Higher-Degree Orthogonal Graph Drawing with Flexibility Constraints

Bachelor Thesis of

Guido Brückner

At the Department of Informatics
Institute of Theoretical Computer Science

Reviewers:	Prof. Dr. Dorothea Wagner
	Prof. Dr. Peter Sanders
Advisors:	Thomas Bläsius
	Dr. Ignaz Rutter

Time Period: June 1st 2013 to September 30th 2013.

Statement of Authorship

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

Karlsruhe, September 30, 2013

Abstract

Much work on orthogonal graph drawing has focused on 4-*planar* graphs, that is planar graphs where all vertices have maximum degree 4. In this work, we study aspects of the *Kandinsky model*, which is a model for orthogonal graph drawings of higher-degree graphs.

First, we examine the decision problem β-EMBEDDABILITY, which asks whether for a given planar graph with a fixed or variable embedding, a drawing in the Kandinsky model exists where every edge has at most β bends. We show that 0-EMBEDDABILITY in the Kandinsky model is equivalent to 0-EMBEDDABILITY in the lower-degree case. We show that 1-EMBEDDABILITY for multigraphs with variable planar embeddings is $\mathcal{NP}$-complete. Then, we show that any simple graph is 1-embeddable, even if it has a fixed planar embedding, and we present a linear-time algorithm for finding a corresponding 1-bend drawing. Furthermore, we show to find a 2-bend Kandinsky drawing of any plane graph in linear time.

Next, we study some restrictions of the bend minimization problem OPTIMALKANDINSKYDRAW, which finds a Kandinsky drawing with the minimum number of bends for a plane graph. We present a linear-time algorithm solving OPTIMALKANDINSKYDRAW for biconnected, outerplanar, inner-triangulated graphs. Then, we give an $\mathcal{O}(n^3)$ time algorithm for finding bend-minimal 1-bend Kandinsky drawings of series-parallel graphs.

Finally, we inquire into new ways to solve OPTIMALKANDINSKY-DRAW using linear programming. We show that for any constant $c \in \mathbb{R}$ a graph exists so that the difference between the real solution and the integer solution of the corresponding linear program is greater than c.

Acknowledgements

I would like to express my sincere gratitude to my advisors Thomas Bläsius and Dr. Ignaz Rutter for their support, guidance and patience, and for the interesting discussions and helpful corrections which played a key role in the development of this work. I would like to extend my appreciation to Prof. Dr. Dorothea Wagner for giving me the opportunity of writing this thesis at her institute.

Contents

1 Introduction

Graphs consist of a set of vertices and a set of edges, each connecting two vertices. Graphs can be used to represent a variety of data, models and systems. For example, graphs are used to model geographic maps, social networks, or software dependencies.

Oftentimes when working with data, it is helpful to visualize it. A graph drawing is a visualization of a graph, where vertices are usually drawn as simple geometric shapes such as circles or rectangles and edges are usually drawn as curves. Graph drawing is a field of theoretical computer science which deals with the automated generation of such graph drawings.

In general, in a graph drawing, vertices and edges may have any shape. For example, vertices might be drawn as polygons of varying size and an edge might be represented as a sequence of arcs, geometric lines or even strings of geometric shapes. Sometimes it is useful to restrict vertex shapes and edge shapes. Orthogonal graph drawing deals with the automated generation of graph drawings where all vertices are represented as squares of equal size and all edges are sequences of horizontal and vertical line segments. In this work, we focus on *planar* graphs, that is graphs which may be drawn in the plane so that edges only intersect at their endpoints. Graphs which are not planar may be *planarized* using some planarization method (e.g. [12]). Orthogonal graph drawings have a number of applications [18], e.g. for PERT charts in project management, for UML diagrams in software engineering, for entity-relationship models in database modeling, for VLSI circuit design [13], [16] and in architecture for floor plan layouts [14].

It is of interest to find *nice* orthogonal graph drawings. What makes a drawing nice is, of course, subject to debate. For example, one might say that a drawing is nice if it can be enclosed in a small area, or if all faces have approximately equal size, or if it has some other aesthetic qualities. In this work, we focus on nuances of minimizing the total number of bends in a drawing.

1.1 Related Work

A restriction often used in orthogonal graph drawing is to assume that the graphs are 4-*planar*, which means that they are planar and all vertices have maximum degree 4.

Additionally, only one edge may be attached to each side of a square vertex. Given this restriction, orthogonal graph drawing is a well-studied problem. Tamassia [15] shows that for a *plane* graph (that is a graph together with a fixed embedding), a drawing with the minimum number of bends can be found in $\mathcal{O}(n^2 \log n)$ time by transforming the bend minimization problem into a minimum cost flow problem. Furthermore, Tamassia and Garg [9] show that the problem of deciding whether for a planar graph there exists an embedding and a corresponding orthogonal graph drawing so that no edge is bent is $\mathcal{NP}$-complete.

Bläsius et al. [4] introduce the generalized problem FlexDraw for 4-planar graphs $G = (V, E)$. In a FlexDraw instance, a function $\text{flex}\colon E \to \mathbb{N}_0$ assigns a flexibility to each edge. The problem is to decide whether there exists an embedding and an orthogonal drawing of G so that every edge e is bent at most $\text{flex}(e)$ times. For positive flexibilities, that is $\text{flex}\colon E \to \mathbb{N}_+$, FlexDraw has been shown to be solvable in $\mathcal{O}(n^2)$ time (this can be optimized even further). As a special case, FlexDraw contains the problem β-Embeddability, where for some $\beta \in \mathbb{N}_0$ every edge e is assigned a flexibility of $\text{flex}(e) = \beta$.

All of these contributions focus on 4-planar graphs. However, many graphs have *higher-degree* vertices, that is vertices v with $\deg(v) > 4$. If vertices should be represented by squares and edges should be represented by orthogonal line segments, a higher-degree vertex has to have at least one side to which more than one edge is attached.

A model for such higher-degree orthogonal graph drawings is the *Kandinsky model*. Drawings in the Kandinsky model, also called *Kandinsky drawings*, have been introduced by Fößmeier and Kaufmann [7] as *planar orthogonal drawings with equal vertex size and non-empty faces* or *podevsnef*.

Fößmeier and Kaufmann [7] claimed to have found a polynomial time algorithm to find bend-minimal Kandinsky drawings. However, Eiglsperger [5] later showed that the presented algorithm is incorrect. It remains unknown whether a polynomial time algorithm for finding Kandinsky drawings with the minimum number of bends exists, even for graphs with a fixed planar embedding. Eiglsperger [5] and Yildiz [18] present approximation algorithms for finding bend-minimal Kandinsky drawings. Both algorithms have an approximation ratio of 2.

1.2 Outline

In this work, we study various aspects of Kandinsky drawings and their automated generation.

After going over some preliminaries in Chapter 2, we first constrain the flexibility of the edges in Chapter 3. Specifically, we consider some restrictions of the decision problem β-Embeddability for $\beta \in \{0, 1, 2\}$. We see that 0-Embeddability in the Kandinsky model is equivalent to the lower-degree case of 4-planar graphs and that we can reuse the existing results for 4-planar graphs. Specifically, 0-Embeddability with variable embeddings is $\mathcal{NP}$-complete. With a fixed embedding, 0-Embeddability is solvable in polynomial time and if a plane graph is 0-embeddable, we can find a 0-bend drawing in polynomial time. For 1-Embeddability and 2-Embeddability, linear-time algorithms for finding 1-bend Kandinsky drawings and 2-bend Kandinsky drawings, respectively (though not bend-minimal), are presented. Furthermore, we show that 1-Embeddability for graphs with variable embeddings is $\mathcal{NP}$-complete.

In Chapter 4, we study Kandinsky drawings of graphs from restricted graph classes. Specifically, we consider the bend minimization problem OptimalKandinskyDraw for biconnected, outerplanar inner-triangulated graphs. We show how to find an optimal drawing of such a graph in linear time and prove that it has $n-2$ bends. For series-parallel graphs we consider the bend minimization problem 1-OptimalKandinskyDraw, which finds a bend-minimal 1-bend Kandinsky drawing, and give a solution algorithm running in $\mathcal{O}(n^3)$ time.

Finally, in Chapter 5, we refocus on solving OptimalKandinskyDraw for general graphs and explore new solution approaches employing mixed integer linear programming. We show that for any constant $c \in \mathbb{R}$, we can find a graph so that the difference between the real solution and the integer solution of the corresponding linear program is greater than c.

2 Preliminaries

In this chapter, we cover some basic notions and definitions. A *graph* $G = (V, E)$ is a tuple of a set of vertices V and a set of edges $E \subseteq V \times V$. A *simple graph* has at most one edge between each pair of vertices. A *multigraph* is a graph which may have one, but also more edges between the same pair of vertices. Such edges are called *multiedges.*

A graph is *connected* if we can find a consecutive path between any two vertices. A *separator* of a connected graph $G = (V, E)$ is a set $S \subseteq V$ so that the graph $(G \setminus S) = (V', E')$, where $V' = V \setminus S$ and $E' = \{(u, v) \in E \colon \{u, v\} \cap V' = \emptyset\}$ is no longer connected. We call a separator $s \in S$ with $|S| = 1$ a *cutvertex* or a *separator vertex* and we call a separator S with $|S| = 2$ a *separator pair.* If a graph does not have a separator vertex, but it does have a separator pair, the graph is *biconnected.*

A *planar graph* is a graph which can be drawn in the plane so that no two edges cross each other. If we delete all edges from such a drawing, the plane is decomposed into a number of disjoint regions, called *faces.* Only one face is unbounded: This face is called the *outer* face. All other faces are bounded and are called *inner* faces. Consider two drawings of the same graph: Two faces – one from each drawing – are considered equal if the ordered cycle of edges incident to the faces are equal. By F, denote the set of faces of the first drawing and by F', denote the set of faces of the second drawing. If there is a bijective binary relation $\sim\ \subseteq F \times F'$ between the faces of the drawings so that for two faces $f \in F$, $f' \in F'$ from $f \sim f'$ it follows that f and f' are equal, the drawings have the same *(combinatorial) embedding.* A planar graph together with a planar embedding is called a *plane* graph. With $\text{faces}(G)$, we denote the set of faces of a plane graph $G = (V, E)$, with $\text{faces}(v)$ we denote the set of faces incident to a vertex $v \in V$, with $\text{faces}(e)$ we denote the set of faces incident to an edge $e \in E$ and with $\text{faces}(f)$ we denote the set of faces adjacent to a face $f \in \text{faces}(G)$. The *contour* of a face f is the subgraph of vertices and edges incident to f.

A plane graph is called *outerplanar* if all vertices are incident to the outer face. A graph G is called *maximal planar* if inserting an edge between any two vertices in G leads to G no longer being planar. A maximal planar graph is also said to be *triangulated.* A plane graph G is called inner-triangulated if all inner faces of G are incident to exactly three vertices.

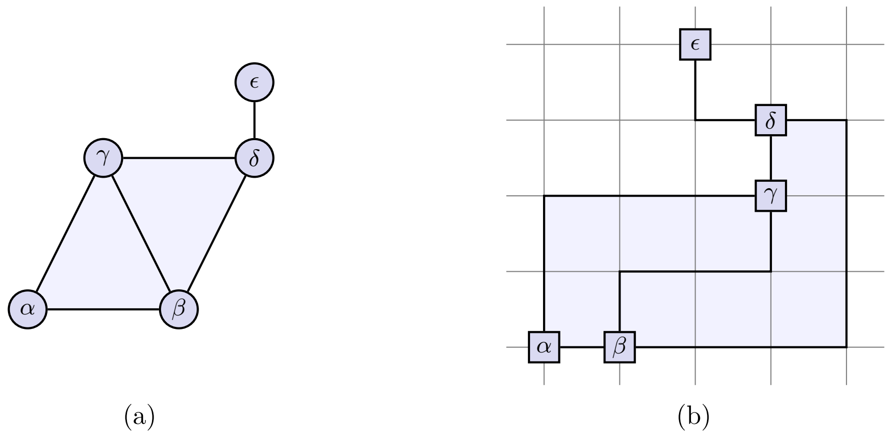


Figure 2.1: An example graph G in (a) and an orthogonal drawing of G in (b). Graph edges are drawn as thick black lines and grid lines are drawn as thin, gray lines.

2.1 The Tamassia Model

Tamassia [15] presents and studies a model for orthogonal graph drawings of graphs with maximum degree 4. In this model, all vertices are placed on the points of a uniform grid. They have equal, square size. Edges are represented by series of horizontal and vertical segments which lie on the grid. They may not touch or cross each other. An edge e only touches a vertex v if v is one of the two endpoints of e.

For an example of an orthogonal graph drawing in the Tamassia model, see Figure 2.1.

2.1.1 Orthogonal Representation

To formally describe an orthogonal drawing of a graph $G = (V, E)$, Tamassia uses an *orthogonal representation*, which is a set $H = \{H_{f_1}, H_{f_2}, \ldots\}$ of *face descriptions*. Every face description H_f is a circularly ordered list $H_f = (h_{e_1}, h_{e_2}, \ldots)$ of *edge descriptions* h_{e_i}. The edge descriptions appear in the order in which they area encountered when going around the contour of f in the "positive" direction, i.e. having the face at one's right. An edge description $h_e \in H_f$ is a triple $h_e = (e, s, a)$ where:

- $e \in E$ is an edge of G.
- s is a binary string. The k-th bit of s represents the k-th bend of e, as it is encountered on the right when going around f in the positive direction. The binary symbols `0` and `1` denote an angle of 90° and 270°, respectively. The empty word ϵ describes an edge with no bends.
- a is an integer in the set $\{90, 180, 270, 360\}$ specifying the angle in f formed by e and its successor edge.

When working with orthogonal representations, a useful tool is the notion of *rotations*. For a directed edge e, we define its rotation $\mathrm{rot}(e)$ as $\mathrm{rot}(e) = \mathrm{zeroes}(s) - \mathrm{ones}(s)$, where

s is the bitstring from the edge description $h_e = (e, s, a) \in H_f$, and f is the face to the right of e. If $e = (u, v)$ and $e' = (v, w)$ are two directed edges so that the face f lies to the right of both e and e', and h_e precedes $h_{e'}$ in H_f, we define the rotation $\text{rot}(e, e') = 2 - a/90$, where a is the value from the edge description $h_e = (e, s, a) \in H_f$. If $\pi = (e_1, e_2, \ldots, e_k)$ is a path in the contour of some face f, that is if all pairs h_{e_i} and $h_{e_{i+1}}$ for $1 \leq i < k$ are successors in H_f, we define the rotation of π as follows:

$$\text{rot}(\pi) = \sum_{i=1}^{k-1} (\text{rot}(e_i) + \text{rot}(e_i, e_{i+1})) + \text{rot}(e_k)$$

Tamassia shows that for any valid orthogonal representation a corresponding orthogonal graph drawing exists [15, Theorem 1]. An orthogonal representation is *valid* if it meets certain criteria:

1. There is an orthogonal graph drawing D of some 4-planar graph so that the orthogonal representation describes D.

2. Each edge must have consistent descriptions in the faces in which it appears. More formally, let $e \in E$ be an edge of G and $\{f, f'\} = \text{faces}(e)$. Let $h_e = (e, s, a) \in H_f$ and $h'_e = (e, s', a') \in H_{f'}$ be the two edge descriptions of e. The string s' can be obtained from s by reversing s and exchanging zeroes and ones.

3. Every face described by H must be a rectilinear polygon. More formally, if for a face f the path $\pi_f = (e_1, e_2, \ldots, e_k)$ denotes the contour path of f in positive direction, the following equation must be true for all faces f:

$$\text{rot}(\pi_f) = \begin{cases} +4 & \text{if } f \text{ is an inner face} \\ -4 & \text{if } f \text{ is the outer face} \end{cases}$$

4. For each vertex v, the sum of angles between pairs of neighboring edges is equal to $360°$. More formally, if $(e_1, e_2, \ldots, e_k, e_{k+1} = e_1, \ldots)$ denotes the cyclic list of edges incident to v in counter-clockwise order, the following equation has to hold true:

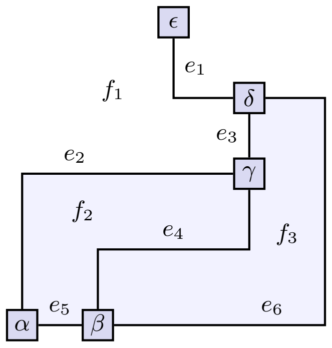
$$\sum_{i=1}^{k} (2 - \text{rot}(e_1, e_2)) = 4$$

For an example of a (valid)[1] orthogonal representation, see Figure 2.2. A plane graph together with an orthogonal representation is called a *configuration*.

2.1.2 The Tamassia Flow Network

Tamassia [15] transforms the problem of computing a bend-minimized drawing of a plane graph $G = (V, E)$ into a minimum cost flow problem, which is solvable in polynomial

[1] In the following, we will not explicitely state that a valid orthogonal representation is valid.



$$\begin{aligned}
H_{f_1} &= ((e_1, \texttt{1}, 90), (e_3, \epsilon, 90), (e_2, \texttt{1}, 270), (e_5, \epsilon, 180), (e_6, \texttt{11}, 180), (e_1, \texttt{0}, 360))\\
H_{f_2} &= ((e_2, \texttt{0}, 90), (e_4, \texttt{01}, 90), (e_5, \epsilon, 90))\\
H_{f_3} &= ((e_3, \epsilon, 90), (e_6, \texttt{00}, 90), (e_4, \texttt{01}, 180))
\end{aligned}$$

Figure 2.2: An orthogonal drawing and its orthogonal representation $H = \{H_{f_1}, H_{f_2}, H_{f_3}\}$ of the graph G shown in Figure 2.1a.

time. To this end, he shows how to transform a flow into an orthogonal representation and vice versa.

The *Tamassia flow network* $T = (N, A, \mathrm{dem}, \mathrm{cap}, \mathrm{cost})$ is constructed as follows. By $F = \mathrm{faces}(G)$, denote the set of faces of G. By N, denote the set of nodes $N = N_V \cup N_F$ in the flow network, where:

$$\begin{aligned}
N_V &= \{n_v \mid v \in V\} && (\textit{vertex nodes})\\
N_F &= \{n_f \mid f \in F\} && (\textit{face nodes})
\end{aligned}$$

By $A = A_{VF} \cup A_{FF}$, we denote the set of directed arcs in the flow network, where A_{VF} are arcs from vertex nodes to face nodes and A_{FF} are arcs between face nodes:

$$\begin{aligned}
A_{VF} &= \bigcup_{v \in V} \{(n_v, n_f) \mid f \in \mathrm{faces}(v)\}\\
A_{FF} &= \bigcup_{f \in F} \{(n_f, n_{f'}) \mid f' \in \mathrm{faces}(f)\}
\end{aligned}$$

If k units flow over an arc $a = (n_v, n_f) \in A_{VF}$, a $(k+1) \cdot 90°$-angle is generated in f at v. If k units flow over an arc $a = (n_f, n_{f'} \in A_{FF}$, the edge e with $\mathrm{faces}(e) = \{f, f'\}$ will have a $k \cdot 90°$-bend in f.

By $\mathrm{dem}\colon A \to \mathbb{Z}$ we denote the *demand function*, which is defined as follows:

$$\begin{aligned}
&\forall n_v \in N_V\colon \mathrm{dem}(n_v) = \deg(n_v) - 4\\
&\forall n_f \in N_F\colon \mathrm{dem}(n_f) = \begin{cases} \mathrm{size}(f) - 4 & \text{if } f \text{ is an inner face}\\ \mathrm{size}(f) + 4 & \text{if } f \text{ is an outer face} \end{cases}
\end{aligned}$$

Note that the definition of the demand function correlates with criterium 3 of valid orthogonal representations, as described above.

By cap: $A \to \{3, \infty\}$ we denote the *capacity function*, which is defined as follows:

$$\forall a \in A\colon \operatorname{cap}(a) = \begin{cases} 3 & a \in A_{VF} \\ \infty & a \in A_{FF} \end{cases}$$

The capacity of 3 for arcs in A_{VF} is based on the fact that flow over an arc $a = (n_v, n_f) \in A_{VF}$ causes a $(k+1) \cdot 90°$-angle. Since that angle obviously cannot be greater than 360°, the flow must not be greater than 3. The capacity of ∞ for arcs in A_{VF} is founded in the fact that edges may be bent an arbitrary number of times.

By cost: $A \to \{0, 1\}$ we denote the *cost function*, which is defined as follows:

$$\forall a \in A\colon \operatorname{cost}(a) = \begin{cases} 0 & a \in A_{VF} \\ 1 & a \in A_{FF} \end{cases}$$

We try to minimize the number of bends: The cost of 1 for arcs in A_{FF} is based on the fact that one unit of flow over an arc in A_{FF} causes one bend in an edge in the drawing. Flow over arcs in A_{VF} does not cause bends in edges, so the cost for those arcs is 0.

A valid flow in T is a function flow: $A \to \mathbb{N}_0$ which has to satisfy the capacity constraints and the demand constraints:

- $\forall a \in A\colon \operatorname{flow}(a) \leq \operatorname{cap}(a)$
- $\forall n \in N\colon \sum\limits_{(u,v) \in A} \operatorname{flow}((u,v)) \quad - \sum\limits_{(v,w) \in A} \operatorname{flow}((v,w)) = \operatorname{dem}(n)$

The cost of a flow f is defined to be $\operatorname{cost}(f) = \sum\limits_{a \in A} f(a) \cdot \operatorname{cost}(a)$.

The minimum cost flow problem consists of finding a valid flow f in T so that for any other valid flow f' the equation $\operatorname{cost}(f') \geq \operatorname{cost}(f)$ holds true. This problem is well-studied and polynomial-time algorithms finding solutions for it are known (e.g. the algorithm proposed by Goldberg and Tarjan [10]).

For an example of a Tamassia flow network, see Figure 2.3.

2.2 The Kandinsky Model

Since only one edge may be attached to each side of a vertex, it is impossible to draw graphs with vertices of degree greater than 4 in the Tamassia model. So, for higher-degree graphs, the Tamassia model is not applicable. The Kandinsky model is a model which admits drawings of higher-degree graphs [7]. As in the Tamassia model, vertices are placed on the points of a uniform grid. Again, the vertices have equal, square size and edges are represented by series of horizontal and vertical straight-line segments. In contrast to Tamassia's model, more than one edge may be attached to a vertex side. Thus, we have to extend the orthogonal representation from the Tamassia model.

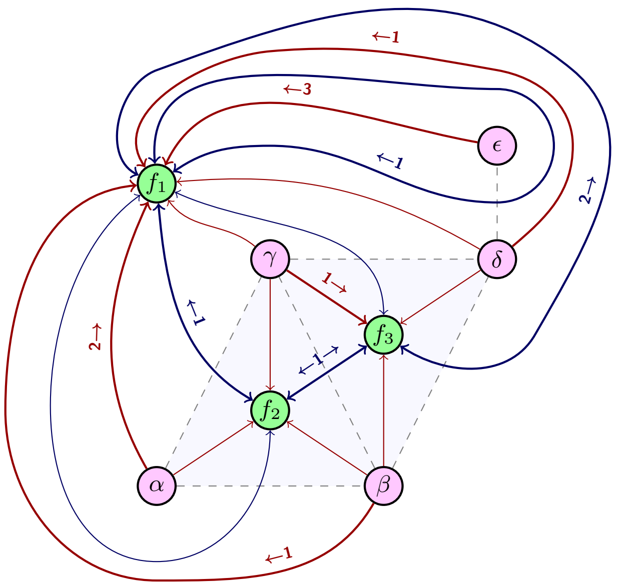


Figure 2.3: The Tamassia flow network T for the graph G shown in Figure 2.1a. The graph G is drawn with dashed lines. Nodes in N_V are drawn in purple, nodes in N_F are drawn in green. Arcs in A_{VF} are drawn in red, arcs in A_{FF} are drawn in blue. Edge labels denote the amount and direction of flow in T. If an edge is unlabeled, there is no flow on that edge. The flow corresponds to the orthogonal representation in Figure 2.2.

2.2.1 Orthogonal Representation

Specifically, for an edge description $h_e = (e, s, a) \in H_f$ we have to expand the value set for a, the integer specifying the angle in f formed by e and its successor edge by the value 0, that is $a \in \{0, 90, 180, 270, 360\}$. For an orthogonal representation of a Kandinsky drawing to be valid, all the criteria from Section 2.1.1 have to be met. Moreover, there is an additional requirement: A Kandinsky drawing must not have any empty faces[2]. A face is considered empty if its contour does not enclose at least one block of the grid. To ensure that a drawing has no empty faces, Fößmeier and Kaufmann state the following:

Lemma 1 ([7, Lemma 4])**.** *Every 0°-angle of a Kandinsky drawing has a unique corresponding 270°-bend.*

This property is also referred to as the *bend-or-end property*. For an example of an orthogonal graph drawing in the Kandinsky model, see Figure 2.4.

[2]The original reason for this requirement is that Fößmeier and Kaufmann [7] conjecture that if empty faces are allowed, the bend-minimization problem becomes $\mathcal{NP}$-hard. Furthermore, empty faces are problematic insofar as there is no way to prohibit subgraphs inside of an empty face, but a drawing with such faces would require large nodes to provide space for the subgraph's drawing.

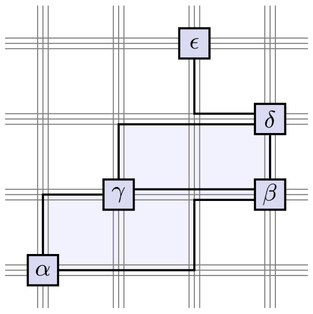


Figure 2.4: A Kandinsky drawing of the graph from Figure 2.1a. Graph edges are drawn as thick black lines and grid lines are drawn as thin, gray lines.

2.2.2 The Kandinsky Flow Network

The optimization problem OPTIMALKANDINSKYDRAW is: Given a plane graph G, find a Kandinsky drawing D of G so that the number of bends in D is minimal among all Kandinsky drawings of G. To solve OPTIMALKANDINSKYDRAW, the *Kandinsky flow network* can be used.

The Kandinsky flow network is an extension of the Tamassia flow network. The main difference is that in the Kandinsky flow network, it has to be possible to represent 0°-angles between two edges at a vertex. As stated by Lemma 1, such a 0°-angle has to have a unique corresponding 270°-bend. In a Kandinsky flow network, a 0°-angle can only be caused by flow through a helper node into the node associated with the vertex where the 0°-angle occurs. Flow can only enter the helper node from the two nodes associated with the faces adjacent to the face in which the 0°-angle occurs. For a visualization, see Figure 2.5. Except for this, the flow networks are similar.

Let $G = (V, E)$ be an undirected plane (multi-)graph. We define the Kandinsky flow network K for G as a tuple $K = (N, A, \text{dem}, \text{cap}, \text{cost}, B)$. By N we denote the set of nodes $N = N_V \cup N_F \cup N_H$ in the flow network, where:

$$N_V = \{n_v \mid v \in V\} \qquad (\textit{vertex nodes})$$

$$N_F = \{n_f \mid f \in F\} \qquad (\textit{face nodes})$$

$$N_H = \bigcup_{v \in V} \{n_{v,f} \mid f \in \text{faces}(v)\} \qquad (\textit{helper nodes})$$

By A we denote the set of directed arcs in the flow network, where:

$$A_{VF} = \bigcup_{v \in V} \{(n_v, n_f) \mid f \in \text{faces}(v)\}$$

$$A_{FF} = \bigcup_{f \in F} \{(n_f, n_{f'}) \mid f' \in \text{faces}(f)\}$$

$$A_{FH} = \bigcup_{f \in F} \{(n_f, n_{v,f'}) \mid \{f, f'\} \subseteq \text{faces}(v) \wedge f' \in \text{faces}(f)\}$$

$$A_{HV} = \bigcup_{v \in V} \{(n_{v,f}, n_v) \mid f \in \text{faces}(v)\}$$

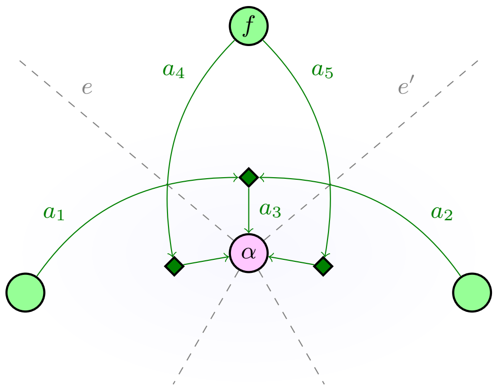


Figure 2.5: Parts of a Kandinsky flow network. If the corresponding Kandinsky drawing should have a 0°-angle in face f between the edges e and e' at the vertex associated with α, one unit of flow has to flow int0 α over the arc a_3. This unit must also flow either over a_1 or a_2, thereby ensuring that a 270°-angle corresponding to the 0°-angle exists. Since an edge cannot be bent in both directions, the arcs e_1 and e_4 are *bundled*, and the arcs e_2 and e_5 are bundled (if two arcs are bundled, only one unit may flow over both arcs cumulatively).

By dem: $N \to \mathbb{Z}$ we denote the demand function, which is defined as follows:

$$\forall n_v \in N_V\colon \operatorname{dem}(n_v) = \deg(n_v) - 4$$
$$\forall n_f \in N_F\colon \operatorname{dem}(n_f) = \begin{cases} \operatorname{size}(f) - 4 & f \text{ is inner face} \\ \operatorname{size}(f) + 4 & f \text{ is outer face} \end{cases}$$
$$\forall n \in N_H\colon \operatorname{dem}(n) = 0$$

By cap: $A \to N_+$ we denote the capacity function, which is defined as follows:

$$\forall a \in A\colon \operatorname{cap}(a) = \begin{cases} 1 & a \in A_{FH} \cup A_{HV} \\ 3 & a \in A_{VF} \\ \infty & a \in A_{FF} \end{cases}$$

By cost: $A \to \{0, 1\}$ we denote the cost function, which assigns a cost per unit of flow to each edge:

$$\forall a \in A\colon \operatorname{cost}(a) = \begin{cases} 0 & a \in A_{VF} \cup A_{HV} \\ 1 & a \in A_{FF} \cup A_{FH} \end{cases}$$

With $B = \{C_1, C_2, \ldots, C_k\}$ we denote a set of *bundles* $C_i = (\mathcal{C}, c)$ where $\mathcal{C} \subseteq A$ and $c \in \mathbb{N}_+$. Bundles allow the restriction of flow across arbitrary sets of arcs. In the following, we will always implicitely assume $B_0 \subseteq B$, where:

$$B_0 = \bigcup_{\substack{(u,v)\in E \\ \text{faces}((u,v))=\{f,f'\}}} \{(\{(n_f, n_{u,f'}), (n_{f'}, n_{u,f})\}, 1), (\{(n_f, n_{v,f'}), (n_{f'}, n_{v,f})\}, 1)\}$$

A valid flow in K is a function flow: $A \to \mathbb{N}_0$ which has to satisfy the capacity constraint, the demand constraints and the bundle constraints:

- $\forall a \in A\colon \text{flow}(a) \leq \text{cap}(a)$
- $\forall n \in N\colon \sum\limits_{(m,n)\in A} \text{flow}((m,n)) \;-\; \sum\limits_{(n,m)\in A} \text{flow}((n,m)) = \text{dem}(n)$
- $\forall\, C = (\mathcal{C}, c) \in B\colon \sum\limits_{a\in\mathcal{C}} \text{flow}(a) \leq c$

The cost of the flow f is defined to be $\text{cost}(f) = \sum\limits_{a\in A} f(a)\cdot\text{cost}(a)$.

Yildiz shows the following lemma:

Lemma 2 ([18, Satz 5])**.** *Let G be a plane graph. For every Kandinsky drawing D of G, there exists an integer flow in the Kandinsky flow network K, so that the flow's cost is equal to the number of bends in D. Furthermore, for every integer flow in K, a drawing D of G exists so that the flow's cost is equal to the number of bends in D.*

So, as with the Tamassia flow network, the Kandinsky minimum cost flow problem consists of finding a valid flow f in K so that for any other valid flow f' the equation $\text{cost}(f) \leq \text{cost}(f')$ holds true. Note that in addition to the standard constraints in a flow network, namely the capacity constraints and the demand constraints, f has to satisfy the bundle constraints. This means that standard algorithms for finding minimum cost flows are not applicable to K. Fößmeier and Kaufmann [7] and Yildiz [18] present approximation algorithms for OptimalKandinskyDraw, both of which have an approxmiation ratio of 2.

For an example of a Kandinsky flow network, see Figure 2.6.

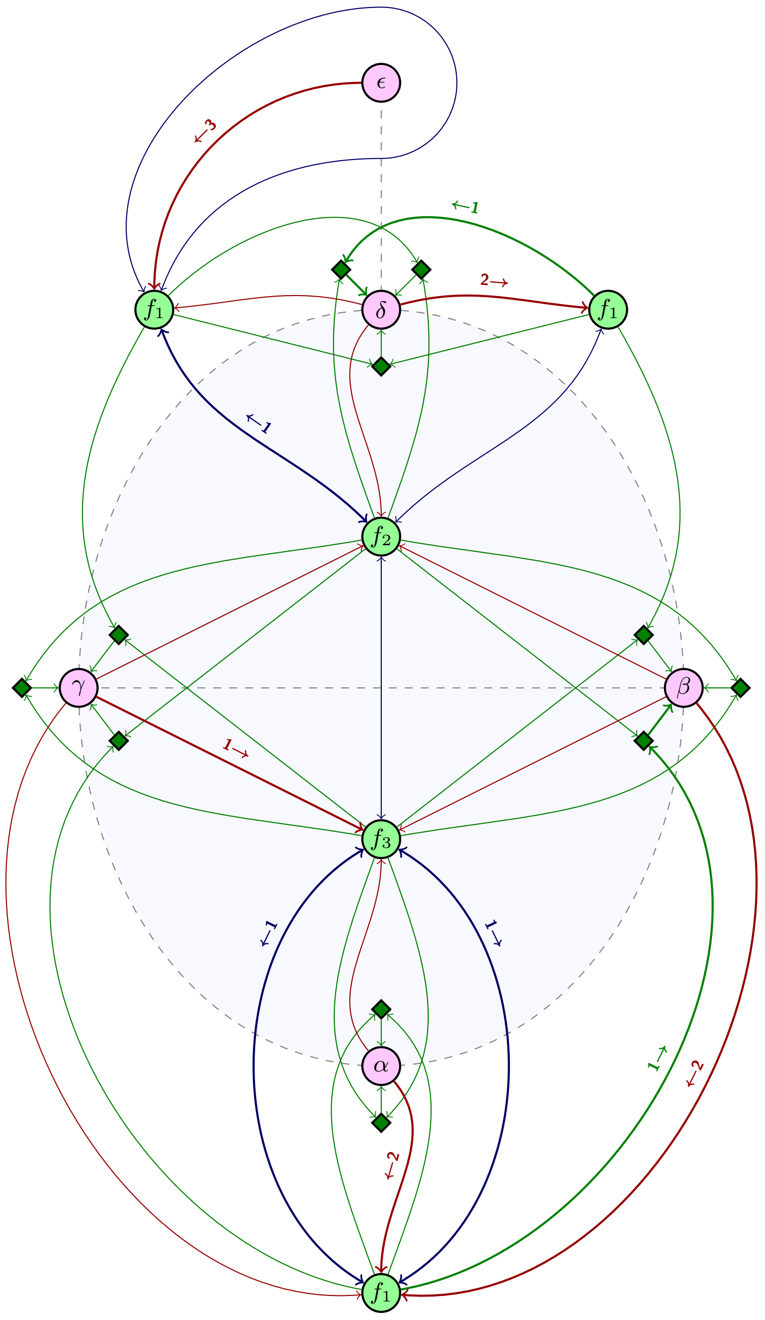


Figure 2.6: Kandinsky flow network K for the graph G shown in Figure 2.1a. The graph G is drawn with dashed lines. Nodes in N_V are drawn in purple, nodes in N_F are drawn in light green and nodes in N_H are drawn in dark green. Arcs in A_{VF} are drawn in red, arcs in A_{FF} are drawn in blue and arcs in $A_{FH} \cup A_{HV}$ are drawn in green. Edge labels denote the amount and direction of flow in K. For the sake of clarity, the node f_1 has been split into three nodes.

3 β-Embeddability

Taking a step back from the general problem OPTIMALKANDINSKYDRAW we attempt to make the problem more manageable by constraining the flexibility of edges, that is the maximum number of bends allowed per edge. For some constant $\beta \in \mathbb{N}_0$, a planar graph is said to be β-embeddable if it admits an orthogonal drawing where all bends have at most β bends. The decision problem β-EMBEDDABILITY can be studied for graphs with fixed or variable embeddings. In this chapter, we explicitely study β-EMBEDDABILITY for $\beta \in \{0, 1, 2\}$. For $\beta \geq 3$, we can reuse the results for $\beta = 2$.

3.1 0-Embeddability

In the Kandinsky model, all edges have the bend-or-end property, which ensures that if two edges portrude from the same side of a vertex at least one of them has to have a bend (this is an implication of Lemma 1). This means that in a 0-bend Kandinsky drawing (a *β-bend drawing* is a drawing where all edges have at most β bends) no more than one edge may portrude from each vertex side. Thus, 0-bend Kandinsky drawings only exist for graphs with maximum degree 4. Orthogonal graph drawing for graphs with maximum degree 4 is a well-studied problem, so we can reuse the corresponding results:

Theorem 1 ([15])**.** *The decision problem* 0-EMBEDDABILITY *for graphs with a fixed embedding is solvable in polynomial time.*

Theorem 2 ([9])**.** *The decision problem* 0-EMBEDDABILITY *for graphs with variable embeddings is $\mathcal{NP}$-complete.*

3.2 1-Embeddability

Now that we have solved 0-EMBEDDABILITY, a logical next step is to study 1-EMBEDDABILITY. We first show that for multigraphs with variable embeddings, 1-EMBEDDABILITY is $\mathcal{NP}$-complete. Then, we show that all plane simple graphs are 1-embeddable.

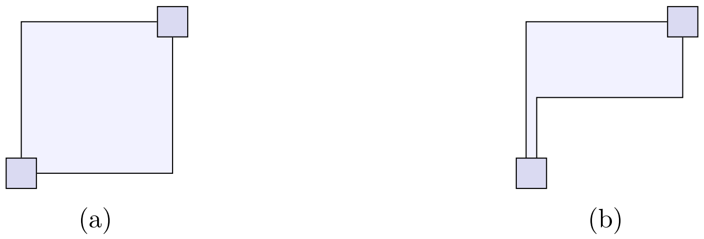


Figure 3.1: There is only one orthogonal representation for $\mathfrak{G}_1$ where every edge has at most one bend, as seen in Figure (a). In Figure (b), there is a 0°-angle at the lower vertex, at the cost of having two bends in an edge.

3.2.1 Multigraphs

In this section, we show that the decision problem 1-Embeddability for planar multigraphs with variable embeddings is $\mathcal{NP}$-complete. The approach is to reuse the result that 0-Embeddability with variable embeddings is $\mathcal{NP}$-complete (see Theorem 2) and then simulate unbendable edges with 1-bend Kandinsky drawings of *rigid* plane multigraphs. In this context, a rigid plane multigraph is a plane multigraph graph G so that for all 1-bend Kandinsky drawings of G we can draw a path $\pi(s,t)$ between two designated vertices s and t through G so that $\pi(s,t)$ is contained within the contours of G and $\text{rot}(\pi(s,t)) = 0$.

We start by constructing a very simple graph: Let $\mathfrak{G}_1 = (V,E)$ be the undirected multigraph consisting of two parallel edges, that is $V = \{u,v\}$ and $E = \{\{u,v\},\{u,v\}\}$. See Figure 3.1 for a visualisation. We state the following:

Lemma 3. *A pair of parallel edges $\mathfrak{G}_1 = (V,E)$ has exactly one valid orthogonal representation $\mathcal{R}$ where every edge has at most one bend.*

Proof. First, note that $\mathfrak{G}_1$ has exactly one planar embedding. Now, consider the Kandinsky flow network K for $\mathfrak{G}_1$. Let f be the inner face of $\mathfrak{G}_1$ and let v_f be the corresponding node in K. Let $E = \{e_1, e_2\}$. For any $e_j \in E$, let A_j be the set of arcs in K with source v_{f_i} causing a bend on e_j. Thus, $|A_j| = 3$. Note that for any arc $a = (v_f, v) \in A$ it is $a \in A_1 \cup A_2$. Since every edge in $\mathfrak{G}_1$ should have at most one bend, every set A_j may contain one arc with flow at most 1. We have $\text{dem}(v_f) = -2$, so in both sets A_j there is exactly one arc with flow 1. This means that there cannot be more flow passing v_f, which means that the flow on all arcs in K with target v_f must be 0. Thus, the internal face spans at most 90° at any vertex. If the internal face were to span 0° at a vertex, there would have to be a non-zero flow over an arc in K associated with an edge e_j. This would cause an additional bend on that edge, so the internal face has to span exactly 90° at both adjacent vertices and the outer face has to span exactly 270° at both adjacent vertices.

This describes the only possible orthogonal representation $\mathcal{R}$ of $\mathfrak{G}_1$ where every edge has at most one bend. A Kandinsky drawing corresponding to $\mathcal{R}$ is shown in Figure 3.1a. □

Next, we construct a plane graph $\mathfrak{G}_2$ with the intention to simulate an edge (u,v) which is unbendable to the right. To this end, we start with three vertices u, a and v

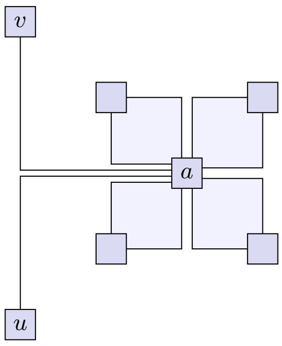


Figure 3.2: The plane graph $\mathfrak{G}_2$.

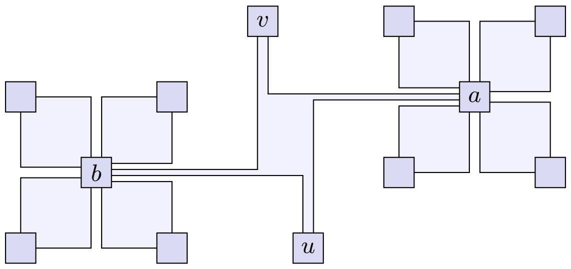


Figure 3.3: The graph $\mathfrak{G}_3$.

and two edges (u, a) and (a, v). Then, we add four vertices and attach them to a via two edges each. We embed $\mathfrak{G}_2$ so that it is outerplanar and that the edges (u, a) and (a, v) are neighbors in the cyclic edge incidency list of a. For a visualisation of $\mathfrak{G}_2$, see Figure 3.2.

Later on, we need this technical lemma:

Lemma 4. *The outer face of $\mathfrak{G}_2$ spans only 0°-angles at vertex a.*

Proof. It follows from Lemma 3 that all faces of size two span a 90°-angle at a. Since there are four such faces, together they span 360°, and since the sum of angles spanned by all faces around a given vertex is always 360°, all other faces, including the outer face, must span 0°-angles at a. □

If $\mathfrak{G}_2$ is embedded as shown in Figure 3.2 and D is a corresponding 1-bend Kandinsky drawing, we can draw a path $\pi(u, v)$ in D so that the equation $\text{rot}(\pi(u, v)) \geq 0$ holds true. This means that in a sense, $\mathfrak{G}_2$ is "unbendable" to the right.

Expanding upon this concept, we now construct a plane graph $\mathfrak{G}_3$ with the intention to simulate an edge (u, v) which is unbendable in both directions. To this end, we start with two instances of $\mathfrak{G}_2$, flip one of them horizontally and merge the two graphs at the vertices u and v. For an illustration of $\mathfrak{G}_3$, see Figure 3.3.

Then, by partly triangulating $\mathfrak{G}_3$, we generate the plane graph $\mathfrak{G}$, which is shown in Figure 3.4.

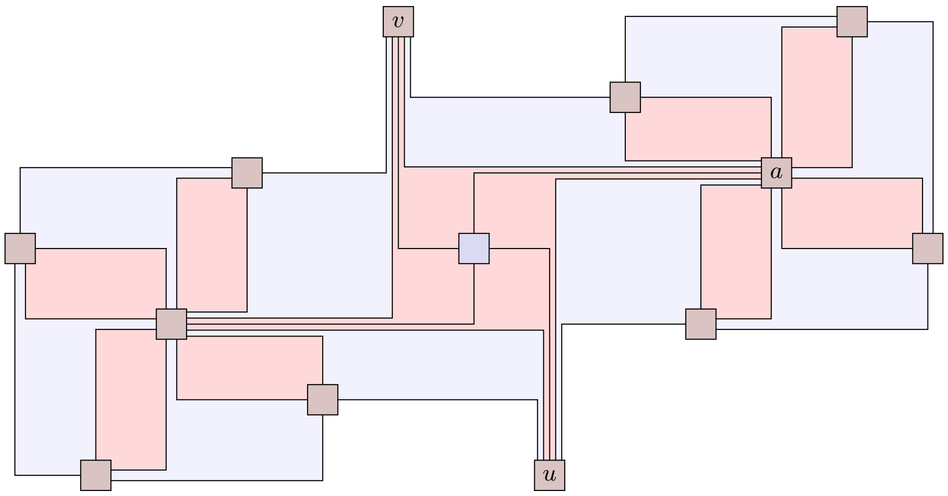


Figure 3.4: The replacement graph $\mathfrak{G}$. The instance of $\mathfrak{G}_3$, on which $\mathfrak{G}$ is based, is shaded in red.

Lemma 5. *The embedding of $\mathfrak{G}$ is fixed up to a flip. Only one face can be chosen as the outer face of $\mathfrak{G}$ so that an orthogonal representation can be constructed where every edge has at most one bend.*

Proof. It is easy to see that $\mathfrak{G}$ is triconnected. Whitney [17] showed that a triconnected graph has a fixed embedding up to a flip and the choice of the outer face.

For the second part, we start by observing that there is one face f_0 of $\mathfrak{G}$ with size$(f_0) = 10$. For all other faces $f \neq f_0$ of $\mathfrak{G}$ it is size$(f) \in \{2, 3\}$. All faces of $\mathfrak{G}$ except for f_0 are adjacent to exactly one of the vertices a or b. Furthermore, all faces $f \neq f_0$, have size$(f) \leq 3$. First, assume that size$(f) = 2$. We embed the contour f in the plane, which leads to a unique configuration (see Lemma 3). In this configuration, the angle formed by the edges at the vertices in the inner face are both 90°. The face f contains a vertex $x \in \{a, b\}$. There are three more faces f_i with size$(f_i) = 2$ incident to x. By assumption, f is the outer face so all other faces f_i have to lie inside the 90°-angle. However, they require an angle of at least 270° to be embedded with at most one bend. Thus, a face of size 2 cannot be the outer face.

Now, assume that size$(f) = 3$. Again, f contains one vertex $x \in \{a, b\}$. If the angle in f formed at x by the incident edges is less than 360°, we have the same problem as in the case of faces of size 2. So, embed the contour of f so that the angle at x in f is 360°, which means that the angle at x in the outer face is 0°. We now show that then one edge of the contour of f has to have more than two bends: Let K be the Kandinsky flow network for $\mathfrak{G}$. In K, for the node v_f associated with the outer face it is dem$(v_f) = 7$. Since f spans a 0°-angle on the outer face, at least one edge of f will be bent and there can be no flow from v_x to v_f. So, seven units must flow into v_f. Since only two edges are not bent, at least five units must flow into v_f from two nodes in K associated with two vertices other than x. Thus, there has to be at least one flow from some node v_y into v_f of amount at least 3. Since dem$(v_y) = 2$, pushing three units from this vertex into v_f

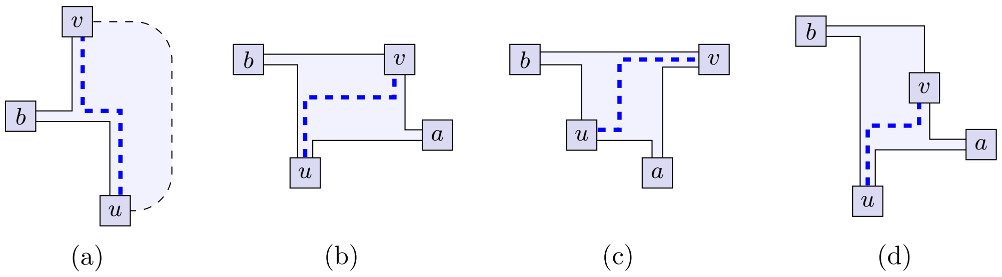


Figure 3.5: Illustration of Lemma 6. Regardless of the orthogonal representation of $\mathfrak{G}$, we may choose a path π with $\mathrm{rot}(\pi) = 0$, indicated as a dashed blue line.

requires receiving one unit from a 0°-angle on the inner face. This 0°-angle would cause an edge to be bent once more, but all edges are already bent. □

While the embedding of $\mathfrak{G}$ is fixed up to a flip, there are lots of 1-bend Kandinsky drawings of $\mathfrak{G}$. We show that regardless of the drawing, we can draw a path with rotation zero through it.

Lemma 6. *Let $\mathcal{R}$ be an orthogonal representation of $\mathfrak{G}$. In a corresponding drawing, it is possible to draw a path π from one endpoint u to the other endpoint v of $\mathfrak{G}$ so that $rot(\pi) = 0$ and π is completely contained within the contours of the outer face of $\mathfrak{G}$.*

Proof. Consider the shape of the edges (u, b) and (b, v) in $\mathfrak{G}$ with respect to $\mathcal{R}$. From Lemma 4 we know that the angle between these two edges is 0°. Thus, there are five possible situations which may be pooled into three classes: the edges are bent in differing directions, both edges are bent in the same direction, or one edge is a straight line while the other edge is bent.

If the edges are bent in differing directions, we may choose a path along these two edges as shown in Figure 3.5a.

Now, let one edge be a straigt line and let the other edge be bent. If the angle in the inner face at u and v is at least 90°, choosing π is trivial. The angle in the inner face cannot be 0° at both u and v. If the angle in the inner face is 0° at u, the angle at v must be 90° (see Figure 3.5b). Analogously, if the angle in the inner face is 0° at v, the angle at u must be 90° (see Figure 3.5c). In both cases, we find the desired path π.

Finally, if both edges are bent in the same direction, we choose π similarily to the case that one of the edges is a straight line; see Figure 3.5d. □

Lemma 6 means that $\mathfrak{G}$ is a rigid graph. We now show that in an orthogonal graph drawing in the Tamassia model, an edge with rotation zero is equivalent to an unbent edge.

Lemma 7. *Let G be a 4-planar graph together with a drawing D of G. In D, let for all paths π corresponding to edges in G be $rot(\pi) = 0$. Then G is admits a rectilinear planar drawing, that is D can be altered in such a way that for all corresponding edges e the equation $bends(e) = 0$ holds true.*

Proof. This follows directly from the construction of the Tamassia flow network. If there is a path π_e in D corresponding to an edge e in G so that $\text{rot}(\pi_e) = 0$ and $\text{bends}(\pi_e) \geq 1$, there is a circular flow in the corresponding Tamassia flow network, which may be removed, leading to an altered drawing D' with the only difference to D that π_e is a straight line in D'. This process can be repeated for all edges e in G, leading to a rectilinear planar drawing. □

Theorem 3. *In the Kandinsky model, the decision problem* 1-EMBEDDABILITY *with variable embeddings is $\mathcal{NP}$-complete.*

Proof. To see that 1-EMBEDDABILITY is $\mathcal{NP}$-hard, note that for a given configuration it is obviously possible to check whether the configuration is a 1-bend drawing.

Now, to see that 1-EMBEDDABILITY is even $\mathcal{NP}$-complete, we show how to transform an instance $\mathcal{I}$ of 0-EMBEDDABILITY into an equivalent instance $\mathcal{I}'$ of 1-EMBEDDABILITY.

Let $G = (V, E)$ be the graph of $\mathcal{I}$. We replace each edge $e = (u, v) \in E$ by a copy of the replacement graph $\mathfrak{G}$ to G and merge $\mathfrak{G}$ and G at the nodes u and v. For every edge this transformation requires constant time, and the entire transformation requires $\mathcal{O}(m)$ time (where $m = |E|$). Since G has degree at most 4, it follows that $m \leq 2n$, so the entire transformation is feasible in $\mathcal{O}(n)$ time. We denote the new graph by G'. Let $\mathcal{I}'$ be the corresponding instance of 1-EMBEDDABILITY. We show that the graph associated with $\mathcal{I}$ is 0-embeddable if and only if the graph associated with $\mathcal{I}'$ 1-embeddable.

First, assume that G admits a rectilinear planar drawing. Then, G' is clearly 1-embeddable in the Kandinsky model – simply replace each edge in G by a replacement graph $\mathfrak{G}$.

Now, assume that G' is 1-embeddable in the Kandinsky model. Let $\mathcal{E}$ be an embedding of G' and let $\mathcal{R}$ be an orthogonal representation of G'.

Note that G' need not be triconnected. This means that there may be more than one embedding of G'. Recall that the replacement graph $\mathfrak{G}$ is triconnected and only has one embedding. Theorem 2 is proved by transforming an instance of NOTALLEQUAL3SAT into an instance $\mathcal{I}$ of 0-EMBEDDABILITY [9]. The graph associated with $\mathcal{I}$ is always biconnected, so in the following we can assume G to be biconnected.

Since G is biconnected, no part of G' may be embedded in an inner face of a replacment graph $\mathfrak{G}$. In a drawing of $\mathcal{R}$, we can replace every drawing of a replacement graph $\mathfrak{G}$ with a path $\pi_{\mathfrak{G}}$ so that $\text{rot}(\pi_{\mathfrak{G}}) = 0$ (Lemma 6). There are no path intersections in $\mathcal{R}$. Since these paths are completely contained in the contours of the outer faces of the replacement graphs, there will be no path crossings in the altered drawings either. For every path π it is $\text{rot}(\pi) = 0$. So, there exists a rectilinear planar drawing of G' (Lemma 7).

Thus, G admits a rectilinear planar drawing. □

3.2.2 Simple Graphs

We show that for any plane simple graph we can find a 1-bend Kandinsky drawing. To this end, we use a decomposition known as a *canonical ordering for graphs* presented by De Fraysseix, Pach and Pollack [6]:

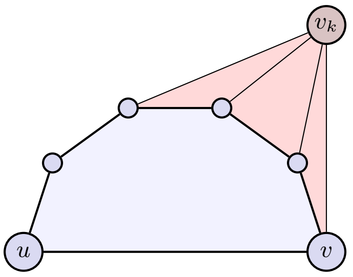


Figure 3.6: Visualization of Lemma 8. The area shaded in blue is the graph G_{k-1}. The corresponding cycle C_{k-1} is drawn thickly. The vertex v_k is the newly added vertex. The blue area together with the red area make up the graph G_k. Note that in G_k, v_k is adjacent to a subinterval of vertices on C_{k-1}.

Lemma 8 ([6, Canonical representation lemma for plane graphs])**.** *Let G be a maximal planar graph embedded in the plane with exterior face u, v, w. Then there exists a labelling of the vertices $v_1 = u, v_2 = v, v_3, \ldots, v_n = w$ meeting the following requirements for every $4 \leq k \leq n$.*

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

Moreover, it has been shown that such a canonical ordering for graphs can be computed in linear time. For a visualization of Lemma 8, see Figure 3.6.

We use Lemma 8 to construct a 1-bend Kandinsky drawing for simple plane graphs. Let G be a plane simple graph. First, triangulate G. This is possible in $\mathcal{O}(n)$ time. Then, compute a canoncial ordering of G, which Lemma 8 guarantees to exist. This, too, is possible in linear time. Starting with $i = 2$, we construct a 1-bend drawing for all subgraphs G_i induced by $v_1, \ldots, v_i$. To prove the correctness of our algorithm, we show that an invariant holds true for the drawings of all G_i. To define the invariant, we need some preliminary definitions. Let $C_i = (v_1^i = u, v_2^i = v, v_3^i, \ldots, v_i^i, v_{i+1}^i = v_1^i = u, \ldots)$ be the cyclic list of vertices incident to the outer face of G_i as encountered when going around the outer face in positive direction. For a vertex v_j^i, we define its *location* $\operatorname{loc}(v_j^i) = (x_j^i, y_j^i)$ in the drawing as a cartesian coordinate tuple. We describe the shape of an edge by a sequence of cartesian coordinates. We are now in the position to define the invariant:

Invariant. *The Kandinsky drawing D_i for the graph G_i ($3 \leq i \leq n$) has the following properties:*

1. *The location of u, v and the edge (u, v) are constant. It is loc$(u) = (0, 1)$, loc$(v) = (1, 0)$ and the shape of (u, v) is $((0, 1), (0, 0), (1, 0))$. Graphically, this means that u, v and the edge between them form the shape of the letter "L".*

2. *For the location (x^i_j, y^i_j) of the vertex v^i_j with $3 \leq j \leq i$ it is $x^i_{j-1} \geq x^i_j \geq x^i_{j+1}$ and $y^i_{j-1} \leq y^i_j \leq y^i_{j+1}$. Graphically, this means that the vertices in D_i are arranged as a "cascade".*

3. *The edge (v^i_j, v^i_{j+1}) has the shape $((x^i_j, y^i_j), (x^i_{j+1}, y^i_j), (x^i_{j+1}, y^i_{j+1}))$. Graphically, this means that (v^i_j, v^i_{j+1}) is attached to the left side of v^i_j, has a 90°-bend in the outer face and is attached to the lower side of v^i_{j+1}.*

We start by describing the drawing D_3. In D_3, let $\text{loc}(u) = (1, 0)$, $\text{loc}(v) = (0, 1)$ and $\text{loc}(v^3_3) = (\frac{1}{2}, \frac{1}{2})$. Furthermore, let the shape of (v, v^3_3) be $((1, 0), (\frac{1}{2}, 0), (\frac{1}{2}, \frac{1}{2}))$ and let the shape of (v^3_3, u) be $((\frac{1}{2}, \frac{1}{2}), (0, \frac{1}{2}), (0, 1))$. A visualization of D_3 is shown in Figure 3.7a. Clearly, the invariant from above is true for D_3.

We now show that from a drawing D_{k-1} with $4 \leq k \leq n$ for which the invariant is true, we can construct a drawing D_k for which the invariant is true. So, start with the drawing D_{k-1}. The task now is to add the vertex v_k and all its incident edges to the drawing so that afterwards, the invariant is still true. Let v_k be adjacent to the vertices $\left\{v^{k-1}_i, \ldots, v^{k-1}_j\right\}$ with $i \leq j$. Set the location of v_k to $\text{loc}(v_k) = (x_k, y_k)$, where:

$$x_k = \frac{x^{k-1}_i + x^{k-1}_{i+1}}{2}$$
$$y_k = \frac{y^{k-1}_{j-1} + y^{k-1}_j}{2}$$

Then, we set the shape of (v^{k-1}_i, v_k) to:

$$(\text{loc}(v^{k-1}_i), (x_k, y^{k-1}_i), \text{loc}(v_k))$$

We set the shape of (v_k, v^{k-1}_j) to:

$$(\text{loc}(v_k), (x^{k-1}_j, y_k), \text{loc}(v^{k-1}_j))$$

For all other edges (v_k, v^{k-1}_h) with $i < h < j$, we set the shape to:

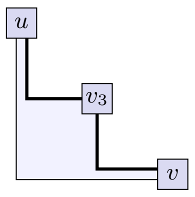
$$(\text{loc}(v_k), (x_k, y^{k-1}_h), \text{loc}(v^{k-1}_h))$$

A visualization of this procedure is shown in Figure 3.7b.

Clearly, a valid Kandinsky drawing D_k is generated and the invariant is still true for this drawing. Since all vertices and edges are only considered once by the algorithm, the runtime lies in $\mathcal{O}(n)$. This leads to the following theorem:

Theorem 4. *Let G be a plane simple graph. Then G is* 1*-embeddable and a* 1*-bend Kandinsky drawing of G may be computed in linear time.*

Note that this procedure does not necessarily work for multigraphs. Consider Figure 3.8. The nodes u and v_3 are connected by two edges, e_1 and e_2. We start by drawing either edge so that it has a 90°-bend in the outer face, in this case e_1. Since no edge may have more than one bend, we have no choice in how we draw e_2. But drawing e_2 will break the invariant of all edges adjacent to the outer face having a 90°-bend in the outer face.



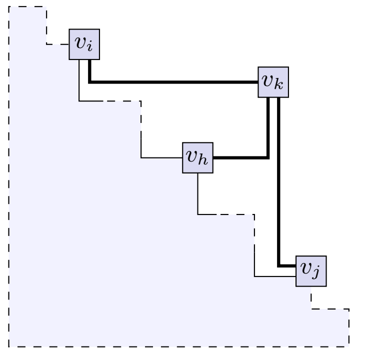


Figure 3.7: Construction of a 1-bend Kandinsky drawing. We start by drawing u and v. Next, add v_3, thereby generating the drawing D_3; see (a). Note that the newly added edges (shown in bold) have 90°-bends in the outer face. We repeat the same process for all remaining vertices v_k, see (b).

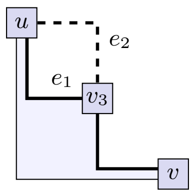


Figure 3.8: Adding multiedges breaks the invariant.

3.3 2-Embeddability

In this section, we study 2-EMBEDDABILITY. Specifically, we show that for any plane multigraph we can find a 2-bend Kandinsky drawing. We use a similar approach to the one we used in the case of 1-EMBEDDABILITY in Section 3.2.2.

We only alter the shape of the edges. Again, we can formulate an invariant:

Invariant. *The Kandinsky drawing D_i for the graph G_i ($3 \leq i \leq n$) has the following properties:*

1. *The location of u, v and the edge (u, v) are constant. It is loc$(u) = (0, 1)$, loc$(v) = (1, 0)$ and the shape of (u, v) is $((0, 1), (0, -\epsilon), (1, -\epsilon), (1, 0))$.*
2. *For the location (x^i_j, y^i_j) of the vertex v^i_j with $3 \leq j \leq i$ it is $x^i_{j-1} \geq x^i_j \geq x^i_{j+1}$ and $y^i_{j-1} \leq y^i_j \leq y^i_{j+1}$. Graphically, this means that the vertices in D_i are arranged as a "cascade".*
3. *The edge (v^i_j, v^i_{j+1}) has the shape $((x^i_j, y^i_j), (x^i_{j+1}, y^i_j - \epsilon), (x^i_{j+1}, y^i_{j+1} - \epsilon), (x^i_{j+1}, y^i_{j+1}))$. Graphically, this means that (v^i_j, v^i_{j+1}) is attached to the lower side of v^i_j, has two 90°-bends in the outer face and is attached to the lower side of v^i_{j+1}.*

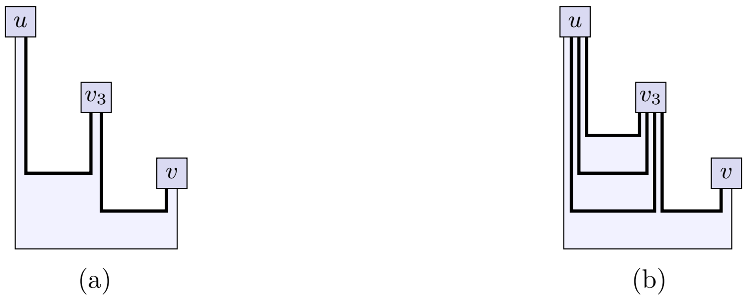


Figure 3.9: Construction of a 2-bend Kandinsky drawing. We start by drawing u and v. Next, we add v_3, see (a). If u and v_3 are connected by multiple edges, we can simply "stack" the edges, see (b).

The variable ϵ is chosen so that the edges do not intersect. Due to the similarity of this procedure to the case of 1-Embeddability, we will not explicitely prove its correctness. For a visualization of the altered procedure, see Figure 3.9.

We deduce the following theorem:

Theorem 5. *Let G be a plane multigraph. Then G is 2-embeddable and a 2-bend Kandinsky drawing of G can be computed in linear time.*

Obviously, for $\beta \geq 3$ this means that any plane multigraph G is β-embeddable and that a β-bend Kandinsky drawing of G can be computed in linear time.

4 Bend Minimization for Special Graph Classes

In the last chapter, we constrained the flexibility of edges, thereby solving some instances of β-EMBEDDABILITY and presented algorithms for finding β-bend Kandinsky drawings.

In this chapter, we try to make the bend minimization problem OPTIMALKANDINSKYDRAW more manageable by constraining the flexibility of edges and restricting the solution to special graph classes.

We begin by considering some very simple graph classes. For simple cycles, we make the following observation:

Lemma 9. *Let G be a simple cycle with $n \geq 2$. There exists a Kandinsky drawing of G with $max(4-n,0)$ bends. No drawing of G with fewer bends exists.*

Proof. This statement is intuitively evident, but we prove it nonetheless as an excercise. Start by observing that G has two faces, namely the inner face f and the outer face $\hat{f}$.

For now, consider the case $n = 2$. Then for the node n_f in the Kandinsky flow network K it is $\mathrm{dem}(n_f) = -2$. Since for all arcs $a_{n'} = (n_f, n')$ in K it is $\mathrm{cost}(a_{n'}) = 1$, any Kandinsky drawing $\mathcal{D}$ of G will have at least two bends. Clearly, a drawing of G with two bends exists, as shown in Figure 3.1a.

For $n \in \{3,4\}$, we can make similar arguments. For $n \geq 5$, each additional vertex can intuitively be placed in the middle of a straight line, causing no additional bends. Moreover, a 0-bend drawing is obviously bend-minimal. □

For trees, we make the following observation:

Lemma 10. *Let $G = (V,E)$ be a tree. There exists a Kandinsky drawing $\mathcal{D}$ of G with $\sum\limits_{v \in V} max(deg(v)-4,0)$ bends. No drawing of G with fewer bends exists.*

Proof. Clearly, such a drawing $\mathcal{D}$ exists. The optimality follows directly from the bend-or-end property of the Kandinsky model. □

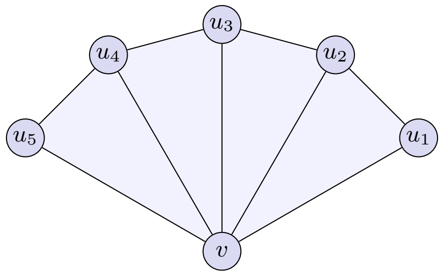


Figure 4.1: A *fan* with vertex v at its center.

4.1 Biconnected, Outerplanar, Inner-Triangulated Graphs

In this section, we present an algorithm solving OPTIMALKANDINSKYDRAW for biconnected, outerplanar, inner-triangulated plane graphs in linear time.

Let G be an outerplanar, inner-triangulated plane graph with $n \geq 3$. Let v be a vertex of G so that all faces of G are incident to v. Then G is referred to as a *fan* with v at its *center* and $(u_1, u_2, \ldots, u_{n-1})$ the sequence of *fan vertices* in the same order as they are encountered when traversing the outer face of G in positive direction. For an example of a fan with six vertices, see Figure 4.1.

We use the special structure of outerplanar, inner-triangulated graphs to find a *superposition* of fans. A set of fans $\mathcal{F} = \{F_1 = (V_1, E_1), F_2, \ldots, F_k\}$ is called a superposition of $G = (V, E)$ if it has the following properties:

- The entire graph G is covered, that is $\bigcup\limits_{i=1}^{k} V_i = V$.
- No face of G is covered by two fans: Let f be an inner face of G. Since G is inner-triangulated, f is incident to three vertices u, v and w. There is exactly one fan $F_i \in \mathcal{F}$ so that $\{u, v, w\} \subseteq V_i$ holds true.

For an example of a superposition of fans, see Figure 4.2.

We now show the following:

Lemma 11. *Let G be an outerplanar, inner-triangulated graph. It is possible to find a superposition of fans $\mathcal{F}$ of G in linear time.*

Proof. We show the existance of such a superposition $\mathcal{F}$ by describing an algorithm which decomposes G into a superposition $\mathcal{F}$.

Start the decomposition by arbitrarily choosing a vertex v, which – since G is outerplanar – lies on the outer face. The vertex v will be the center vertex of a newly constructed fan F. First, define the *active* edge $e_1 = (v, u_1)$ as the successor edge of v on the outer face of G in counter-clockwise order. The active edge e_1 is incident to two faces, namely the outer face and an inner face f_1. Add f_1 to the fan. Since G is inner-triangulated, f_1 is incident to three vertices v, u_1 and u_2. Then, update the active edge to be $e_2 = (v, u_2)$. We repeat this procedure until the active edge becomes the predecessor edge of v on the

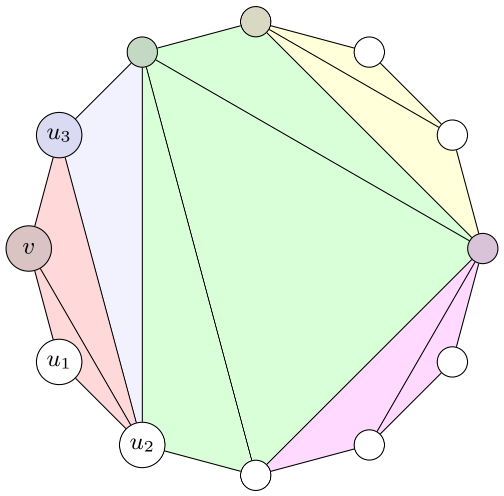


Figure 4.2: Example of an outerplanar, inner-triangulated graph and a superposition of fans thereof. The colored areas represent the fans, the colored vertices are at their respective centers.

outer face of G in counter-clockwise order, that is $e_k = (v, u_k)$. The fan F now consists of $k-1$ faces f_i. Any edge $e = (u_i, u_{i+1})$ is incident to a face f_i in F and to some other face $\hat{f}_i$. If $\hat{f}_i$ is not the outer face, $\hat{f}_i$ belongs to a *rest graph*. Note that any rest graph is also an outerplanar, inner-triangulated graph G'. We recursively apply our procedure on G', starting with the vertex u_{i+1}, yielding a superposition $\mathcal{F}'$. Then $\mathcal{F} = \mathcal{F}' \cup \{F\}$ is a superposition of G. Since any fan consists of at least one face and the number of faces in G is finite, the procedure terminates. The algorithm's runtime is obviously proportional to the number of faces in G, and thus in $\mathcal{O}(n)$. □

For an example of the decomposition algorithm, consider Figure 4.2. We begin by choosing the vertex v as the first fan's center, which means that the edge (v, u_1) becomes the first active edge. We then add the face $\{v, u_1, u_2\}$ to the fan and make (v, u_2) the active edge. Likewise, we then add the face $\{v, u_2, u_3\}$. At this point, (v, u_3) becomes the active edge and since this edge is the predecessor edge of a in counter-clockwise order, the first fan is complete. In Figure 4.2, this first fan is shaded in red. The edge (u_1, u_2) is not incident to a rest graph, but the edge (u_2, u_3) is. Here, the rest graph is the entire graph, except for the first fan shaded in red. This means that we recursively apply the algorithm on the rest graph, starting with the vertex u_3. Ultimately, the algorithm finds a superposition of five fans (each of which is shaded with a distinct color in Figure 4.2).

Consider a superposition of fans $\mathcal{F} = \{F_1, F_2, \ldots, F_k\}$ of some outerplanar, inner-triangulated graph G. Two fans $F_i = (V_i, E_i)$ and $F_j = (V_j, E_j)$ are said to *conflict* if $E_i \cap E_j \neq \emptyset$. Any edge in $E_i \cap E_j$ is a *conflicting edge*. We then define the undirected *conflict graph* $G^\times = (V^\times, E^\times)$ of $\mathcal{F}$, where $V^\times = \{v_{F_i} \mid F_i \in \mathcal{F}\}$ and $(v_{F_i}, v_{F_j}) \in E^\times$ if and only if the fans F_i and F_j are conflicting. The conflict graphs of the superpositions found by the algorithm from Lemma 11 have a special structure:

Lemma 12. *Let G be an outerplanar, inner-triangulated graph, let $\mathcal{F}$ be a superposition of fans of G found by the algorithm presented in Lemma 11 and let $G^\times$ be the conflict graph of $\mathcal{F}$. Then $G^\times$ is a tree and we root it at v_{F_1}.*

Proof. The graph $G^\times$ is a tree if it does not contain a cycle. Clearly, if $G^\times$ contained a cycle, G would not be outerplanar. ☐

Now that we have shown how to decompose a biconnected, outer-planar, inner-triangulated graph G, let us proceed with drawing G. We start by defining two Kandinsky drawings $\dot{D}(F)$ and $\ddot{D}(F)$ for every fan F.

Let F be a fan of n vertices. We define the locations of all vertices w as follows:

$$\operatorname{loc}(w) = \begin{cases} (n-2,1) & \text{if } w = v \\ (0,1) & \text{if } w = u_1 \\ (i-1,0) & \text{if } w = u_i \text{ for } 1 < i \leq n-1 \end{cases}$$

The coordinate sequences $\pi(e)$ describing the edges $e = (v, u_i)$ incident to the center vertex v are defined as follows:

$$\pi((v,u_i)) = \begin{cases} (\operatorname{loc}(v), \operatorname{loc}(u_i)) & \text{if } i \in \{1, n-1\} \\ (\operatorname{loc}(v), (i-1,1), \operatorname{loc}(u_i)) & \text{for } 1 < i < n-1 \end{cases}$$

The coordinate sequences $\pi(e)$ describing the edges $e = (u_i, u_{i+1})$ between fan vertices are defined as follows:

$$\pi(u_i, u_{i+1}) = \begin{cases} (\operatorname{loc}(u_i), \operatorname{loc}(u_{i+1})) & \text{for } 1 < i < n-1 \\ ((0,1),(0,0),(1,0)) & \text{if } i = 1 \end{cases}$$

Obviously, the generated drawing $\dot{D}(F)$ is a valid Kandinsky drawing and has $n-2$ bends. For an example of such a drawing $\dot{D}(F)$, see Figure 4.3b. We state the following:

Lemma 13. *Let F be a fan. There exists a Kandinsky drawing of F with $n-2$ bends. No drawing with fewer bends exists.*

Proof. Clearly, a Kandinsky drawing of F with $n-2$ bends exists, namely $\dot{D}(F)$.

To see the optimality of $\dot{D}(F)$, consider the Kandinsky flow network K for F. All inner faces f have size 3, so for the corresponding Kandinsky flow network nodes n_f it is $\operatorname{dem}(n_f) = -1$. Every unit flowing from n_f causes one bend because for every arc $a_{n'} = (n_f, n')$ it is $\operatorname{cost}(a_{n'}) = 1$. Since a fan has $n-2$ faces, it follows that any Kandinsky drawing of F has at least $n-2$ bends. ☐

The drawing $\ddot{D}(F)$ is a small variation of the drawing $\dot{D}(F)$. To generate the drawing $\ddot{D}(F)$ for a fan F, start with the drawing $\dot{D}(F)$. We then reposition u_1 so that $\operatorname{loc}(u_1) = (0,2)$. For $\ddot{D}(F)$ to be a Kandinsky drawing, the shape of the edge (v, u_1) has to be redefined to $(\operatorname{loc}(v), (0,1), \operatorname{loc}(u_{n-1}))$. Obviously, the generated drawing $\ddot{D}(F)$ is a valid Kandinsky drawing and has $n-1$ bends, that is one more bend than an optimal drawing. For an example of such a drawing $\ddot{D}(F)$, see Figure 4.3c.

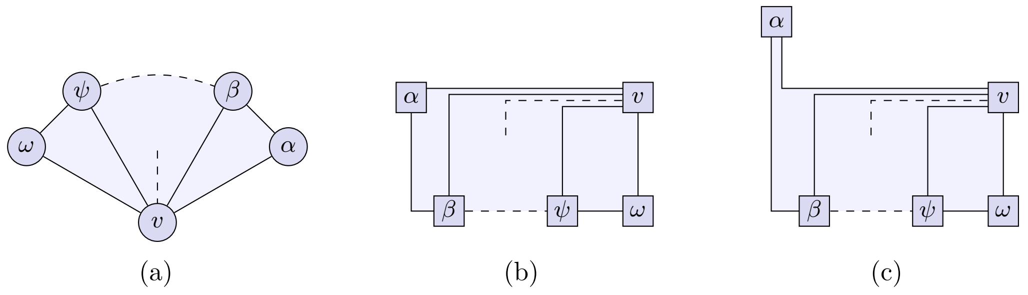


Figure 4.3: A fan F in (a), where the sequence of fan vertices is $(u_1 = \alpha, u_2 = \beta, \ldots, u_{n-2} = \psi, u_{n-1} = \omega)$, and the corresponding Kandinsky drawings $\dot{D}(F)$ in (b) and $\ddot{D}(F)$ in (c).

For the sake of clarity, it is useful to introduce a naming scheme for edges in a fan drawing $\dot{D}(F)$ and $\ddot{D}(F)$: We call the edge (v, u_1) the *upper* edge, the edge (v, u_{n-1}) the *right* edge and all other edges incident to v *inner* edges. Furthermore, we call the edge (u_1, u_2) the *corner* edge and all other edges u_i, u_{i+1} *lower* edges.

Consider a biconnected, outerplanar, inner-triangulated plane graph G, a superposition of fans $\mathcal{F}$ (Lemma 11) of G and the corresponding conflict graph $G^\times$. Consider F, the fan associated with the root of $G^\times$ and F', a fan associated with a child of the root in $G^\times$. Let e be the conflicting edge of F and F'. In the drawing $\dot{D}(F)$, e is either the corner edge or a lower edge. To see this, note that e clearly is not an inner edge. It also is not the upper edge or the right edge, because these edges are incident to the outer face and F. In the drawing $\dot{D}(F')$, e is the upper edge by definition. The same considerations are true for drawings $\ddot{D}(F)$ and $\ddot{D}(F')$. We can extend the argument for any two conflicting fans by considering the subtree rooted at the parent fan's node. This leads to the following lemma:

Lemma 14. *Let F and F' be two conflicting graphs so that $v_{F'}$ is a child node of v_F in $G^\times$. The conflicting edge of F and F' is a corner edge or a lower edge in $\dot{D}(F)$ and $\ddot{D}(F)$, and it is the upper edge in $\dot{D}(F')$ and $\ddot{D}(F')$.*

Again, consider the conflict tree $G^\times$. For all nodes v_{F_i} in $G^\times$, we compute the drawing $D(F_i)$, which is either $\dot{D}(F_i)$ or $\ddot{D}(F_i)$. We use a top-down approach: Starting with F_1, the fan associated with the root node of $G^\times$, we set $D(F_1) = \dot{D}(F_1)$. Let F_i and F_j be any two conflicting fans so that without loss of generality v_{F_j} is a child node of v_{F_i}. If the conflicting edge e_j is a lower edge in $D(F_i)$, we set $D(F_j) = \dot{D}(F_j)$. Otherwise, if e_j is the corner edge in $D(F_i)$, we set $D(F_j) = \ddot{D}(F_j)$. Clearly, the individual drawings $D(F_i)$ can be merged into a single drawing $D(G)$. Such a drawing is shown in Figure 4.4.

We now claim the following:

Lemma 15. *Let $G = (V, E)$ be a biconnected outerplanar, inner-triangulated plane graph. There exists a Kandinsky drawing of G with $n - 2$ bends and no drawing with fewer bends exists.*

Proof. The sought-after Kandinsky drawing is the drawing $D(G)$, which is computed as described above. We now need to show that $D(G)$ has $n - 2$ bends. To see this, note that

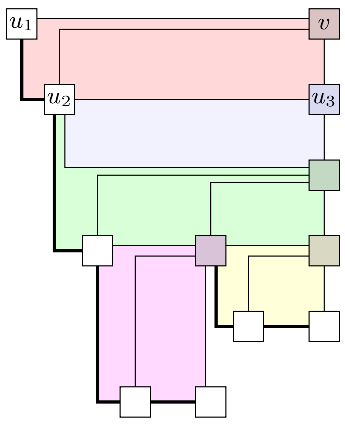


Figure 4.4: Kandinsky drawing of the graph shown in Figure 4.2. Two rest graphs, namely the purple and yellow area are attached to the green area. Edges to which another rest graph might be attached are drawn in bold.

the number of bends in $D(G)$ is not equal to the sum of bends in all drawings $D(F_i)$, because some bends are counted twice. Specifically, this occurs if two fans F_i and F_j conflict (w.l.o.g., let v_{F_j} be a child node of v_{F_i} in $G^\times$) and the conflicting edge e_j is the corner edge in $D(F_i)$. Note that this occurs if and only if $D(F_j) = \ddot{D}(F_j)$. Coincidentally, the drawing $\ddot{D}(F_j)$ has exactly one more bend than the drawing $\dot{D}(F_j)$. So, instead of counting $|V_j| - 1$ bends for a drawing $D(F_j) = \ddot{D}(F_j)$, we count $|V_i| - 2$ bends for all drawings $D(F_i)$. Thus, for the cumulated number of bends we know:

$$\sum_i (|V_i| - 2) = \sum_i |V_i| - 2k$$

Since $G^\times$ is a tree, we count two vertices twice for each pair of conflicting fans. Since there are $k - 1$ pairs of conflicting fans, we know:

$$\begin{aligned}\sum_i |V_i| - 2k &= |V| + 2(k-1) - 2k \\ &= |V| - 2 \\ &= n - 2\end{aligned}$$

To see the optimality of $D(G)$, realize that any drawing with fewer bends would induce a drawing of a fan with k nodes with less than $k - 2$ bends. In Lemma 13, we have shown that this is impossible. □

This leads to the following theorem:

Theorem 6. *The bend minimization problem* OptimalKandinskyDraw *is solvable in linear time for biconnected, outerplanar, inner-triangulated graphs. Every optimal drawing has $n - 2$ bends.*

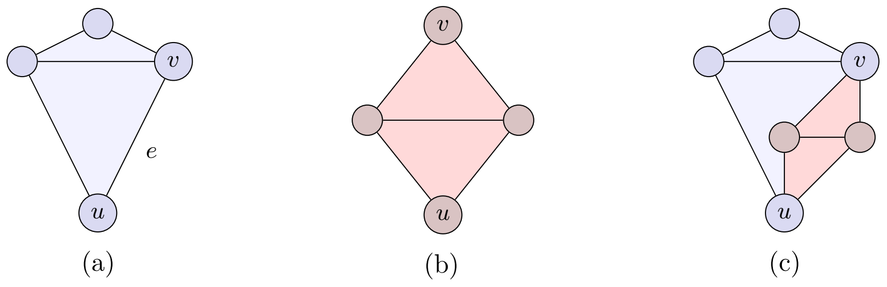


Figure 4.5: Illustration of the concept of virtual edges. Let G be the plane graph shown in (a) and let G' be the plane graph shown in (b) so that $V \cap V' = \{u, v\}$. Then we can define the edge $e = (u, v)$ to be a virtual edge, and merge G' into G at e. The result of this is shown in (c).

4.2 Series-Parallel Graphs

In this section, we demonstrate how to compute bend-minimal 1-bend drawings of series-parallel graphs. This solves the optimization problem of 1-OptimalKandinskyDraw: Given a series-parallel plane graph G, find a Kandinsky drawing D so that every edge in D is bent at most once and the total number of bends in D is minimal among all 1-bend Kandinsky drawings of G.

To this end, we start by introducing a series-parallel decomposition tree $\mathcal{T}$ for any series-parallel plane graph. This tree is quite similar to the SPQR-tree introduced by Di Battista and Tamassia [3]. Unlike an SPQR-tree, the series-parallel decompositon tree does not represent all planar embeddings of a series-parallel graph, but rather a specific, fixed embedding.

The series-parallel decomposition makes use of the concept *virtual edges*. Let $G = (V, E)$ and $G' = (V', E')$ be two plane graphs with the fixed embeddings $\mathcal{E}$ and $\mathcal{E}'$, respectively. Let $e = (u, v) \in E$ be an edge and $V \cap V' = \{u, v\}$. Then we can define e to be a virtual edge. The graph G' can then be merged into G at e by replacing the occurence of e in G with G'. More formally, we say that merging G' into G at e yields a new plane graph $\hat{G} = (\hat{V}, \hat{E})$ with the fixed embedding $\hat{\mathcal{E}}$ with the following properties:

- The graphs are merged, that is $\hat{V} = V \cup V'$ and $\hat{E} = (E \cup E') \setminus \{e\}$.
- The restriction of $\hat{\mathcal{E}}$ to $\hat{G} \setminus V$ matches the restriction of $\mathcal{E}'$ to G'.
- Likewise, the restriction of $\hat{\mathcal{E}}$ to $\hat{G} \setminus V'$ matches the restriction of $\mathcal{E}$ to $G \setminus \{e\}$.

For a visualization of the concept of virtual edges, see Figure 4.5.

Let $G = (V, E)$ be a series-parallel plane graph with the fixed embedding $\mathcal{E}$. The series-parallel decomposition tree $\mathcal{T}$ of $G = (V, E)$ has three kinds of nodes, namely S-, P- and Q-nodes.

If G consists of a simple edge, that is $V = \{s, t\}$ and $E = \{(s, t)\}$, we say that G has the *poles* s and t, which we write as $G_{s,t}$. Then, $\mathcal{T}$ consists of a single *Q-node* μ_q. The Q-node μ_q is associated with $G = \text{graph}(\mu_q)$. Note that all leaves of $\mathcal{T}$ are Q-nodes.

If G is biconnected, it has a separator pair $S = \{s, t\}$ which splits G into two subgraphs G' and G''. These subgraphs may be composited in parallel to reconstruct G. All three graphs have the same poles, namely the nodes in S, and we write $G_{s,t}$, $G'_{s,t}$, $G''_{s,t}$. Then, $\mathcal{T}$ is rooted at a P-node μ_p. The P-node μ_p has two children μ'_p and μ''_p, which are the roots of the series-parallel decomposition trees $\mathcal{T}'$ and $\mathcal{T}''$ for G' and G'', respectively. The P-node is associated with a plane *skeleton graph* $\operatorname{skel}(\mu_p) = (\{s,t\}, \{e_1 = \{s,t\}, e_2 = \{s,t\}\})$ (mind the multiset!). Both edges e_1 and e_2 of the skeleton graph are virtual edges. The plane graph $G = \operatorname{graph}(\mu_p)$ can be retrieved by merging $\operatorname{graph}(\mu'_p)$ into $\operatorname{skel}(\mu_p)$ at e_1 and $\operatorname{graph}(\mu''_p)$ into $\operatorname{skel}(\mu_p)$ at e_2.

If G has a separator vertex s, it splits G into two subgraphs $G'_{r,s}$ and $G''_{s,t}$. These subgraphs may be composited in series to reconstruct $G_{r,t}$. Then, $\mathcal{T}$ is rooted at an *S-node* μ_s. The S-node μ_s has two children μ'_s and μ''_s, which are the roots of the series-parallel decomposition trees $\mathcal{T}'$ and $\mathcal{T}''$ for G' and G'', respectively. The S-node is associated with a plane *skeleton graph* $\operatorname{skel}(\mu_s) = (\{r,s,t\}, \{e_1 = (r,s), e_2 = (s,t)\})$. Both edges e_1 and e_2 of the skeleton graph are virtual edges. The plane graph $G = \operatorname{graph}(\mu_s)$ can be retrieved by merging $\operatorname{graph}(\mu'_s)$ into $\operatorname{skel}(\mu_s)$ at e_1 and $\operatorname{graph}(\mu''_s)$ into $\operatorname{skel}(\mu_s)$ at e_2.

Due to the similarity of the series-parallel decomposition tree $\mathcal{T}$ to SPQR-trees, we can reuse the results for SPQR-trees and state the following:

Lemma 16. *The series-parallel composition tree for a series-parallel graph has linear size and can be constructed in linear time.*

This follows from a similar result for SPQR-trees shown by Gutwenger and Mutzel [11].

The general idea of the algorithm solving 1-OptimalKandinskyDraw is to use dynamic programming: In order to compute the optimal Kandinsky drawing of a plane series-parallel graph, we first construct the series-parallel decomposition tree $\mathcal{T}$. Then, for all Q-nodes μ_q we calculate all possible configurations for the associated graphs $\operatorname{graph}(\mu_q)$. For S-nodes and P-nodes we calculate all possible combinations of configurations of the subgraphs and store them. Finally, for the root node, we pick the configuration with the smallest number of bends. Since all possible configurations have been considered, the optimal configuration will be found in the end.

In the rest of this section, we demonstrate how to perform the necessary computations efficiently. The main idea behind this is that it is not necessary to consider the entire, possibly large child graphs, but instead focus on a few relevant properties. We can then reuse results which have been precomputed for graphs with the same relevant properties, thus saving time.

As a first step, we define a binary equivalence relation of configurations $\sim$, where two configurations $\mathcal{K}_1$ and $\mathcal{K}_2$ are related with respect to $\sim$ if in any drawing containing $\mathcal{K}_1$, we can replace $\mathcal{K}_1$ by $\mathcal{K}_2$. More formally, let G be a biconnected graph and $\mathcal{K}$ a configuration of G. Let G_1 be a subgraph of G so that G_1 may be separated from the rest of G by a separator pair $\{s,t\}$ and let the restriction of $\mathcal{K}$ to G_1 be $\mathcal{K}_1$. Then we can replace G_1 by some graph G_2, thereby generating an altered graph $G' = G \setminus G_1 \cup G_2$, so that G_2 may be separated from the rest of G' by $\{s,t\}$, and find a configuration $\mathcal{K}'$ so that the restriction of $\mathcal{K}'$ to G_2 equals $\mathcal{K}_2$ and the restrictions of $\mathcal{K}$ and $\mathcal{K}'$ to $G' \setminus G_2$ are equal.

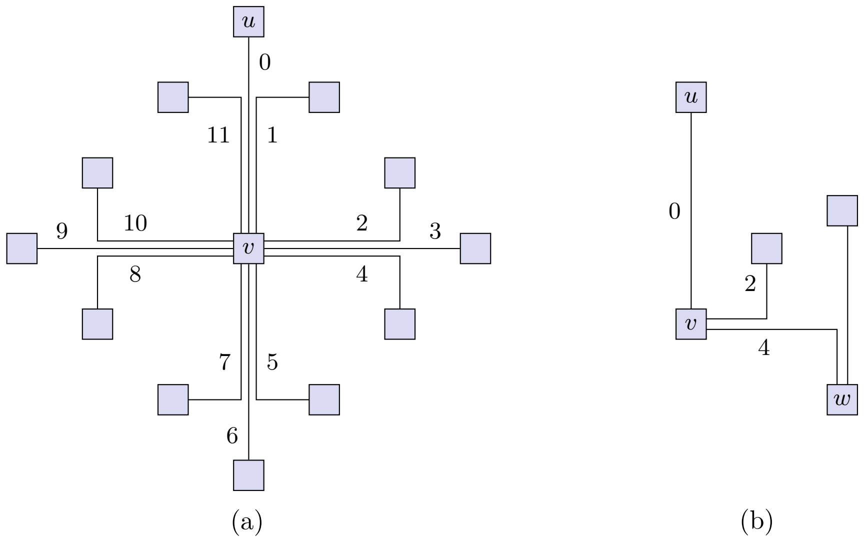


Figure 4.6: In (a), a configuration $\mathcal{K}$ of a simple graph where all edge shapes are present. The value next to an edge e is shape(e) Note that for the closest shape σ' for some shape σ it might be $|\sigma - \sigma'| > 1$. For example, in (b) the closest shape in counter-clockwise order for the edge (v, w) is not 3, but 2.

We show that whether two configurations are related depends on the *shape* of some edges. In the case of 1-Embeddability, an edge adjacent to some vertex v has one of twelve shapes. For a given configuration $\mathcal{K} = (G, \mathcal{R})$ of a graph $G = (V, E)$, we define a function shape: $E \to [0, 11]$ assigning a shape to an edge, as shown in Figure 4.6.

Let $\mathcal{K} = (G, \mathcal{R})$ with $G = (V, E)$ be a configuration. Let $e = (v, u) \in E$ be an edge with shape$(e) = \sigma$. We call a shape σ' the *closest* shape to e in clockwise order at v if there is an edge $e' = (v, u') \in E$ with shape$(e') = \sigma'$ (or it is possible to add such an edge to $\mathcal{K}$) and for no other edge $\bar{e} = (v, \bar{u}) \in E$ it is $\sigma \leq$ shape$(\bar{e}) < \sigma'$ (or it is impossible to add such an edge to $\mathcal{K}$). The closest shape in counter-clockwise order is defined analogously.

Let G_1 and G_2 be two graphs so that $G_1 \cap G_2 = \{s, t\}$. For $i \in \{1, 2\}$, let $\mathcal{K}_i = (G_i, \mathcal{R}_i)$ be a configuration of G_i, so that s and t lie on the outer face of $\mathcal{K}_i$. Let $\pi_i = (v_1^i = s, v_2^i, \dots, v_{j_i-1}^i, v_{j_i}^i = t, v_{j_i+1}^i, \dots, v_{k_i-1}^i, v_{k_i}^i)$ be the sequence of vertices as encountered when traversing the outer face of $\mathcal{K}_i$ in positive direction. By $\pi_i(s, t)$ we denote the subsequence of π_i from s to t and by $\pi_i(t, s)$ we denote the subsequence of π_i from t to s. Neither $\pi_i(s, t)$ nor $\pi_i(t, s)$ should consist of exactly one edge with a 270° angle in the outer face. We call configurations which have such paths *prohibited*. A configuration which is not prohibited is *admissible*. Let σ_i^1 be the closest shape to $(v_{k_i}^i, s)$ at s in counter-clockwise order and let σ_i^2 be the closest shape to (s, v_2^i) at s in clockwise order. Analogously, let σ_i^3 be the closest shape to $(v_{j_i-1}^i, t)$ at t in counter-clockwise order and let σ_i^4 be the closest shape to $(t, v_{j_i+1}^i)$ at t in clockwise order. Let $\sigma_i = (\sigma_i^1, \sigma_i^2, \sigma_i^3, \sigma_i^4)$. If $\sigma_1 = \sigma_2$, we say that $\mathcal{K}_1$ and $\mathcal{K}_2$ have *equal shapes*. If rot$(\pi_1(s, t)) =$ rot$(\pi_2(s, t))$, we say that $\mathcal{K}_1$ and $\mathcal{K}_2$ have *equal rotations*. We now claim the following:

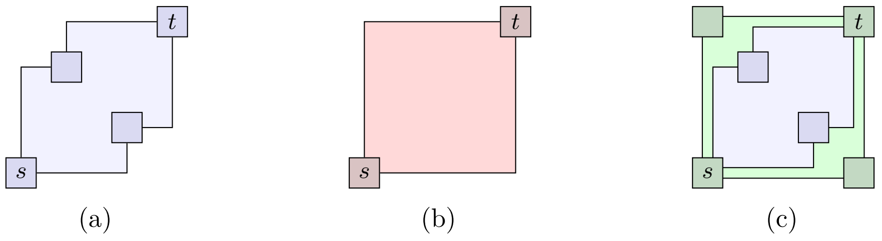


Figure 4.7: If we did not prohibit some paths $\pi(s,t)$ in Lemma 17, the configuration $\mathcal{K}$, shown in (a), and the configuration $\mathcal{K}'$, shown in (b) would be in the same equivalence class. Consider the configuration $\hat{\mathcal{K}}$ shown in (c). Clearly, the configuration $\mathcal{K}$ can't be replaced by $\mathcal{K}'$ in $\hat{\mathcal{K}}$.

Lemma 17. *If two admissible configurations $\mathcal{K}_1$ and $\mathcal{K}_2$ have equal shapes and equal rotations, it follows $\mathcal{K}_1 \sim \mathcal{K}_2$. Conversely, if it is $\mathcal{K}_1 \sim \mathcal{K}_2$, the two configurations have equal shapes and equal rotations.*

Proof. We start with showing that if two configurations $\mathcal{K}_1$ and $\mathcal{K}_2$ have equal shapes and equal rotations, it follows $\mathcal{K}_1 \sim \mathcal{K}_1$. Let G be a graph so that G_1 is a subgraph of G. Let $\mathcal{K}$ be a configuration of G so that the restriction of $\mathcal{K}$ to G_1 equals $\mathcal{K}_1$. We start by deleting $\mathcal{K}_1$ from $\mathcal{K}$. By definition, we can insert the edges $(v^2_{k_2}, s)$, (s, v^2_2), $(v^2_{j_2-1}, t)$ and $(t, v^2_{j_2+1})$ from $\mathcal{K}_2$ into $\mathcal{K}$. Since $\text{rot}(\pi_1(s,t)) = \text{rot}(\pi_2(s,t))$ and $\sigma_1 = \sigma_2$ we know $\text{rot}(\pi_1(t,s)) = \text{rot}(\pi_2(t,s))$. Because of this, we can add all of G_2 into $\mathcal{K}$. To see this, recall that G_2 is connected to the rest of G only through s and t. This means that in the Kandinsky flow network K for G, any flow between G_2 and the rest of G has to flow through s, t and the two faces adjacent to the paths $\pi_2(s,t)$ and $\pi_2(t,s)$. The size of this flow via the two faces depends only on the rotation of the respective path – which is equal in G_1 and G_2. The flow through the vertices s and t depends only on the shape of these incident edges. This leads to a special case. If one path π consists of only one edge with a 270°-angle in the outer face, this angle may not be assigned to 0°-angles at both s and t (remember Lemma 1). This is why we only allow admissible configurations (we deal with prohibited configurations later on).

Following the above argument in reverse order, we realize that if two configurations $\mathcal{K}_1$ and $\mathcal{K}_2$ are related with respect to $\sim$, they have equal shapes and equal rotations. □

As we have seen, the proof for Lemma 17 does not necessarily work for a prohibited configuration $\mathcal{K}$, that is if in $\mathcal{K}$ either path $\pi(s,t)$ or $\pi(t,s)$ is a path of length 1 with a 270°angle in the outer face. For an example, see Figure 4.7.

The solution to this problem is to manually sort such configurations into two equivalence classes, as shown in Figure 4.8. Since all configurations are sorted into a constant number of equivalence classes, all of the following considerations remain correct.

Consider a graph $G = (V, E)$ and a configuration $\mathcal{K}$ of some subgraph of G. As shown in Lemma 17, all configurations $\mathcal{K}' \in [\mathcal{K}]_\sim$ have the following properties:

- $\sigma_{\mathcal{K}'} = \sigma_{\mathcal{K}}$. There are twelve possible values for each of the four variables $\sigma^i_{\mathcal{K}'}$ in $\sigma_{\mathcal{K}'}$, which means that there are $12^4 \in \mathcal{O}(1)$ possible values for $\sigma_{\mathcal{K}'}$.

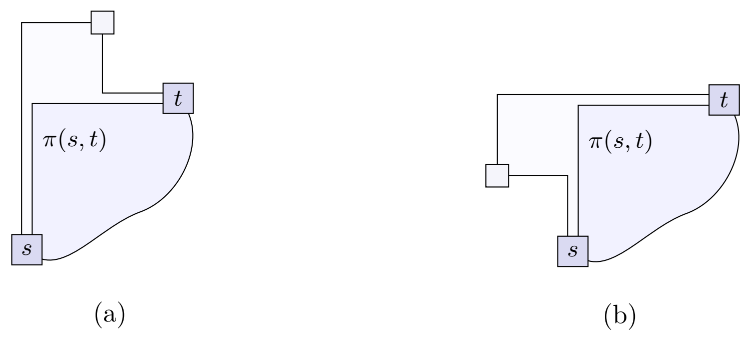


Figure 4.8: The closest shapes for paths prohibited in Lemma 17. The configuration $\mathcal{K}$, shaded in blue, thus belongs to two equivalence classes $\mathfrak{K}_{\sigma,\tau}$.

- $\operatorname{rot}(\pi_{\mathcal{K}'}(s,t) = \operatorname{rot}(\pi_{\mathcal{K}}(s,t)) = \tau$. Since all edges have at most one bend and there are at most $n = |V|$ vertices in the graph associated with $\mathcal{K}'$, the minimum and maximum values for τ lie in $\mathcal{O}(n)$.

This means that there are $\mathcal{O}(n)$ equivalence classes of $\sim$. We refer to an equivalence class $[\mathcal{K}']_\sim$ with $\mathfrak{K}_{\sigma,\tau}$. Obviously, it is possible to compute which equivalence class a configuration belongs to in $\mathcal{O}(n)$ time: σ can be computed in constant time and τ in linear time.

Next, we show how to find simple representantives of an equivalence class $\mathfrak{K}_{\sigma,\tau}$. Start with six nodes $v_k, s, v_2, v_{j-1}, t, v_{j+1}$ and four edges $e_1 = (v_k, s)$, $e_2 = (s, v_2)$, $e_3 = (v_{j-1}, t)$, $e_4 = (t, v_{j+1})$. Shape the edges e_i so that the equation $\operatorname{shape}(e_i) = \sigma_i$ holds true. Now, add a set of nodes $\{v_3, v_4, \ldots, v_{j-2}\}$ and a set of edges $\{(v_i, v_{i+1}) \mid 2 \leq i < j-1\}$ in such a way that for the resulting path $\pi = (s, v_2, v_3, \ldots, v_{j-1}, t)$ it is $\operatorname{rot}(\pi) = \tau$. This can be achieved in a number of ways. For example, we might choose that all angles in the outer face formed between two successive edges on π are 180° and that all added edges have exactly one bend in the same direction. More formally, in terms of the orthogonal representation, we require $h_e = (e, s, 180)$ for all edge descriptions $h_e \in H_f$ of all edges $e \in \{(v_i, v_{i+1}) \mid 2 \leq i < j-1\}$, where s may globally be either `0` or `1` for all edges and f denotes the outer face. Note that from $\tau \in \mathcal{O}(n)$ it follows that $j \in \mathcal{O}(n)$. We can use a similar procedure to add a path $\bar{\pi} = (v_{j+1}, v_{j+2}, \ldots, v_{k-1}, v_k)$. The resulting graph has linear size and can be constructed in linear time.

Since there are $\mathcal{O}(n)$ equivalence classes of $\sim$ and the construction of a small representative of an equivalence class is feasible in $\mathcal{O}(n)$ time, we require $\mathcal{O}(n^2)$ time to build a list of one small representative for each equivalence class.

Next, we build a lookup table A with one entry $A[\mathfrak{K}_{\sigma_i,\tau_j}, \mathfrak{K}_{\sigma_j,\tau_j}]$ for every combination of two equivalence classes $\mathfrak{K}_{\sigma_i,\tau_i}$ and $\mathfrak{K}_{\sigma_j,\tau_j}$.

The value of $A[\mathfrak{K}_{\sigma_i,\tau_i}, \mathfrak{K}_{\sigma_j,\tau_j}]$ is a tuple $(\mathfrak{K}_{\sigma_s,\tau_s}, \mathfrak{K}_{\sigma_p,\tau_p})$, where $\mathfrak{K}_{\sigma_s,\tau_s}$ is the equivalence class so that by performing a series composition with a configuration in $\mathfrak{K}_{\sigma_i,\tau_i}$ and a configuration in $\mathfrak{K}_{\sigma_j,\tau_j}$ we can generate a configuration in $\mathfrak{K}_{\sigma_s,\tau_s}$. Likewise, $\mathfrak{K}_{\sigma_p,\tau_p}$ is an equivalence classes so that by performing a parallel composition with a configuration in $\mathfrak{K}_{\sigma_i,\tau_i}$ and a configuration in $\mathfrak{K}_{\sigma_j,\tau_j}$ we can generate a configuration in $\mathfrak{K}_{\sigma_p,\tau_p}$. If configurations in $\mathfrak{K}_{\sigma_i,\tau_i}$ and $\mathfrak{K}_{\sigma_j,\tau_j}$ cannot be merged in series or in parallel, we write $\mathfrak{K}_{\sigma_s,\sigma_s} = \mathfrak{K}_{\emptyset}$ or $\mathfrak{K}_{\sigma_s,\sigma_s} = \mathfrak{K}_{\emptyset}$, respectively.

Since there are $\mathcal{O}(n)$ equivalence classes there are $\mathcal{O}(n^2)$ entries in A. A value $A[\mathfrak{K}_{\sigma_i,\tau_i}, \mathfrak{K}_{\sigma_j,\tau_j}]$ can be computed in constant time: First, consider merging two configurations $\mathcal{K}_1 \in \mathfrak{K}_{\sigma_i,\tau_i}$ and $\mathcal{K}_2 \in \mathfrak{K}_{\sigma_j,\tau_j}$ in series. We only have to take into account the shapes σ_i and σ_j to find out whether this is possible. For $\{\alpha, \beta, \gamma\} \subset [0, 11]$, we define the relationship:

$$\alpha \leq_\circ \beta \leq_\circ \gamma \Leftrightarrow \beta \in \{(\alpha + \delta) \bmod 12 \mid \delta \in [0, \min\{\epsilon \mid (\alpha + \epsilon) \bmod 12 = \gamma\}]\}$$

Let $\mathcal{K}_i$ and $\mathcal{K}_j$ be configurations of two graphs with the poles $\{s_i, t_i = v\}$ and $\{s_j = v, t_j\}$, that is they share a pole v. These configurations can be merged in series at v if the equation $\sigma_j^1 \leq_\circ \sigma_j^2 \leq_\circ \sigma_i^3 \leq_\circ \sigma_i^4 \leq_\circ \sigma_j^1$ holds true. For a visualization, see Figure 4.9a. Obviously, checking this condition is possible in constant time.

Now, let $\mathcal{K}_i$ and $\mathcal{K}_j$ be configurations of two graphs with the shared poles $s_i = s_j = s$ and $t_i = t_j = t$. We try to merge these configurations in parallel so that $\pi_i(s,t)$ and $\pi_j(t,s)$ lie on the outer face of the merged configuration, and $\pi_i(t,s)$ and $\pi_j(s,t)$ lie on the boundary of a newly created inner face of the merged configuration. This is possible if the following criteria are met:

1. The edges do not conflict at s, that is $\sigma_i^1 \leq_\circ \sigma_i^2 \leq_\circ \sigma_j^1 \leq_\circ \sigma_j^2 \leq_\circ \sigma_i^1$.
2. The edges do not conflict at t, that is $\sigma_i^3 \leq_\circ \sigma_i^4 \leq_\circ \sigma_j^4 \leq_\circ \sigma_j^4 \leq_\circ \sigma_i^3$.
3. The newly created inner face of the merged configuration must be a rectilinear polygon. If e_i and e_i' (e_j and e_j') are the first and last edges of the path $\pi_i(t,s)$ ($\pi_j(s,t)$), this is the case if the following equation holds:

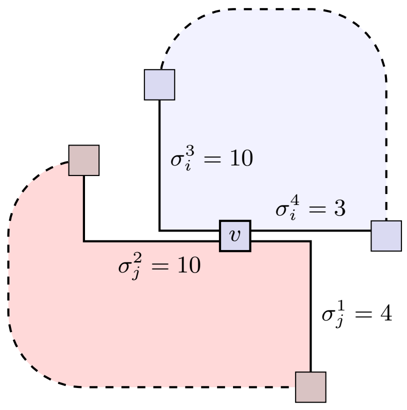
$$\operatorname{rot}(\pi_i(t,s)) + \operatorname{rot}(e_i', e_j) + \operatorname{rot}(\pi_j(s,t)) + \operatorname{rot}(e_j', e_i) = 4$$

For a visualization of properties one and two, see Figure 4.9b.

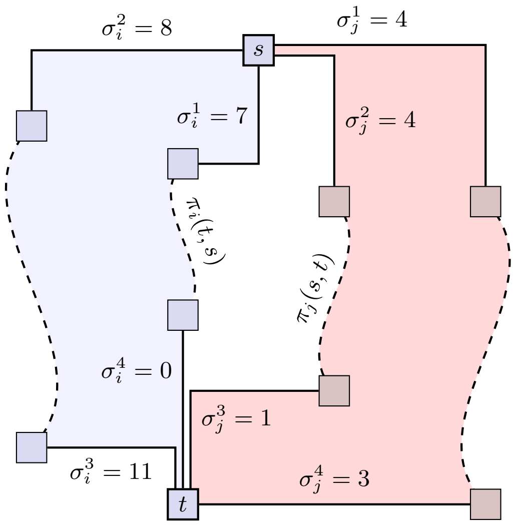
Again, checking these criteria is possible in constant time. To see this, first note that $\operatorname{rot}(\pi_j(s,t)) = \tau_j$ and τ_j is a known value. Next, both $\operatorname{rot}(e_i', e_j)$ and $\operatorname{rot}(e_j', e_i)$ can be computed in constant time from σ_i and σ_j. Finally, $\operatorname{rot}(\pi_i(t,s))$ can be computed from τ_i and σ_i.

To find the values σ and τ for the composite configuration, constant time is needed: It is $\sigma = (\sigma_j^1, \sigma_i^2, \sigma_i^3, \sigma_j^4)$ and $\tau = \tau_i$. This means that A can be computed in $\mathcal{O}(n^2)$ time.

Now, consider the series-parallel decomposition tree $\mathcal{T}$ of a plane series-parallel graph G. We start by generating all possible configurations for every Q-node (note that in $\mathcal{T}$ all leaves are Q-nodes). Store the number of bends required (since we're dealing with 1-Embeddability, this value will be either 0 or 1) in a new table, or ∞ if no suitable configuration exists. Clearly, this is possible in linear time. Now, for any S- or P-node μ, consider all configurations $\mathcal{K}_{1,i}$ and $\mathcal{K}_{2,i}$ of the child graphs graph(μ') and graph(μ''). To find out whether $\mathcal{K}_{1,i}$ and $\mathcal{K}_{2,i}$ may be merged in series or in parallel, simply look up the value $A[[\mathcal{K}_{1,i}]_\sim, [\mathcal{K}_{2,i}]_\sim]$. If it is possible to merge the configurations, calculate the number of bends required by looking up the stored values and adding them, then, if the



(a) Series composition of two configurations which share a pole v. The relationship $\sigma_j^1 \leq_\circ \sigma_j^2 \leq_\circ \sigma_i^3 \leq_\circ \sigma_i^4 \leq_\circ \sigma_j^1$ has to hold true.



(b) Parallel composition of two configurations which share both poles s and t. The relationships $\sigma_i^1 \leq_\circ \sigma_i^2 \leq_\circ \sigma_j^1 \leq_\circ \sigma_j^2 \leq_\circ \sigma_i^1$ and $\sigma_i^3 \leq_\circ \sigma_i^4 \leq_\circ \sigma_j^3 \leq_\circ \sigma_j^4 \leq_\circ \sigma_i^1$ have to hold true.

Figure 4.9

new value is smaller than the previous value, store the new configuration and the new value in the table. Clearly, this operation is feasible in $\mathcal{O}(n^2)$ time.

Recall Lemma 16. Since $\mathcal{T}$ has linear size, the entire algorithm has a runtime in $\mathcal{O}(n^3)$. Furthermore, since we calculate all possible configurations, the ideal configuration will be found by this procedure.

This leads to the following theorem:

Theorem 7. *The optimization problem* 1-OptimalKandinskyDraw *can be solved in* $\mathcal{O}(n^3)$ *time for series-parallel graphs.*

5 Solving OptimalKandinskyDraw Using Linear Programming

We can use linear programming to solve the Kandinsky flow network problem, and thereby the optimization problem OptimalKandinskyDraw.

Definition A Kandinsky flow network $K = (N, A, \mathrm{dem}, \mathrm{cap}, \mathrm{cost}, B)$ is a standard flow network with additional bundle constraints B. For every arc $a \in A$ we introduce a variable X_a. For every variable X_a add an equation which ensures that the capacity constraint is satisfied:

$$0 \leq X_a \leq \mathrm{cap}(a)$$

For every node $v \in N$ add an equation which ensures that the demand constraint is satisfied:

$$\sum_{(u,v)\in A} X_{(u,v)} \quad - \sum_{(v,w)\in A} X_{(v,w)} = \mathrm{dem}(v)$$

For every bundle $C = (\mathcal{C}, c) \in B$ add an equation which ensures that the bundle constraint is satisfied:

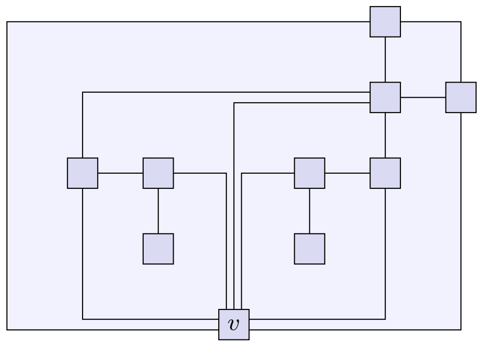
$$\sum_{a\in\mathcal{C}} X_a \leq c$$

Finally, define the cost of the solution $X = \{X_a \mid a \in A\}$:

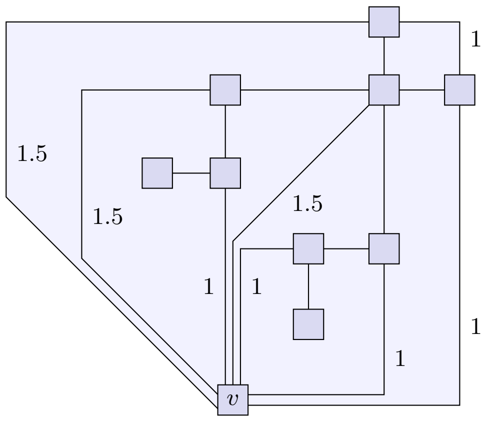
$$\mathrm{cost}(X) = \sum_{a\in A} X_a \cdot \mathrm{cost}(a)$$

Note that all equations are linear. Thus, the Kandinsky minimum cost flow problem may indeed be formulated as a linear program.

Results Linear programming can now be used to find a solution X so that $\mathrm{cost}(X)$ is minimized. From this solution, we can construct a flow by using the relationship $\mathrm{flow}(a) = X_a$.



(a)



(b)

Figure 5.1: Drawings of a graph. The drawing in (a) is a bend-minimized Kandinsky drawing with integer flow. The drawing in (b) is a bend-minimized Kandinsky drawing with non-integer flows. Edges e in (b) are annotated with the amount of flow flowing over arcs asociated with e in the corresponding Kandinsky flow network. In (b), the value 1 for the unbent edge stems from the fact that in the flow network, half a unit flow over arcs associated with the edge in opposing directions, thus “cancelling” out each other.

In the Kandinsky flow network, all flows must have integer values, which means that we require $X_a \in \mathbb{Z}$. The linear program then becomes an *integer* linear program. As shown by Garey and Johnson [8], in general integer linear programming is $\mathcal{NP}$-complete.

However, we might decide to relax the requirements and allow non-integer solutions as well, that is $X_a \in \mathbb{R}$. In this case, the problem becomes solvable in polynomial time. It is interesting to examine whether a non-integer solution allows us to make meaningful statements about the integer solution as well. We investigate whether there is a constant $c \in \mathbb{R}$ so that for a given solution X_F of the linear program we can state $X_I \leq X_F + c$ for solutions X_I of the integer linear program.

Such a constant does not exist. Consider the graph in Figure 5.1. Figure 5.1a is a bend-minimal Kandinsky drawing of the graph with 10 bends. In Figure 5.1b, non-integer solutions are permitted, which leads to a solution with cost 9.5. Let G_1 be the plane graph shown in Figure 5.1. Recursively define a family of graphs $\mathcal{G} = \{G_1, G_2, \ldots\}$ by $G_n = G_{n-1} \cup G_1 \cup \{v_{G_{n-1}}, v_{G_1}\}$. Then $G_n \in \mathcal{G}$ has cost $n \cdot 10$ in the case of an integer solution and $n \cdot 9.5$ in the case of a non-integer solution.

This leads to the following theorem:

Theorem 8. *There is a family $\mathcal{G}$ of plane graphs so that for any constant $c \in \mathbb{R}$ there is a graph $G \in \mathcal{G}$ so that $X_I - X_F > c$, where X_I is the solution for the integer linear program for K_G and X_F is the solution for the linear program.*

Acknowledgements During the experimenting phase leading to the results presented in this chapter, *OGDF* [2], the *Open Graph Drawing Framework* was used for creating and manipulating graphs and *Gurobi Optimizer* [1] was used for solving linear programs.

6 Conclusion

In this thesis, we studied various aspects of Kandinsky drawings.

Contribution First, in Chapter 3, we have seen that due to the bend-or-end property of the Kandinsky model, the decision problem 0-Embeddability for Kandinsky drawings is equivalent to 0-Embeddability for lower-degree graphs. It has been shown that for fixed embeddings, the problem is solvable in polynomial time and that for variable embeddings, the problem becomes $\mathcal{NP}$-complete. Next, we demonstrated how to find 1-bend Kandinsky drawings of any simple planar graph, and we showed that solving 1-Embeddability with variable embeddings is $\mathcal{NP}$-complete. Furthermore, we presented a linear-time algorithm for finding 2-bend drawings of any planar graph.

Next, in Chapter 4, we gave a linear-time algorithm for finding bend-minimal Kandinsky drawings of biconnected, outerplanar, inner-triangulated graphs. Additionally, we showed how to find bend-minimal 1-bend Kandinsky drawings of series-parallel graphs in $\mathcal{O}(n^3)$ time.

Finally, in Chapter 5, we explored a new approach to solving OptimalKandinsky-Draw. We found that for any constant $c \in \mathbb{R}$, we can find a graph so that the difference between the linear solution and the integer solution of the linear program corresponding to the OptimalKandinskyDraw instance is greater than c.

Outlook With many questions answered, lots of new questions have been raised.

We have presented an algorithm finding bend-minimal 1-bend Kandinsky drawings in $\mathcal{O}(n^3)$ time. Can this runtime be improved? Is it possible to use a similar approach for solving β-OptimalKandinskyDraw for series-parallel graphs? And finally, can we generalize the flexibility of edges, thereby solving OptimalKandinskyFlexDraw?

Then, we showed that any planar simple graph is 1-embeddable. But some planar multigraphs are 1-embeddable as well. Is there a simple characterization for these 1-embeddable multigraphs?

Moreover, it might be interesting to further investigate the use of linear programming for solving OptimalKandinskyDraw. Specifically, given a real solution X_F, what kind of statement can we make about the integer solution X_I? We have shown that there is a family of graphs for which the equation $X_I = 10/9.5 \cdot X_F$ holds true. Is there some constant $c \in \mathbb{R}$ so that the equation $X_I \leq c \cdot X_F$ holds true for all graphs?

With so many open questions, higher-degree orthogonal graph drawing remains a hot research topic.

Bibliography

[1] Gurobi Optimizer. `http://www.gurobi.com`. Accessed 09/30/2013.

[2] Open Graph Drawing Framework. `http://www.ogdf.net`. Accessed 09/30/2013.

[3] Giuseppe Di Battista and Roberto Tamassia. On-line maintenance of triconnected components with SPQR-trees. *Algorithmica*, 15(4):302–318, 1996.

[4] Thomas Bläsius, Marcus Krug, Ignaz Rutter, and Dorothea Wagner. Orthogonal grahp drawing with flexibility constraints. *Algorithmica*, 2012.

[5] Markus Eiglsperger, Carsten Gutwenger, Michael Kaufmann, Joachim Kupke, Michael Jünger, Sebastian Leipert, Karsten Klein, Petra Mutzel, and Martin Siebenhaller. Automatic layout of UML class diagrams in orthogonal style. *Information Visualization*, 3(3):189–208, 2004.

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

[7] Ulrich Fößmeier and Michael Kaufmann. Drawing high degree graphs with low bend numbers. In Franz Brandenburg, editor, *Graph Drawing: Symposium on Graph Drawing, GD95*, Lecture Notes in Computer Science, 1996.

[8] Michael R. Garey and David S. Johnson. *Computers and Intractability.* W. H. Freeman & Co. New York, NY, USA, 1979.

[9] Ashim Garg and Roberto Tamassia. On the computational complexity of upward an rectilinear planarity testing. *SIAM Journal on Computing*, 31(2):601–625, 2001.

[10] Andrew V. Goldberg and Robert E. Tarjan. Finding minimum-cost circulations by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[11] Carsten Gutwenger and Petra Mutzel. A linear time implementation of SPQR-trees. In Joe Marks, editor, *Graph Drawing: Sumposium on Graph Drawing, GD2000*, Lecture Notes in Computer Science, 2001.

[12] Carsten Gutwenger, Petra Mutzel, and René Weiskircher. Inserting an edge into a planar graph. In *Algorithmica*, pages 246–255. ACM Press, 2000.

[13] Charles Eric Leiserson. *Area efficient VLSI computation.* PhD thesis, Carnegie Mellon University, 1981.

[14] Robin S. Liggett and William J. Mitchell. Optimal space planning in practice. *Computer-Aided Design*, 13(5):277–288, 1981.

[15] Roberto Tamassia. On embedding a graph in the grid with the minimum number of bends. *SIAM Journal on Computing*, 16(3):421–444, 1987.

[16] Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Transactions on Computers*, C-30(2):135–140, 1981.

[17] Hassler Whitney. Non-separable and planar graphs. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 17, pages 122–125, 1931.

[18] Canan Yildiz. *Knickminimales Orthogonales Zeichnen Planarer Graphen im Kandinsky Modell.* PhD thesis, Technische Universität Wien, 2005.