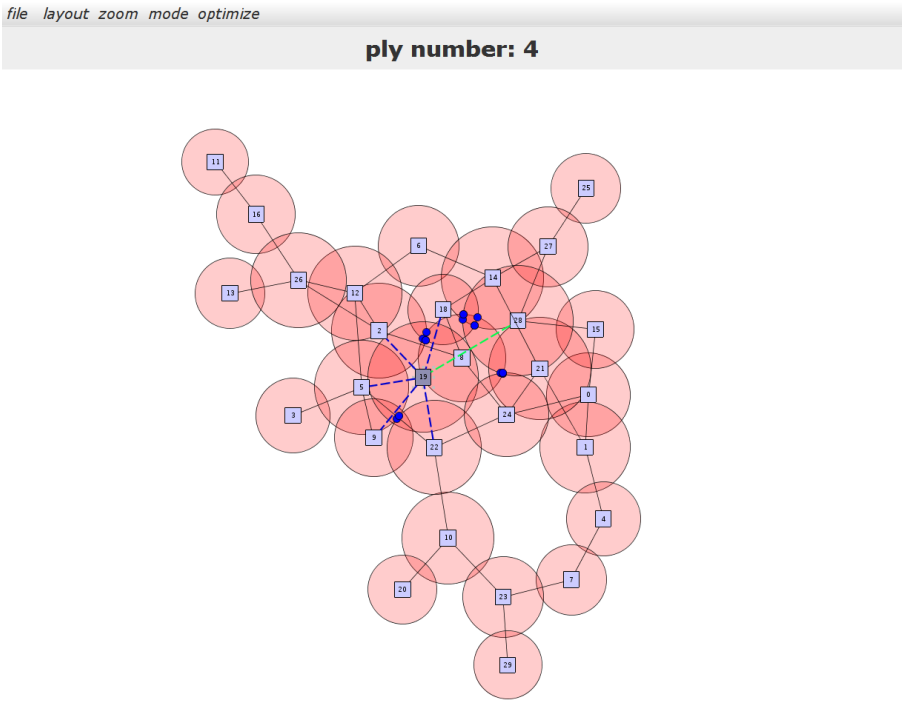


Figure 3.5: The graphical user interface of the ply-exploration tool. A graph with 30 vertices and 40 edges is drawn. The *ply-number* of this drawing is 4. Note that the *ply-disks* are drawn, as well as indicators where the maximum *ply-number* occurs. Additionally, one vertex in the center is marked, and the incident edges are marked blue, whereas the longest edge is yellow. The options a user can choose from are file, layout, zoom, mode and optimize.

ZOOM

For exploration and further analysis of a given drawing, we implemented a zoom function. It allows the user to focus on local parts in a big graph. For easy access this function is bound to the "+" or "-" button as well as the mouse wheel. The displayed part of the graph can be panned by using the left mouse button.

Additionally, the indicators for the maximum ply-regions keep a constant size on the screen, regardless of zooming. In that way, looking at a large graph, the user can observe the regions with the maximal *ply-number*. This feature is presented in the Figures 3.9 and 3.10, where the same drawing in different zoom levels is shown. To reset the zoom we implemented a function to fit the drawing to the view.

```
node
[
    id 0
    label "0"
    graphics
    [
        x 380.77130126953125
        y 585.5761108398438
        w 30.0
        h 30.0
        type "rectangle"
        fill "#CCCCFF"
        outline "#000000"
    ]
]
```

Figure 3.6: The representation of a vertex in `gml` format. The information contains the coordinates in the plane, as well as layout properties regarding the representation of the vertex.

```
edge
[
    source 1
    target 4
    grapics
    [
        fill "#000000"
        targetArrow "none"
    ]
]
```

Figure 3.7: The representation of an edge in `gml` format. The source and the target vertices are given. Note that we consider undirected graphs.

```
<node id="n5">
      <data key="d0" >
        <y:ShapeNode>
          <y:Geometry  x="489.5" y="335.5" width="30.0"
                height="30.0"/>
          <y:Fill color="#FFFF00"  transparent="false"/>
          <y:BorderStyle type="line" width="1.0" color="#000000" />
          <y:NodeLabel x="13.0" y="13.0" width="4.0" height="4.0"
                visible="true" alignment="center"
                fontFamily="Dialog" fontSize="12" fontStyle="plain"
                textColor="#000000" backgroundColor="#FFFFFF"
                modelName="internal" modelPosition="c"
                autoSizePolicy="content">
          </y:NodeLabel>
          <y:Shape type="ellipse"/>
        </y:ShapeNode>
      </data>
    </node>
    ...
<edge id="e96" source="n89" target="n88">
      <data key="d1" >
        <y:PolyLineEdge>
          <y:Path sx="0.0" sy="0.0" tx="0.0" ty="0.0"/>
          <y:LineStyle type="line" width="1.0" color="#000000" />
          <y:Arrows source="none" target="standard"/>
          <y:BendStyle smoothed="false"/>
        </y:PolyLineEdge>
      </data>
    </edge>
```

Figure 3.8: The representation of a vertex and an edge in `graphml` format. This data contains the coordinates of the vertex in the plane as well as layout properties of vertices and edges.

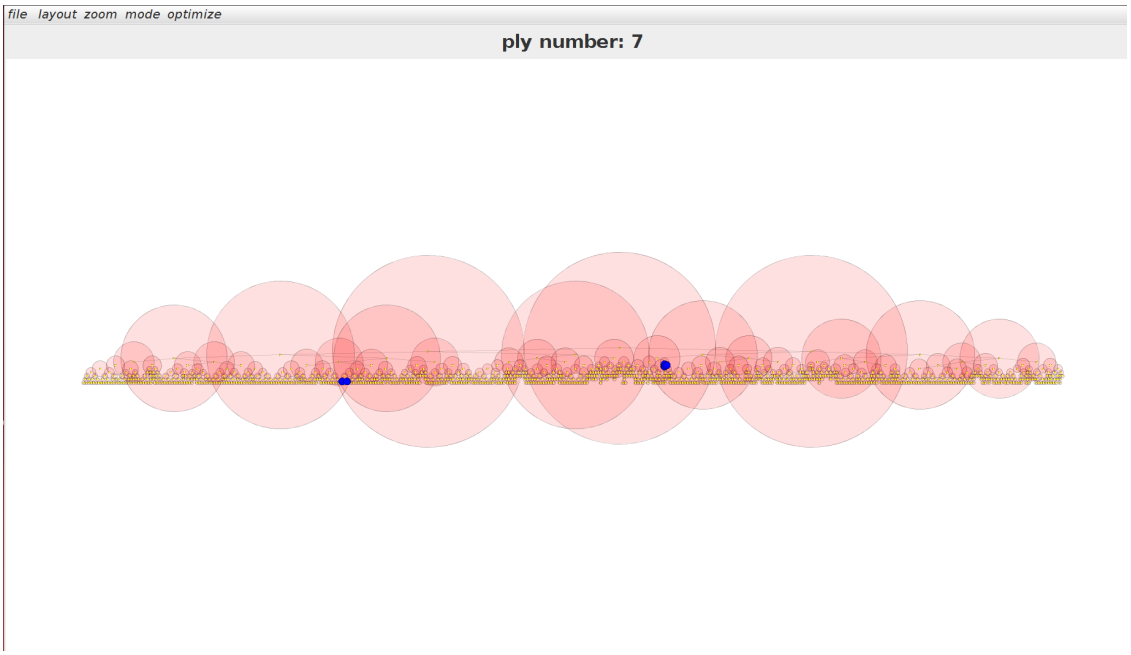


Figure 3.9: A binary tree with 300 vertices is drawn. The *ply-number* of this drawing is 7. Note that the indicators for the maximum ply-regions are observable. Note that the size does not change, when zooming on the two left indicators. This scene is shown in the next Figure 3.10.

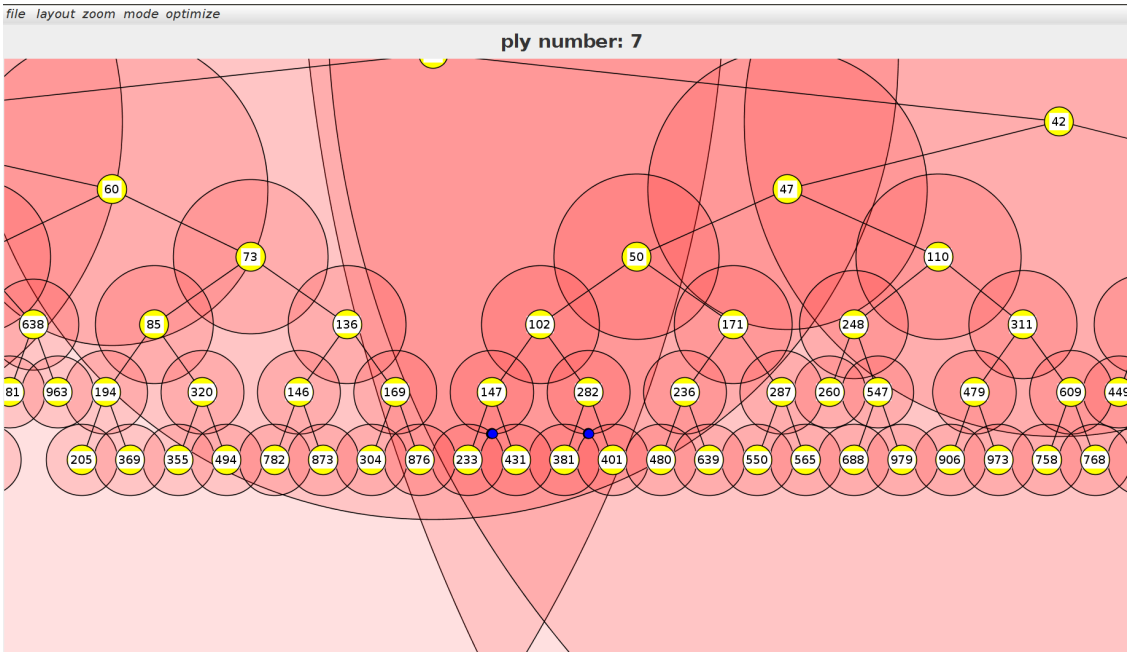


Figure 3.10: A part of the drawing from Figure 3.9 is shown, namely the indicators for the ply-regions which admit the *ply-number* 7.

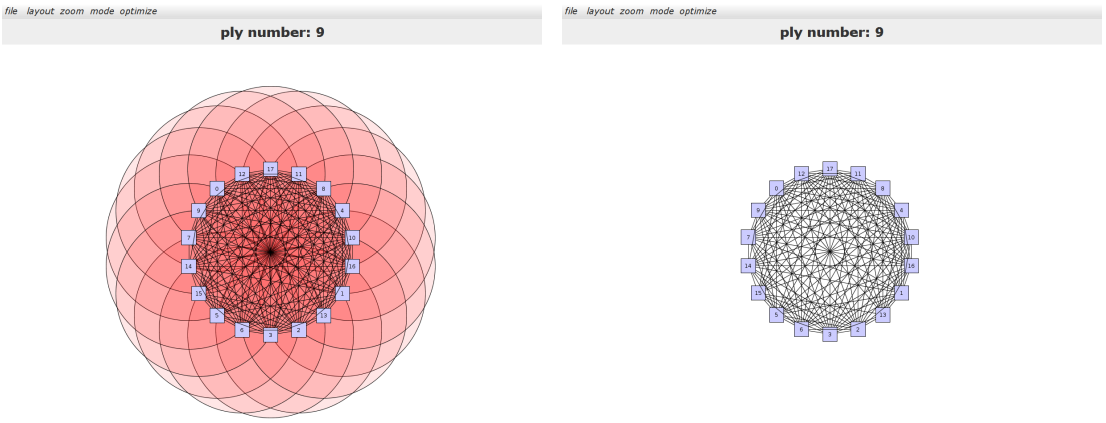


Figure 3.11: The complete graph K_{18} is drawn on a circle. In the right Figure the *ply-disks* are not drawn. Disabling the *ply-disks* enhances the structure of the graph.

MODE

During inspection and exploration regarding the *ply-number* of a graph, it turns out to be helpful to indicate the regions with the maximum *ply-number*, if the user is interested in manual modification, or is interested in the *ply-disks*, which contribute to the *ply-number* of the drawing. The indicators allow the user to quickly identify the regions of interest observing the outline of the drawing. In other cases, the indicators are not necessary, so we allow to toggle the indication of the maximum ply-regions.

Modification of large and highly symmetric graphs sometimes results in a jiggle of the view. These graphs turn out to be hard instances for the algorithm to compute the *ply-number*. To guarantee a smooth modification by the user, the instant feedback on modifications can be toggled off. The user can complete the modifications smoothly and the *ply-number* will be computed once the modification is finished. This does not affect the *ply-disks*, but the indicators of the maximum ply-regions and the *ply-number* will be updated after the modification.

As a last feature we can disable the drawn *ply-disks*, if the user is interested in the structure of the graph, without getting distracted by the *ply-disks*. If we want to examine dense graphs, the difference between edges and the boundary of the *ply-disks* can be confused by the user. Compare the exactly same drawings in Figure 3.11.

3.4 Layout algorithms

Dependent on the users interest, we use the structural information to draw the graph given in the input file allowing to examine the drawing. For further investiga-

tion regarding the *ply-number* of the graph, we provide different layout algorithms, namely organic, circular and random. In our experimental Section 3.6 we will show that the organic layout is most effective on sparse graphs and the circular layout is effective on dense graphs. These three layouts are provided by the `yFiles` library. In the following we present structural properties of the different layout types.

In general, organic layouts are most suited to draw sparse graphs, whereas in dense graphs the vertices tend to form clusters. When investigating drawings regarding their *ply-number*, we observed that drawings with a high edge-uniformity and evenly distributed vertices generally have low *ply-numbers*. For sparse graphs, the organic layout produces drawings with these properties. In contrast to the organic layout, where dense regions form clusters and the placement of vertices seems to be chaotic, highly connected components cluster very symmetrically in the circular layout. We added a randomized placement of the vertices, as a reference for our experiments to approve our results. Additionally, since our automated approaches to minimize the *ply-number* are heuristics, it was helpful to construct drawings, which are not biased by one of the other two approaches or the initial drawing.

ORGANIC

The organic layout is a standard force-directed layout algorithm and is implemented by a spring embedder. This method has been first proposed by Fruchterman and Reingold [47]. A spring-embedder simulates attractive forces along edges and repulsive forces between vertices, as a physical model of springs. Next to the attractive and repulsive forces, there is a parameter, called the spring-stiffness. This parameter allows to influence the variety of edge-lengths throughout the drawing. It is a very common approach to draw straight-line drawings, which result in a natural way to draw a graph [62]. In general, spring embedders can be further extended to satisfy a number of aesthetic criteria, as maximizing crossing angles between edges or enforcing an equal distribution of the vertices. An example is shown in Figure 3.12, the sparse regions (in the lower left part of the drawing) contain very few overlapping *ply-disks* in contrast to denser regions (close to the center), where the vertices are clustered.

CIRCULAR

In the circular layout technique, the vertices are partitioned by connectivity structure in the graph. Highly connected components are placed along a circle. Each partition is placed on a different circle. Initially this layout was developed to emphasize group and tree structures in networks [81]. Recall the proof of the guaranteed *ply-number* of at most $\lfloor \frac{n}{2} \rfloor$, this can be achieved in this layout style by Theorem 2.4. We present a graph drawn in the circular layout in Figure 3.13. Note that the longest edge of the drawing is close to the center of the circle.

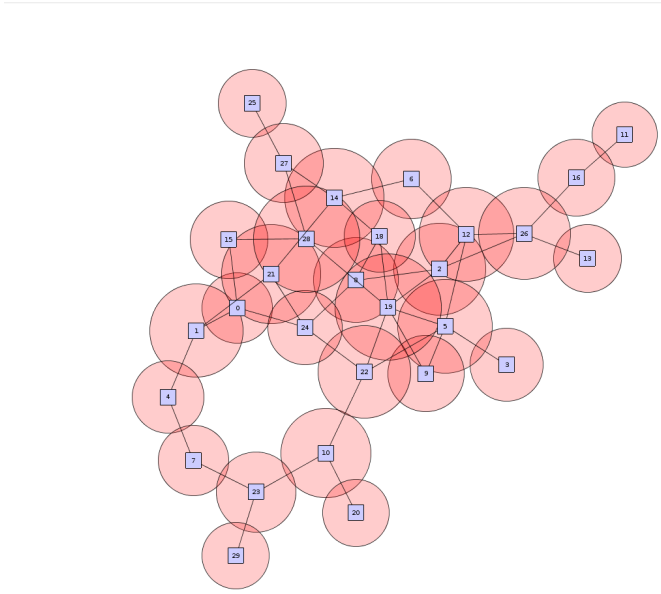


Figure 3.12: The graph is drawn with the organic layout style. The *ply-number* of this drawing is 4. Observe that in sparse regions are very few overlaps i.e. in the bottom left part of the drawing. Denser regions tend to form clusters, i.e. in the center of the drawing.

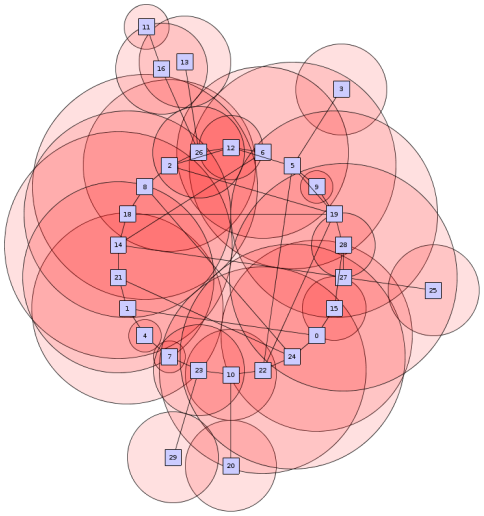


Figure 3.13: A drawing with circular layout style. The *ply-number* of this drawing is 7. Singletons are drawn at the outside and the longest edges in this drawing pass through the center of the circle, on which the vertices are placed regularly.

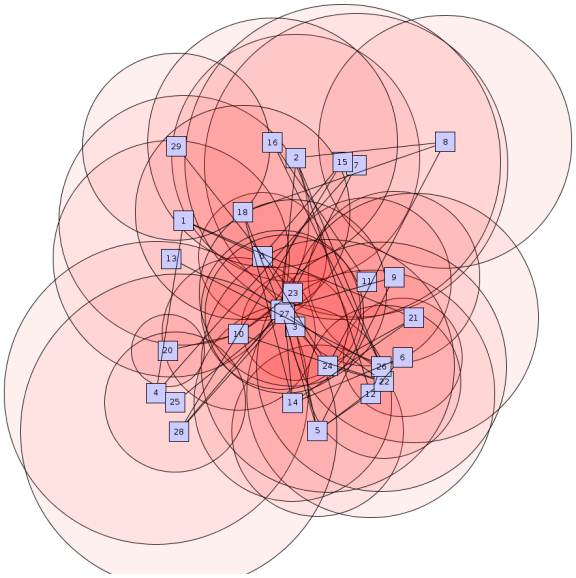


Figure 3.14: A drawing with random layout style. The vertices are placed randomly. We can observe a large number of overlapping *ply-disks*.

RANDOM

The algorithm places all vertices at x and y coordinates which are chosen uniformly at random. Since the coordinates are chosen uniformly, we can see the underlying rectangle, if we have a large amount of vertices. Furthermore, the edges are likely to cross and the *ply-number* is likely to be close to the maximum of n. Even though the randomized placement of the vertices has no benefit in terms of minimal *ply-number*, we can exclude side effects in the computation algorithm, which often occur in symmetrical layouts like the circular one.

3.5 Implementation & Ply Computation

We developed an efficient algorithm to compute the *ply-number* and our tool to support the understanding of the *ply-number* of a drawing of a graph. The main algorithmic challenge is to compute the *ply-number* of a given drawing efficiently in a way that the feedback can be given instantly when the user modifies the drawing. The algorithm, which is based on the plane-sweep technique, will be briefly described in the following paragraphs. We will start with the general concept of plane-sweep algorithms and present the modification to compute the *ply-number* in detail in Section 3.5.2.

Our tool relies on the `yFiles` graph drawing library (Version 2.13), a detailed API can be found at [99]. The `yFiles` library offers a large amount of graph-

drawing tools. Our main functionality is provided by the classes `Graph2D` and `Graph2DView`. The `Graph2D` class describes the structure of the graph, as composed of its vertices and edges. `Graph2DView` is used to display the `Graph2D` object in the plane. Together with `NodeRealizer` and `EdgeRealizer`, it represents a drawing of the graph. `Graph2DView` admits a canvas on which zooming and modifications i.e. moving vertices manually are supported. These modifications are interpreted by this class directly. The drawing can further be modified using the interface `Backgrounddrawables`. This interface allows to draw *ply-disks* for every vertex as well as indicators for the maximum ply-regions. Using the class `Graph2DView` we could implement the zoom function, adjust the colors of the edges dependent on possible vertex selections, and keep track of manual movement of vertices by the user. The layouts a user can choose from are named organic, circular and random, which are computed on the `Graph2D` object.

We conclude this section with a discussion on precision errors using the data type `double` throughout our implementation. The previous implementations [7][30] used the `ApFloat` library. We will point out the differences between the different approaches.

Recall that the *ply-number* of a drawing is defined as the maximum number of overlapping *ply-disks* for any coordinate in $\mathbb{R}^2$. To compute the *ply-number* of a drawing we need to identify the maximal overlaps of *ply-disks* in the plane. The problem to compute the *ply-number* for a given drawing can be reduced to identify the maximal number of overlapping circles for any point. Since this is a geometric problem we used a plane sweep technique like suggested in [7] and [30]. In the following we present the basic concept of the algorithm, which was initially designed to calculate all the intersections of a set of straight segments. Then we will point out how to compute the *ply-number* by identifying intersections between circular segments. Furthermore, we will describe the similarities and adjustments to the original plane-sweep algorithm.

3.5.1 Line Segment Intersection

The idea of the plane-sweep algorithm was introduced by Shamos and Hoey [80] and improved by Bentley and Ottman [10] and proved to be a powerful technique in computational geometry. To present the ideas we follow the description from the textbook [28]. The initial task of this algorithm is formulated as follows:

Given a set $S = \{s_1, ..., s_n\}$ of n straight-line segments in the plane, detect all intersections. In the following we assume general positioning of the line segments in the plane, that is

1. no two endpoints of line segments share the same $x-$coordinate
2. no two crossings share the same $x-$coordinate

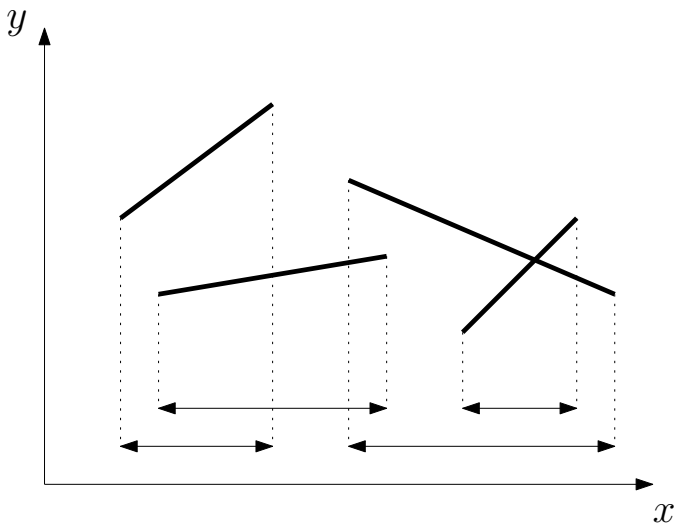


Figure 3.15: A crossing of two line segments can only occur in the union of their $x-$intervals.

3. no two line segments lie on top of each other

4. no three line segments intersect at the same coordinate.

An intuitive solution to this problem would be to consider every pair of segments and test whether they intersect, which would result in a quadratic time complexity. In order to achieve a faster algorithm we can observe the following: A crossing of two line segments can only occur in the intersection of their x-intervals. The $x-$interval is defined as the projection of the line segment to the $x-$axis as presented in Figure 3.15. Thereby we only check for intersections between segments, whose $x-$intervals overlap, that is pairs of segments for which there exists a vertical line intersecting both segments.

Sweeping the plane from left to right with a conceptual vertical line L, called *sweep-line*, we reduce the candidates to check for intersections to the segments, which are currently intersecting L. These segments are called active segments.

The *status* of the sweep line L is the set of active segments ordered from top to bottom along L. The status can change during the sweep at specific coordinates. These coordinates are called *events*. An event can be one of the following and between any two consecutive events the status remains unchanged:

1. at the leftmost coordinate of a segment, the segment will be marked active

2. at the rightmost coordinate of a segment, the segment will be marked inactive

3. at an intersection the order of segments swap in the sweep line status.

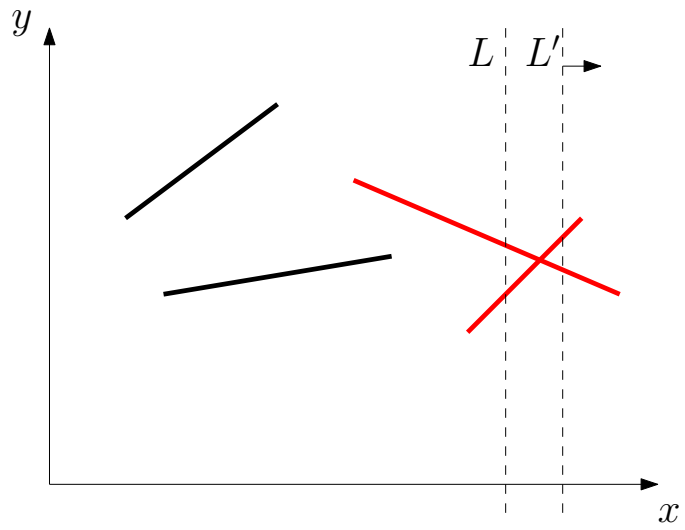


Figure 3.16: The segments crossing the vertical sweep line L are marked in red. Whenever two line segments cross there exists a vertical line crossing both at the same $x-$coordinate. At a crossing of two active line segments the order from top to bottom is swapped and the staus L is updated to status L'.

At an intersection of two line segments, the order from top to bottom will change in the sweep line status. This is illustrated in Figure 3.16, where the order of active segments is swapped between the status of L and the status of L'.

The second observation is that two line segments can have an intersection only if they are consecutive in the sweep-line status. Thus, whenever a segment is marked active, it will be checked for intersections with the segment above and the segment below in the sweep line status. Note that a new active segment requires at most two tests for intersections since it has at most two direct neighbours in the sweep line status. Marking a segment as inactive requires at most one test for intersections, since the top and bottom neighbour might not have been neighboured before. Finally an intersection, and thereby a swap in the order along L, requires at most two tests for intersections, since both candidates might get a new neighbour.

The algorithm is executed as follows: Initially for every segment there is one event at the leftmost and one event for the rightmost coordinate. These events are stored in an event-queue ordered by the x-coordinate. For each event we add the segment to the sweep line status, if the event is a leftmost coordinate. The new active segment will be added according to its intersection with the sweep line L and will be tested for intersections with its neighbours. If there is an intersection, the intersection coordinate will be added to the event-queue. If the event is a rightmost coordinate, the segment will be removed from the sweep line status and the neighbours of the removed segment will be checked for intersections, if not done at some state already. Again, if there is an intersection it will be added to the event-

queue. If the event is an intersection of two segments, we report the intersection and the two segments will be swapped in the order of segments in the status and we will check for intersections with the new neighbours, if not done already.

If the event-queue is empty, we have swept the whole plane from left to right and the set of active events is empty. As the invariant of this algorithm, we can guarantee that every crossing to the left of the sweep line L has been detected. Note that during the sweep each pair of neighboured segments have to be tested for intersections exactly once.

As data structures for the event queue and the y-structure balanced binary search trees are used. This can be done, since a total ordering can be enforced. Thereby adding or removing events can be done in $O(\log\ n)$ time as well as the search neighbour operation. The total running time is $O((n + k)\ \log\ n)$ where n is the number of segments and k is the number of intersections.

3.5.2 Computation of the *ply-number*

Recall the initial problem: Given a straight-line drawing Γ of a graph $G = (V, E)$ in the plane, we can easily compute the set of *ply-disks* $D = \{D_v | v \in V\}$. Every disk D_v is associated with the vertex v at position (x_v, y_v) and radius r_v. The *ply-number* of the drawing is the maximum number of overlapping *ply-disks* in the plane.

We can reformulate this problem such that the plane-sweep algorithm computes the *ply-number*. Therefore we state some observations on the *ply-number* of drawings. Sweeping upon a set of *ply-disks* along the x-coorinate, the *ply-number* can change, if we observe a leftmost coordinate of a *ply-disk*, the rightmost coordinate of a *ply-disk* or at an intersection of two *ply-disks*, as suggested in Figure 3.17. Formally, the ply-value can change whenever a disk D_v starts at $(x_v - r_v, y_v)$, ends at $(x_v + r_v, y_v)$ or if there is an intersection of two disks. Each disk can be represented by two halfcircles by cutting the disk at a horizontal line through the center. Thereby one halfcircle corresponds to the top part of the *ply-disk* and one halfcircle to the bottom part. Observe that the two halfcircles corresponding to one *ply-disk* enclose a region where the ply-value is at least one.

In contrast to line-segment intersection, we are now looking at halfcircle intersection, where a halfcircle is either a top or a bottom part of a *ply-disk*. The *sweep-line status* is represented as an ordered list of opening and closing halfcircles along the vertical sweep line L in a ply drawing. For each state at a fixed x-coordinate every halfcircle has a specific y-coordinate and the order if fixed. For each halfcircle we remember the current ply-value above and below. Note that the ply-value above and below any halfcircle differs by exactly one. The ply-values associated with a halfcircle can change, if there is an intersection and thereby a reordering of the halfcircles or if halfcircles are inserted or removed from the sweep-line status. In any state between two events the order of halfcircles is fixed and thereby the ply-value

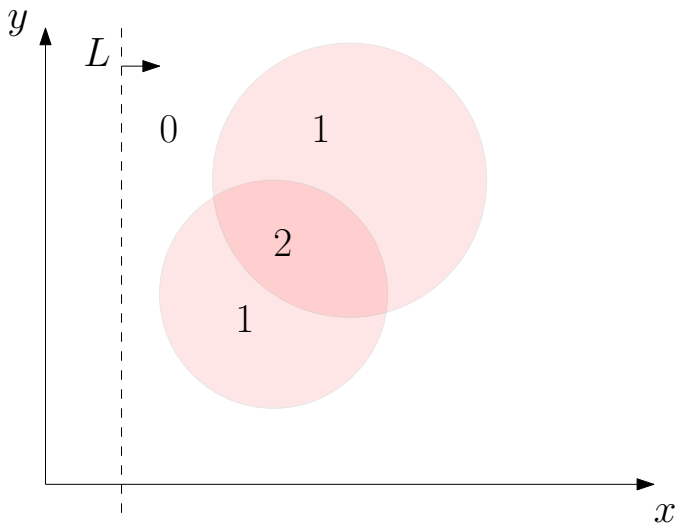


Figure 3.17: The *ply-number* for a drawing can change at the leftmost coordinate, at the rightmost coordinate or if two *ply-disks* intersect. The ply-values for the different regions are given.

can be determined by counting top and bottom halfcircles. The ply-value cannot change between any two events. The *ply-number* of the drawing is the maximum number over all ply-values as presented in Figure 3.18. Note that two disks might intersect if and only if any two corresponding halfcircles occur next to each other in the vertical structure. Furthermore, any two disks D_u and D_v can intersect at most twice. To keep the computational effort minimal the intersection of disks is calculated once the first time any two halfcircles appear next to each other in the vertical structure.

Now we will describe the type of events that occur during the plane-sweep algorithm and how the sweep line status is updated at these events.

A **start event** is defined at the leftmost coordinate for every *ply-disk* D_v. For the sweep line status we insert two halfcircles, one corresponding to the top part of the *ply-disk* (H_v^t) and one corresponding to the bottom part of the *ply-disk* (H_v^b). Both *ply-disks* have the same $y-$coordinate in the sweep line status and we define the top part to be above the bottom part consistently to the order of intersections along the sweep line L. For each halfcircle we associate a current ply-value. For newly inserted halfcircles, the ply-value in between is defined as the ply-value above (below respectively) plus one, i.e presented in Figure 3.19. The halfcircles are tested for intersections with their immediate neighbours. If there is an intersection, this intersection is added as an intersection event. As a special case, it might occur, that there already exists another halfcircle sharing the same (x, y)-coordinates as the leftmost coordinate of the *ply-disk*. In this scenario, there exists an implicit intersection event at the same coordinates as the start event. We add the new

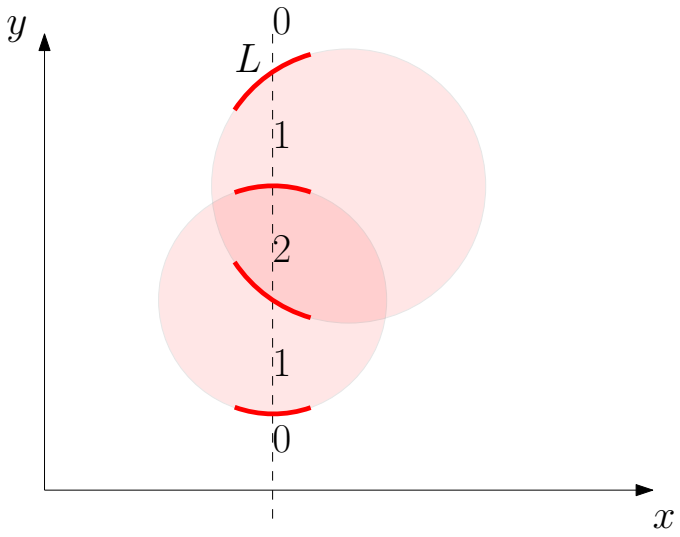


Figure 3.18: The status of the sweep line L contains the active halficrcles, partially drawn in red and the *ply-number* associated to the regions between the halfcircles. Note that the ply-value right above and below a halfcircle can differ at most by 1 and the value is always 0 at the top and the bottom of L.

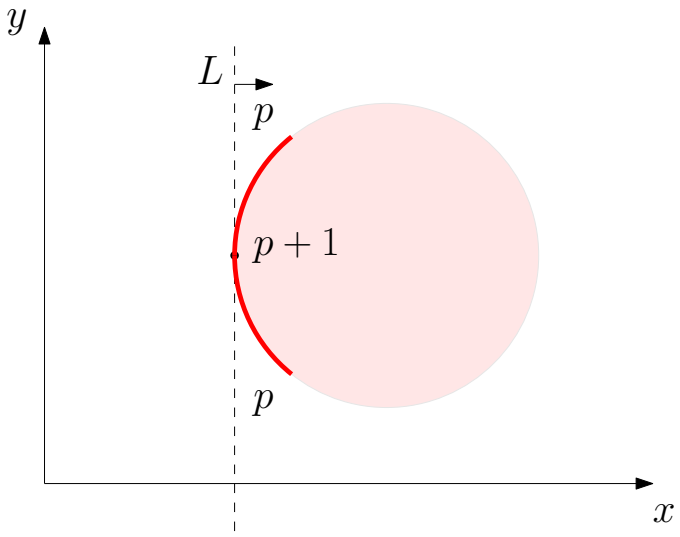


Figure 3.19: At a **start event** associated with the *ply-disk* D_v, we insert the two halfcircles H_v^t and H_v^b into the sweep line status. The former ply-value p of the region is increased to $p + 1$ between the two halfcircles.

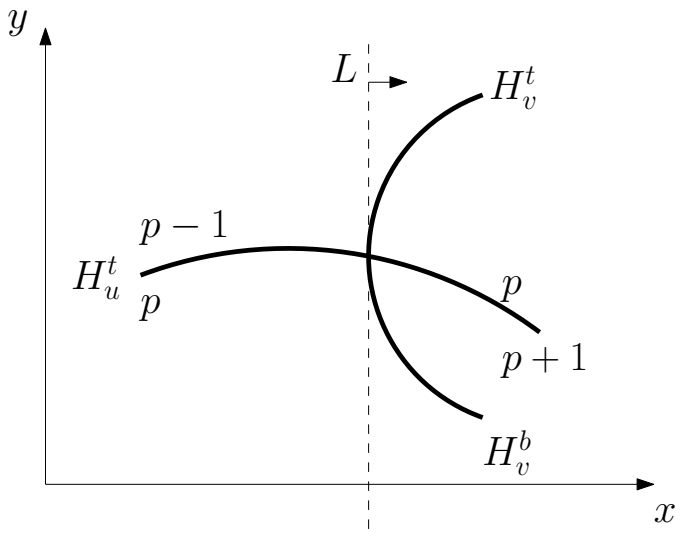


Figure 3.20: The start event corresponds to an intersection event of H_u^t and either H_v^b or H_v^t. The newly inserted halfcircles are placed above (below respectively) the halfcircles that are present at the corresponding y-coordinate. The ply-values of the halfcircles in between are increased by one.

halfcircles by inserting them above (below respectively) the present halfcircles as illustrated in Figure 3.20. The ply-values of the halfcircles between the newly inserted are increased by one, whereas the new ply-values are set by the regions directly above and below.

An **end event** is defined as the rightmost coordinate $x_v + r_v$ for any disk D_v. At the rightmost coordinate of a *ply-disk*, the region admitting a ply-value $p+1$ is enclosed and the remaining region beyond the removed *ply-disk* has the ply-value p as shown in Figure 3.21. Accordingly, we remove the corresponding halfcircles H_v^t and H_v^b from the sweep line status. To ensure consistency, the newly neighboured halfcircles are tested for intersections. Ideally the halfcircles to remove are already consecutive in the sweep line status. It might happen that there exist halfcircles in between, for example if the x-coordinate is shared by an intersection event with either H_v^t or H_v^b. For any halfcircle crossing an end event, we decrease its ply-value as presented in Figure 3.22. Further intersections with a removed halfcircle are neglected.

Whenever two halfcircles intersect, there is an **intersection event**. An intersection relates to a swap of halfcircles in the sweep line status. Furthermore, at such an event the ply-value either increases or decreases dependent on the type of intersection. First we will describe the intersection of two bottom halfcircles. The update of the stored ply-values depend on the relative positioning of the sweep line status. Intersecting the halfcircle H_v^b with the halfcircle H_u^b, they have to occur next to each other in the sweep line status. Let H_v^b be w.l.o.g. below H_u^b, then during

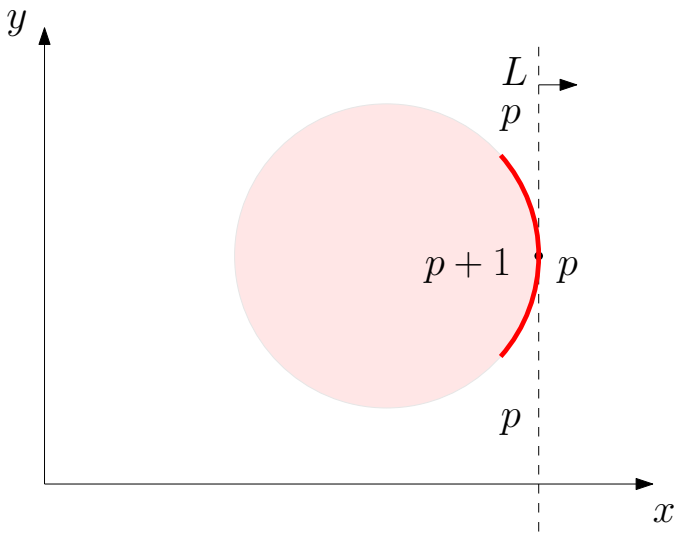


Figure 3.21: At an **end event** associated with the *ply-disk* D_v, we remove the two halfcircles H_v^t and H_v^b from the sweep line status. This encloses the region with ply-value $p+1$, leaving the ply-value p.

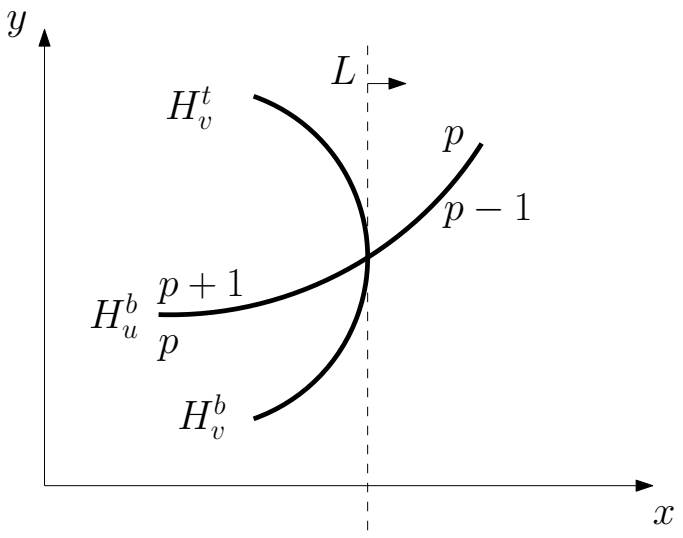


Figure 3.22: An end event is executed even if there is a halfcircle H_v^b in between. The ply-values of the halfcircle between H_v^t and H_v^b are decreased by one, since it either intersects H_v^t or H_v^b. If we encounter an intersection event with any of the removed halfcircles, it will be neglected later.

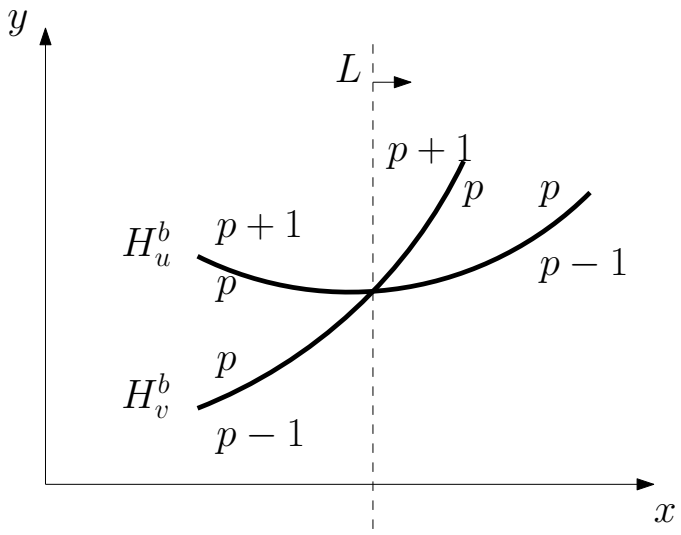


Figure 3.23: An event of two intersecting botom halfcircles is drawn, where H_v^b is below H_u^b in the sweep line structure. For each halfcircle the ply-value below and above is stored. Note that this value has to be consistent during all computational steps. The halfcircle below enters the *ply-disk* D_u and thereby its ply-values increase by one at this intersection. Respectively the ply-values of the halfcircle H_u^b decrease by one. By symmetry this applies to top halfcircles intersecting each other too, where the relative positioning in the sweep line status is reversed.

the intersection, the halfcircle H_v^b enters the *ply-disk* D_u and thus the ply-values stored for H_v^b increase by one. Symmetrically the ply-values stored for H_u^b decrease by one. By symmetry this applies to the top halfcircles intersecting each other too, where the relative positioning in the sweep line status is reversed.

The intersection of bottom and top halfcircles again depend on their relative positioning. A bottom halfcircle intersecting a top halfcircle from below correspond to an intersection of *ply-disks* at the rightmost coordinate of their intersection. Therefore, the ply-value associated to the halfcircle is reduced. The top halfcircle in this case reduces its ply-values too, for the same reason. An illustration of this case is shown in Figure 3.24. A bottom halfcircle intersecting a top halfcircle from above, enters the *ply-disk* corresponding to the top halfcircle and thereby increases its ply-values. Similarly the top halfcircle enters the *ply-disk* corresponding to the bottom halfcircle. Both ply-values increase as presented in Figure 3.25. Note that a swap of halfcircles is possible if and only if the halfcircles are consecutive in the sweep line status.

In the event-queue the events are stored and sorted by their x-coordinate of occurrence. In general positioning the x-coordinates are unique for the events, where in our case there can be several events at the same coordinate. If there are several events at the same coordinate, we execute the different types in the following

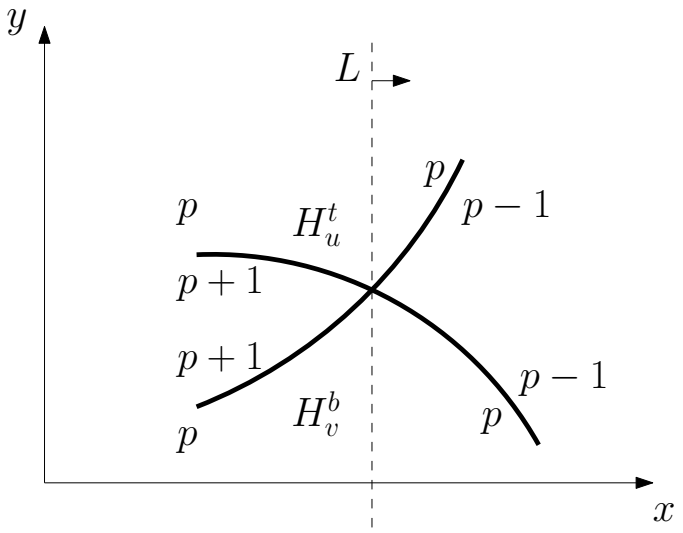


Figure 3.24: The intersection of a bottom halfcircle and a top halfcircle, where the relative positioning of the bottom halfcircle is below the top halfcircle. Here both halfcircles decrease their ply-values.

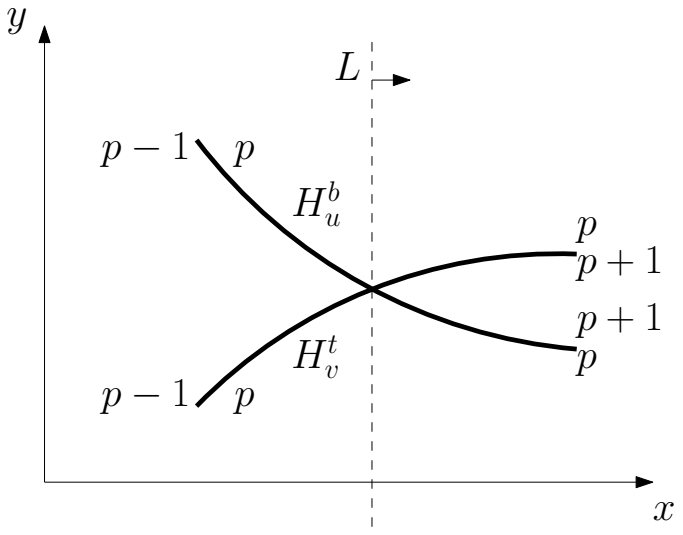


Figure 3.25: A bottom halfcircle intersecting a top halfcircle from above. In this case the ply-values for both halfcircles increase, since they form a new overlap of *ply-disks*.

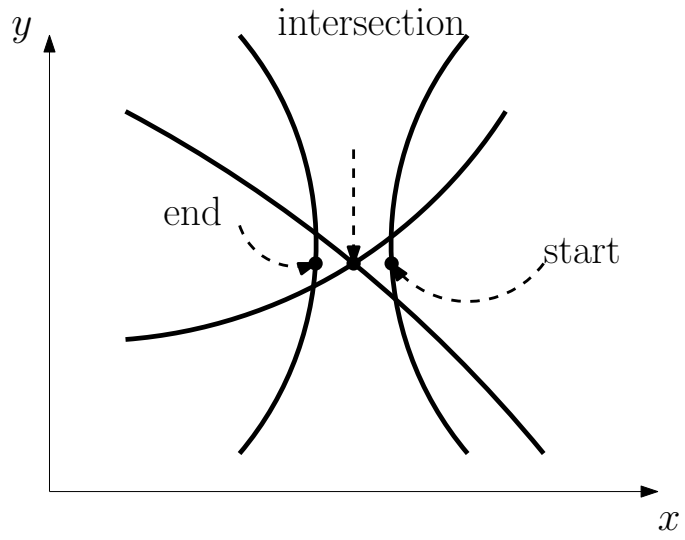


Figure 3.26: Several events at the same (x, y)-coordinates are taken care of in the following order: We prioritize end-events and then we solve all intersection events and at last we insert new halfcircles according to the start event.

order: end, intersection, start as presented in Figure 3.26. We prioritize end events since we simply remove both components, even if in the actual state there are other halfcircles in between. We decrease the ply-value at each intermediate segment by one. We confirm this rule by the definition of "open" *ply-disks*, such that even if there is an intersection at the corresponding $x-$coordinate which might increase the *ply-number*, we do not want to have the ending disk to be involved.

We have no proper way to determine which intersection event should be executed first, when we identify them. There might be several intersection events sharing the same coordinates. Therefore we allow intersection events to be executed if and only if the corresponding halfcircles are adjacent in the sweep line status. Assume we have the intersection event intersecting the two halfcircles H_v and H_u but in the current sweep line status there exists another halfcircle H_w in between. In that case, we can conclude that there exists the intersection event intersecting the intermediate halfcircle H_w with either H_v or H_u and the intersection events are not ordered correctly in the event-queue. As we describe in the following Section 3.5.4, this can have several reasons. If we encounter such an intersection event, we cannot execute in the actual status, we linearly search for the next event and try the postponed event again at the next state of the sweep-line. If we have several events at the same x-coordinate, especially if we have several events at the same (x, y)-coordinate, we want to state that the order of execution is irrelevant as long as we only swap neighboured halfcircles.

The third priority are start events and internally the start events are ordered by the radii of the corresponding *ply-disks*. A smaller radius has higher priority than

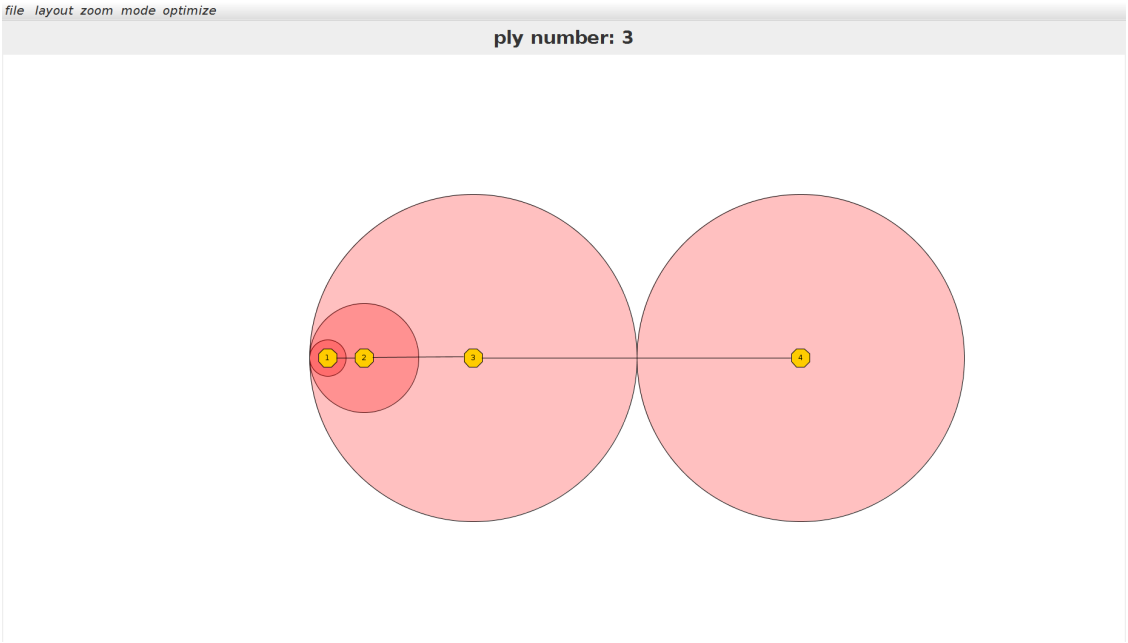


Figure 3.27: A graph with 4 vertices is shown. The *ply-disks* of the vertices 1 to 3 start at the same coordinate.

a larger radius. The rule to add new halfcircles to the sweep line status is to check for existing halfcircles at the same $y-$coordinate and insert the top halfcircle above and the bottom halfcircle below respectively. The ply-values for each halfcircle in between is increased by one and we set the ply-values for the inserted halfcircles accordingly. This requires the start-events to be ordered internally by their radii as shown in Figure 3.27.

3.5.3 Computing Intersections

Let the disk D_u be the *ply-disk* with center $u = (u_x, u_y)$ and radius r_u intersecting the *ply-disk* D_v with center v. Furthermore, let w.l.o.g. be $u_x > v_x$ and by *dist* we denote the distance between u and v. Note that $r_u + r_v > dist$, since otherwise the circles are disjoint, as well as $dist + r_u > r_v$ and $dist + r_v > r_u$ since otherwise one disk is in the interior of the other or if equality holds there is exactly one intersection point, which we neglect in the computation.

Note that for any two intersecting circles there exist two right-angled triangles, one with the lengths ℓ, r_u and a, and the second one with the lengths $(dist - a), \ell$ and r_v as shown in Figure 3.28.

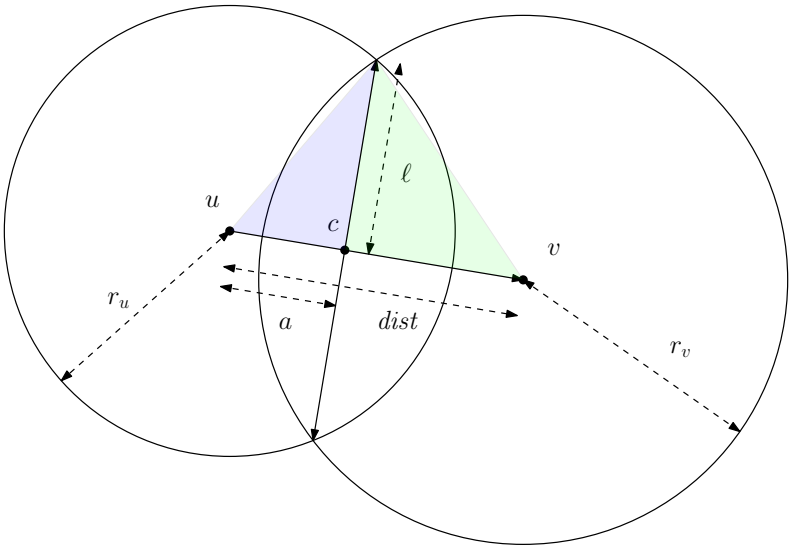


Figure 3.28: Two intersecting disks form two right triangles, one with the lengths r_u, ℓ and a (blue) and the second one with the length r_v, ℓ and $(dist - a)$ (green).

For the two triangles we get the formulas:

$$r_u^2 = \ell^2 + a^2$$
$$r_v^2 = \ell^2 + (dist - a)^2.$$

Solving the equations to ℓ^2 and plugging in yields:

$$r_u^2 - a^2 = r_v^2 - (dist - a)^2$$
$$\Leftrightarrow a = \frac{r_u^2 - r_v^2 + dist^2}{2 \cdot dist}$$

Furthermore, we can determine ℓ as:

$$\ell = \sqrt{|r_u^2 - a^2|}$$

For the computation of the intersections we use the vector $\overrightarrow{uv}$ which we scale to have the length a. The vector $\overrightarrow{c} = \overrightarrow{u} + \overrightarrow{uv} \cdot \frac{a}{dist}$ denotes the central coordinate between the two intersection points of the disks D_u and D_v. As a last step, we construct an orthonormal vector $\overrightarrow{h}$ with $||h|| = 1$ and $\overrightarrow{h} \times \overrightarrow{uv} = 0$ and scale it to have the length ℓ. The intersections i_1 and i_2 can now be described as

$$i_{1,2} = \overrightarrow{c} \pm \ell \cdot \overrightarrow{h}$$

and the relative positioning of the intersections regarding the center of the disks determine, whether the corresponding top or bottom halfcircles will intersect.

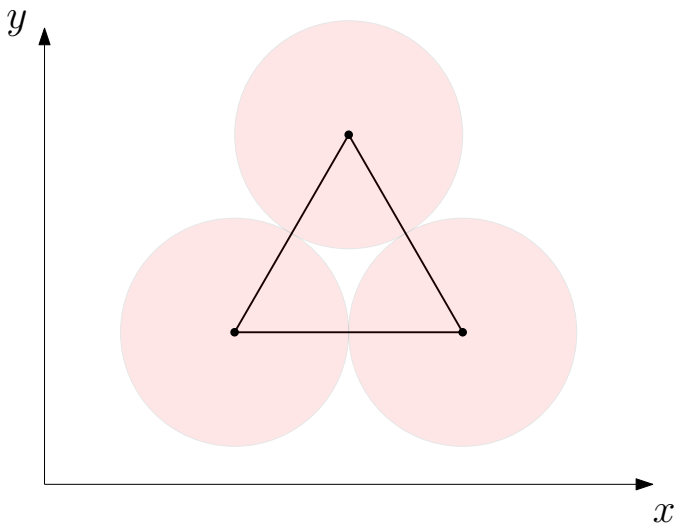


Figure 3.29: A complete graph K_3 drawn as an equilateral triangle admits a drawing with *ply-number* 1. Furthermore, it requires irrational coordinates which can not be represented by common datatypes. Therefore the algorithm to compute the *ply-number* computes the *ply-number* of two for any drawing of K_3.

3.5.4 Precision Problems & Debugging

Naturally thinking of an easy case to start with are graphs that admit a drawing with a *ply-number* of 1. This case is easy to describe and points out the difficulty of computing the *ply-number*: A graph that admits a drawing with *ply-number* 1 has no overlapping *ply-disks* and can be drawn such that every edge has equal length l and any two vertices are at a distance of at least l.

Suppose that there exists a drawing Γ and a vertex v with different edge lengths $|(u, v)| = l_1$ and $|(w, v)| = l_2$ and let w.l.o.g $l_1 > l_2$. $\frac{l_2}{2}$ is a lower bound on the *ply-disk* D_w and since $\frac{l_1}{2} + \frac{l_2}{2} > l_2$ the *ply-disks* D_v and D_w intersect and the *ply-number* of Γ is ≥ 2. Furthermore, since the radius for any *ply-disk* D_v is $\frac{l}{2}$, the distance $\text{dist}(v, w)$ between any two vertices has to be $\geq l$.

The complete graph K_3 admits a drawing with *ply-number* 1, since the vertices can be placed on an equilateral triangle. Computing a drawing of K_3 with the vertices u, v, and w, some coordinates must be irrational, since otherwise the condition $\text{dist}(u, v) = \text{dist}(u, w) = \text{dist}(v, w)$ is violated as shown in Figure 3.29.

Since a computer is limited in the representation of numbers, we run into precision problems, since the computer would often need an infinite precision to represent a drawing with *ply-number* 1. Additionally, the calculation of coordinates for intersection points of circles involves precise arithmetic and is likely to result in irrational coordinates.

During the implementation of the plane sweep algorithm we experienced a number of precision errors in the sweep line status. To check for consistency at the

```
Element 3 : TOP 10, current Y position: 1307.011187049395
Element 4 : TOP 7,  current Y position: 1295.4695805456354
Element 5 : TOP 1,  current Y position: 1235.4794682409479
Element 6 : BOT 10, current Y position: 1163.82255318498
Element 7 : TOP 66, current Y position: 1163.8225531849826
Element 8 : TOP 83, current Y position: 976.3626815537044
Element 9 : BOT 39, current Y position: 959.5613407831479
```

Figure 3.30: The actual event is an intersection between H_{10}^b and H_{66}^t. The execution of the intersection event and thereby the swap of these halfcircles will recover the correct order of the halfcircles, whereas the current status is inconsistent.

```
  Element 87 : BOT 43, current Y position: 969.0000050810148
  Element 88 : TOP 23, current Y position: 969.0000050810147
  Element 89 : BOT 62, current Y position: 969.0000027201414
>>Element 90 : TOP 28, current Y position: 969.0000027201404 <<
  Element 91 : BOT 72, current Y position: 969.000002720141
>>Element 92 : TOP 99, current Y position: 969.0000027201406 <<
  Element 93 : BOT 9,  current Y position: 969.0000013131587
  Element 94 : TOP 47, current Y position: 969.0000013131586
  Element 95 : BOT 22, current Y position: 968.9999972798596
  Element 96 : TOP 61, current Y position: 968.9999972798594
```

Figure 3.31: The current intersection event indicates an intersection between H_{28}^t and H_{99}^t but there still exists the halfcircle H_{72}^b. The next event solves this issue, since it is the event indicating the intersection between H_{28}^t and H_{72}^b.

current status, we compute the y-coordinates for every halfcircle. As an example we present a part of the sweep line status where the current event is an intersection event between the two halfcircles H_{10}^b and H_{66}^t. By definition of an intersection point, the (x, y)-coordinates of both halfcircles have to be equal, whereas the top halfcircle already admits a higher $y-$coordinate. One of two reasons might have happened. Either the x- or the y-coordinate is not exact due to rounding. The current status is shown in Figure 3.30.

More importantly, these precision errors can induce events which cannot be handled consistently, for example an intersection-event that requires a swap of halfcircles, which are not neighboured in the current state. In the following scenario, the current intersection event indicates an intersection between H_{28}^t and H_{99}^t but there still exists the halfcircle H_{72}^b in between. Additionally, the $y-$coordinate suggests that the halfcircle H_{72}^b should be above H_{28}^t.

Our solution to this scenario is linearly searching for the closest consistent event.

This event will be executed and we jump back to the unresolved one. We repeat this until it can be resolved. The following intersection event swaps H_{28}^t and H_{72}^b, so if we revisit the initial intersection event, it is now solvable. The number of revisited events will be tracked as **postponed events**. In the results section we evaluate this delay and describe graphs where this periodically occurs.

During the plane-sweep algorithm halfcircles are checked for intersections, once they occur next to each other in the sweep line status. Here another type of precision problem might occur, if we encounter two halfcircles next to each other. It might have happened that their first intersection has been at a previous x-coordinate, if there was a rounding error while computing the intersection event which causes the halfcircles to be next to each other. When encountering two halfcircles for the first time, we allow a short distance before the actual sweepline. The distance is set to 0.0001. This range seems to be sufficient, since the precision errors we observed were around 0.00001. A similar problem involves the leftmost and the rightmost coordinates of the *ply-disk*. If there is an intersection exactly at the start coordinate we cannot distinguish whether there is an intersection with the bottom or top part of the halfcircles. Ultimately we want to enclose these halfcircles entering the newly inserted *ply-disk* between the corresponding halfcircles. Therefore, we allow intersections with the top halfcircle, if the other halfcircle is above in the sweep-line status and allow an intersection with the bottom halfcircle if it is below in the sweep-line status, respectively.

Previous Applications

In previous applications [7, 30] precision errors were tackled by increasing the precision using the Apfloat library. The `apfloat` library allows to define the number of digits, which will be calculated to be precise. This allows calculation on up to 1000 digit decimal precision. On the downside these arithmetics require high computational effort and long runtimes.

These implementations suggested that if there is an inconsistent intersection event, there had to be a precision error. To determine the correct order of intersection events, the decimal precision of the coordinates were iteratively increased in [7]. In [30] the decimal precision was set to 20 digits and between the events they added consistency tests. If an intersection event was encountered, they recalculate possible intersections to resolve this error. The main issue in these implementations occurs, if by a rounding error, the $x-$coordinate of an intersection is incorrect.

Furthermore, they do only allow the removal of neighboured halfcircles if an end event is queued, this favours intersection events where in our implementation these intersection events will be neglected. In fact, we encountered some drawn instances, where the *ply-numbers* by the two computations differ.

3.6 Experiments

We applied some experiments to evaluate our algorithms. We compare the different layouts a user can choose from regarding to the *ply-number* of the resulting drawing on a variety of different graphs. We start with a description of the graph sets we used, stating the number of graphs, the number of vertices and the density, meaning the edge to node ratio. Additionally, we compare our implementation in terms of *ply-number*, running time, and the number of events executed during the plane-sweep algorithm with the previous implementation by De Luca et al. [30]. As a third part, we present and evaluate an approach to minimize the *ply-number* for a given drawing automatically.

3.6.1 Datasets

Rome graphs The first set is the set of graph also known as *Rome graphs*. The Rome graphs are a large set of graphs publicly available at [84]. They have been introduced to provide benchmark data for graph drawing algorithms [95]. The set consists of roughly 11500 graphs, where each graph consists of 10 to 100 vertices. The ratio of edges per node differs from 1 to 2. The files only contain structural information so we have to compute an initial layout. We use this set to compare the different layouts on sparse graphs.

For each layout we will compute the *ply-number* to present an intuition to the different layouts on sparse graphs and the implications on the *ply-number*. Additionally, we present some statistics on the computational speed of the algorithm as well as the number of events which are executed during the plane-sweep algorithm. In the following we will refer to this set of graphs as **ROMEdata**.

Randomly generated graphs The second set includes 60 graphs with 100 vertices each. The set includes 5 trees with low and 5 trees with high degree inner vertices. For the other graphs the number of edges is set to 150, 250, 500, 650, 800, 1000, 1200, 1500, 2000 and 4000. These graphs are assigned randomly, meaning we choose two vertices randomly and add the edge between them until the number of edges is reached. During this process we maintain the simplicity of the graph. This set of graphs induces a higher variety in terms of density than the **ROMEdata**. In the following we refer to this set as **RANDdata**.

FM3 drawings The third set will be referred to as **FM3data**. This set of graphs was kindly provided by the authors of the experimental study [30]. Each graph was drawn using the fast multipole multilevel method (**FM3**) of Hachul and Jünger [53] which is among the most effective force-directed algorithms in the literature [54]. The **FM3** produces drawings with low *ply-numbers* on average and was one of the best performing algorithms in the experimental study by De Luca et al. [30]. Since

this set was used in previous applications, we use this set to compare computation time and to evaluate our optimization approach and whether we can identify drawings having a smaller *ply-number* than the drawings by the **FM3**. Furthermore, we observed that the **FM3** drawings have a small average edge length. The average edge length of these drawings is close to 50 and all vertices are placed on integer coordinates. We suggest that small average edge lengths induce a higher probability of precision errors and thereby computationally more complicated instances.

The **FM3data** can be subdivided into three graph classes. It contains 50 caterpillars. Recall that a caterpillar is defined as a path of vertices and for each vertex at the path there might be some degree one vertices connected to it. The degree one vertices in the caterpillar are called legs or leafs. Caterpillars with n vertices have $n-1$ edges. The caterpillars have 250, 300, 350, 400, and 450 vertices, where for each number of vertices there are 10 caterpillars in the set.

The second subset of **FM3data** is a set of planar graphs. Again the number of vertices is one of 250, 300, 350, 400 and 450 where for each number of vertices there exist 10 graphs for the density 1.5 and the density of 2 each.

The largest and most diverse subset are the general graphs. The graphs are simple, connected and the edges are chosen by a uniform random distribution. There are no further restrictions to the graphs. The number of vertices is again from 250 to 400 whereas the densities vary from 1.5 to 2.5.

3.6.2 *Ply-number* for different layouts

We compare the different layouts provided by our tool, namely organic, circular, and random on the **ROMEdata**. Plotting the *ply-number* of the drawing against the number of vertices shows a large variance throughout graphs with the same number of vertices in the organic and circular layout. We can observe a strong linear correlation between the number of vertices and the *ply-number* in the random placement of the vertices, cf. Figure 3.32.

The same holds true for the relation between the density of the graphs and the *ply-number* of their drawings. Drawings of graphs with the organic layout have the smallest *ply-numbers* overall i.e. less than 13, whereas the drawings with the circular layout perform a little weaker i.e. *ply-numbers* less than 25. In both layouts the *ply-number* and the number of vertices or the density do not suggest any strong correlation.

Comparing the *ply-number* regarding the density for the organic or circular layout has no significance on this set of graphs, since the variance is high. The random placement shows a large variance independent of the density of the graphs.

On tree-like graphs, meaning graphs with a density close to 1, the circular and the organic layouts perform very similar, whereas the organic layout produces drawings with lower *ply-number* on average on graphs with density between 1.2 and 1.8.

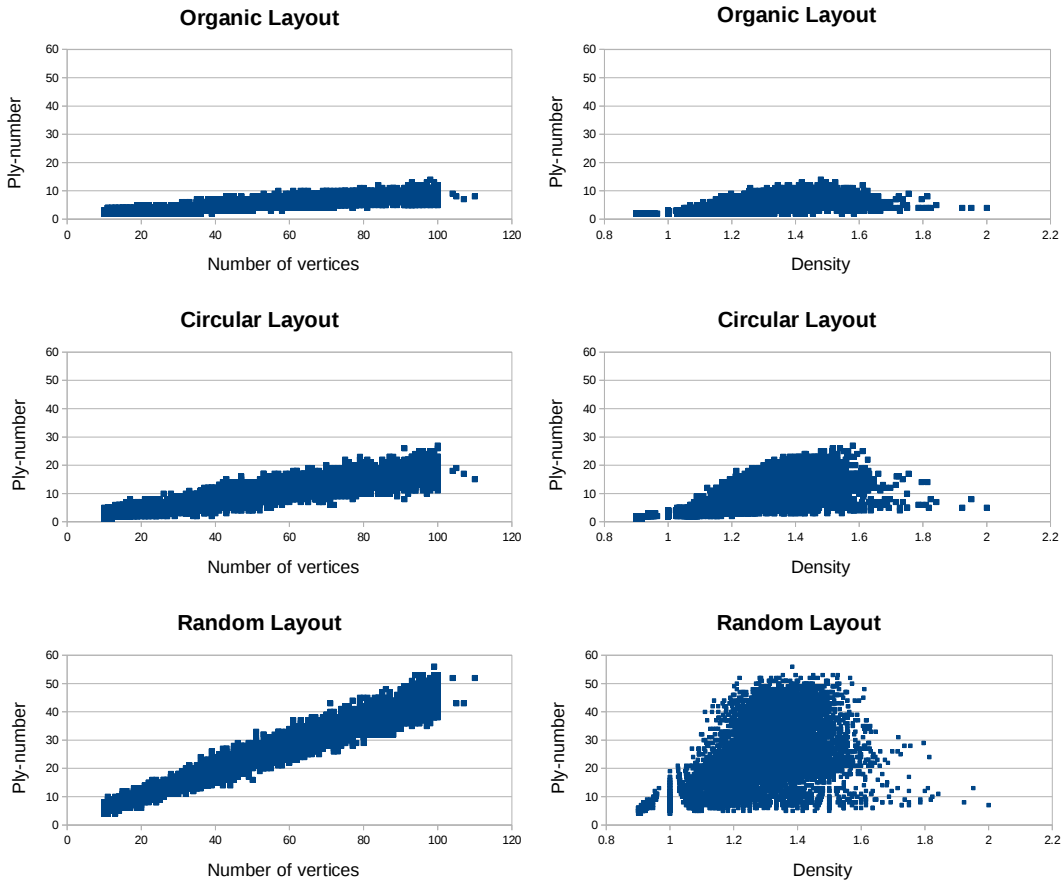


Figure 3.32: The graphs of **ROMEdata** are drawn with different layouts, namely organic, circular, and random. For each layout, the *ply-number* of the drawing is plotted against the number of vertices (left) and the density of the graph (right). We can observe a strong correlation between the *ply-number* of the drawing and the number of vertices in the random layout, and a weak correlation in the circular style.

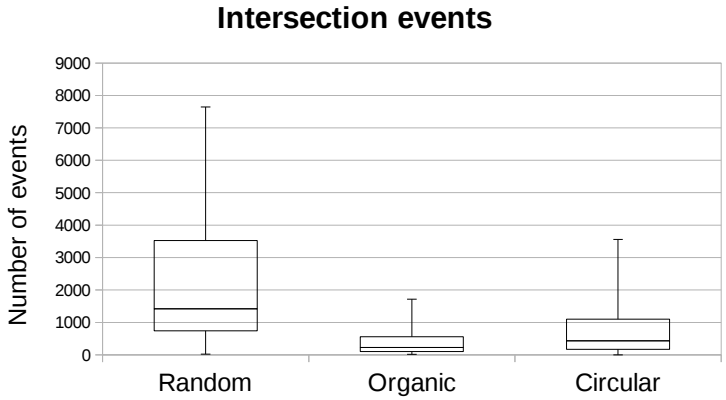


Figure 3.33: The number of events during the computation of the *ply-number* is shown for the different layouts. The variance in the drawings by the organic layout is low, whereas the variance in the drawings by the random layout is high. Clearly the organic layout has the lowest average number of events during the plane-sweep algorithm. The exact medians of intersection events are 212 for the organic, 432 for the circular, and 1419 for the random layout.

In this paragraph we evaluate the *Rome graphs* with focus on the number of events and the computation time. Note that the total number of events can be determined as the number of start events plus the number of end events plus the number of intersections. The number of start and end events is equal to the number of vertices in the graph and thereby the same for each layout algorithm. The differences are determined only by the number of intersection events. We present the variance in the number of intersection events for the different layout styles in Figure 3.33.

The computation time highly depends on the number of intersection events. This number highly differs throughout the different layouts. We indicate the range of intersection events, as well as the median and the upper and lower quartiles in the figure. The error bars mark the maximum and the minimum number of events. We can observe that the variance in the organic layout is small, in comparison to the random layout, which has a large variance. The circular layout has a higher number of intersection events on average in comparison to the organic layout.

On **ROMEdata** the time spent computing the *ply-number* for each drawing ranges from < 1 to 27 milliseconds, where a drawing in the organic layout takes 0.68 milliseconds on average, a drawing in the circular layout 1.2 milliseconds and a drawing in the random layout on average 3.3 milliseconds.

In this paragraph we evaluate the randomly generated graphs with focus on the number of events and the computation time. In the random data we have

a larger variety in densities. We compute the *ply-number* on the **RANDdata**. We present the average numbers for different densities in Table 3.1. The table is sorted by different densities and the average *ply-numbers* for the different layouts are reported. Additionally, we state computation time and the total number of events.

In the table we can see that the computation time is proportional to the number of events. Whereas we observe a relation of roughly 1000 events need 1.5 milliseconds to compute. The number of events for all layouts increases by the increasing densities. Overall the random layout has the highest number of events on average and the organic layout has an equal number of events in high densities. Throughout the different densities, the random layout has an almost constant number of events and computation time. At densities larger than 8 the circular layout meets the theoretical upper bound of the *ply-number* of $\frac{|V|}{2}$ and in these graphs the number of postponed events is noteworthy. While it can be neglected in the organic and random layout, the number of postponed events in the circular layout is consequential and highly affects the computation time. In the circular layout the highly symmetric placement of the vertices causes many events to share an x-coordinate. The number of postponed events, sequentially counts all events which cannot be processed instantly and thereby tends to highly exceed the number of total events. Additionally, we report the maximal number of consecutively postponed events indicating the maximal distance to the first solvable event. The average number of the linear distance is presented as the value in brackets.

We can observe that the spring-embedding algorithms produce drawings with low *ply-number* on sparse graphs, whereas on graphs with a high density the computed *ply-numbers* are very similar to just randomly placing the vertices. On the dense graphs the circular layout meets the theoretical upper bound of $\frac{|V|}{2}$. Even though the number of vertices is constant, the *ply-number* increases with the density in the random layout.

To confirm the accuracy of our ply-computation algorithm, we compare our results to the previous implementation of De Luca et al. [30]. All computations were done on the initial layout, since the previous computation does not support the change of layouts. In all tested graphs the *ply-number* was equal, whereas we can observe a large difference in the number of events and the computation time. To compare the computation time, the average of the complete set of **RANDdata** in our implementation is 8.8 milliseconds. This is a significant improvement in comparison to the average computation time of the implementation [30] which was around 27 seconds. The number of events on average differs by the factor of ~ 3, namely on average over **RANDdata** by the factor of 2.8.

Table 3.1: The table presents the results of the ply-computation on **RANDdata**. For each density we tested 5 graphs where every graph consists of exactly 100 vertices. We can observe a strong correlation between the number of events and the computation time. The number of postponed events increases with the density in the circular layout.

Density	Layout	Ply	Time in ms	Events total	postponed (max)
1	Organic	4.8	1.2	475.2	0 (0)
	Circular	3.8	1	408	0 (0)
	Random	40.6	8.8	5893.8	0 (0)
1.5	Organic	11.6	2.4	1404	0 (0)
	Circular	22.4	5.6	2736.8	0 (0)
	Random	45.4	11	6598	0 (0)
2.5	Organic	21	5.2	3321.2	0 (0)
	Circular	36.4	11	6070	0 (0)
	Random	56.4	17	8122.8	0 (0)
5	Organic	41.8	10.8	6584.8	0 (0)
	Circular	46	10	8722.8	16.4 (7)
	Random	71	16	9366.8	0 (0)
6.5	Organic	48	10.6	7556.4	0 (0)
	Circular	47.8	13	9149.2	4.8 (4)
	Random	78.8	15	9603.6	0 (0)
8	Organic	60.8	10.8	8654.4	0 (0)
	Circular	48.8	12	9357.6	27.2 (14)
	Random	80.4	13	9721.6	0 (0)
10	Organic	69.4	14.6	9149.2	0 (0)
	Circular	49.2	14	9547.6	66.2 (21)
	Random	82.2	14	9818.4	0 (0)
12	Organic	76.8	11	9607.2	0 (0)
	Circular	50	12	9685.6	118.8 (30)
	Random	85	15	9856.4	0 (0)
15	Organic	84.4	13.6	9767.2	0 (0)
	Circular	50	16	9772.4	539.6 (68) ←
	Random	87.8	16	9936	0 (0)
20	Organic	91	16.2	9895.2	0 (0)
	Circular	50	13	9860	761.4 (79) ←
	Random	92.2	15	10004	0 (0)
40	Organic	95.8	15.2	10055.2	0 (0)
	Circular	50	31	9991.2	16349 (438) ←
	Random	96.8	14	10085	0 (0)

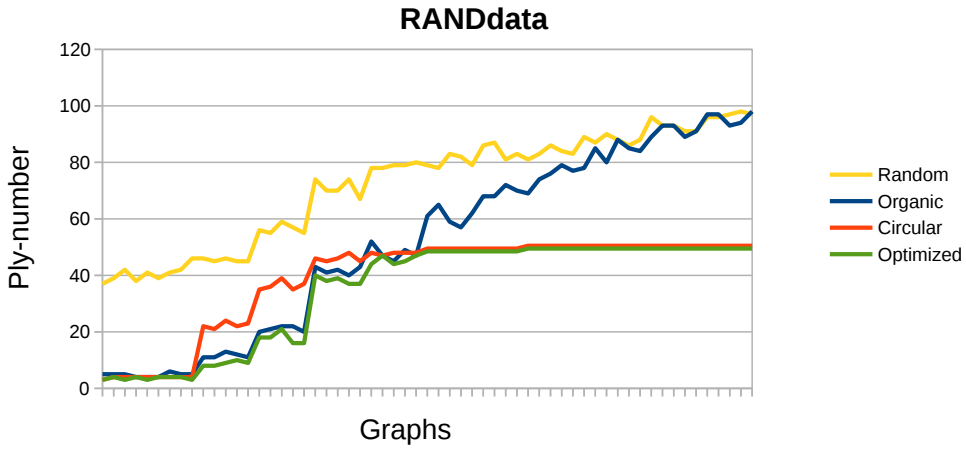


Figure 3.34: The graphs of **RANDdata** ordered by their density are plotted against the *ply-number* of the drawing. We observe that the organic layout produces low *ply-numbers* for low densities, whereas at higher densities the circular layout outperforms the organic layout. In very dense graphs the organic layout performs evenly as the random layout.

3.6.3 Comparison on the FM3 drawing dataset

The **FM3data** were kindly provided by the authors of [30] and were found to be drawings with low *ply-number*. We take this set as benchmark to compare the implementations. We will present results in form of the number of events for the two implementations and state the average computation time. In the two implementations the *ply-numbers* are exactly the same, which is an improvement to our publication [56].

To compare the algorithms, the *Apfloat* decimal precision was set to 20 digits. This value was used in the experiments of [30]. To clarify the behaviour, we present the data split by the type of graphs and according to their density.

Caterpillars There are 50 caterpillars in **FM3data**, ranging in the number of vertices from 250 to 450. As a reminder, caterpillars are graphs, which consist of a path called the spine and degree one vertices which are called legs connected to vertices along the path. An example is presented in Figure 3.35. The drawings of the caterpillars are produced by using the **FM3** method. We present the results of the computation in Table 3.2 where we compare our implementation to the implementation by [30]. First of all, we want to point out the difference in computation time and the total number of events. Arguing that the number of events should be constant for a given drawing, we can explain the difference in number of events by the different approaches to handle inconsistencies. The previous algorithm [30] introduces a number of redundant events to detect and handle inconsistencies. As a summary, our implementation clearly reduces the computation for the *ply-number* from seconds to milliseconds.

To compare the **FM3** layout method to the previous methods we present the average *ply-number* and the average computation time in Table 3.5 on the optimization section. We can observe that in general the **FM3** layout produces low *ply-numbers* on this set of graphs, similar to the organic and the circular layout. The three layout methods are also comparable in computation time.

Planar graphs In **FM3data** are 50 planar graphs ranging again from 250 to 450 vertices and the densities between 1.5 and 2. Presenting the results in dependency on the density, we can again observe the factor ~ 3 difference in the number of events and the reduction of the computation time from seconds to milliseconds. We split the density at 1.7 and the results are shown in Table 3.3 . The average *ply-number* indicates a correlation to the density rather than the number of vertices of the graph. To show that the **FM3** and the organic layout perform equally well on this set, we present Figure 3.36. The circular layout performs slightly weaker on these graphs whereas the performance was similar on the set of caterpillars.

For low densities the circular layout produces higher ply drawings, where the organic layout or the **FM3** method produces similar *ply-numbers*. On average the

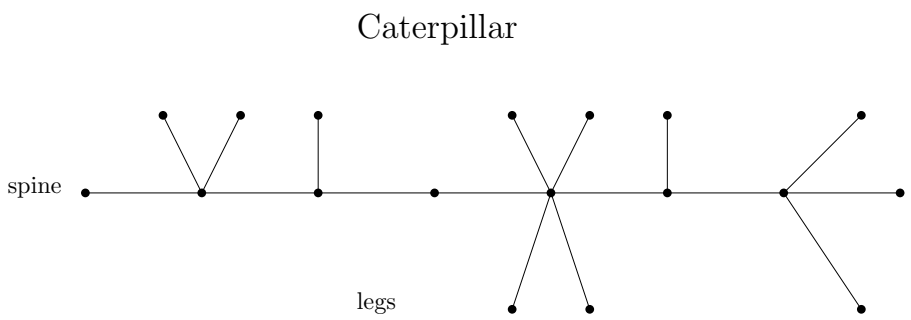


Figure 3.35: A caterpillar consists of a path called the spine and degree one vertices attached to the vertices along the spine which are called legs. This graph consists of 18 vertices.

Table 3.2: The caterpillars of **FM3data**. Each subset with 250 to 450 vertices contains 10 graphs. During all experiments the *ply-numbers* for both implementations were the same. The table presents average values for each set of graphs.

		[30]		**Our Tool**	
Vertices	Ply	Events	Time(ms)	Events	Time(ms)
250	3.8	2692.4	1328.1	1122.6	3.5
300	4.3	3510.4	1831.9	1430	1.6
350	4.5	3827	1883.7	1602.4	1.9
400	4.6	4564.9	2291.5	1879.3	2.4
450	4.3	5032.8	2581	2110	2.3

drawings generated by **FM3** have slightly higher ply than the organic layout. The average *ply-number* in the organic layout is 9 for graphs with density ≤ 1.7 and 9.7 for higher density. Again, note the difference in number of events and computation time, keeping the accuracy of the computation on the same level.

General graphs The remaining subset of **FM3data** consists of graphs without further requirements. They have 250 to 450 vertices and the densities vary from 1.5 to 2.5. During the computation of the *ply-number* on this set, we encountered one instance where the *ply-number* of the two implementations differed by one. Our explanation for this circumstance is our preference to handle end-events first. In such a case, our algorithm tends to underestimate the *ply-number*, since an intersection event at the same coordinate might increase the *ply-number* by one, whereas the end-event reduces the *ply-number*. To conclude this paragraph, we present the interesting result on different layouts on the third subset of **FM3data** presented in Figure 3.37. Note that again the *ply-numbers* on **FM3** and organic layout are very similar. Observe that the stairs in the plot indicate the jump between the densities from 1.5 to 2.5 for each set of graphs.

Table 3.3: The planar graphs of **FM3data**. The values show the average results for each subset. Both implementations always computed the same *ply-numbers*.

		[30]		Our Tool	
Density	Ply	Events	Time(ms)	Events	Time(ms)
≤ 1.7	9.6	9878.6	8444.2	3434.6	3.7
> 1.7	11.4	9625.9	9625.9	3609.4	3.8

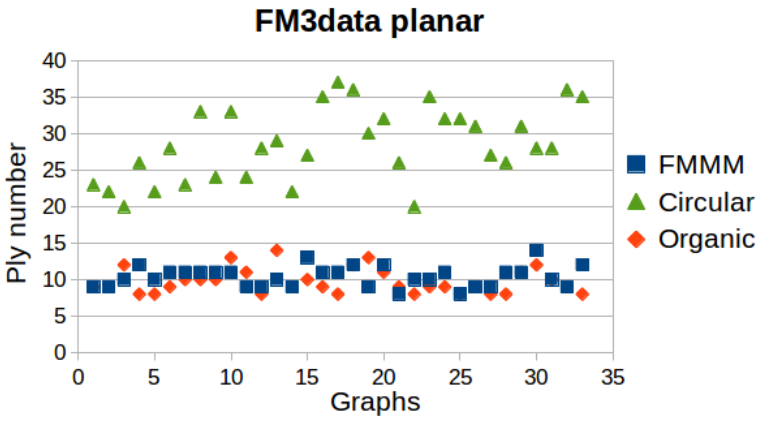


Figure 3.36: For each planar graph of **FM3data** the computed *ply-numbers* for each of the three layouts is plotted. Note that the organic and the FM3 drawings have similar *ply-numbers*, while the circular layout produces higher *ply-numbers* on low density graphs.

Table 3.4: The average results of the general graphs of **FM3data** are presented. Note that the *ply-numbers* in brackets indicate a different result of the algorithms. These cases have a high number of postponed events which points to a difficult instance to compute.

			[30]		Our Tool	
Density	Vertices	Ply	Events	Time(ms)	Events	Time(ms)
1.5	250	18	18334	23955	6430	7.4
	300	19.8	25688.3	38454.8	8950.8	13.2
	350	23.7	34140.5	52829.1	11949.7	15.3
	400	25.4	43543.4	72928.5	15227	19.2
	450	28 (27.9)	55643.2	100653.2	19395.6	31.6
2.5	250	38.1	47248	92192.7	16539.1	21.5
	300	45.4	68070.5	147113.6	23892.3	33.2
	350	51.4	90943.4	217999.9	31850.1	42.6
	400	59.3	118188.7	309601.8	40606.9	53.1
	450	64.3	148973.3	426993.5	51640.4	69.4

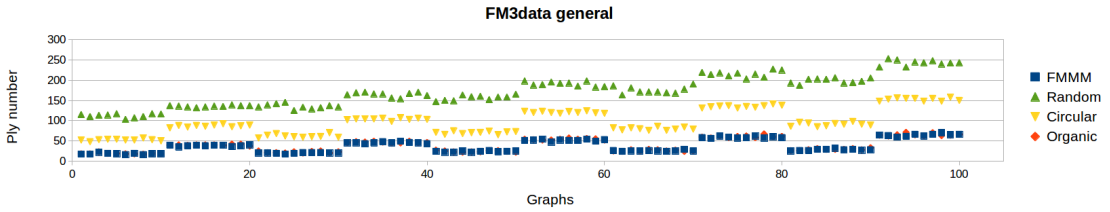


Figure 3.37: The set of general graphs can be subdivided in 5 subsets consisting of graphs with 200, 250, ... , 450 vertices. Each subset can be divided in 10 graphs with density 1.5 and 10 graphs with density 2.5. The figure indicates that the **FM3** algorithm and the organic layout produce similar drawings regarding the *ply-number*.

3.6.4 Ply Minimization

We observed that the **FM3** and the organic layout produce drawings with low *ply-number* on sparse graphs, and the circular layout produces low ply drawings on dense graphs. An interesting question to ask is whether we can identify drawings which have a lower *ply-number* for a given graph in comparison to the previous methods. In the following we present our strategies to create drawings with low *ply-numbers* whereas we can give no guarantees if that is the minimal *ply-number* over all drawings for a graph. Afterwards we present the results on the evaluation of our strategies on **FM3data** and **ROMEdata**.

Strategies

Since we have a guaranteed upper bound of *ply-number* $\frac{|V|}{2}$ for any graph, we use this upper bound and apply the circular layout, whenever we cannot identify drawings whose *ply-numbers* are smaller than $\frac{|V|}{2}$.

For the interactivity in our tool, we present a workflow to achieve a drawing with low *ply-number* which is directly accessible. We start with the current drawing of the graph, preferably with the organic layout, since it has presented itself to produce drawings with low *ply-number* on sparse graphs. We observe that the *ply-number* represents equal edge-lengths distributions and equal vertex distributions in the plane. Following these observations, we tuned a new spring embedder based on Fruchterman and Reingold [47], similar as it was suggested in [30]. We tuned the parameters to produce drawings with less ply by strengthening the repulsive forces between unconnected vertices. We can show that on sparse graphs our tuned spring embedding algorithm outputs drawings with lower *ply-number* on average than the organic layout of the **FM3** method on sparse graphs. On dense graphs we have the guaranteed upper bound. Within the tool we can iterate several steps and try to reduce the *ply-number* of the drawing.

Having a closer look to these drawings, we can observe that in many cases there exist only a few regions which contribute to the maximum *ply-number* of the drawing. The *ply-number* can often be reduced by moving some vertices locally. We could identify some rules for degree one and two vertices, but beyond these we could not identify a deterministic way to move the vertices, since local changes might affect other ply-regions far away.

Results

On the **ROMEdata** we can present the advantages of our methods in comparison to the organic layout. On average we could reduce the *ply-number* from 6.3 to 5.1 in the modified setting. The achieved *ply-numbers* for these graphs are presented in Figure 3.38. The spring embedder to create drawings with low *ply-number* is

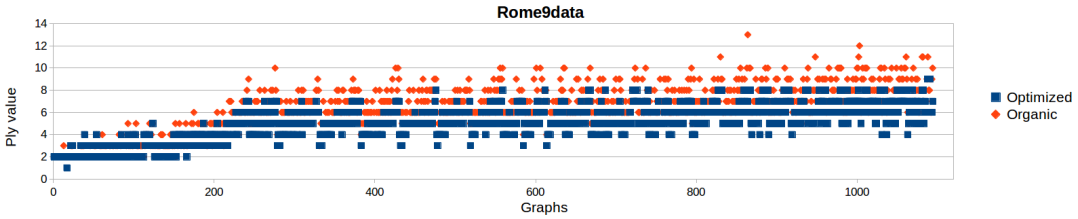


Figure 3.38: For each graph in the **ROMEdata** set the organic and the improved *ply-number* are illustrated. The graphs are ordered by the number of vertices.

Table 3.5: The average values of the ply-computation for the different layouts on the subset of caterpillars on **FM3data** are shown. We can observe that on these graphs the three layout methods produce very similar *ply-numbers* whereas we can reduce the *ply-number* by one on average if we try to minimize the *ply-number*. We cannot state the computation time for the optimization approach, since our approach computes several layouts for every graph.

Layout	Average *ply-number*	Average time in ms
FM3	4.3	3.14
Organic	3.92	4
Circular	4.24	3.68
Random	120	101
Optimized	3.24	

executed several times with 500 iterations each. Thereby, we do not state the computation time.

On **RANDdata** the results can be seen in Figure 3.34. For dense graphs, we perform equally to the circular layout, whereas between the densities of 1.5 and 5 we can improve in comparison to the other layouts.

In the experimental study of the *ply-number* [30], one of the results was the strength of the **FM3** algorithm to produce low ply drawings. To test our ply minimization approach, we evaluate our algorithm on **FM3data** and compare the *ply-number* to the results of our workflow.

The set of caterpillars, which had an average *ply-number* of 4.3 with the **FM3** drawn graphs, we improved with our approach to an average *ply-number* of 3.3 as presented in Table 3.5.

On the set of planar graphs, we could improve the *ply-number* on average from 10.4 to 8.8 and on the set of general graphs we could improve from 37.3 to 36.7. Note that overall we could improve the *ply-number* by 1, which on sparse graphs corresponds to an improvement of about 20 %. The results of our experiments are presented in Figure 3.39.

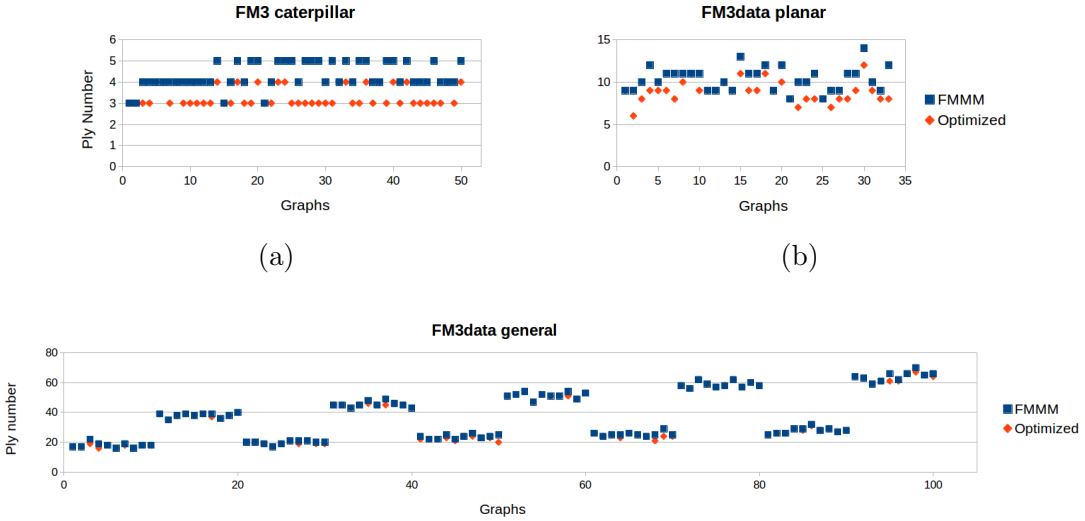


Figure 3.39: The plots present the minimization results on **FM3data**. Note that the axis change in scale throughout the plots. (a) the *ply-number* of the caterpillars. (b) the *ply-number* of the planar graphs. (c) the *ply-number* of the general graphs.

3.7 Discussion & Conclusion

In this section we will discuss our tool in terms of usability, computation of the *ply-number* in different layouts, improvement and comparison to previous implementations as well as the effectiveness of our ply-minimization approaches. As a last paragraph we conclude the advantages of our tool.

We have developed an intuitive tool to examine the *ply-number* for a given drawing. It is possible to load and store drawings of graphs as well as just loading structural information of a graph. The user can easily get an understanding of the *ply-number* as a parameter to evaluate the aesthetics of a drawing.

Our ply-computation allows to give instant feedback to the user during modification of the drawing. This supports the understanding on the influence of minor local changes might effect other ply-regions. A local change can influence the *ply-disk* of any adjacent vertex if it induces a new longest edge and thereby a different *ply-disk* radius for another vertex. By providing information on the regions where the maximum *ply-number* occurs, we furthermore enable the user to identify meaningful modifications.

In our experimental section we have presented results on the *ply-number* computation for a large variety of graphs and drawings. We examined different layout methods and can conclude that placing the vertices at random positions generates an almost worst case scenario. Recall that the *ply-number* can be at most $|V| - 1$.

Analysing the spring-embedder layouts, we can conclude that their effectiveness to produce low *ply-number* drawings shines on graphs with low density. This circumstance confirms the results of the experimental study on the *ply-number* by Felice de Luca et al. [30]. When it comes to denser graphs, we could observe the equilibrium between the spring-embedder algorithm and the circular layout to be at the density between 5 and 6.5. At this point both layout methods produce drawings with equal *ply-numbers*.

Figure 3.34 indicates, that for densities larger than 6.5 the circular drawings clearly have lower *ply-numbers* than the drawings by the organic layout. For very dense graphs, the spring embedding approaches do not have a significant advantage in comparison to just randomly placing vertices. In very dense graphs the circular layout meets the upper bound of $\frac{|V|}{2}$. This is by now the best we can do.

We want to point out the correlation between the number of events and the *ply-number* for graphs with the same amount of vertices. Since the number of vertices is constant, the difference in the number of events is purely influenced by intersection events. We can indicate an implication between the number of intersection events and the *ply-number*. Additionally, the number of events highly correlates with the density of the graphs, recall Table 3.4. Accordingly, the number of events in dense graphs support the observation that the organic layout produces similar *ply-numbers* as the random layout.

The effect of denser graphs having a larger amount of intersection events can be explained by the fact that more edges tend to induce larger radii of the *ply-disks* and thereby more intersections, even though the number of disks is constant. The increasing number of events for denser graphs can be observed in both evaluated implementations.

To compute the *ply-number* of a given drawing we use the plane-sweep technique used in the provious implementations [7, 30]. We introduced a different event handling as prioritizing end over intersection over start events. Another major difference was the handling of prevision errors and using the primitive data type `double`. To deal with precision error we look ahead in the priority queue for a event that can be handled consistently rather than recomputing the coordinates of the involved intersections.

We observed precision errors especially in very dense graphs in the **RANDdata** with the circular layout. In this setting we had a large amount of **postponed events**, meaning events that could not be executed instantly. Recall that the circular layout is highly symmetric and thereby causes many events to share an x-coordinate. Furthermore, we cannot guarantee that due to the structure we found every intersection. It might occur that, due to rounding inaccuracies, an intersection is not identified correctly, whereas for the majority of graphs the number of postponed events can be neglected. In the circular layout the radii of the *ply-disks* are likely to be irrational numbers and are thus prone for precision errors. Note that we simply count all events in this parameter and thus the number tends to

be large. An event might need several prior intersections events to be resolved and might be counted several times. After identification of a resolvable event, we simply reinsert the postponed events into the priority queue and try again. This number is purely an indicator whether precision errors occurred during the computation.

We can conclude that low *ply-numbers* admit a visually pleasing way to draw a graph. This confirms the results of De Luca et. al. [30]. Note that the *ply-number* does not relate to the crossing number of a drawing. For some planar graphs it can be beneficial to draw crossings. This relates to the paradigm of force-directed layouts [47, 62].

We compare our algorithm to the previous implementation by de Luca et al. [30] and we want to point out mainly three important observations. Clearly, we could reduce the computation time from seconds to milliseconds as presented in Tables 3.3, 3.4 and 3.5. We can identify several reasons for this improvement. First of all, the number of total events differs by the factor of ~ 3. The implementation by [30] introduces additional events to ensure the consistency during the computation. In contrast to our solution, they can detect and resolve inconsistencies immediately. Investigating purely the number of events might be a reason for a small time difference but clearly does not explain the huge difference in computation time.

We suggest that the main improvement in computation time is based on the fact that our implementation uses the primitive data type `double` in contrast to the data type `ApFloat`. The data type `ApFloat` allows to compute values at an arbitrarily chosen arithmetic precision at the cost of computational speed. Recall that we identified one drawing where the *ply-number* differed by one in both computations. Even though we investigated this drawing carefully, we could not identify the true *ply-number* of this instance. We suggest that prioritizing end-events in our implementation might be the reason that the *ply-number* differs by one. We did not observe any postponed events in this instance which would indicate precision errors during the computation whereby using `ApFloat` would allow a higher precision in the computation.

In contrast to the previous implementation, our focus was the examination of drawings regarding their *ply-number*. Therefore we needed a fast algorithm to provide feedback to the user instantaneously. The previous implementation was designed for computation of the *ply-number* for given drawings and the computation time component was not at their focus and thereby their algorithm was not applicable for our purposes.

To evaluate the accuracy of the algorithms, we had a look to the number of postponed events. The idea was that a large number of postponed events indicates the occurrence of precision errors and thereby our algorithm is likely to report an underestimation of the *ply-number*. We suggest to state the number of postponed events as measurement of likelihood of the correctness of the result. As it turned out, we cannot apply this reasoning to the drawing in which the results differ between the implementations.

In our publication at GD 2017 [56] we detected a few computations with a high number of **postponed events** during the analysis of the **FM3data**, which did correspond to differences in the computation of the *ply-number*. We were able to improve our results in comparison to our publication by identification of a minor bug and were able to resolve all inconsistencies between the algorithms except one. The **FM3** algorithm tends to have small average edge lengths throughout the drawings and places the vertices on integer coordinates. Both circumstances are likely to induce irrational radii for the *ply-disks* and thereby are likely to induce precision errors. We conjecture that on **FM3data** the accuracy of the computation might be increased by scaling a given drawing. Unfortunately, we cannot support this hypothesis by experimental data.

The layout algorithms provided in our tool tend to have larger average edge lengths. As we conjecture, this increases the accuracy of our algorithm and might be an explanation for the computation error on the graph. All in all we present an algorithm which can compete in accuracy of the computed result and is very fast.

In this paragraph we will discuss our approach to identify drawings with lower *ply-number*. We have evaluated earlier that in case of sparse graphs spring embedding algorithms create drawings with low *ply-numbers*. At densities larger than 6.5 the circular layout, and thereby the upper bound of $\frac{|V|}{2}$, seems to achieve a good value regarding the *ply-number*. Comparing the organic layout and our ply-minization approach on **ROMEdata**, we could reduce the *ply-number* on average from 4.3 to 3.3. Overall, **ROMEdata** consists of sparse graphs. By tuning a spring embedding algorithm we could use the observation that the maximum *ply-number* of a drawing occurred in a few regions and thereby a decrease in the *ply-number* could often be achieved by moving a few vertices locally.

According to the experimental study [30], the **FM3** layout produces drawings with low *ply-number*. We could confirm that result and were even able to identify drawings with lower *ply-number* with our spring-embedder. For the **FM3data** we could reduce again the *ply-number* by one over all graphs on average, as we presented in Figure 3.39.

On the subset of caterpillars and the planar graphs, we could achieve a better improvement in contrast to the general graphs. For the caterpillars the computed *ply-number* still ranges up to 4 and we could improve the average *ply-number* including to be around 3.24 whereas we know that caterpillars can be drawn with *ply-number* ≤ 2 [3]. Further examination on these graphs suggests that our methods are often able to construct drawings with *ply-number* 2 given a suitable start configuration and enough time. Since we gave a strict time limit during the experiments, we did not manage to produce many ply 2 drawings on this set.

On the set of general graphs, namely **FM3data**, we were not able to improve the *ply-number* on average whereas we identified equally good results. The average values were 37.33 for the **FM3** layout and 37.78 for our approach. In contrast to

the other two subsets, this difference is minor. This indicates that on these type of graphs spring-embedding algorithms tend to reach their limits in terms of creating drawings with low *ply-number* due to the graph's densities.

In very dense graphs, purely using the spring embedding algorithm faces the same issue tending to similar results as the random layout. So for dense graphs we can at least guarantee the upper bound by using the circular layout, which is included in our optimization. This can be observed in Figure 3.34, where for larger densities the optimization and the circular layout have the same values.

As a last feature our tool provides the user with our adjusted spring embedder and the possibility to enforce equal edge lengths. Enforcing equal edge length ℓ and the minimal distance between any two vertices to be ℓ as well can be interpreted as a test, whether the graph can be drawn with ply 1 iff this enforcement converges to a stable drawing. Stability in this sense means that the vertices stop moving after some iterations. In case of a non-stable outcome, the graph is most likely not ply 1 drawable. Remember, during our experiments, due to precision errors, we did not observe ply 1 drawings by automated layout methods. We observed a strong convergence by including strong forces in this equal edge length approach.

Overall, our optimization process involves several iterative computational steps using spring embedding algorithms and computation of the *ply-number* in between. By using these methods and adjusting the vertices manually, it is possible for the user to reduce the *ply-number* even further by moving few vertices, since due to a previous observation there often exist only few regions with maximal ply. Our indication of the region with maximum *ply-number* for a given drawing turns out to be very useful for this matter.

As a short conclusion on this chapter we want to point out the following. We introduced a fast ply-computation algorithm which is able to give instant feedback to user interaction, e.g. whenever the drawing of a graph is modified. We were successfully able to reduce the computation time from seconds to milliseconds. Our tool is equipped with basic layout algorithms and automated minimization techniques that are intuitively to use.

The tool can be used to get a deeper understanding of several graph classes e.g. according to the question if there exists a lower bound on the *ply-number*. Since we discovered some drawings where the investigated implementations reported different *ply-numbers*, it might be necessary to include further examination and identification of the exact reasons.

Since we are somehow neglecting or ignoring precision errors in some cases, the implementation providing higher precision in the computation might be used as verification. Furthermore, we see some potential to improve the minimization methods. Further evaluation and experiments will be necessary to observe the influence of scaling to our computations and whether scaling can actually improve the accuracy of the computations.

Chapter 4

Bar Visibility Beyond Planarity

In the wide field of graph drawing, graph theory and combinatorics there exist several well-studied representation models for graphs. These models describe specific properties of the drawing. Formally, a representation in graph drawing defines rules to draw a graph and thereby enforces the graphs, which can be drawn under a specific rule set, to require certain properties.

A well-known example of graphs are planar graphs. Planar graphs can be drawn without any crossing edges and have at most $3n - 6$ edges in total where n is the number of vertices. Furthermore, they cannot contain minors of the complete graph K_5 or the complete bipartite graph $K_{3,3}$. Such drawings can be constructed in linear time on a grid of quadratic size [29, 79].

In *bar-visibility* drawings vertices of a graph are represented as pairwise disjoint horizontal line segments in the plane called bars. We say that two segments u and v are visible to each other if there exists a vertical rectangle of non-zero width such that the opposing sides of this rectangle are subsets of u and v and the rectangle does not intersect any other bar. In other words we can connect the bars u and v by a vertical edge segment without intersecting or touching any other bar due to the non-zero width condition.

This representation was introduced in [50] for VLSI layout testing. VLSI layouts have a huge importance in chip design. In chip design it is preferred to realize connections via straight segments and it is beneficial to avoid crossings to map the components to a grid [43, 70, 87]. The components itself will be realized in a rectangular shape and is realized by resizing in one dimension.

Defining an edge segment between two bars if and only if they are visible to each other we obtain a *strong bar-visibility* representation. This model has been studied in [98] showing that a planar graph has a strong *bar-visibility* representation if and only if it has an s-t ordering. Any planar graph has a weak *bar-visibility* representation [98] which relaxes the condition that an edge between any two bars might exist if the bars are visible to each other. Formally, in the weak model the edges of the

Figure 4.1: (a) is a planar drawing of a graph and its *bar-visibility* representation is presented in (b). A vertex is represented as a horizontal bar and edges are visualized as vertical lines of sight indicated by dots.

graph are a subset of the visibility relations whereas in the strong model the edges of the graph are equal to the visibility relations of the bars. Visibility representations of planar graphs in $O(n^2)$ area can be constructed in linear time [77, 87].

In graph drawing graphs beyond planarity have gained increasing interest in recent years. Partially motivated by applications of network visualisation, it has become crucial to compute readable drawings of non-planar graphs for humans. Cognitive experiments by Huang et al [58] indicate that a high angular resolution of crossing edges does not affect human understanding of graph drawings.

Recall that, planar graphs can be drawn in the plane without any crossing edges. Extending the graphs by the allowance of crossing edges we encounter the graphs beyond planarity. Important approaches on graphs beyond planarity are k-planarity [71], bar k-visibility [31, 55], k-quasi planarity [1] and right-angle-crossings (RAC)[37].

A graph is called k-planar if it can be drawn in the plane such that any edge is crossed at most k times. Furthermore, a graph is called k-quasi planar if no k edges pairwise intersect. It is known that any k-planar graph is a $(k+1)$-quasi planar graph [2]. In RAC drawings we consider straight-line edges and the crossing angle of any two edges is always 90°. RAC graphs have at most $4n - 10$ edges [37] and the recognition problem for RAC graphs is known to be NP-hard [5].

More recently, bar k-visibility and especially bar 1-visibility have been studied i.e. by Brandenburg [15], Evans et al. [44] and Sultana et al. [83].

Bar k-visibility representation is the extension of *bar-visibility* beyond planarity. In bar k-visibility representations vertex segments are allowed to "see" through k other bars. Even though the edges in this representation formally do not cross, the routing of edges through bars correspond to crossing edges in other representations. Moreover, every 1-planar graph is bar 1-visible [15].

In this chapter we present and extend our results published in [16] where we

refine the model and introduce bar (k, j)-visibility representations of graphs. In this notation, k defines the maximal number of intersections per edge segment and j denotes the maximal number of intersections per bar. Using the refined definition, (weak) *bar-visibility* graphs are bar $(0, 0)$-visibility graphs, bar k-visibility graphs as defined by Evans et al. [44] correspond to bar (k, ∞)-visibility graphs and bar 1-visibility graphs as defined by Brandenburg [15] are bar $(1, 1)$-visibility graphs.

In the following we will present our results focusing on bar $(1, j)$-visibility graphs. Among others, we present upper bounds on the density for bar $(1, 2)$- and bar $(1, 3)$-visibility graphs. We continue by presenting a structure with very limited *bar-visibility* representation and use this structure to derive edge maximal graphs with $5n + O(1)$ edges where the general upper bound is $6n - 20$ [31]. Finally, we state our observations on bar $(k, 1)$-visibility graphs.

4.1 Previous Work

Bar-visibility graphs were initially introduced by Garey et al. [50] where edges are represented by an unobstructed line of sight. It has been shown that planar graphs admit *bar-visibility* representations. Allowing the edges to cross bars, extends this model. This extension has been recently studied and as the main previous results we will shortly summarize the results of Brandenburg [15], Evans et al. [44] and Sultana et al. [83].

In [83] the authors give an $O(n)$ time algorithm to construct *bar 1-visibility* drawings of some 1-planar graphs, in particular diagonal grid graphs, maximal outer 1-planar graphs, recursive quadrangle graphs and pseudo double wheel 1-planar graphs. Grid graphs are embedded graphs forming a grid with quadrangular faces. In the class of diagonal grid graphs the diagonals are drawn in each quadrangular face. These graphs are known to be maximal 1-planar graphs with $4n - 8$ edges. A maximal outer 1-planar graph is a graph where all vertices are incident to the outer face. By the additional maximality constraint it is not possible to add any edge without violating the outer 1-planarity properties. Recall that, each edge is crossed at most once in 1-planarity. Recursive quadrangle graphs follow a recursive construction pattern as the name suggests. We start with an quadrangular face and either add the diagonals or an internal cycle of length 4. The vertices of this cycle are then connected to the corners of the surrounding face such that there are 5 newly constructed quadrangular faces. These graphs are again maximal 1-planar graphs.

The introduced notation of bar k-visibility defines the number k of bars which can be crossed by a single edge. The number of crossings per bar is not limited in this definition. Recall that in our definition this corresponds to bar $(1, \infty)$-visibility representations. Dean et al. [31] present a tight upper bound of the number of edges in bar 1-visibility graphs to be $6n - 20$.

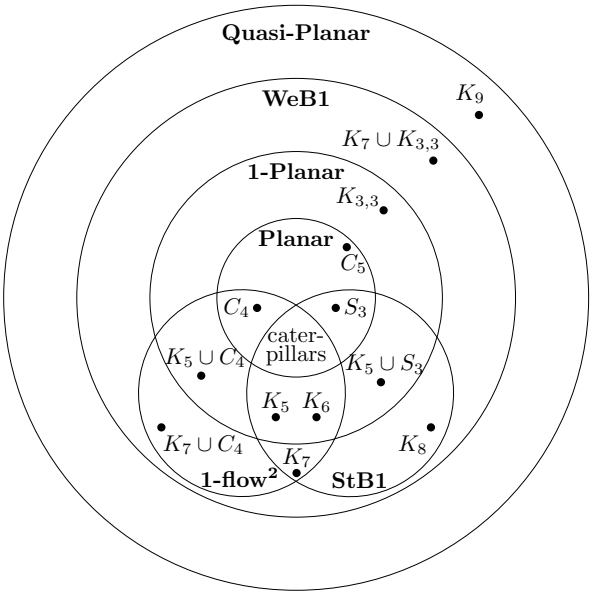


Figure 4.2: The relative relationship of strong (StB1) and weak (WeB1) bar 1-visibility graphs within the class of nearly planar graphs as presented in [44].

This work is extended by Evans et al. [44] where the authors focus on the case that k equals 1 and deduce relations to nearly planar graphs, meaning graphs which are 1-planar or quasi-planar. In general, 1-planar graphs have at most one crossing per edge and in quasi-planar graphs no three mutually different edges cross each other. The authors show that squares of planar 1-flow networks are weak bar 1-visibility graphs. In general, a k-flow network describes a directed planar graph, where for any vertex v $\min\{\text{indegree}(v), \text{outdegree}(v)\} \leq k$ holds [88]. Recall that a square of a graph can be obtained by adding edges between any two vertices with distance 2 in the original graph. Formally, let $G = (V, E)$, then $G^2 = (V, E \cup E')$ where any edge $(u, v) \in E'$ such that there exists a vertex $t \in V$ with $(u, t) \in E$ and $(t, v) \in E$.

Furthermore, any 1-planar graph admits a weak bar 1-visibility drawing (WeB1). Investigating strong bar 1-visibility (StB1) drawings, the authors deduce a relationship within other graph classes presented in Figure 4.2 with existential example graphs for each class.

Further restricting the model of *bar-visibility* representations, Brandenburg restricts the number of crossings per bar to be at most 1 [15]. This relates to bar $(1, 1)$-visibility drawings in our model. These graphs have at most $4n - 8$ edges which coincides with maximal 1-planar graphs. It is clearly a subclass of the WeB1 graphs, since the complete graph with 7 vertices missing one edge, namely $K_7 - e$, is edge maximal in this case since every internal bar has exactly one crossing as shown in Figure 4.3a. [15] presents a linear time algorithm to construct a bar

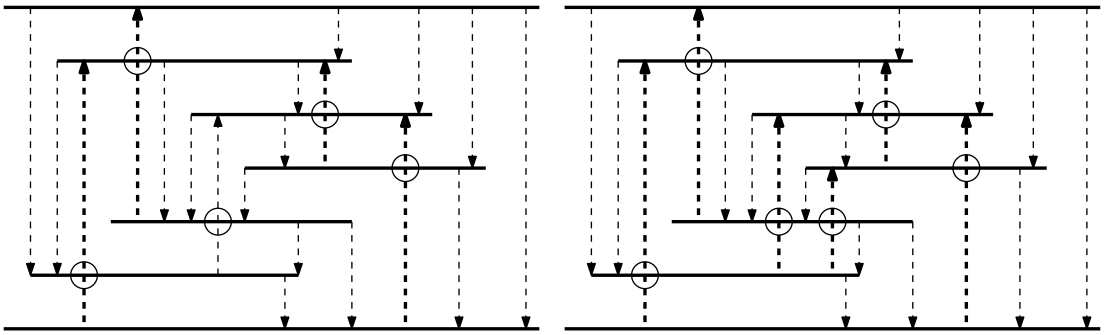


Figure 4.3: (a) is a bar $(1, 1)$-visibility drawing of an edge maximal graph, namely $K_7 - e$. In (b) the complete graph K_7 is drawn in a bar $(1, 2)$-visibility representation. Note that inserting the missing edge induces a crossing since there exist $3n - 6$ planar edges. For clarification and counting purposes, the planar edges are indicated by downward pointing arrows (non-planar upward) and the crossing points at the bars are marked by circles.

$(1, 1)$-visibility drawing given a 1-planar embedding in $O(n^2)$ area. Furthermore, they give an example for an edge maximal bar $(1, 1)$-visibility graph which is not 1-planar, namely $K_7 - e$. Recall that by definition in Brandenburg [15], the drawing in Figure 4.3a is a bar 1-visibility drawing and the drawing in Figure 4.3b is not. By the definition in Sultana et al. [83] both drawings are bar 1-visibility drawings.

4.2 Maximal bar $(1, j)$-visibility representations

A *maximal graph* G belongs to a class $\mathcal{G}$ such that adding any edge e would violate the properties of $\mathcal{G}$, namely $G \in \mathcal{G}$ and $G + e \notin \mathcal{G}$. In this section we present existential upper bounds on the number of edges for dense bar $(1, j)$-visibility representations for $j = 0, 1, 2, 3$ and $j \geq 4$.

Note that any bar $(1, 0)$-visibility representation where each bar is crossed at most once, is a bar $(0, 0)$-visibility representation since edges are not allowed to cross any bar and thereby no bar is crossed. These are exactly the planar graphs with $3n - 6$ edges [98]. Considering triangulated planar graphs, this bound is tight.

We observe that for any bar $(1, 1)$-visibility representation, the upper bound of edges can be determined as the set of planar edges, namely $3n - 6$ plus the number of bars, since every bar is crossed at most once. By the general structure of *bar-visibility* representations, the topmost and the bottommost vertex cannot be crossed at all. We can deduce the maximum number of edges in any bar $(1, 1)$-visibility drawing to be at most $3n - 6 + (n - 2) = 4n - 8$. This number coincides with the maximal number of edges in any 1-planar graph. Brandenburg gave an example of $K_7 - e$ as shown in Figure 4.3a, which is edge-maximal in bar $(1, 1)$-visibility but

not 1-planar. Furthermore, the graph $K_7 - e$ can be subsequently augmented by vertices of degree 4. Thereby, this bound is tight for any $n \geq 7$ [15].

Bar $(1, 2)$-visibility representations have a theoretical upper bound of $5n - 10$ edges since every internal bar can be crossed at most twice. We give a construction for graphs with $5n - 12$ edges which leaves a small gap.

Lemma 4.1. For every odd $n \geq 9$ there exists a graph G_n with a bar $(1, 2)$-visibility representation and $5n - 12$ edges.

Proof. Let the graph $G = (V, E)$ have $n = 2k + 1$ vertices where $n \geq 9$. We will first place the vertex segments denoted as $v_i = (l_i, r_i, y_i)$ where l_i is the x-coordinate of the leftmost point of the segment v_i, r_i the rightmost x-coordinate and y_i the y-coordinate respectively. In the following we will use indexing according to the y-coordinate i.e. $y_i = i$ for all vertices.

In the following we introduce the central structure G'_9 with $n = 9$ vertices and $5n - 17 = 28$ vertical edge segments. Afterwards we give an inductive construction to add two vertices and 10 edge segments. As a last step we show how to elongate the outermost vertex segments to increase the number of edges from $5n - 17$ to $5n - 12$.

G'_9 is constructed such that every bar is crossed exactly twice except for the central bar and the two outer bars. Moreover, every planar face is triangulated, except the outer face which has size 4. Thereby G'_9 has 20 planar edges and 8 crossing edges and is presented in Figure 4.4a. In G'_9 for the even numbered vertex segments the leftmost coordinates start alternating from top to bottom. v_5 as the central vertex segment has its leftmost coordinate and then the remaining odd numbered vertices have their left endpoints alternating from top to bottom.

For the rightmost coordinates of the bars, the vertex segments v_6, v_5 and v_4 build a staircase where the bar v_8 has its rightmost coordinate between the rightmost coordinates of v_6 and v_5. Then, the remaining even numbered bar v_2 has its rightmost coordinate, and finally the remaining odd numbered vertices end alternating starting with v_7 in the center, c.f. Figure 4.4a.

For the inductive addition of vertices we want to keep as invariant that the four outermost vertex segments do not have crossings. This is important within the last step, where we describe how to add the remaining edges to achieve a drawing with $5n - 12$ edges in total.

Depending on the orientation of the outermost vertices, namely whether they extend the drawing to the left or the right side, we choose the opposite side to add two vertices inductively. Assume w.l.o.g. that the outermost bars are oriented to the right side. Two vertices, namely v_{10} and v_{11} are added to obtain G'_{11} as presented in Figure 4.5a. The newly added vertex segments are oriented to the left. Both vertices exceed the second to outer vertices and the upper vertex segment namely v_{11} exceeds v_{10} to the left whereas v_{10} exceeds v_{11} to the right. We add 6 planar edges according to the visibility of v_{10} and v_{11} as well as two crossing edges

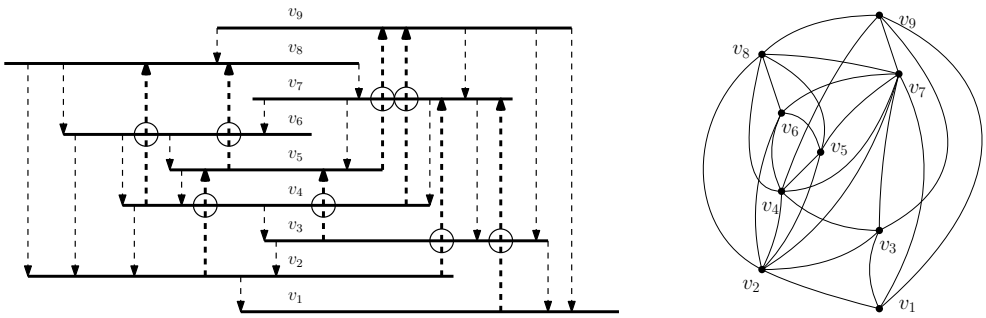


Figure 4.4: (a) The central structure of a maximally dense bar $(1, 2)$-visibility representation is shown. The planar edges are directed downwards and the crossing edges are directed upwards for reading purposes. Additionally the crossings are marked by small circles. Note that except for the four outermost and the central vertex segment v_5, all bars are crossed exactly twice. This structure has $5 \cdot 9 - 17 = 28$ edges. (b) Shows the same structure with vertices as dots and curved edges.

for v_2 and v_8. In total we add 10 edge segments when adding two vertices. Note that, due to the invariant, v_9 and v_1 are not crossed at all. By symmetry we obtain G'_{13} using the same rules shown in Figure 4.5b.

For simplicity we draw only the newly added edges in Figure 4.5a and Figure 4.5b. Note that the structure at the right and left side of the drawing is maintained such that this inductive step can be repeated.

In the ith iteration we have constructed the graph G'_{9+2i} with $n = 9 + 2i$ vertices and $5n - 17$ edges in a bar $(1, 2)$-visibility representation. For simplicity let us assume that i is even. The case that i is odd can be handled analogously by symmetry. As we present exemplary in Figure 4.6, we elongate the outermost vertices to overlap the complete drawing. By construction, the second to outer bars can be crossed exactly twice and we can add one additional planar edge segment. In total, we add 5 more edges in this post processing step to construct the graph G_n from G'_n. G_n has a bar $(1, 2)$-visibility representation and exactly $5n-12$ edges. □

Next, we will augment the graph G'_n to be a bar $(1, 3)$-visibility representation. The upper bound for the number of edges can be estimated by the number of planar edges $(3n-6)$ plus 3 edges crossing each internal bar $(3 \cdot (n-2))$. The upper bound for the number of edges thereby is $6n - 12$ which is close to the lower bound of $6n - 20$ edges for bar $(1, \infty)$-visibility representations [31].

Lemma 4.2. For every odd $n \geq 9$ there exists a graph G_n with $6n - 21$ edges, which has a bar $(1, 3)$-visibility representation.

Proof. We start with an augmentation of the basic structure of the bar $(1, 2)$-visibility graph G'_9 to achieve a bar $(1, 3)$-visibility representation with $6n - 24$

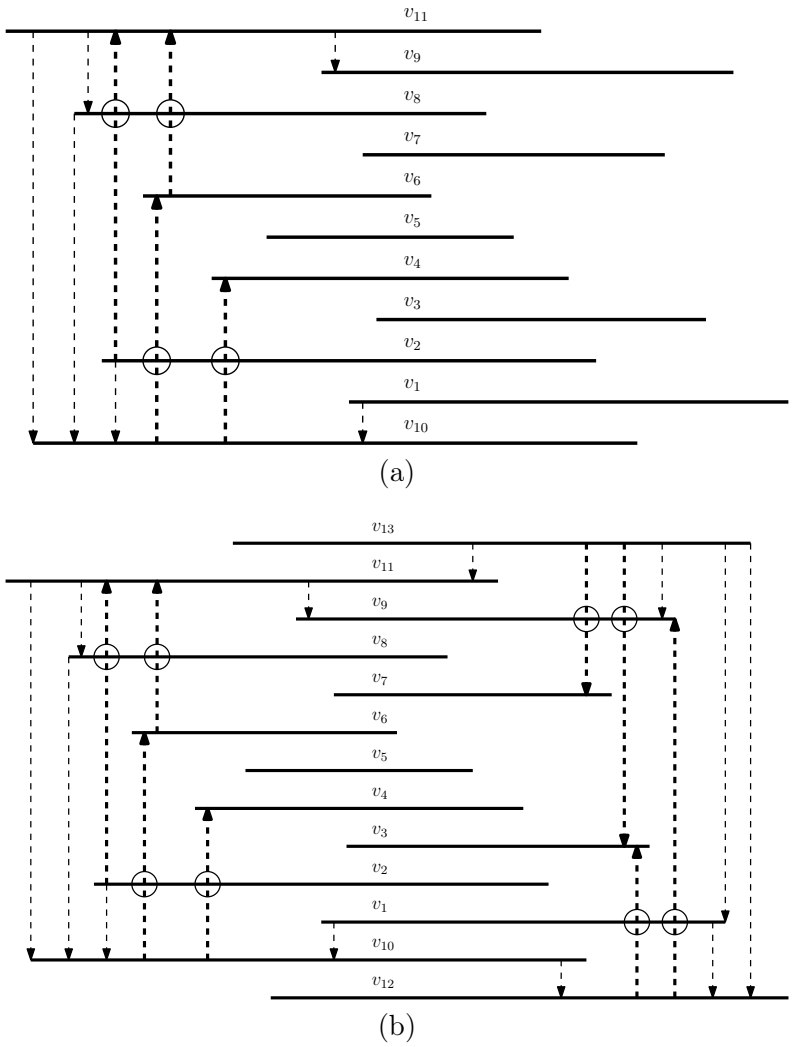


Figure 4.5: Starting from G'_9 we obtain G'_{11} (a) by adding two vertices v_{10} and v_{11} on the left side. Adding inductively two more vertices to G'_{11}, we obtain G'_{13} (b). For every 2 newly inserted vertices we add 10 edges. Note that for readability, only the newly added edge segments are drawn in both figures.

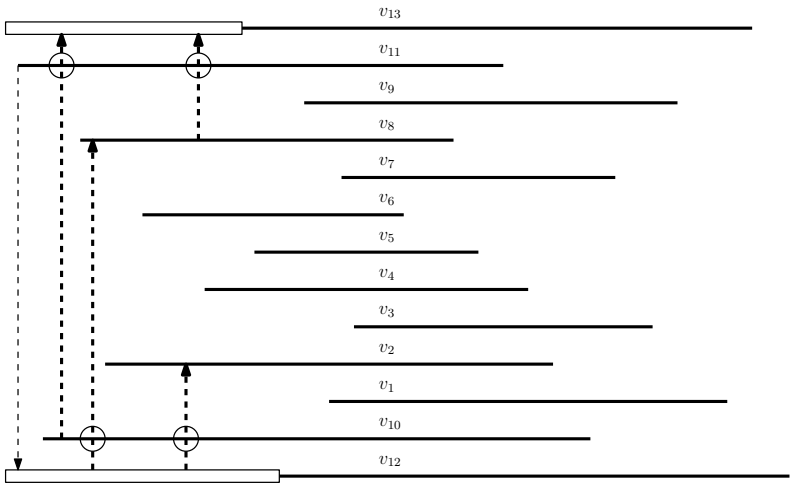


Figure 4.6: The elongation of the two outermost vertices to construct G_{13} from G'_{13} is shown. There are 5 newly added edges, namely one planar edge segment and the previously uncrossed vertices are crossed exactly twice. In general, the graph G_n has $5n - 17 + 5 = 5n - 12$ edges.

edges. We will use a very similar inductive approach and present the adjusted steps to achieve a bar $(1, 3)$-visibility representation with $6n - 21$ edges. Recall that now every vertex segment can be crossed three times.

We start with G'_9 from the previous proof and add two edges, namely (v_1, v_4) and (v_6, v_9). For simplicity we will only draw the crossing edge segments and omit the $3n - 7$ planar edges. The planar edges will stay exactly the same as in the previous proof. We obtain a bar $(1, 3)$-visibility drawing with $n = 9$ vertices and $5n - 15 = 6n - 24$ edges as shown in Figure 4.7. Note that the second to outer bar is crossed once in this representation.

We augment the inductive step such that for any two newly added vertices, we add 12 new edges as shown in Figure 4.8a and Figure 4.8b. Note that every vertex segment is crossed at most three times and thereby we maintain a bar $(1, 3)$-visibility representation of G'_n with $6n - 23$ edges.

In analogy to the previous proof, we conclude with the elongation of the topmost and bottommost vertex segments to add the 3 missing edges as presented previously in Figure 4.6. We can only add 3 more edges, since we already added 2 edges, namely (v_{12}, v_2) and (v_8, v_{13}) during the inductive step. In the final graph, the central vertex and the outermost vertices are not crossed at all. 4 bars close to the center have exactly 2 crossing edges as well as the 2 bars which are placed second to outer. The remaining $(n - 9)$ vertices have exactly 3 crossings each. Together with $3n - 6$ planar edges, we obtain $3n - 6 + (n - 9) \cdot 3 + 6 \cdot 2 + 3 \cdot 0 = 6n - 21$. Thereby, we gave an construction for a bar $(1, 3)$-visibility representation of G_n with $6n - 24 + 3 = 6n - 21$ edges for a given odd $n \geq 9$. □

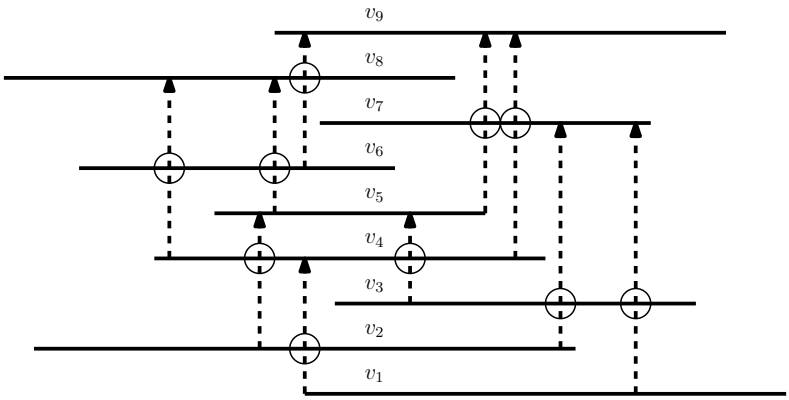


Figure 4.7: The augmented bar $(1,3)$-visibility basic structure G'_9 with two additional edge segments, namely (v_1, v_4) and (v_6, v_9) with $6 \cdot 9 - 24 = 30$ edges. (of which are $3n-7$ planar)

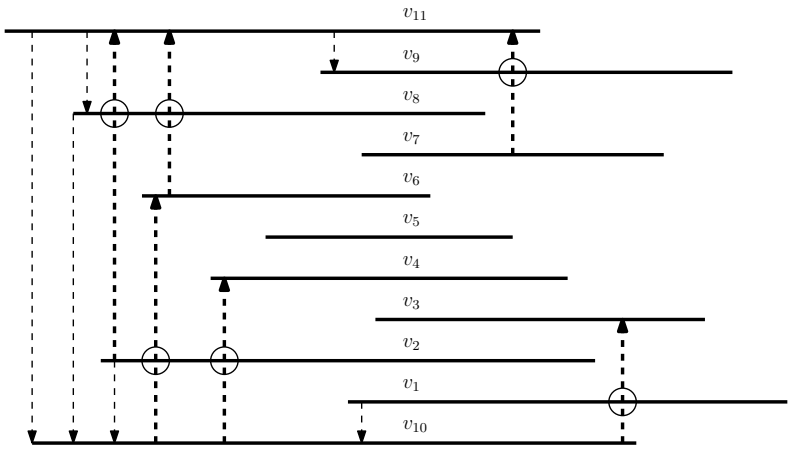
We can show, that three more crossings suffice to reach $34 = 6 \cdot n - 20$ edges if $n = 9$ in Figure 4.9, whereas we were able to generalize the drawing to $6n - 21$ edges for any odd $n > 9$ for bar $(1,3)$-visibility graphs. Unfortunately, we were not able to identify a general way to construct edge maximal bar $(1,3)$-visibility graphs with $6n - 20$ edges but we can show that bar $(1,4)$-visibility graphs admit such a generalization. Thereby, 4 crossings per bar are sufficient to construct edge maximal *bar-visibility* graphs G_n for odd n.

Lemma 4.3. For every odd $n \geq 9$ there exists a graph with a bar $(1,4)$-visibility representation with $6n - 20$ edges.

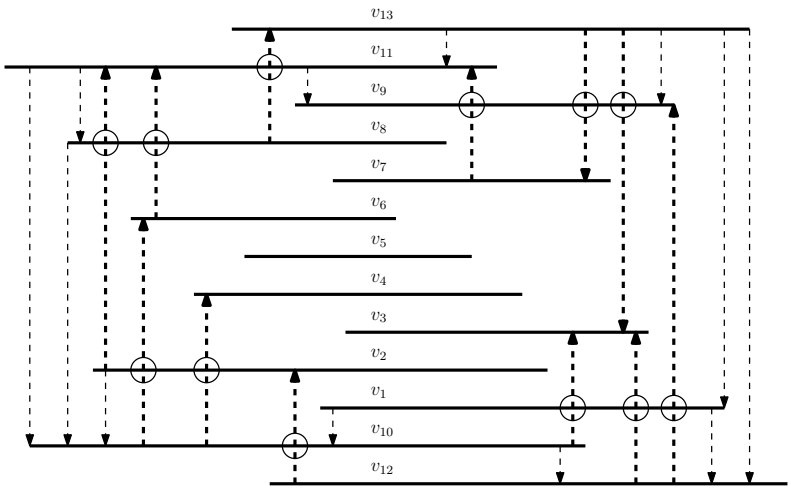
Proof. Since we presented a graph in Figure 4.9 with 9 vertices and $6 \cdot 9 - 20$ edges, we can augment the inductive step based on this graph. The difference is an elongation of v_5 to the left and of v_6 and v_8 to the right. The basic structure and the first two inductive steps are presented in Figure 4.10. The elongations are marked by rectangles. The procedure inductively follows the same pattern as in the previous proofs. By the same argumentation as in bar $(1,3)$-visibility graphs with $6n - 21$ edges, we add one additional edge crossing v_2 and obtain a bar $(1,4)$-visibility drawing with $6n - 20$ edges which matches the general upper bound for any bar $(1,j)$-visibility representation for $j \geq 4$ [31].

□

Corollary 4.4. There exist maximal bar (1,4)-visibility graphs with $6n - 20$ edges and only 1 bar is crossed 4 times.



(a)



(b)

Figure 4.8: In the bar $(1, 3)$-visibility presentation with $6n - 24$ edges of G'_9 we add two vertices and 6 edges in every inductive step according to the adjustment of the central structure in Figure 4.7 G'_{11} (a) and G'_{13} (b) are bar $(1, 3)$-visibility graphs. Note that the vertex segments v_2 and v_8 are crossed 3 times in total. The placement of the new vertex segments is the same as in the proof of Lemma 4.1 .

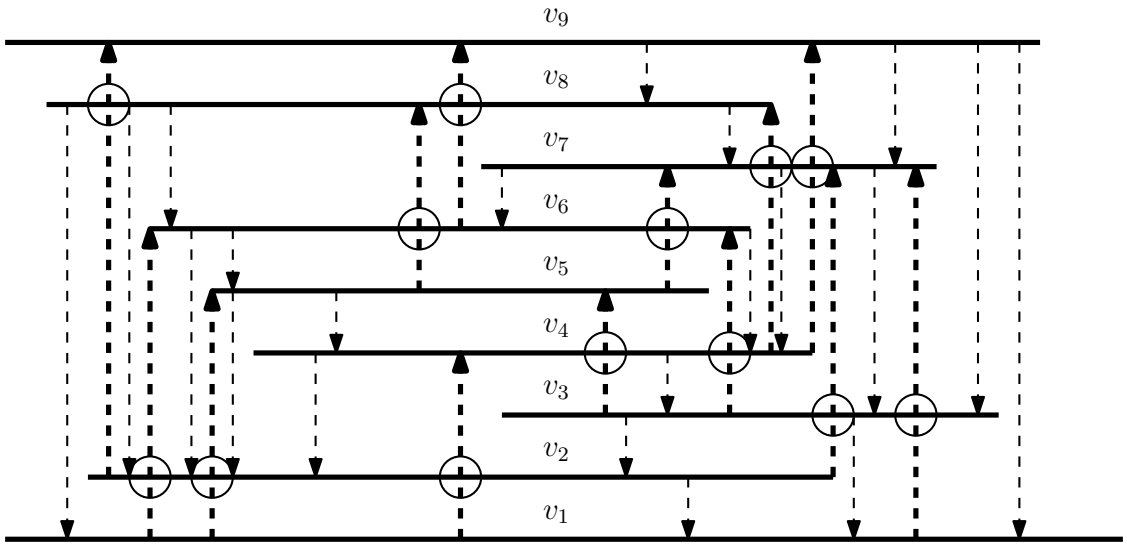


Figure 4.9: A bar $(1, 3)$-visibility drawing with $n = 9$ vertices and exactly $6 \cdot 9 - 20 = 34$ edges. The $3n - 6 = 21$ planar edges are drawn as dashed arrows pointing downwards and the 13 crossing edges are pointing upwards. Each crossing is indicated by a circle.

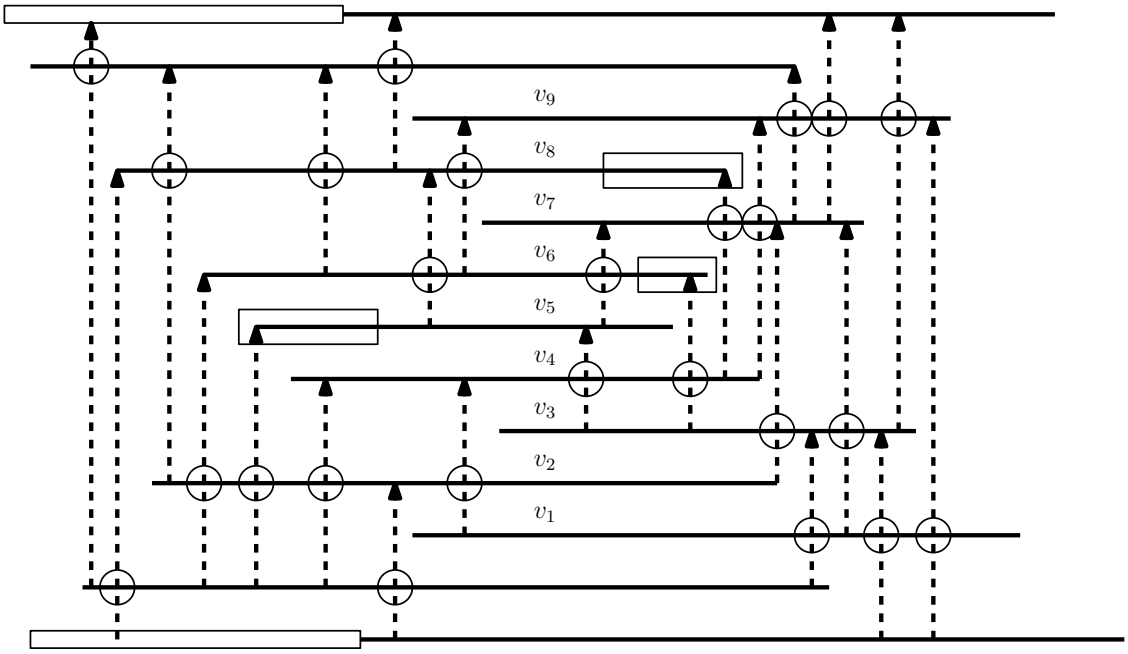


Figure 4.10: A bar $(1, 4)$-visibility drawing of a graph with $6n - 20$ edges. This drawing can be generalized to any odd $n \geq 9$ by inductively adding outer vertices as presented in the previous lemmas. Again the final step to elongate the outermost vertices is the same. The adjustments to the central structure are indicated by bars in rectangles, whereas the rectangles at the outermost vertices indicate their elongation. In this drawing the only bar crossed 4 times is v_2.

4.3 Infinite hierarchy of bar $(1, j)$-visibility graphs

In this section we present an infinite hierarchy of bar $(1, j)$-visibility graphs by arguing on complete bipartite graphs. First we will neglect the geometric properties of a *bar-visibility* representation by representing vertices as convex regions and allowing edges to be continuous curves. These edges are allowed to surpass one vertex-region but are not allowed to cross each other.

We represent vertices as disjoint blobs. Edges are continuous simple curves and connect the boundaries of adjacent blobs. Furthermore, edges are not allowed to cross any other edges but they might pass through one blob. We call the intersections between blobs and edges ports. If an edge is incident to a blob they share exactly one port. An edge passing through a blob shares two ports with the blob. To cross a blob we require another port to be between the two ports of this edge. This is necessary to prevent edges from entering and leaving a port immediately and avoid face intersections this way. Note that since edges are not allowed to cross each other an edge passing through a blob subdivides this blob into two parts. Another edge passing through the same blob has to pass through either part. A face in the blob graph is defined by an alternating series of edge parts and blob boundary parts. Thereby an edge passing through a blob splits two existing faces into four faces. Note that these faces have to share parts of the same blob rather than parts of the same edge. An example of a blob drawing is shown in Figure 4.11.

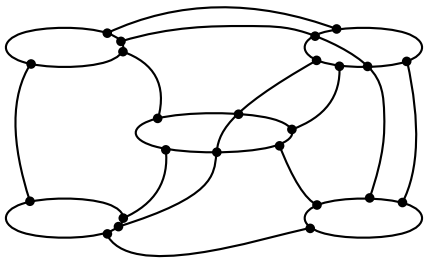


Figure 4.11: In a blob graph the vertices are represented as blobs. The intersection of edges and blobs are called ports. An edge and its incident blob share exactly one port and an edge passing through a blob induces exactly two ports. A face is defined by an alternating sequence of edge and blob parts.

Lemma 4.5. Any bar $(1, j)$-visibility graph can be drawn as blob graph.

Proof. Given a bar $(1, j)$ visibility drawing of a graph G, we can transform every bar to a blob, by assigning an interior region to any bar. Thereby we can guarantee that the blobs are disjoint. The edges stay vertical straight-line segments connecting the boundaries of the blobs. Thus, the edges do not cross each other and since we had a bar $(1, j)$-visibility representation any edge passes through at most one blob. □

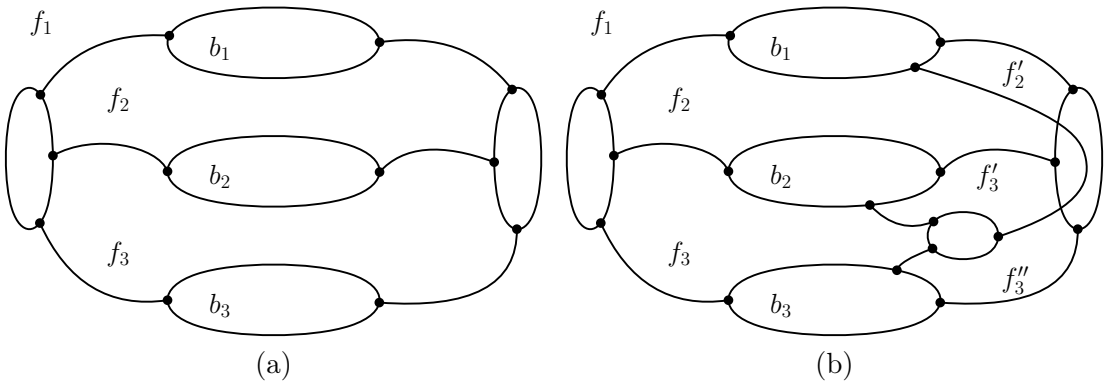


Figure 4.12: Drawing the complete bipartite graph $K_{3,2}$ planar (a) we obtain three faces, namely f_1, f_2, f_3. Any further vertex of $K_{3,n}$ has to be drawn in one of these. The placement of a vertex in f_3 (b) subdivides the faces f_3 and f_2. Note that it is not possible to place any vertex in f_2'.

We argue that in a complete bipartite graph $K_{3,n}$ for $n > 8$ any newly added vertex has a crossing edge with at least one vertex of the first set and thereby the graph $K_{3,(8+3j)}$ admits a bar $(1, j)$-visibility representation but not a bar $(1, (j-1))$-visibility drawing.

Lemma 4.6. For $n > 8$ in $K_{3,n}$ any newly added vertex has an edge crossing one of the three vertices from the first set, namely b_1, b_2 or b_3.

Proof. We will start with a drawing of $K_{3,2}$ in a topological *bar-visibility* representation neglecting the geometric properties as shown in Figure 4.12a. Let us assume for the contrary that the blobs b_1, b_2 and b_3 are not allowed to be crossed by any edge. We obtain three faces, namely f_1, f_2 and f_3 where we can add further vertices. The placement of a third blob in w.l.o.g f_3 will subdivide f_3 and a second face, let us say f_2 as shown in Figure 4.12b. Furthermore, we can place another blob either in f_3' or in f_3'' since otherwise we would induce a crossing of edges. Let us place the next blob in f_3'. This configuration is illustrated in Figure 4.13a. Note that the remaining face to place vertices is f_3'''. Placing one more blob there, we obtain Figure 4.13b.

We can use symmetrical arguments to deduce that there are at most 3 more blobs either in f_1, f_2 or f_3. For this observation we use the property that the initially placed blob is crossed three times. This prohibits the routing of further edges through this blob. Thereby we can place at most 3 more blobs in either face as indicated in Figure 4.14a. This configuration is realizable in a *bar-visibility* drawing, as shown in Figure 4.14b. Now any further vertex in $K_{3,n}$, $n > 8$ has to induce at least one crossing edge on one of the b blobs. By Lemma 4.5 this holds true for *bar-visibility* representations. □

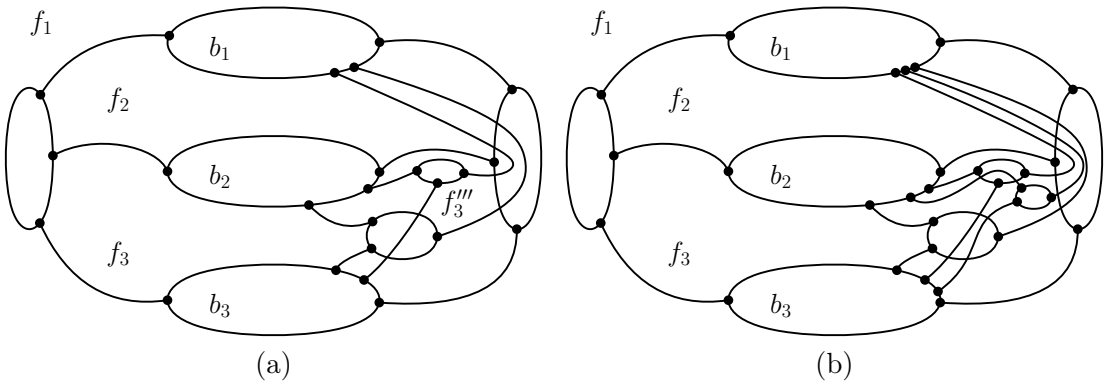


Figure 4.13: The last remaining face to place vertices is f_3''' (a). Placing a vertex there (b) induces a third intersection with the left vertex-region. In this subregion there cannot be any further vertices.

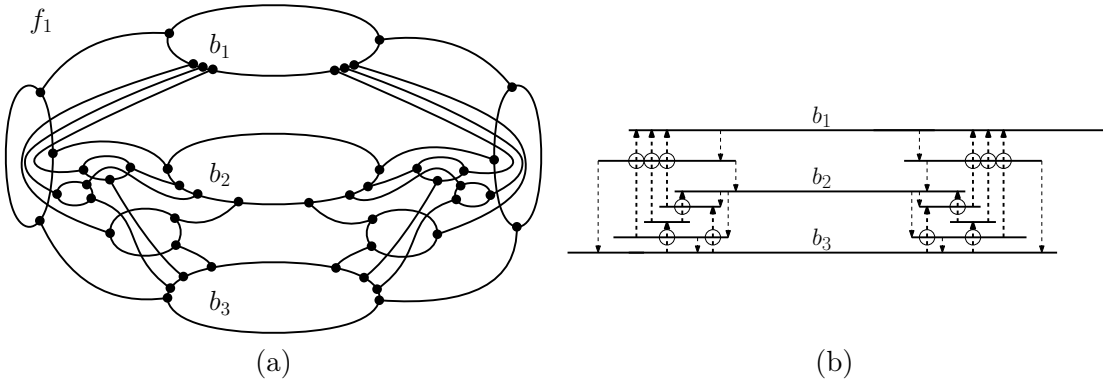


Figure 4.14: By topological argumentation, any further vertex has to induce a crossing edge in either b_1, b_2 or b_3 (a). This structure can be realized in a *bar-visibility* representation (b). Note that neither of b_1, b_2, b_3 are crossed.

Theorem 4.7. *The complete bipartite graph $K_{3,n}$ with $n = 8 + (3j + 1)$ has a bar $(1, j + 1)$-visibility representation but no bar $(1, j)$-visibility representation.*

Proof. Following up the previous proof, the set of newly added vertex segments crossing the bar b_1 is called S_1, and the vertex segments in S_3 have an edge crossing b_3, as presented in Figure 4.15. Note that the vertex segment b_2 can be crossed either from below or above, thereby we introduce the sets S_2 and S_2'. Let us assume we have a bar $(1, j)$-visibility graph $K_{3,n}$ then for each set $|S_1| \leq j, |S_2 \cup S_2'| \leq j$ and $|S_3| \leq j$ must hold. Since we have $n = 8 + (3j + 1)$ vertices at least one of $S_1, S_2 \cup S_2', S_3$ must contain at least $j + 1$ vertices and thereby the graph is bar $(1, j + 1)$-visible but not bar $(1, j)$-visible.

□

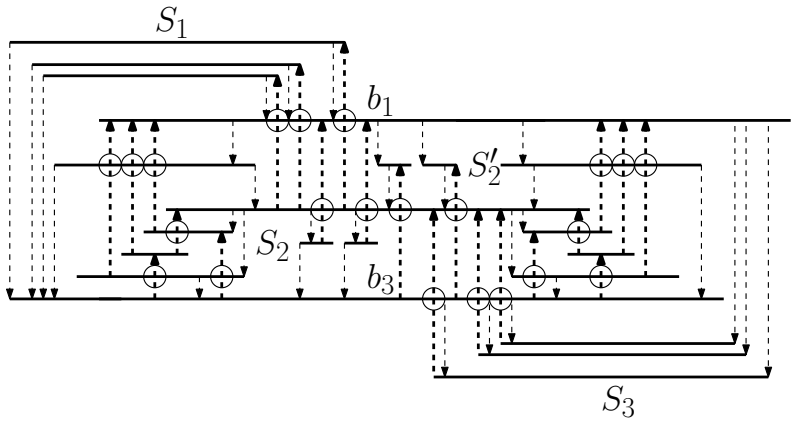


Figure 4.15: A *bar-visibility* drawing of $K_{3,n}$ where $n-8$ vertices are distributed among $S_1, S_2 \cup S_2'$ and S_3. One edge of every vertex in S_1 crosses the bar b_1, $(S_2 \cup S_2')$ crosses b_2 and S_3 crosses b_3 respectively. Let j be the smallest cardinality of these sets, then the drawing is a bar $(1, \min\{3, j\})$-visibility drawing.

Corollary 4.8. There is a infinite hierarchy of classes of bar $(1, j)$-visibility graphs.

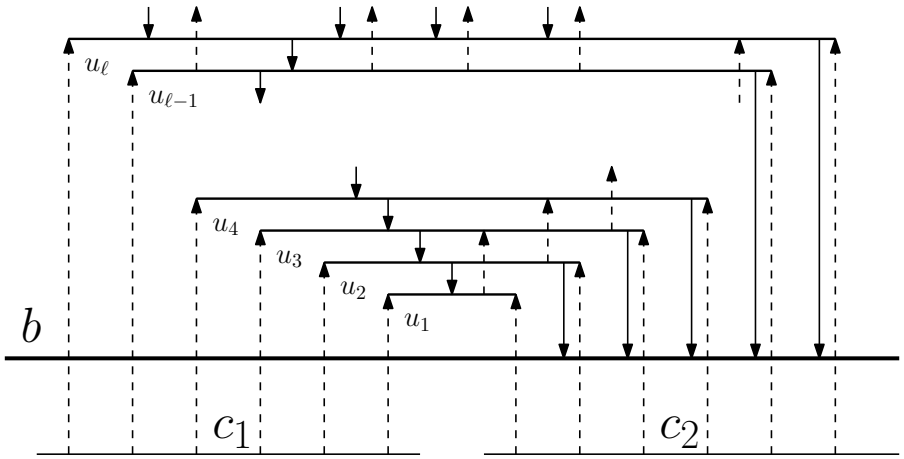


Figure 4.16: Every internal vertex segment $u_i, i > 2$, in the stack of size ℓ, has 5 edges connected to the bottom of each bar, namely connecting to c_1, c_2, b, u_{i-1} and u_{i-2}.

4.4 Maximal bar $(1, j)$-visibility graphs with low density

In this section we present a family of maximal bar $(1, \infty)$-visibility graphs whose number of edges is in the order of $5n+c$, c is a constant. Recall that a maximal graph G belongs to a class $\mathcal{G}$ such that adding any edge e would violate the properties of $\mathcal{G}$, namely $G \in \mathcal{G}$ and $G + e \notin \mathcal{G}$. To ensure the limited number of edges, we need a graph with a very restricted *bar-visibility* representation.

In the following we introduce a graph with a restricted *bar-visibility* representation. Furthermore, we describe how to include a stack $U = \{u_1, \ldots, u_\ell\}$ of ℓ bars such that every bar in the stack, except for the outermost ones, has exactly 5 edges. The general idea of this stack is presented in Figure 4.16, where we have to make sure that the crossed bar b in the bottom is unique. Note that the two topmost vertices and the two bottommost vertices might have more than 5 edges each which will be included in the constant. For counting purposes we state that every internal segment u_i, $i > 2$, has 5 edges connected to the bottom of each bar, namely connecting to u_{i-1}, u_{i-2}, b, c_1 and c_2.

In a restricted *bar-visibility* representation we can guarantee a partial vertical ordering of the bars. This we can use to state the relative placement for most of the bars, meaning we have a limited number of exceptions.

Recall the construction for the infinite hierarchy of bar $(1, j)$-visibility graphs using the complete bipartite graph $K_{3,n}$. We call the first set $B = \{b_1, b_2, b_3\}$. We argue on possible placements of the second set, of bars to be either above, between or below the three vertices from B. In the following we will define the second set to form a path and argue on two additional bars, which we will call h_1 and h_2. We

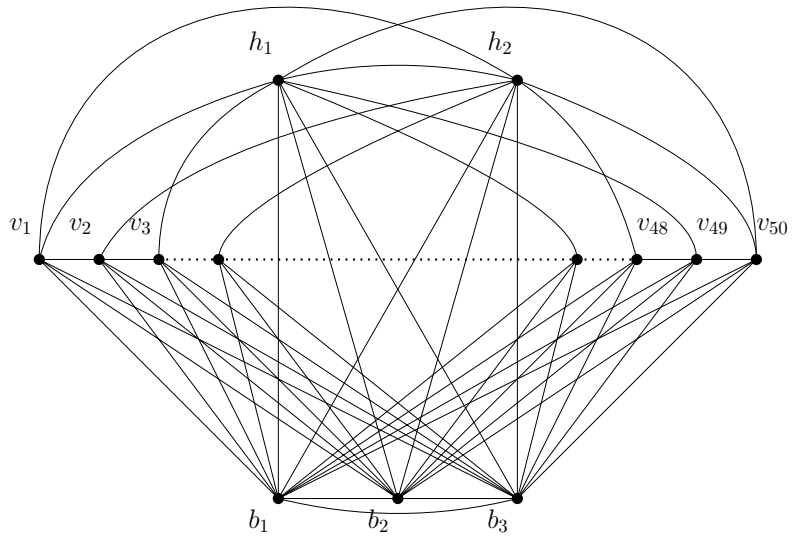


Figure 4.17: The vertices b_1, b_2, b_3 form a complete bipartite graph $K_{3,50}$ with the vertices $\{v_1, v_2, \ldots, v_{50}\}$. The vertices $\{v_1, v_2, \ldots, v_50\}$ form a path and of this subset the odd-numbered vertices are connected to h_1 and the even-numbered vertices to h_2. Furthermore, the vertices b_1, b_2, b_3, h_1, h_2 form a K_5. We add two more edges, namely (v_1, h_2) and (v_{50}, h_1). In [16] we refer to this graph as ship-gadget, where we intended to prove the NP-hardness of the recognition of bar $(1, j)$-visibility graphs.

conclude that this graph has a restricted bar-visibility representation, such that the vertical order of the three vertices, namely b_1, b_2, b_3 and h_1, h_2 is fixed. Additionally, most of the vertices from the second are contained in the common interval of these. This imples that most of these vertices have to be placed vertically between b_1 and b_3.

Then, we apply a stacking argument as indicated in Figure 4.16. We will ensure that the bar $b = b_2$ is unique. Therefore, we start with the complete graph $K_{3,50}$. Note that, 50 is an overestimation of vertices for argumentation purposes. We order the 3 vertices $\{b_1, b_2, b_3\}$ by their y-coordinate so b_2 is the central bar in any drawing. We name the 50 bars $\{v_1, v_2, \ldots, v_{50}\} = C$ and add the edges (v_i, v_{i+1}) for $1 \leq i \leq 49$, so there exists a path. Recall that by Lemma 4.6 in the previous section, at least 42 vertices in C have an edge crossing one of the b's. Since, as presented in the previous section, the 42 bars are placed in either above, between or below b_1, b_2, b_3 (cf. Figure 4.15).

Now, we introduce 2 vertices $\{h_1, h_2\} = H$, and add the edges such that v_i is connected to h_1 if i is odd and connected to h_2 if i is even. Furthermore, we add edges (h_i, b_j) for any i and j so h_1, h_2, b_1, b_2 and b_3 form a complete graph K_5. An illustration of the setting is presented in Figure 4.17. Note that in any *bar-visibility* representation there exists a common interval $I = B \cap H$, since they form a K_5. Recall that, by Helly's Theorem, a set of pairwise intersecting bars has a

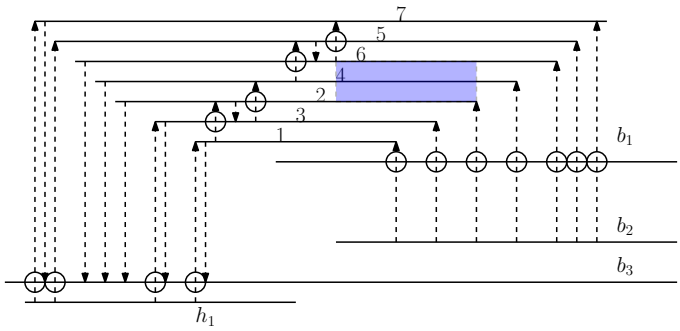


Figure 4.18: Bars crossing the outermost vertex b_1 (or symmetrically b_3) have to be stacked. Let the crossing edges be at the rightmost coordinate of each bar. Since the bars form a path we can place one of h_1 or h_2 at the left (planar) edges. The remaining area to place h_2 is marked as a blue rectangle. Note that we cannot place any further vertex in this configuration.

common intersection. Furthermore, this implies distinct y-coordinates of the bars in $B \cup H$. At last we add the edges (v_1, h_2) and (v_{50}, h_1). Note that the vertices in $B \cup H \cup \{v_1, v_{50}\}$ form a $K_7 - e$, where the edge $e = (v_1, v_{50})$ is missing.

Lemma 4.9. There are at most 7 consecutive vertices in C, whose edge (v_i, b_2) is crossing b_1, namely placed above B, and by symmetry at most 7 vertices below B.

Proof. Assume there are 7 consecutive vertices placed above B and each of the edges (v_i, b_2) cross the segment b_1. W.l.o.g. we place the bar h_1 connected to the odd-numbered vertices close to b_3 and realize their connections. Recall that the bars in C form a path and note that in bar $(1, \infty)$-visibility graphs we can alternate the numbering of vertices in a stack for a few vertices in a way that even-numbered bars are close to each other. Let us furthermore observe that all the rightmost edges of v_i are the crossing edges.

We can place at most 7 consecutive vertices, and the placement of h_2 is limited to be inside the stack as shown in Figure 4.18. In this configuration the placement of h_2 is limited to be in the marked area. Hence the total number of consecutive vertices above B cannot be larger than 7 in total. Additionally observe, that regardless of the placement of h_2 we cannot extend the path by any further vertex.

Alternatively to placing h_2 between the vertices in C, we can place h_2 on top of C. That way, the number of bars from C at one side is at most 5 as presented in Figure 4.19. Note that the vertices v_5 and v_6 might be above h_2 which would be the same setting as in the previous case, since they would have to exceed h_2 to the left to ensure the edge (v_5, b_2). Again there cannot be more than 7 vertices in total. □

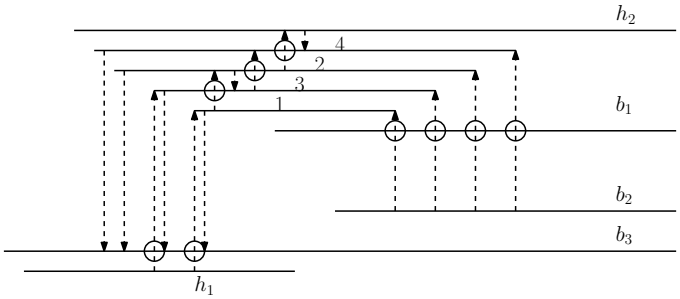


Figure 4.19: Placing h_2 above the stack of vertices leads to at most 4 proper placed bars. The vertices v_5 and v_6 might be placed above h_2 which results in a similar configuration as the previous case since they have to exceed h_2 at both sides.

Lemma 4.10. At least $50 - 2 \cdot (4 + 7) = 28$ vertices have to be vertically between b_1 and b_3.

Proof. Recall that there can be at most 4 vertices in C whose connections to b_1, b_2, b_3 are planar at the right and the left side respectively. Furthermore, by Lemma 4.9, there can be at most 7 vertices either above or below B. The remaining vertices have to be in the common intersection of $B \cap H$. □

Lemma 4.11. A consecutive sequence C of vertices in the common interval of B and H alternate above and below b_2, the middle segment, and the vertices h_1, h_2 have to be above and below b_1, b_2, b_3.

Proof. Recall that the vertices in C have alternating connections to h_1 and h_2 by their indices and most of them have an edge crossing b_2 to connect to either b_1 or b_3. Furthermore, by Lemma 4.10 there are at least 28 of the 50 vertices in C with this property.

The even vertices have to be placed in the center between the four bars of h_2, b_1, b_2 and b_3 as indicated in Figure 4.20. The positioning between b_1 and b_2 is chosen without loss of generality by symmetry. Note that by now the bars b_1 and h_2 are ambiguous. The edges either cross h_1 to connect to b_1 or vice versa. The same argumentation holds true for the odd vertices.

By placing the even vertices between b_2 and b_3, we determine the vertical order to be h_1, b_1, b_2, b_3, h_2 from top to bottom. Since h_2 has to be on the opposite side of b_1 and b_2. Furthermore, note that between two even vertices there is exactly one odd vertex between b_2 and b_1, whose length of the bar is limited by the placement of the crossing edge connecting to b_1. Recall that the edge connecting the even vertices to b_1 has to cross b_2. These edges enforce an order on the alternating vertices along the path.

A valid configuration is given in Figure 4.21. Note that not all of the vertices can be between b_1 and b_2 since there is no valid placement of the vertices h_1 and

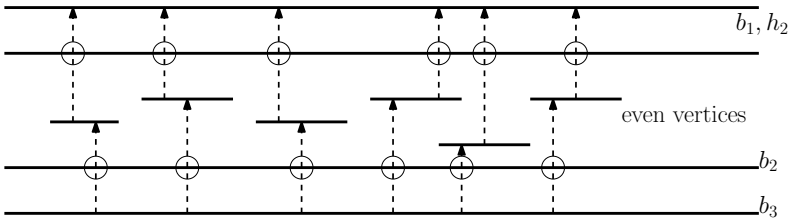


Figure 4.20: The even-numbered vertices have to be placed between the four bars of b_1, b_2, b_3 and h_2. By symmetry we can argue w.l.o.g. that the edges connecting to b_3 have to cross through b_2. Note that the bars b_1 and h_2 are ambiguous without considering the odd-numbered vertices.

h_2 in this case. Additionally, h_1 and h_2 cannot be placed between b_1 and b_3 and not both on the same side since there exist crossing edges in between the bars in C. For the same reason the bars in C cannot be extended infinitely to either side. Furthermore, the bars are ordered from left to right in either increasing or decreasing order, dependent on the side the first two bars are completely in the common interval of h_1, b_1, b_2, b_3, h_2. □

Lemma 4.12. v_1 and v_{50} cannot be in the common interval $I = H \cap B$ and thereby have to be at one of the sides.

Proof. Assume the contrary, that is either v_1 or v_{50} is in the common interval I of $H \cap B$. Let w.l.o.g. v_1 be in the common interval I. Thus, there are at least 3 bars of $H \cup B$ vertically above (or below) v_1. Since v_1 is completely in I there is at least one edge that cannot be realized in a bar $(1, \infty)$-representation since it would have to cross at least two bars. Thereby, the vertices v_1 and v_{50} cannot be placed within I. □

Lemma 4.13. The graph G has a restricted *bar-visibility* representation. And we can choose v_{24} to be c_1 and v_{26} to be c_2 in Figure 4.16.

Proof. In any bar $(1, \infty)$-visibility drawing of G, by Lemma 4.12 the bars of v_1 and v_{50} cannot be within the common interval I of $H \cup B$ and thus have to be at either side of the drawing. Furthermore, by Lemma 4.10 there are at most 20 bars not in the common interval beside v_1 and v_{50}. Thereby the remaining bars are placed in I, especially the bars v_{24}, v_{25} and v_{26} are placed alternating between b_1 and b_3. The edges (v_{24}, h_2) and (v_{26}, h_2) are crossing one of b_1 or b_3. □

Thus we can use these bars to add a stack between the bars of b_1 and h_2, where each vertex is connected to v_{24} and v_{26}.

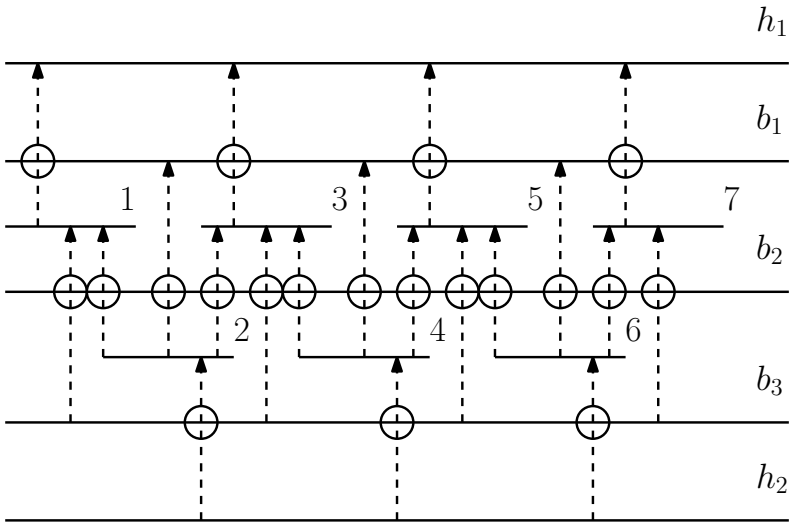


Figure 4.21: Edges for any bar in C have to cross b_2, the central b-vertex, at least once. In the valid configuration of alternating vertices by odd and even indices the vertices h_1 and h_2 are above (below) the b-vertices. Note that the bars in C cannot be extended to either right or left infinitely since there exist crossing edges to both sides. Note that for clarification we name the vertices of an internal subsequence $1, \ldots, 7$ and do not consider the bars placed at the sides.

Theorem 4.14. *There exist maximal bar $(1, \infty)$-visibility graphs with n vertices and $5n + c$ edges, where c is a constant.*

Proof. Lemma 4.13 concludes the restricted visibility representation of the graph G with 55 vertices. It ensures the properties of $c_1 = v_{24}$ and $c_2 = v_{26}$, which are required to place a stack of ℓ vertices. To argue on maximal graphs, we know that the basic structure with $n = 55 + 2$ has at most $6n - 20$ edges. The stack of $\ell - 2$ bars has exactly 5ℓ edges. Recall that the counted edges always attach to the bars from below. In total we have $n' = n + \ell$ vertices and at most $6n - 20 + 5\ell$ edges. For large ℓ, n' converges to ℓ and the number of edges is equal to $5n' + c$, where c is a constant. □

4.5 On bar $(k, 1)$-visibility graphs

Now we consider the second parameter, namely j in bar (k, j)-visibility graphs to be fixed at $j = 1$. Especially we look for graphs that are bar $(k, 1)$-visible for $k > 1$ but not bar $(1, 1)$-visible. In these *bar-visibility* graphs any edge is allowed to cross up to k bars and each bar is allowed to be crossed at most once. Note that these graphs are very sparse for large k. It is difficult to show that a graph is not bar $(1, 1)$-visible not using the sparsity constraint. Note that a drawing with more than $n - 2$ crossings on bars can not be bar $(k, 1)$-visible.

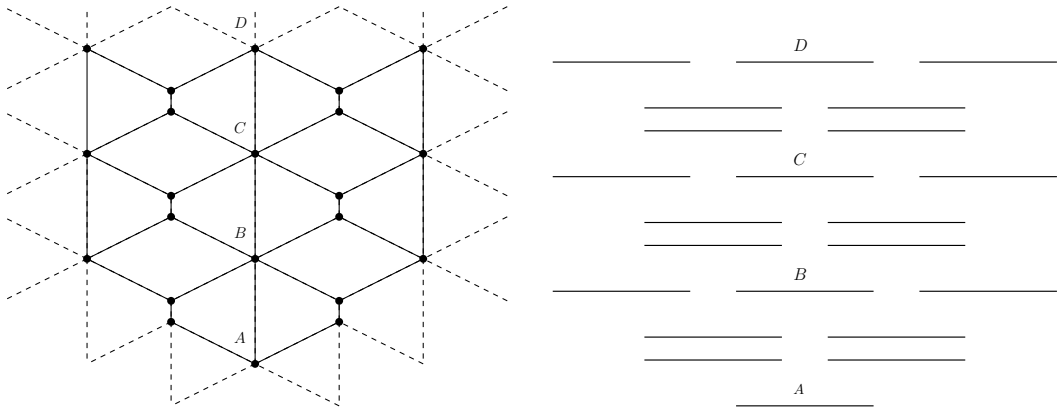


Figure 4.22: (a) A planar subgraph of the embedded quadrangular grid graph G_n and (b) its *bar-visibility* prepresentation.

Theorem 4.15. *Bar $(k, 1)$-visibility for $k \geq 1$ graphs have at most $4n - 8$ edges.*

Proof. In any bar $(k, 1)$-visibility representation all bars except the outer ones can be crossed once. Therefore the graph has at most $(n - 2)$ crossing edges. Together with $3n - 6$ planar edges the graph can have at most $4n - 8$ edges. This bound is tight for maximal bar $(1, 1)$-visibility graphs. □

Corollary 4.16. Bar $(1, 1)$-visibility graphs are a subset of bar $(k, 1)$-visibility graphs for $k \geq 1$.

Proof. By definition any bar is crossed at most once and each edge crosses at most $1 \leq k$ bars. □

Theorem 4.17. *There exist embedded graphs that are bar $(k, 1)$-visible but not bar $(1, 1)$-visible for $k > 1$.*

Proof. Let the graph G_n be a quadrangular grid graph with diagonals with a fixed embedding. An embedding defines the cyclic order of edges around every vertex. Thereby, the planar quadrangular faces are well defined. Furthermore, we define the graph of Figure 4.22a to be a subgraph of G_n. Note that any quadrangulation with diagonals is an optimal 1-planar graph and has exactly $4n - 8$ edges. By [15], this graph is bar $(1, 1)$-visible and the embedding is shown in Figure 4.22b. Note that in any quadrangular face in a *bar-visibility* representation has one topmost and one bottommost vertex and the two middle vertices can be either on one side or on both sides (cf. Figure 4.23). Recall that we consider optimal bar $(1, 1)$-visibility graphs with $4n - 8$ edges and thereby every bar except the outermost bars is crossed exactly once. Furthermore, inserting the diagonals in the *bar-visibility* representation any crossing edge is associated to a middle vertex of the face.

Figure 4.23: Inserting the diagonals (blue) into the quadrangular faces induces a crossing at one of the middle vertices for any face. Note that the assignment of crossing edges to bars is ambiguous.

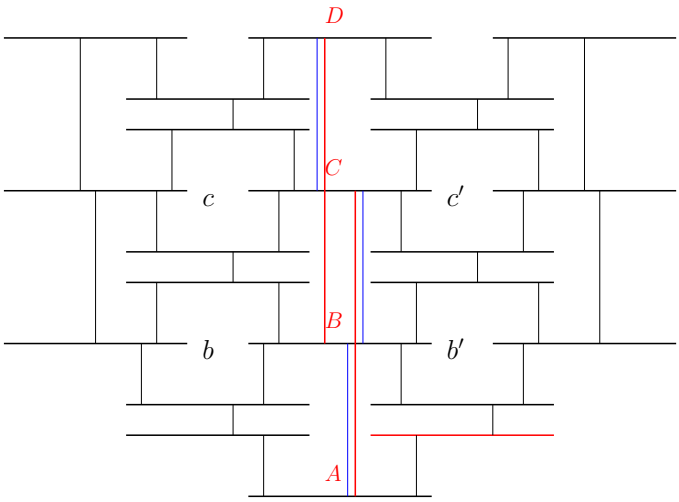


Figure 4.24: After removing one diagonal edge in either quadrangular face c or c' and in either of b and b' we can insert the edges (A, C) and (B, D) to construct an optimal bar $(1, 1)$-visible graph G'_n.

We now alter the bar $(1, 1)$-visibility representation of G_n in Figure 4.22b to get a maximal bar $(1, 1)$ visibility graph G'_n. For the bars (vertices) B and C we remove one of the diagonals of the faces where B and C occur as middle bars. Thereby we ensure that B and C can be drawn crossing-free. Then, we add the edge (A, C) crossing the bar B and the edge (B, D) crossing the bar C. G'_n still has $4n - 8$ edges and every internal bar is crossed exactly once and thereby G'_n is an optimal bar $(1, 1)$-visibility graph. G'_n is sketched in Figure 4.24. By optimality for any crossing edge in G'_n the planar edges along the crossing edges exist. In Figure 4.25 the crossing edge is drawn in red and the planar edges are drawn in blue.

Furthermore, we alter G'_n by replacing the previously added crossing edges (A, C) and (B, D) by a long edge (A, D) to achieve the graph G''_n. Note that due to the embedding of G_n the vertical order of the bars A, B, C, D and the orientation of the faces between them is fixed. Clearly, G''_n is bar $(2, 1)$-visible.

For the sake of contradiction assume that G''_n is bar $(1, 1)$-visible. Since G'_n is

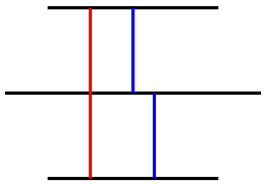


Figure 4.25: In any optimal bar $(1,1)$-visible drawing the two planar edges (blue) along any crossing edge (red) exist.

optimal bar $(1,1)$-visible and the edges (A,C) and (B,D) are crossing edges in G''_n, G''_n has $3n-6$ planar edges.

By the fixed vertical order of A,B,C,D and the presence of $3n-6$ planar edges, the edge (A,D) has to be a crossing edge and has to cross either B or C. Without loss of generality assume that the crossed bar is B but not C (the other case is analogous). By the fact that there exist two planar edges next to any crossing edge in bar $(1,1)$-visibility drawings, we can reinsert the edge (B,D) as a planar edge but then we have $3n-5$ planar edges in total, which is a contradiction to Euler's Formula.

□

4.6 Conclusion and open problems

To conclude this chapter, we will shortly list the results on bar $(1,j)$-visibility graphs. We focus on bar $(1,j)$ visibility representations and investigate properties of maximal graphs. Recall that a maximal graph G belongs to a class of graphs $\mathcal{G}$ such that adding any edge e to G violates the properties of $\mathcal{G}$. We inherit the maximal density for bar $(1,j)$-visibility graphs. Knowing the upper bound of $6n-20$ edges for bar $(1,\infty)$-visibility graphs by Dean et al. [31], we gave a construction for bar $(1,4)$-visibility graphs with exactly this number of edges. For general bar $(1,2)$-visibility graphs we presented graphs with $5n-12$ edges, whereas there is a small gap to the optimal upper bound of $5n-10$. We construct graphs with $6n-21$ edges for general bar $(1,3)$-visibility representations with the special case of $6n-20$ edges if $n=9$. Thereby, we continued our work in [16] and answered the question whether there exist bar $(1,3)$-visibility graphs with n vertices and $6n-20$ edges.

We present an infinite hierarchy of bar $(1,j)$-visibility graphs and show that bar $(1,j)$-visibility graphs are a subset of bar $(1,j+1)$-visibility graphs but not vice versa. This result extends our publication [16], since we are able to answer the remaining question: Can the hierarchy be completed for the small values of j?

As a further result we give a construction of a graph which has a restricted bar $(1,\infty)$-representation and show that there exist edge-maximal graphs with $5n+O(1)$ edges.

We start the investigation of bar $(k, 1)$-visibility graphs. This class of graphs has quite interesting properties regarding the sparsity. Even though, these graphs are at most as dense as bar $(1, 1)$-visibility graphs they are a class of graphs beyond planarity, meaning the upper bound for the number of edges is $4n-8$. We conclude that bar $(1, 1)$-visibility graphs are clearly the most dense subset of bar $(k, 1)$-visibility graphs for larger k.

Next, we want to state some thoughts and formulate a conjecture extending the Theorem 4.17:

Conjecture 4.18. There exist graphs that are bar $(k, 1)$-visible for $k > 1$ but not bar $(1, 1)$-visible.

The difference between this conjecture and the previous Theorem 4.17 is the given embedding of the graph. We present the construction for class of graphs G_n, $n \geq 4$, presented in Figure 4.26, that has a bar $(k, 1)$-visibility representation and we conjecture that this graph has no bar $(1, 1)$-visibility drawing for a fixed k.

The construction of G_n is as follows:
We argue on two vertices, namely A and B, with the edge (A, B) and a cycle of size $(1 + n \cdot (k + 1))$. Now we connect every vertex in the cycle with an edge to both A and B, an example is presented in Figure 4.26a. Then, as shown in Figure 4.26b, we add a sequence of $(k + 1)$-hops, connecting every $(k + 1)$th vertex in the circle starting and ending at the last vertex $v_{(1+n\cdot(k+1))}$. For each vertex of a hop we add the edge connecting its predecessor and its successor and the edge $(v_2, v_{n\cdot(k+1)})$. The last edge $(v_2, v_{n\cdot(k+1)})$ surrounds the first and the last vertex. An illustration is given in Figure 4.26c. This construction admits a bar $(k, 1)$-visibility representation. In Figure 4.27 we present two different bar $(3, 1)$-visibility representations of the exemplary graph from Figure 4.26. Conveniently, we draw only the crossing edges in this drawing.

From our understanding of bar $(k, 1)$-visibility drawings we have two major ways to realize the segments belonging to the cycle. We can either stack the bars as a staircase or as a pyramid and connect the hops crossing k bars. Both extreme cases are presented in Figure 4.27 and we observe that a mixture of these configurations is realizable. The placement of A and B depends on the structure and most of the vertices are either between them or above. Note that in both configurations every bar except for the outermost ones is crossed exactly once.

To support the conjecture, first of all we want to recall that the vertices A and B have a common overlap. By this common overlap we can deduce that most of the vertices are either vertically between A and B or above (resp. below). If the vertices of the cycle are represented as bars in a staircase as in Figure 4.27a the majority of the vertices have to connect to A on the left side and to B on the right side as indicated in Figure 4.28. The same implication would hold in the pyramid setting from Figure 4.27b. Note that it is likely to mix both configurations having a pyramid on the top part and a staircase between A and B. Nonetheless many

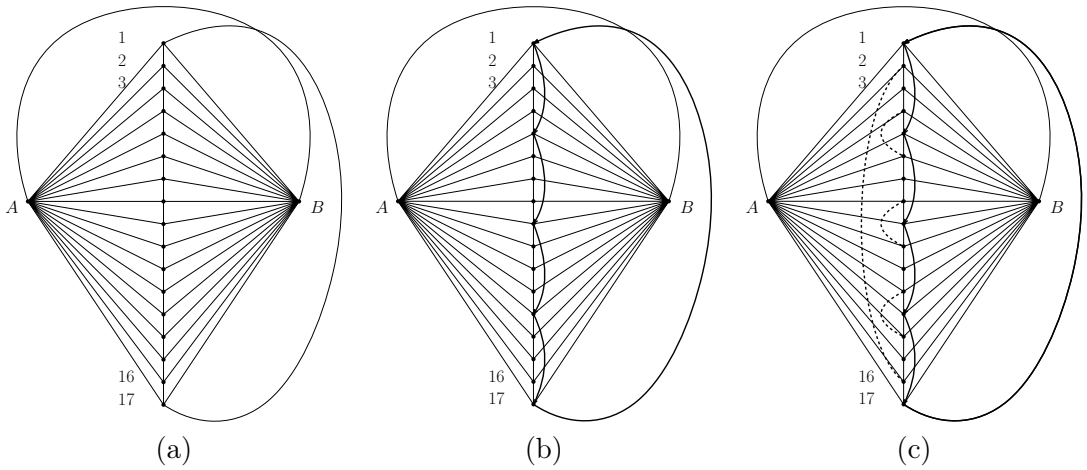


Figure 4.26: (a) shows a circle with $17 = 1 + 4 \cdot (3 + 1)$ vertices where each vertex is connected to A and B. Additionally, the edge (A, B) is drawn. In (b) we added a sequence of 4 hops, that is connecting every 4th vertex starting and ending at vertex v_1. Finally, we add the dashed edges sourrounding the vertices used in the hops in (c).

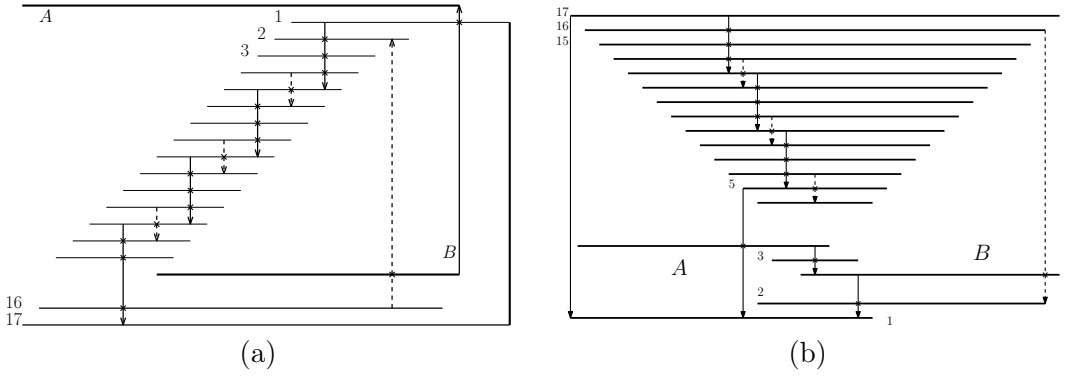


Figure 4.27: Two different bar $(3, 1)$-visibility representations of the graph G_n with 19 vertices from Figure 4.26. Note that every bar except for the outermost bars is crossed exactly once in both representations.

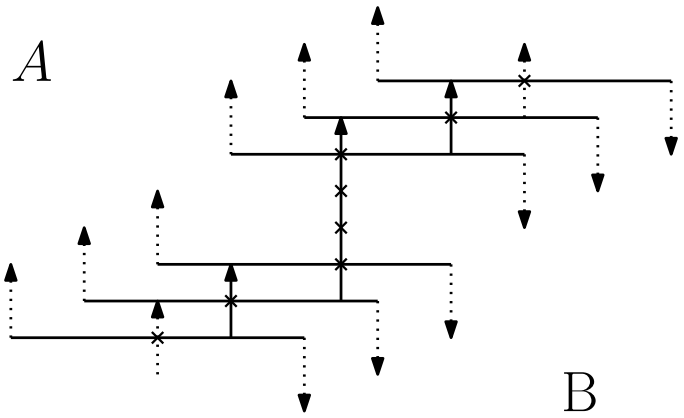


Figure 4.28: Bars which are located between A and B have to connect to both vertices. The majority of these vertices connect at the right side to B and at the left side to A.

vertices in either configuration should imply a subset of bars which is vertically ordered by their occurrence in the cycle.

A fixed vertical ordering of a sufficiently large subset of bars in either configuration would imply bar $(k, 1)$-visibility and not bar $(1, 1)$-visibility.

To conclude our work on bar (k, j)-visibility graphs, recall that we focused mainly on bar $(1, j)$-visibility graphs. We continued with graphs, where j is fixed to be $j = 1$ instead of k, namely bar $(k, 1)$-visibility graphs.

Recall that the number of edges in bar $(1, \infty)$ is at most $6n - 20$, by the work of Dean et al. [31]. In bar $(1, 1)$-visibility graphs the number of edges is bound by $4n - 8$ [15]. Furthermore, bar $(1, 1)$-visbility graphs include the class of 1-planar graphs. Extending this model to bar $(k, 1)$-visibility representations, for $k \geq 1$ we describe a class of graphs, which includes all bar $(1, 1)$-visibility graphs. Therefore, bar $(k, 1)$-visibility graphs have the same upper bound in the number of edges, namely $4n - 8$. This leads to a class of graphs beyond planarity with few edges.

Given this upper bound for the class of bar $(k, 1)$-visibility for $k \geq 1$ and the knowledge, that bar $(1, 1)$-visibility graphs are a subset of these graphs, the further investigation of this class of graphs would be interesting.

In a fan-planar graph an edge is allowed to cross multiple other edges, as long as the crossed edges share a common vertex. In the corresponding *bar-visibility* representation, it might suffice to cross one bar, rather than a fan of edges. We would be interested if fan-planar graphs can be drawn as bar (k, j)-visibility representation with small values for k and j.

To the best of our knowledge, there is still no efficient algorithm to draw bar (k, j)-visibility graphs for $k \geq 2$. We conjecture the recognition of bar $(1, j)$-visibility graphs to be NP-hard, even for fixed j due to their close relation of bar $(1, 1)$-visibility graphs to 1-planar graphs. A remaining task is to develop an ap-

proximation algorithm or a heuristic to minimize j, k or both parameters in given drawings.

We conclude this chapter with a short list of open problems:

1. Can we further describe the class of bar $(k, 1)$-visibility graphs? Is there a relation of these to other known graph classes?
2. Can we relate fan-planar graphs to bar (k, j) visibility graphs?
3. Can every bar $(1, 1)$-visibility graph be drawn efficiently allowing more crossings, e.g. as bar $(1, 2)$- or bar $(2, 2)$-visibility representation?
4. Can we derive bounds on the number of edges for general bar (k, j)-visibility graphs, especially for $k = 2$?

Chapter 5

Summary

In this thesis we present new insights on three topics in graph drawing. The *ply-number* was initially introduced by Eppstein et al. [42] during their study of road networks. Further investigation by De Luca et al. [30] suggest the *ply-number* as measurement for aesthetically pleasing drawings of graphs. We present new theoretical insights on drawings with low *ply-number*. We extend our research to the *vertex-ply-number* and focus on *empty-ply* drawings, namely drawings with *vertex-ply-number* 1. In our third chapter, we propose an efficient algorithm to compute the *ply-number* for a given drawing. Our algorithm is experimentally evaluated and directly compared to previous implementations by [30]. We introduce a workflow to optimize drawings regarding the new parameter. In the fourth chapter we present our work on *bar-visibility* representations beyond planarity. For planar graphs it is known that they have planar *bar-visibility* representations [98], as well as for 1-planar graphs [83]. We introduce a new notation, namely bar (k, j)-visibility representations, where each bar can be crossed at most j times and any edge can cross at most k bars. This allows us to clarify the different notations of bar k-visibility graphs used in [31], [83] and [15] in Section 4.1. Especially, we continue our work published in [16] and answer some of our remaining open problems.

In the beginning of Chapter 2 we give a summary on the previous work on the *ply-number* of drawings [3, 4, 36, 42]. Then, we present results on graphs which have drawings with bounded *ply-number*, namely *ply-number* 1 and 2. Drawings with *ply-number* 1 have very restrictive properties such as uniform edge lengths and uniform vertex distribution. Thereby, drawings with *ply-number* 1 are polynomial in area. We conclude that the correspondence to unit-disk contact representation and thereby the recognition of graphs that can be drawn with *ply-number* 1 is NP-hard.

Graphs that can be drawn with *ply-number* 2 include stars, binary trees and caterpillars. Recall that in drawings with *ply-number* 2 at most two *ply-disks* over-

lap in any coordinate and we observe that for binary trees the ratio r between the lengths of two incident edges can be $\frac{1}{2}$. We give constructions to obtain drawings with *ply-number* 2 for the named graph classes in Section 2.3.2. Recalll that drawings with *ply-number* 2 might require exponential area.

We state in Theorem 2.4 that any graph with n vertices can be drawn with *ply-number* less or equal to $\frac{n}{2}$ and prove this bound considering complete graphs with n vertices. Additionally, we have polynomial area requirements for these drawings.

We continue by pointing out the relation between the *ply-number* and the *vertex-ply-number* of drawings in Section 2.5. Recall that the *vertex-ply-number* denotes the maximal number of *ply-disks* any vertex in the drawing is contained in. Note that the *vertex-ply-number* of a drawing cannot exceed the *ply-number* of the drawing. In particular we show that any drawing with *vertex-ply-number* h has a *ply-number* less than $5h$ (recall Theorem 2.5). For further examination of the *vertex-ply-number* we focus on *empty-ply* drawings. In any *empty-ply* drawing the ratio between edges incident to the same vertex have a edge-length ratio of at most 2. This fact directly translates to the radii of *ply-disks* of adjacent vertices. Nonetheless, the area requirements can be exponential, since the ratio between the shortest and the longest edge in the drawing of a graph G can be as large as 2^d, depending on the diameter d of G. Additionally, we discovered that the square of a graph with *ply-number* 1 admits an *empty-ply* drawing. This does not hold for graphs with *ply-number* > 1, as we present a counterexample. Furthermore, we give a proof that the maximum vertex-degree in any *empty-ply* drawing is less than 24 (cf. Theorem 2.12).

We continue to name graphs with and without *empty-ply* drawings in Section 2.7. We present drawings for the star $K_{1,24}$, the complete graph K_7, and the complete bipartite graphs $K_{2,12}, K_{3,9}$, and $K_{4,6}$.

We present an exhaustive case distinction to prove that the complete graph K_8 does not admit an *empty-ply* drawing. Thereby, for any $n \geq 8$ the complete graph K_n is not *empty-ply*. Applying the same techniques, we are able to show that the complete bipartite graph $K_{2,15}$ does not admit an *empty-ply* drawing. Thus the remaining question, whether the graphs $K_{2,13}$ and $K_{2,14}$ have *empty-ply* drawings, is still open.

We conjecture that $K_{2,13}$ does not admit an *empty-ply* drawing and it might be proven by refining the argumentation in Theorem 2.21. The argumentation on the *ply-disks* could be more restrictive, since during our argumentation we rather extensively use the *cover-disks*. Thereby, the result is an overestimation in terms of accuracy arguing on any drawing. Although not proven, it is reasonable to conjecture that the *empty-ply* drawings of K_7 and $K_{4,6}$ are unique with respect to scaling and rotating since, in contrast to the other known complete and complete bipartite graphs, these drawings do not allow any movement of vertices without violating the *empty-ply* property.

In Chapter 3 we present our algorithmic approach to compute the *ply-number* of a given drawing. There have been some attempts to solve this task before, see [7] and [30]. We present a supportive tool to give the user an intuitive understanding of the *ply-number* as aesthetic criterion. The user can see the impact of modification of the drawing on the *ply-number*. We implemented different layout techniques, namely organic, circular, and random placement of the vertices.

Beside the basic functionality, described in Section 3.3, we improved the existing algorithm to compute the *ply-number* of a given drawing. This is an important step to enable instant feedback to the users actions. Our algorithm is significantly faster maintaining the same accuracy as previous implementations [56]. In fact we are able to reduce the time spent on the computation of the *ply-number* from seconds to milliseconds. This result is presented in Section 3.6.3.

Our algorithm uses the plane-sweep technique to compute intersections of *ply-disks* and keeps track of the current number of intersecting disks, namely the ply-value. The maximum over all ply-values is the *ply-number* of the drawing. For details recall Section 3.5. In contrast to the previous applications, we handle inconsistencies regarding precision errors differently. While in previous implementations [7, 30] the precision of the computation was increased, we focus on keeping the *ply-disks* invariant intact. That is the order of closing and opening *ply-disks* along a vertical line can only be changed by computed intersections. This technique allows us to use computationally effective operations on the data structure `double`.

To confirm the use of our algorithm, we perform experiments on the well known Rome graphs, and set of randomly generated graphs. The set are described in Section 3.6.1 in detail. Furthermore, we compare our implementation to the implementation in [30] on a set of drawings kindly provided by Felice De Luca. These graphs are drawn using the multipole multilevel method by Hachul and Jünger [53].

On the Rome graphs, we identify a linear correlation between the number of vertices and the *ply-number*, rather than a correlation between the density of the graphs and the *ply-number*. Nonetheless, the density of the graphs clearly has a minor impact on the *ply-number* of the drawings. Furthermore, we confirm that the time spent on computation is mainly determined by the number of intersections of *ply-disks* in the drawing. Throughout the comparison of the different layout algorithms on this set, we can conclude that in sparse graphs the organic layout gives drawings with the lowest *ply-number* and the least number of intersections. As expected, the randomized placement of vertices yield the highest *ply-number*s and highest number of intersecting *ply-disks*. Interestingly, the circular layout gives the most problematic intersections due to the symmetric placement of the vertices, and thereby a high likelihood of precision errors. The precision errors have a huge impact on the computation time, namely on average the circular layout took by far the longest time to compute the *ply-number* on the compared layouts. This can also be confirmed by the large amount of postponed events in these computations.

We use the provided set **FM3data** to compare the speed of the algorithms and

confirm that we reduce the overall computation time from seconds to milliseconds in Section 3.6.3. This allows us to further investigate a heuristic optimization approach, which requires many computations of the *ply-number* for various drawings of the graphs.

Since the **FM3data** drawings have been shown to have low *ply-numbers* in [30], we perform another experiment on this set in Section 3.6.4. We use a tuned spring-embedder to further optimize the drawings. On average we can reduce the *ply-number* on this set from 4.3 to 3.3, which leads to the conclusion that there can be even better results. Recall the upper bound of $\frac{n}{2}$, which is presented in Section 2.4, we use the circular layout as fall-back strategy, whenever the *ply-number* exceeded this bound. This gave us an advantage on dense graphs, where the force directed algorithms do perform similarly to just randomly placing vertices.

The upper bound of $\frac{n}{2}$ is a good starting point for further optimization, since dense graphs are challenging for force directed algorithms [62, 85]. Furthermore, this indicates that the circular layout in dense graphs is an appropriate approach to draw graphs with respect to aesthetic criteria in straight-line drawings.

In order to optimize drawings of sparse graphs we can observe that the organic layout performed similar to the multilevel multipole method by Hachul and Jünger [53]. Recall that the multilevel multipole method places vertices on an integer grid with small average edge lengths. Thereby, the occurrence of precision errors and thus a negative effect on the computation is likely. In our tool, the organic layout has larger average edge lengths. We conjecture that we can increase the accuracy of computations by scaling, but cannot confirm this thesis by experimental data.

In Chapter 4, we present the generalization to bar (k, j)-visibility representations to clarify the different definitions of bar 1-visibility being used in [15, 31, 44] and [83]. Recall that the definitions relate to bar $(1, \infty)$-visibility or bar $(1, 1)$-visibility representations.

In particular, we present results on bar (k, j)-visibility representations for fixed k, namely $k = 1$. We know the upper bound of $6n - 20$ edges for any bar $(1, \infty)$-visibility representation by Dean et al. [31] and we investigate bar $(1, j)$-visibility representations for fixed j regarding their maximality. Recall that a maximal graph G belongs to a class of graphs $\mathcal{G}$ such that adding any edge e would violate the properties of $\mathcal{G}$. In particular, we construct maximal graphs with bar $(1, 2)$, bar $(1, 3)$-visibility representations and proved that bar $(1, 4)$-visible graphs are sufficient to construct maximal graphs with $6n - 20$ edges. We observe that there exist maximal bar $(1, 4)$-visibility graphs with $6n - 20$ edges with exactly 1 bar that is crossed 4 times and all other bars have less crossings.

Furthermore, we show in Section 4.3 that there exists an infinite hierarchy of bar $(1, j)$ graphs, meaning that for any $j > 1$ there exist graphs that have a bar $(1, j)$-visibility representation but not a bar $(1, (j - 1))$-visibility representation.

Thereby, we extend our work [16] and complete the argumentation for small values of j. We use topological arguments, neglecting the geometric properties of bar $(1, j)$-visibility representations in this part of the thesis.

We give a sparse class of maximal bar $(1, \infty)$-visible graphs with about $5n$ edges in Section 4.4. For this purpose we identify a graph with a very restricted *bar-visibility* representation. Most of the vertices of a path, where any vertex is connected to 3 base vertices and alternating connected to 2 vertices, have to be placed between the 5 vertices in an alternating fashion. From this restricted base structure, we can conclude that there exist sparse graphs with maximal bar $(1, \infty)$-representation.

In the final section we present our results on the investigation on bar (k, j)-visibility representations with fixed $j = 1$. We know that the number of edges in this graph class is less or equal to the number of edges in maximal bar $(1, 1)$-visibility graphs, namely $4n - 8$. Furthermore, we discover that these graphs are a sparse class of graphs class beyond planarity. In contrast to most of the investigated graph classes beyond planarity, which have the common property that they are dense. We know that the bar $(1, 1)$-visibility graphs include all 1-planar graphs [31, 83] and any bar $(k, 1)$, where $k \geq 1$, includes all bar $(1, 1)$-visibility graphs.

The sparsity constraint is a commonly used property to argue on graph classes beyond planarity and therefore, bar $(k, 1)$-visibility graphs, as a sparse set of graphs, should be investigated in further research.

5.1 Open Problems

We want to conclude this thesis with a short list on open problems, arising from our work. We presented *empty-ply* drawings for several graphs, namely K_7, $K_{2,12}, K_{3,9}$ and $K_{4,6}$ in Section 2.7. Arising from this, we would be interested is we can close the gap on complete bipartite graphs, especially if $K_{2,13}$ or $K_{2,14}$ have *empty-ply* drawings. Are there similar results for other complete bipartite graphs as $K_{n,m}$ for larger values of n?

During our work on the *ply-number* of drawings, we encounter the conjecture by Stephen Kobourov that any 3-regular planar graph has an *empty-ply* representation. We tested a few instances of these graphs and it seems that the conjecture holds true, where we do not have a proof in any sense. Interestingly most of the drawings where not planar. This would be a good result, since by now we only know a few graphs that have *empty-ply* drawings rather than a class of graphs.

We want to recall the question for *k-vertex-ply* drawings. Here, any vertex can be contained in at most k *ply-disks*. This corresponds to the relaxation of k-proximity drawings, where at most k vertices are allowed to be in any proximity drawing. In this topic there exist concepts of k-Gabriel and k-relative- neighbourhood drawings [65].

Clearly, by Theorem 2.5 any graph with *vertex-ply-number* h has a *ply-number* at

most $5h$ and the *vertex-ply-number* is at most equal to the *ply-number* in any drawing. Can we describe drawings where these values are always equal? Can we describe drawings with *vertex-ply-number* h and *ply-number* $5h$?

We introduced a workflow to optimize the *ply-number* of a given drawing. We can formulate rules for low degree vertices to minimize their *ply-disk* 's radius and thereby their contribution to the *ply-number* of the drawing. For example vertices with degree 1 can be placed close to their adjacent vertex to minimize the *ply-disk* of this vertex. We think that the optimization of drawings regarding their *ply-number* can be further improved.

For the algorithmic part, recall that we mention the possible influence of scaling to the accuracy and the number of precision errors in our ply-computation, which we cannot support by experiments by now. It would be interesting, if the number of precision errors can be efficiently reduced just by scaling the drawing.

Furthermore, we mentioned in Section 3.7 that spring-embedding algorithms with enforced equal edge lengths seem to converge if and only if the drawing has *ply-number* 1. This might be extended to a test for the NP-hard recognition problem of Unit-Disk contact representation. Note that this is still a heuristic.

By [44] we have the relationship between bar $(1, \infty)$-visibility representations in both, strong and weak, models to planar, 1-planar and quasi-planar graphs. Even though 1-planar graphs are well categorized for a long time, a complete characterization of optimal 2-planar and 3-planar graphs has been done recently [9]. Can we extend results on these graphs to bar (k, j)-visibility representations with constant $k > 1$ or $j > 1$? In particular, do all 2-planar graphs have a bar $(2, 2)$-visibility representation? Following up our work, can we derive bounds on the number of edges for general bar (k, j)-visibility graphs, especially for $k = 2$? Clearly, any complete graph with n vertices has a bar $(n-4, n-4)$-visibility representation. This bound can be derived by drawing all vertices above each other and allowing the topmost and bottommost vertices to be connected by planar edges. The remaining $n-4$ vertices simply connect through all the other vertices.

We started the investigation of bar $(k, 1)$-visibility graphs. We would be interested in the characterization of these graphs. By now, we know that they are a class of sparse graphs beyond planarity. Can we relate bar $(k, 1)$-visibility graphs to known graph classes? We think that fan-planar graphs might be realizable as bar (k, j)-visible drawings with small values for k and j. Recall that in a fan-planar graph, an edge can cross multiple other edges as long as all the crossed edges share a common vertex. This is likely to relate to a crossed bar. The detailed relationship has yet to be discovered.

Bibliography

[1] Pankaj K. Agarwal, Boris Aronov, János Pach, Richard Pollack, and Micha Sharir. Quasi-planar graphs have a linear number of edges. *Combinatorica*, 17(1):1–9, 1997.

[2] Patrizio Angelini, Michael A. Bekos, Franz J. Brandenburg, Giordano Da Lozzo, Giuseppe Di Battista, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ignaz Rutter. On the relationship between k-planar and k-quasi-planar graphs. In Hans L. Bodlaender and Gerhard J. Woeginger, editors, *Graph-Theoretic Concepts in Computer Science - 43rd International Workshop, WG 2017, Eindhoven, The Netherlands, June 21-23, 2017, Revised Selected Papers*, volume 10520 of *Lecture Notes in Computer Science*, pages 59–74. Springer, 2017.

[3] Patrizio Angelini, Michael A. Bekos, Till Bruckdorfer, Jaroslav Hancl Jr., Michael Kaufmann, Stephen G. Kobourov, Antonios Symvonis, and Pavel Valtr. Low ply drawings of trees. In Yifan Hu and Martin Nöllenburg, editors, *Graph Drawing and Network Visualization - 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers*, volume 9801 of *Lecture Notes in Computer Science*, pages 236–248. Springer, 2016.

[4] Patrizio Angelini, Steven Chaplick, Felice De Luca, Jirí Fiala, Jaroslav Hancl Jr., Niklas Heinsohn, Michael Kaufmann, Stephen G. Kobourov, Jan Kratochvíl, and Pavel Valtr. On vertex- and empty-ply proximity drawings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization - 25th International Symposium, GD 2017, Boston, MA, USA, September 25-27, 2017, Revised Selected Papers*, volume 10692 of *Lecture Notes in Computer Science*, pages 24–37. Springer, 2017.

[5] Evmorfia N. Argyriou, Michael A. Bekos, and Antonios Symvonis. The straight-line RAC drawing problem is np-hard. *J. Graph Algorithms Appl.*, 16(2):569–597, 2012.

[6] Mikhail J. Atallah. *Algorithms and theory of computation handbook*. CRC Press, 1999.

[7] Lukas Bachus. *Ply*. Bachelor thesis, University of Tübingen, 2016.

[8] Alexandru T Balaban. Applications of graph theory in chemistry. *Journal of chemical information and computer sciences*, 25(3):334–343, 1985.

[9] Michael A. Bekos, Michael Kaufmann, and Chrysanthi N. Raftopoulou. On optimal 2- and 3-planar graphs. In Boris Aronov and Matthew J. Katz, editors, *33rd International Symposium on Computational Geometry, SoCG 2017, July 4-7, 2017, Brisbane, Australia*, volume 77 of *LIPIcs*, pages 16:1–16:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[10] Jon Louis Bentley and Thomas Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Trans. Computers*, 28(9):643–647, 1979.

[11] Danail Bonchev. *Chemical graph theory: introduction and fundamentals*, volume 1. CRC Press, 1991.

[12] Stephen P Borgatti, Ajay Mehra, Daniel J Brass, and Giuseppe Labianca. Network analysis in the social sciences. *science*, 323(5916):892–895, 2009.

[13] Prosenjit Bose, Giuseppe Di Battista, William Lenhart, and Giuseppe Liotta. Proximity constraints and representable trees. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 10-12, 1994, Proceedings*, volume 894 of *Lecture Notes in Computer Science*, pages 340–351. Springer, 1994.

[14] Prosenjit Bose, William Lenhart, and Giuseppe Liotta. Characterizing proximity trees. *Algorithmica*, 16(1):83–110, 1996.

[15] Franz J. Brandenburg. 1-visibility representations of 1-planar graphs. *J. Graph Algorithms Appl.*, 18(3):421–438, 2014.

[16] Franz J. Brandenburg, Niklas Heinsohn, Michael Kaufmann, and Daniel Neuwirth. On bar (1, j)-visibility graphs - (extended abstract). In M. Sohel Rahman and Etsuji Tomita, editors, *WALCOM: Algorithms and Computation - 9th International Workshop, WALCOM 2015, Dhaka, Bangladesh, February 26-28, 2015. Proceedings*, volume 8973 of *Lecture Notes in Computer Science*, pages 246–257. Springer, 2015.

[17] Ulrik Brandes, Markus Eiglsperger, Jürgen Lerner, and Christian Pich. Graph markup language (graphml). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 517–541. Chapman and Hall/CRC, 2013.

[18] Heinz Breu and David G. Kirkpatrick. Unit disk graph recognition is np-hard. *Comput. Geom.*, 9(1-2):3–24, 1998.

[19] Till Bruckdorfer, Sabine Cornelsen, Carsten Gutwenger, Michael Kaufmann, Fabrizio Montecchiani, Martin Nöllenburg, and Alexander Wolff. Progress on partial edge drawings. In Walter Didimo and Maurizio Patrignani, editors, *Graph Drawing - 20th International Symposium, GD 2012, Redmond, WA, USA, September 19-21, 2012, Revised Selected Papers*, volume 7704 of *Lecture Notes in Computer Science*, pages 67–78. Springer, 2012.

[20] Till Bruckdorfer, Sabine Cornelsen, Carsten Gutwenger, Michael Kaufmann, Fabrizio Montecchiani, Martin Nöllenburg, and Alexander Wolff. Progress on partial edge drawings. *J. Graph Algorithms Appl.*, 21(4):757–786, 2017.

[21] Till Bruckdorfer and Michael Kaufmann. Mad at edge crossings? break the edges! In Evangelos Kranakis, Danny Krizanc, and Flaminia L. Luccio, editors, *Fun with Algorithms - 6th International Conference, FUN 2012, Venice, Italy, June 4-6, 2012. Proceedings*, volume 7288 of *Lecture Notes in Computer Science*, pages 40–50. Springer, 2012.

[22] Till Martin Bruckdorfer. *Schematics of Graphs and Hypergraphs*. PhD thesis, University of Tübingen, 2016.

[23] Michael Burch, Robin Woods, Rudolf Netzel, and Daniel Weiskopf. The challenges of designing metro maps. In Nadia Magnenat-Thalmann, Paul Richard, Lars Linsen, Alexandru Telea, Sebastiano Battiato, Francisco H. Imai, and José Braz, editors, *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2016) - Volume 2: IVAPP, Rome, Italy, February 27-29, 2016.*, pages 197–204. SciTePress, 2016.

[24] Paul Butler. Visualizing facebook friends. 2010. `https://paulbutler.org/archives/visualizing-facebook-friends/`.

[25] Sergio Cabello and Bojan Mohar. Adding one edge to planar graphs makes crossing number and 1-planarity hard. *SIAM J. Comput.*, 42(5):1803–1829, 2013.

[26] Markus Chimani, Stefan Felsner, Stephen G. Kobourov, Torsten Ueckerdt, Pavel Valtr, and Alexander Wolff. On the maximum crossing number. *J. Graph Algorithms Appl.*, 22(1):67–87, 2018.

[27] Markus Chimani, Carsten Gutwenger, Michael Jünger, Gunnar W. Klau, Karsten Klein, and Petra Mutzel. The open graph drawing framework

(OGDF). In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 543–569. Chapman and Hall/CRC, 2013.

[28] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition.* Springer, 2008.

[29] Hubert de Fraysseix, János Pach, and Richard Pollack. How to draw a planar graph on a grid. *Combinatorica*, 10(1):41–51, 1990.

[30] Felice De Luca, Emilio Di Giacomo, Walter Didimo, Stephen G. Kobourov, and Giuseppe Liotta. An experimental study on the ply number of straight-line drawings. In Sheung-Hung Poon, Md. Saidur Rahman, and Hsu-Chun Yen, editors, *WALCOM 2017*, volume 10167 of *Lecture Notes in Computer Science*, pages 135–148. Springer, 2017.

[31] Alice M. Dean, William S. Evans, Ellen Gethner, Joshua D. Laison, Mohammad Ali Safari, and William T. Trotter. Bar k-visibility graphs. *J. Graph Algorithms Appl.*, 11(1):45–59, 2007.

[32] B. Delaunay. Sur la sphère vide. a la mémoire de Georges Voronoi. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et naturelles*, 6:793 – 800, 1934.

[33] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom.*, 4:235–282, 1994.

[34] Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs.* Prentice-Hall, 1999.

[35] Giuseppe Di Battista, Giuseppe Liotta, and Sue Whitesides. The strength of weak proximity. *J. Discrete Algorithms*, 4(3):384–400, 2006.

[36] Emilio Di Giacomo, Walter Didimo, Seok-Hee Hong, Michael Kaufmann, Stephen G. Kobourov, Giuseppe Liotta, Kazuo Misue, Antonios Symvonis, and Hsu-Chun Yen. Low ply graph drawing. In Nikolaos G. Bourbakis, George A. Tsihrintzis, and Maria Virvou, editors, *6th International Conference on Information, Intelligence, Systems and Applications, IISA 2015, Corfu, Greece, July 6-8, 2015*, pages 1–6. IEEE, 2015.

[37] Walter Didimo, Peter Eades, and Giuseppe Liotta. Drawing graphs with right angle crossings. *Theor. Comput. Sci.*, 412(39):5156–5166, 2011.

[38] Verkehrsverbund Neckar Alb Donau. 2018. `https://www.naldo.de/fileadmin/media/download/teilnetzplaene/2018-Tuebingen.pdf`.

[39] Christian A. Duncan and Michael T. Goodrich. Planar orthogonal and polyline drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 223–246. Chapman and Hall/CRC, 2013.

[40] Tim Dwyer, Bongshin Lee, Danyel Fisher, Kori Inkpen Quinn, Petra Isenberg, George G. Robertson, and Chris North. A comparison of user-generated and automatic graph layouts. *IEEE Trans. Vis. Comput. Graph.*, 15(6):961–968, 2009.

[41] Peter Eades and Sue Whitesides. The logic engine and the realization problem for nearest neighbor graphs. *Theor. Comput. Sci.*, 169(1):23–37, 1996.

[42] David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In Walid G. Aref, Mohamed F. Mokbel, and Markus Schneider, editors, *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, page 16. ACM, 2008.

[43] MARZIEH ESKANDARI and MAHDIEH YEGANEH. S-visibility problem in vlsi chip design. *Turkish Journal of Electrical Engineering & Computer Sciences*, 25(5):3960–3969, 2017.

[44] William S. Evans, Michael Kaufmann, William Lenhart, Giuseppe Liotta, Tamara Mchedlidze, and Stephen K. Wismath. Bar 1-visibility graphs and their relation to other nearly planar graphs. *CoRR*, abs/1312.5520, 2013.

[45] Sándor P. Fekete, Michael E. Houle, and Sue Whitesides. The wobbly logic engine: Proving hardness of non-rigid geometric graph representation problems. In Giuseppe Di Battista, editor, *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*, volume 1353 of *Lecture Notes in Computer Science*, pages 272–283. Springer, 1997.

[46] Arne Frick, Andreas Ludwig, and Heiko Mehldau. A fast adaptive layout algorithm for undirected graphs. In Roberto Tamassia and Ioannis G. Tollis, editors, *Graph Drawing, DIMACS International Workshop, GD '94, Princeton, New Jersey, USA, October 10-12, 1994, Proceedings*, volume 894 of *Lecture Notes in Computer Science*, pages 388–403. Springer, 1994.

[47] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Softw., Pract. Exper.*, 21(11):1129–1164, 1991.

[48] K Ruben Gabriel and Robert R Sokal. A new statistical approach to geographic variation analysis. *Systematic zoology*, 18(3):259–278, 1969.

[49] Emden R. Gansner, Yehuda Koren, and Stephen C. North. Graph drawing by stress majorization. In János Pach, editor, *Graph Drawing, 12th International Symposium, GD 2004, New York, NY, USA, September 29 - October 2, 2004, Revised Selected Papers*, volume 3383 of *Lecture Notes in Computer Science*, pages 239–250. Springer, 2004.

[50] M. R. Garey, David S. Johnson, and H. C. So. An application of graph coloring to printed circuit testing (working paper). In *16th Annual Symposium on Foundations of Computer Science, Berkeley, California, USA, October 13-15, 1975*, pages 178–183. IEEE Computer Society, 1975.

[51] Jacob E. Goodman and Joseph O'Rourke, editors. *Handbook of Discrete and Computational Geometry, Second Edition*. Chapman and Hall/CRC, 2004.

[52] Alexander Grigoriev and Hans L. Bodlaender. Algorithms for graphs embeddable with few crossings per edge. *Algorithmica*, 49(1):1–11, 2007.

[53] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In János Pach, editor, *Graph Drawing, 12th International Symposium, GD 2004, New York, NY, USA, September 29 - October 2, 2004, Revised Selected Papers*, volume 3383 of *Lecture Notes in Computer Science*, pages 285–295. Springer, 2004.

[54] Stefan Hachul and Michael Jünger. Large-graph layout algorithms at work: An experimental study. *J. Graph Algorithms Appl.*, 11(2):345–369, 2007.

[55] Stephen G. Hartke, Jennifer Vandenbussche, and Paul S. Wenger. Further results on bar k-visibility graphs. *SIAM J. Discrete Math.*, 21(2):523–531, 2007.

[56] Niklas Heinsohn and Michael Kaufmann. An interactive tool to explore and improve the ply number of drawings. In Fabrizio Frati and Kwan-Liu Ma, editors, *Graph Drawing and Network Visualization - 25th International Symposium, GD 2017, Boston, MA, USA, September 25-27, 2017, Revised Selected Papers*, volume 10692 of *Lecture Notes in Computer Science*, pages 38–51. Springer, 2017.

[57] Michael Himsolt. Gml: A portable graph file format. *Html page under http://www. fmi. uni-passau. de/graphlet/gml/gml-tr. html, Universität Passau*, 1997.

[58] Weidong Huang, Seok-Hee Hong, and Peter Eades. Effects of crossing angles. In *PacificVis*, pages 41–46, 2008.

[59] Weidong Huang and Mao Lin Huang. Exploring the relative importance of crossing number and crossing angle. In Guozhong Dai, Kang Zhang, Mao Lin Huang, Hongan Wang, Xiaoru Yuan, Linmi Tao, and Wei Chen, editors, *2010 International Symposium on Visual Information Communication, VINCI '10, Beijing, China - September 28 - 29, 2010*, page 10. ACM, 2010.

[60] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Process. Lett.*, 31(1):7–15, 1989.

[61] Jyrki Katajainen and Olli Nevalainen. Computing relative neighbourhood graphs in the plane. *Pattern Recognition*, 19(3):221–228, 1986.

[62] Stephen G. Kobourov. Force-directed drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 383–408. Chapman and Hall/CRC, 2013.

[63] Stephen G. Kobourov, Sergey Pupyrev, and Bahador Saket. Are crossings important for drawing large graphs? In Christian A. Duncan and Antonios Symvonis, editors, *Graph Drawing - 22nd International Symposium, GD 2014, Würzburg, Germany, September 24-26, 2014, Revised Selected Papers*, volume 8871 of *Lecture Notes in Computer Science*, pages 234–245. Springer, 2014.

[64] Vladimir P. Korzhik and Bojan Mohar. Minimal obstructions for 1-immersions and hardness of 1-planarity testing. *Journal of Graph Theory*, 72(1):30–71, 2013.

[65] Giuseppe Liotta. Proximity drawings. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 115–154. Chapman and Hall/CRC, 2013.

[66] John M Lowenstein. *Citric acid cycle.* Elsevier, 1969.

[67] David W Matula and Robert R Sokal. Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane. *Geographical analysis*, 12(3):205–222, 1980.

[68] Andreas Noack. Energy models for graph clustering. *J. Graph Algorithms Appl.*, 11(2):453–480, 2007.

[69] Atsuyuki Okabe, Toshiaki Satoh, Takehiro Furuta, Atsuo Suzuki, and K. Okano. Generalized network voronoi diagrams: Concepts, computational methods, and applications. *International Journal of Geographical Information Science*, 22(9):965–994, 2008.

[70] Ralph HJM Otten. Graph representations in interactive layout design. In *IEEE Internat. Symp. on Circuits and Systems, 1978*, pages 914–918, 1978.

[71] János Pach and Géza Tóth. Graphs drawn with few crossings per edge. *Combinatorica*, 17(3):427–439, 1997.

[72] Helen C. Purchase. Which aesthetic has the greatest effect on human understanding? In Giuseppe Di Battista, editor, *Graph Drawing, 5th International Symposium, GD '97, Rome, Italy, September 18-20, 1997, Proceedings*, volume 1353 of *Lecture Notes in Computer Science*, pages 248–261. Springer, 1997.

[73] Helen C. Purchase, Robert F. Cohen, and Murray I. James. Validating graph drawing aesthetics. In Franz-Josef Brandenburg, editor, *Graph Drawing, Symposium on Graph Drawing, GD '95, Passau, Germany, September 20-22, 1995, Proceedings*, volume 1027 of *Lecture Notes in Computer Science*, pages 435–446. Springer, 1995.

[74] Helen C. Purchase, Christopher Pilcher, and Beryl Plimmer. Graph drawing aesthetics - created by users, not algorithms. *IEEE Trans. Vis. Comput. Graph.*, 18(1):81–92, 2012.

[75] Helen C. Purchase, Beryl Plimmer, Rosemary Baker, and Christopher Pilcher. Graph drawing aesthetics in user-sketched graph layouts. In Christof Lutteroth and Paul R. Calder, editors, *User Interfaces 2010, AUIC 2010, Brisbane, Australia, January 2010*, volume 106 of *CRPIT*, pages 80–88. Australian Computer Society, 2010.

[76] Gerhard Ringel. Ein sechsfarbenproblem auf der kugel. In *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, volume 29, pages 107–117. Springer, 1965.

[77] Pierre Rosenstiehl and Robert Endre Tarjan. Rectilinear planar layouts and bipolar orientations of planar graphs. *Discrete & Computational Geometry*, 1:343–353, 1986.

[78] Martine D. F. Schlag, Yuh-Zen Liao, and C. K. Wong. An algorithm for optimal two-dimensional compaction of VLSI layouts. *Integration*, 1(2-3):179–209, 1983.

[79] Walter Schnyder. Embedding planar graphs on the grid. In David S. Johnson, editor, *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms, 22-24 January 1990, San Francisco, California, USA.*, pages 138–148. SIAM, 1990.

[80] Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *17th Annual Symposium on Foundations of Computer Science, Houston, Texas, USA, 25-27 October 1976*, pages 208–215. IEEE Computer Society, 1976.

[81] Janet M. Six and Ioannis G. Tollis. Circular drawing algorithms. In Roberto Tamassia, editor, *Handbook on Graph Drawing and Visualization.*, pages 285–315. Chapman and Hall/CRC, 2013.

[82] Qiao Su, Tianbing Guan, and Haitao Lv. Siderophore biosynthesis coordinately modulated the virulence-associated interactive metabolome of uropathogenic escherichia coli and human urine. *Scientific reports*, 6:24099, 2016.

[83] Shaheena Sultana, Md. Saidur Rahman, Arpita Roy, and Suraiya Tairin. Bar 1-visibility drawings of 1-planar graphs. In Prosenjit Gupta and Christos D. Zaroliagis, editors, *Applied Algorithms - First International Conference, ICAA 2014, Kolkata, India, January 13-15, 2014. Proceedings*, volume 8321 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2014.

[84] Graph Drawing Symposium. Rome graphs. 2008. `http://www.graphdrawing.org`.

[85] Roberto Tamassia, editor. *Handbook on Graph Drawing and Visualization.* Chapman and Hall/CRC, 2013.

[86] Roberto Tamassia and Giuseppe Liotta. Graph drawing. In Jacob E. Goodman and Joseph O'Rourke, editors, *Handbook of Discrete and Computational Geometry, Second Edition.*, pages 1163–1185. Chapman and Hall/CRC, 2004.

[87] Roberto Tamassia and Ioannis G. Tollis. A unified approach a visibility representation of planar graphs. *Discrete & Computational Geometry*, 1:321–341, 1986.

[88] Robert Endre Tarjan. *Data structures and network algorithms*, volume 44. Siam, 1983.

[89] Mikko Tommila. A C++ high performance arbitrary precision arithmetic package, 2003.

[90] Godfried T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern Recognition*, 12(4):261 – 268, 1980.

[91] Nenad Trinajstic. *Chemical graph theory.* Routledge, 2018.

[92] Saraswathi Vishveshwara, KV Brinda, and N Kannan. Protein structure: insights from graph theory. *Journal of Theoretical and Computational Chemistry*, 1(01):187–211, 2002.

[93] Jiri Vlach and Kishore Singhal. *Computer Methods for Circuit Analysis and Design*. John Wiley & Sons, Inc., New York, NY, USA, 1983.

[94] Colin Ware, Helen C. Purchase, Linda Colpoys, and Matthew McGill. Cognitive measurements of graph aesthetics. *Information Visualization*, 1(2):103–110, 2002.

[95] Emo Welzl, Giuseppe Di Battista, Ashim Garg, Giuseppe Liotta, Roberto Tamassia, Emanuele Tassinari, and Francesco Vargiu. An experimental comparison of four graph drawing algorithms. *Comput. Geom.*, 7:303–325, 1997.

[96] Max Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische forschung*, 4(1):301–350, 1923.

[97] Roland Wiese, Markus Eiglsperger, and Michael Kaufmann. *yFiles* - visualization and automatic layout of graphs. In *Graph Drawing Software*, pages 173–191. 2004.

[98] Stephen K. Wismath. Characterizing bar line-of-sight graphs. In Joseph O'Rourke, editor, *Proceedings of the First Annual Symposium on Computational Geometry, Baltimore, Maryland, USA, June 5-7, 1985*, pages 147–152. ACM, 1985.

[99] yWorks. yfiles for java developer's guide. 2018. `docs.yworks.com/yfiles/doc/developers-guide/index.html`.

Index