Florian Ebenführer, BSc

Realizability of Rotation Systems

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematical Computer Science

submitted to

Graz University of Technology

Supervisor

Assoc.Prof. Dipl.Ing. Dr.techn. Oswin Aichholzer

Institute of Software Technology

Second Supervisor

Ass.Prof. Dipl.-Ing. Dr.techn. Birgit Vogtenhuber

Graz, March 2017

Dedicated to Mia Sophie Scharrer

Acknowledgments

First of all I want to thank my supervisor, Oswin Aichholzer, for his commitment, support, and patience during my work on this thesis. I also like to express my gratitude to my co-supervisor Birgit Vogtenhuber, who was so kind to read through my thesis and helped me to improve the quality of my work. Another person I like to thank is Irene Parada for discussing many ideas and results regarding the content of my thesis. At this point I like to appreciate the many hours of discussion together with Oswin Aichholzer, Birgit Vogtenhuber, and Irene Parada. The subjects of this thesis derive from joint research with these three persons.

Most of all I want to thank my parents, Manfred and Eva, for their immense support during my years as a student. Without their help, my education at the university would not have been possible.

I also want to thank Johanna. I am very grateful for your encouragement throughout my whole years of study.

Abstract

In this thesis we consider good drawings and semi-good drawings of the complete graph. These are special classes of graph drawings, which have restricted crossing properties. These properties include that two edges sharing a vertex must not cross. We investigate a tool to analyze these drawings, called the rotation system. It is a fact that the rotation system determines a good drawing uniquely with respect to some special properties.

We analyze rotation systems with respect to their realizability as good or semi-good drawings. We argue that it might be very difficult to generate a random good-drawing. Then we inspect how we can modify rotation systems to keep some properties that are necessary for realizability. We introduce such an operation and analyze its applications for generating random good drawings.

In addition to good drawings we consider semi-good drawings. We show that for a rotation system to be drawable as a semi-good drawing it is not sufficient that every sub-rotation system of four vertices is realizable. We present an example that fulfills this requirement and prove that it is not semi-realizable.

Last but not least, we show some related results regarding semi-good drawings.

Zusammenfassung

In dieser Arbeit befassen wir uns mit “Good Drawings” und “Semi-Good Drawings” des vollständigen Graphens. Das sind spezielle Klassen von Zeichnungen von Graphen, die eingeschränkte Kreuzungseigenschaften besitzen. Diese Eigenschaften beinhalten, dass sich zwei Kanten die einen gemeinsamen Knoten besitzen nicht kreuzen dürfen. Wir beschreiben eine Methode um solche Zeichnungen zu analysieren. Dazu verwenden wir das “Rotation System”. Es ist bekannt, dass ein “Good Drawing” durch sein “Rotation System”, bezogen auf gewisse Eigenschaften, eindeutig bestimmt ist.

Wir analysieren “Rotation Systems” und deren Eigenschaften, die dafür verantwortlich sind ob sie als ein “Good Drawing” beziehungsweise ein “Semi-Good Drawing” zeichenbar sind. Wir argumentieren, dass das Erzeugen von zufälligen “Rotation Systems” die als “Good Drawings” realisierbar sind kein einfaches Problem ist. Weiters untersuchen wir, wie wir “Rotation Systems” verändern können um gewisse Eigenschaften beizubehalten, die für die Zeichenbarkeit notwendig sind.

Zusätzlich zu “Good Drawings” beschäftigen wir uns mit “Semi-Good Drawings”. Wir zeigen, dass es für ein “Rotation System” nicht ausreicht, dass alle “Teil-Rotation Systems” aus vier Knoten zeichenbar sind, um als ein “Semi-Good Drawing” zeichenbar zu sein. Wir präsentieren ein Beispiel das diese Voraussetzung erfüllt und beweisen formal, dass es keine Realisierung als “Semi-Good Drawing” besitzt.

Zum Schluss präsentieren wir einige zusätzliche Resultate zu “Semi-Good Drawings”.

Contents

1 Introduction

In 1736, Leonard Euler stated the historical problem of the *Seven Bridges of Königsberg* and laid therefore the foundations of graph theory [26]. Euler lived in the town Königsberg, which was located on four islands that were connected by seven bridges. He asked whether there is a walk through the town, which crosses each of those bridges exactly once, and which ends at the starting point. To solve this problem, Euler reduced it to its essentials. The only important aspects in this problem are the islands, the bridges, and how the bridges connect the islands. These are the basic components of an abstract graph.

Definition 1.1 (Graph) *[10, p. 1] A graph is an ordered pair $G = (V, E)$ consisting of two sets V and E. The set V contains the vertices of the graph. The elements in the set E are 2-element subsets $\{v, w\}$ of V called edges and connect the two vertices v and w.*

We call an edge and a vertex *incident*, if the edge has the vertex as an endpoint. We call a pair of vertices *adjacent*, if they are connected by an edge. Whenever the graph G is not obvious from the context, we denote the set of vertices of G by $V(G)$ and its set of edges by $E(G)$.

Definition 1.2 (Cycle) *[10, p. 2] A cycle in a graph $G = (V, E)$ is a sequence*

$v_1, v_2, \dots, v_k$ of vertices of V such that $\{v_i, v_{i+1}\}$ is an edge for $1 \leq i < k$ and $v_1 = v_k$.

The field of *Graph Drawing* deals with the representation of abstract graphs as drawings in the plane. Usually, the vertices are drawn as points and the edges are drawn as curves connecting two points. We call such a representation an embedding or a drawing of the graph in the plane. For an abstract graph, infinitely many different representations are possible. For instance, we can select an arbitrary set of points for the vertices of the graph and draw the edges of a graph with straight lines. An important property of every drawing of an abstract graph is the number of crossings its edges produce. If a graph is drawable without crossings, we call the graph *planar*.

Definition 1.3 (Planar Graph) *[10, p. 3] A graph is planar if it can be drawn in the plane without crossing edges.*

The property that a planar graph can be drawn crossing-free in the plane also holds for a drawing on the sphere.

Definition 1.4 (Complete Graph / Complete Bipartite Graph)
We call a graph G a complete graph if for every two vertices $v, w \in V, v \neq w$, there exists an edge $e = \{v, w\} \in E$. The complete graph with n vertices is denoted with K_n.

We call a graph G a complete bipartite graph if its set of vertices can be partitioned into two sets V_1 and V_2 such that $E = V_1 \times V_2$. The complete bipartite graph with m vertices in the first set and n vertices in the second set is denoted by $K_{m,n}$.

In 1930 Kuratowski studied the properties that lead to the fact that a graph is not planar. He came up with two minimal graphs, which are not planar

[10, p. 5]. The first one is K_5, the complete graph with five vertices. The second one is $K_{3,3}$, the complete bipartite graph with three vertices in each partition. Kuratowski showed that these two minimal examples are essentially the only possibilities that prevent a graph from being drawable crossing-free. More exactly, he showed that a graph G is planar if and only if G contains no subgraph that is a subdivision of K_5 or $K_{3,3}$. A subdivision of a graph G is obtained by placing new vertices of degree two on the edges of G.

This means that for every $n \geq 5$, K_n has no crossing-free embedding in the Euclidean plane. One goal for a nice drawing often is to have as few crossings as possible. One area of research in the field of graph drawing is to analyze how many crossings a drawing of a graph needs at least. For this purpose we have to define which kind of crossings we count and which drawings we consider, because there exist many different definitions [21]. For instance we could only consider drawings with straight lines and the minimal number of crossings among all these drawings. In this thesis we define the crossing number as in [21] and count all crossings and consider all drawings.

Definition 1.5 (Crossing Number) *[21] The crossing number $cr(G)$ of a graph G is the minimum number of crossings among all drawings of G. The number of crossings of a drawing $D(G)$ is the number of all proper intersections of two edges.*

The first who came up with a conjecture for the crossing number of the complete graph with n vertices was Hill. His conjecture consisted of two upper bounds, according to the parity of n. Hill and Harary stated this conjecture in their paper in 1962 [12]. Their two formulas were then combined to a single formula by Blažek and Koman who proved that this is indeed an upper bound for the crossings number of K_n [9]. Guy proved the conjecture for complete graphs with up to ten vertices [11].

Conjecture 1.1 (Harary-Hill-Guy Conjecture) *The crossing number $cr(K_n)$ of the complete graph K_n satisfies the following equality:*

$$cr(K_n) = \frac{1}{4} \left\lfloor \frac{n}{2} \right\rfloor \left\lfloor \frac{n-1}{2} \right\rfloor \left\lfloor \frac{n-2}{2} \right\rfloor \left\lfloor \frac{n-3}{2} \right\rfloor$$

It took almost 50 years until in 2007 Pan and Richter proved that the conjecture is true for the complete graph with up to twelve vertices [23]. For 13 vertices Aichholzer et al. showed that the crossing number is $cr(K_{13}) \geq 223$ [1]. The result for 13 vertices used a database that enumerates all good drawings of the complete graph with up to nine vertices. To enumerate the drawings, they made use of the definition of the rotation system of a drawing, which identifies any drawing uniquely with respect to its crossing properties. Up to now it is not known if Conjecture 1.1 is true for graphs with arbitrary size however, during the last few years, the conjecture has been proven for many classes of drawings [2]. We note that already the crossing number $cr(K_8) = 18$ cannot be achieved with straight line drawings [12]. To achieve this number we need curved edges.

1.1 Outline of the Thesis

In Chapter 2 we start with some basic definitions from the field of Graph Drawing, which we will need throughout the thesis. We define, among others, the important terms of good and semi-good drawings, as well as the concept of (realizable) rotation systems.

In Chapter 3 we motivate that realizable rotation systems are very rare among all rotation systems of the same size and that it might be very difficult to generate them at random. We analyze different approaches of how to generate a random realizable rotation system. Then we introduce an operation for making local changes in rotation systems without violating certain properties. We call this operation a *good double-flip*.

In Chapter 4 we consider good double-flips in detail. We show what is possible with these flips in terms of relabeling a good drawing. We also present our results concerning the connectivity of the according flipgraph of all realizable rotation systems.

Chapter 5 is dedicated to the class of semi-good drawings. We present how we are able to decide whether a given rotation system is drawable as a semi-good drawing and present a rotation system for which we show that it is not drawable as a semi-good drawing, although it has some good properties.

Finally, in Chapter 6 we summarize some related results we obtained during the work on this thesis and give a conclusion about our results.

2 Preliminaries

2.1 Good Drawings

A *drawing* of a graph is an embedding of it in the Euclidean plane or on the sphere, in the following intuitively nice way.

Definition 2.1 (Drawing) *[20] A drawing of a graph in the plane is a mapping f with the following properties.*

- *The vertices are mapped to distinct points in the plane.*
- *Every edge (u, v) is mapped to a non-self-intersecting curve that connects the two points $f(u)$ and $f(v)$ and does not pass through the image of any other vertex.*
- *Every intersection of two curves is either an image of a common vertex or a proper crossing, which means that the curve passes from one side of the second curve to its other side, thereby (locally) intersecting the other curve in exactly one point.*

For simplicity, wherever it is clear from the context, we identify vertices and edges with the points and curves they are mapped to and also denote them in the same way. In general, we do not distinguish between a vertex/edge and the point/curve it is mapped to.

Definition 2.2 (Cell) *A drawing of a graph decomposes the plane into connected regions. We denote these regions as cells.*

Definition 2.3 (Good Drawing) *[7] A good drawing of a graph is a drawing of the graph with the property that every pair of edges in the drawing intersects in at most one point (either in the interior of both edges, forming a proper crossing, or at a common end-point).*

Let us have a look at this definition in detail. A *good drawing* is a natural definition of a nice drawing, since violations of its rules are locally resolvable with a reduction of the number of crossings. The definition of a good drawing gives us a natural restriction to drawings that are possible relevant for crossing minimizing problems. That is two edges intersect in at most one point. The left example in Figure 2.1 is no good drawing, because the two edges intersect in a crossing and in one of their endpoints. We can change the order of the edges incident to vertex 1 and draw this example without crossings. This is the first indication that the order in which the edges leave the vertices plays a role for good drawings.

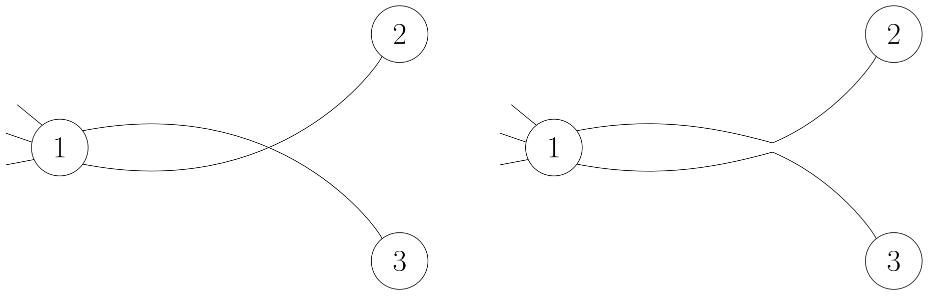


Figure 2.1: Example how to exchange the order to remove crossings

The second important property of a good drawing is that two edges must not cross two or more times. Figure 2.2 shows a drawing where this rule is violated. This drawing can be redrawn in a way that the two edges cross only once. This can be done without changing the order of the edges at the vertices.

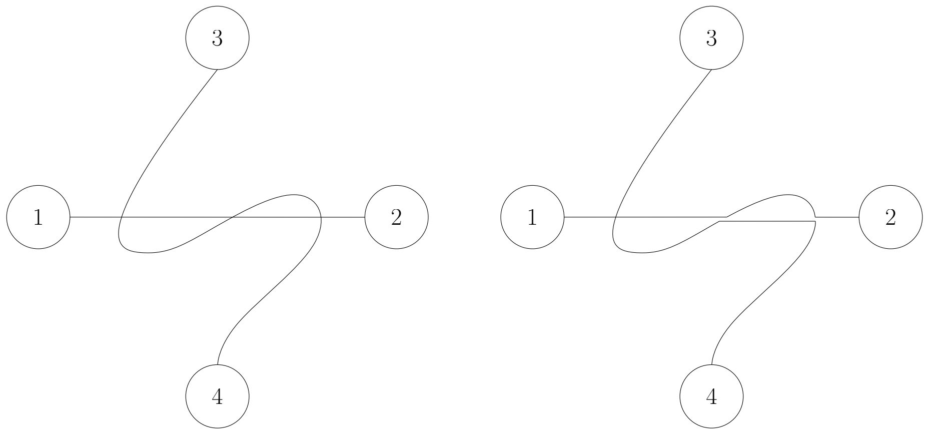


Figure 2.2: Example how to switch the path to remove multiple crossings

This possibility of locally resolving crossings motivates that good drawings are indeed the only drawings that are possible relevant for crossing minimizing drawings.

When dealing with drawings of the complete graph, we have the problem that there are infinitely many different drawings of K_n. One approach for attacking this problem is to group this infinite number of drawings of K_n into groups that are equivalent with respect to their crossing properties. So we have to clarify how to compare good drawings. Looking at the drawings of K_4 in Figure 2.3, we could say that these are two different drawings. The left one is a drawing with straight edges and the right one is a drawing with curved edges. However, when looking at them in a more abstract way, these two drawings are the same in some way: Both drawings have the same amount of crossings and even the same edges cross, namely, edge $(1,4)$ and edge $(2,3)$. Another important indicator for their similarity is that both drawings split the plane or the sphere into the same cells. This means that every cell in the left drawing corresponds to a cell in the right drawing that is bounded by the same edges in the same order. The third important identity of these two drawings is that the order of the edges at every vertex is the same in both drawings. In a formal mathematical sense, we distinguish between two types of similarity of good drawings, namely, weak isomorphism and strong isomorphism.

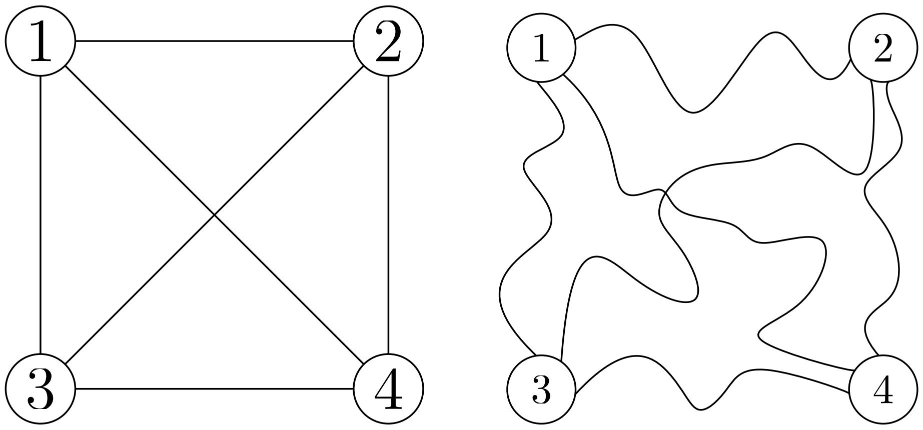


Figure 2.3: Two isomorphic drawings of K_4

Definition 2.4 (Weak Isomorphism) *[16] Two good drawings G and H are weakly isomorphic if there exists an incidence preserving one-to-one correspondence between $V(G)$ and $V(H)$ such that two edges of G cross if and only if the corresponding two edges of H cross.*

Definition 2.5 (Strong Isomorphism) *[16] Two drawings G and H are strongly isomorphic if there exists a homeomorphism of the sphere, which transforms G into H.*

Remark 2.1 *The two definitions of isomorphism above are for unlabeled drawings.*

Let us consider some examples. In Figure 2.4 we can see another pair of strongly isomorphic good drawings of K_4. Although the definition of isomorphism is given for unlabeled drawings, we give names to the vertices. The names are given in a way that we can see the isomorphism between the vertices easily. Although we did not draw the vertices in the same manner, it can be seen that both drawings have the same crossings. Therefore they are weakly isomorphic. Also, both drawings split the plane, and therefore the sphere, into the same number of cells. A one-to-one correspondence between the cells of the first and the second combinatorial structure can be found, so that every cell is bounded by the same edges in the same order. Therefore both drawings are also strongly isomorphic. We marked this correspondence between the cells with the grey numbers. In the left drawing the cell with four edges on its boundary is unbounded (grey number 5). In the right drawing the cell with the number 5 is bounded. Note that the fact whether a cell is bounded or not does not play a role, because strong isomorphism is defined on the sphere where no unbounded cell exists.

For good drawings with at most five vertices, weak isomorphism and strong isomorphism are equivalent. That means that there exists no pair of good drawings with at most five vertices, which are weakly isomorphic but not strongly isomorphic.

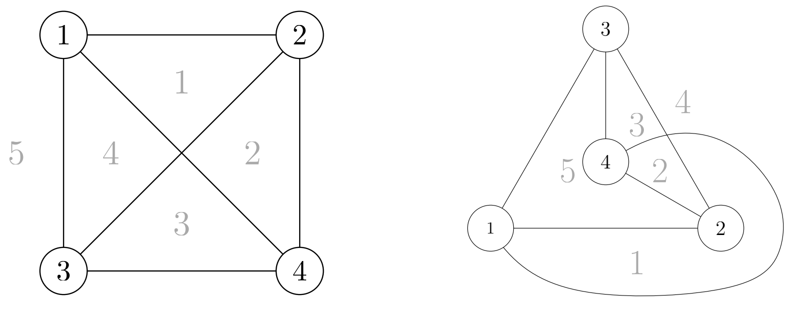


Figure 2.4: Two isomorphic drawings of K_4

Now we concentrate on the order of the edges at the vertices. Since two edges that are incident to the same vertex must not cross, this order influences the way a graph can be drawn. Again, we point this out in an example. Figure 2.5 shows three incomplete drawings of K_4. In every drawing the order of the edges at all vertices is fixed. In the left drawing the missing edge $(2, 3)$ starts at 2 inside the triangle $\triangle 124$. To obtain a good drawing, the only way this edge can leave $\triangle 124$ is by crossing the edge $(1, 4)$, because $(2, 3)$ shares a vertex with the other two edges that are bounding this cell.

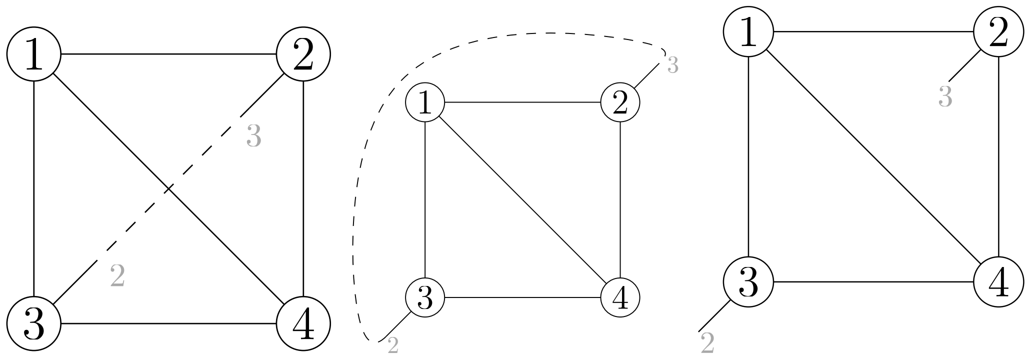


Figure 2.5: Three rotations of vertex 2 and 3 of K_4

In the middle drawing the edge $(2, 3)$ has to start at 2 in the unbounded cell. This edge is not allowed to cross any of the four edges that are bounding this cell. So the only way to connect to vertex 3 is without crossing any edge. Note that it does not play a role whether the edge $(2, 3)$ is drawn on the upper or lower side of the vertices, because strong isomorphism is defined on the sphere. So both cases result in good drawings, which are strongly isomorphic and therefore also weakly isomorphic.

In the right drawing the order of the edges at vertices 2 and 3 is chosen in a way that the edge from 2 to 3 cannot be drawn without violating the conditions for a good drawing. The edge has to start at vertex 3 in the unbounded cell. Since it has to end inside the triangle $\triangle 124$ at vertex 2, it has to cross an edge bounding the unbounded cell. Due to the rules of good drawings this is not possible.

So we have seen that the order of the edges at the vertices influences the way how a graph can be drawn as a good drawing. In fact, in a drawing of K_4, this order defines whether two distinct edges cross or not. (Compare [20]).

Remark 2.2 *For a good drawing, it does not matter how the vertices are arranged in the plane. We can place the vertices in convex position and realize all possible good drawings on this arrangement. This is also the standard method how we present examples of good drawings throughout this thesis.*

For the topics considered in this thesis any good drawings that are weakly isomorphic are essentially the same. Hence, from now on, we consider good drawings as different if and only if they are not weakly isomorphic.

2.2 Rotation Systems

To distinguish good drawings, we use the method of computing its rotation systems, which is a unique representation of it with respect to its crossing properties.

Definition 2.6 (Rotation System) *[16] Let D be a good drawing of a labeled graph G. The rotation of a vertex $v \in V(G)$ is the clockwise cyclic order of all edges incident to v. The rotation system of a good drawing consists of the rotation of every vertex $v \in V(G)$. We denote the rotation system of a good drawing D of G with $\mathcal{R}(D)$.*

We can label the edges that emanate from vertex v by their other end-vertex. So the rotation of a vertex v can be represented as the order of its adjacent vertices. If G is a complete graph, every vertex v is connected to every other vertex and the rotation of v consists of all vertices of G except v itself. Therefore it is a cyclic permutation of the elements of $V(G)\setminus\{v\}$.

In a labeled drawing the rotation of every vertex is uniquely defined. In the following we always represent the rotation of a vertex by listing its adjacent vertices as a tuple in the order the edges emanate from the vertex, starting with the vertex with the alphabetically smallest label. Further, for a drawing with vertices $\{1, \dots, n\}$ we represent its rotation system as the concatenation of the rotations of its vertices, starting with vertex 1 and ending with vertex n. Therefore in a complete graph this gives us a string of $n(n-1)$ numbers.

In 2009, Kynčl stated the following proposition on the connection between rotation systems and good drawings.

Proposition 2.1 (Kynčl 2009) *[14]*

1. *The rotation system of a good drawing of a complete graph uniquely determines, which pairs of edges cross. Therefore it follows that two good drawings of a complete graph with the same rotation system are weakly isomorphic.*
2. *If two good drawings of a complete graph are weakly isomorphic, then their rotation systems are either the same or inverse.*

We say that a pair $\mathcal{R}(D)$, $\mathcal{R}(E)$ of rotation systems is inverse if the rotation of every vertex in $\mathcal{R}(D)$ is the inverse of the corresponding vertex in $\mathcal{R}(E)$. Figure 2.6 shows two weakly isomorphic drawings that have inverted rotation systems. The right drawing is the mirroring of the left one. The rotation system of the right one is then the inverse of the left one. This is the reason why inverse rotation systems are considered in Proposition 2.1.

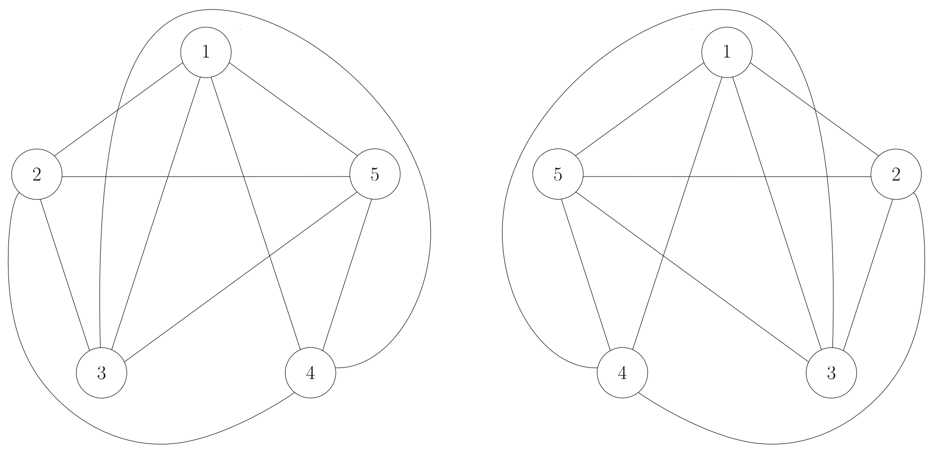


Figure 2.6: Two mirrored drawings of K_5 with inverted rotation systems

Proposition 2.1 gives us a tool to compare and enumerate all weak isomorphism classes of good drawings by concentrating on rotation systems. Note that we can determine whether two edges cross in a good drawing by checking the sub-rotation system of the four incident vertices of the two edges. Proposition 2.1 is stated for labeled drawings. Since we are only interested in the structure of a drawing with respect to its crossings and do not bother about different labelings, we need a well defined rotation system for an unlabeled drawing. This is obtained by the fingerprint of a good drawing.

Definition 2.7 (Fingerprint of a Good Drawing) *[1] Let $D(K_n)$ be an unlabeled good drawing of the complete graph K_n. We label the vertices of the drawing with the values 1 to n. Consider the rotation system of this labeled drawing, represented as a string of $n(n-1)$ numbers. We also take into account the string that we get when we invert the rotation system. The fingerprint of the good drawing* *is the lexicographic minimal string (inverted or non-inverted) among all different labelings of the drawing.*

Since Definition 2.4 of weak isomorphism is defined for unlabeled drawings, relabeling does not change the weak isomorphism class. Definition 2.7 implies that in the fingerprint, the rotation of vertex 1 has the sequence $(2\ 3\ 4\ \ldots\ n)$ and the rotation of every other vertex starts with 1.

While relabeling a labeled good drawing does not affect a good drawing, it does change its rotation system. Similarly, mirroring a labeled good drawing leads to a weakly isomorphic good drawing with inverted rotation system. In analogy to the weak isomorphism class for good drawings we can also define an equivalence relation for rotation systems.

Definition 2.8 (Equivalent Rotation Systems) *[1] Let $\mathcal{R}$ and $\mathcal{P}$ be two rotation systems of labeled drawings of K_n. We say that $\mathcal{R}$ and $\mathcal{P}$ are equivalent, if $\mathcal{R}$ can be obtained from $\mathcal{P}$ by relabeling $\mathcal{P}$ or the inverse of $\mathcal{P}$.*

We denote the set of equivalence classes of rotation systems with n points as $\mathfrak{R}_n$.

By Proposition 2.1, equivalent rotation systems correspond to weakly isomorphic good drawings.

We will use the term fingerprint also for the rotation system that is lexicographically minimal among all equivalent systems. As we defined weakly isomorphic good drawings and equivalent rotation systems, we can speak of the weak isomorphism class of good drawings and the equivalence class of rotation systems. If two drawings are in the same weak isomorphism class, then their rotation systems lie in the same equivalence class. This follows directly from Proposition 2.1. Mostly we will talk about rotation systems and good drawings and refer to their associated classes of equivalent rotation systems and weakly isomorphic good drawings, respectively.

Observation 2.1 *Two unlabeled good drawings are weakly isomorphic if and only if they have the same fingerprint.*

Since good drawings and rotation systems are connected by Proposition 2.1, the relevant properties of a good drawing for our topics can be obtained from the rotation system. Therefore, we will often not distinguish between good drawings and their associated rotation systems. As we have seen in Figure 2.5, the rotation system of K_4 defines whether two distinct edges cross or not. If we know that two edges cross and how they cross, then we can give the rotation system of the four involved vertices. We state this fact as a lemma and need a definition before.

Definition 2.9 (Rotation of a Crossing) *[22, p. 19] Let $D(G)$ be a drawing of a graph G. The rotation of a crossing in $D(G)$ is the cyclic order of the four segments of the two edges involved in the crossing.*

Lemma 2.1 *Consider a labeled good drawing of K_4, where two edges e_1 and e_2 cross. The rotation system of the good drawing is uniquely determined by the two crossing edges and the rotation of the crossing.*

Proof: Figure 2.7 shows all different labeled good drawings of K_4 with one crossing. It can be easily checked that all these drawings have different rotation systems. We can also see that in each drawing two different edges cross, or at least the rotation of the crossing is different. This concludes the proof. □

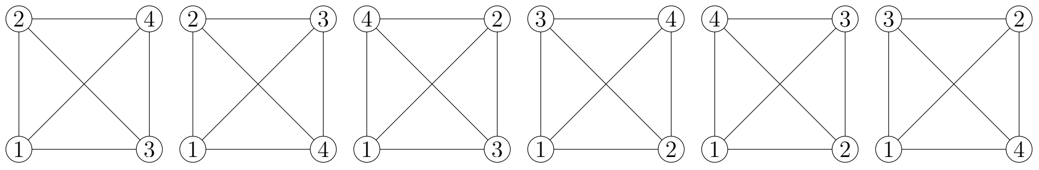


Figure 2.7: All different labeled drawings of K_4 with one crossing

Another important property of a rotation system with four vertices is that in a good drawing it is possible to compute the rotation of the fourth vertex, given the rotations of the other three vertices. Hence, if we want to construct a good drawing with four vertices, we only need the rotations of three vertices. The rotation of the fourth vertex is then determined uniquely. This fact can be easily proven by considering all different rotation systems of good drawings with four vertices, similar to the proof of Lemma 2.1. The following example illustrates the usage of this property for one case. Given is the following incomplete rotation system of the vertices 1 to 3:

$$\begin{aligned} 1 &: \; 2\,3\,4 \\ 2 &: \; 1\,4\,3 \\ 3 &: \; 1\,2\,4 \end{aligned}$$

This incomplete rotation system can only lead to the good drawing given in Figure 2.8. Therefore the rotation of vertex 4 is uniquely determined by

$$4 : \; 1\,3\,2.$$

Up to now, we looked at rotation systems that are derived of good drawings of the complete graph. We can also construct rotation systems without knowing whether there exist good drawings with this rotation systems. In the majority of this work we will look at rotation systems from this point of view. Note that $\mathfrak{R}_n$ contains all equivalence classes of rotation systems, no matter if there is a good drawing having this rotation system. For such a

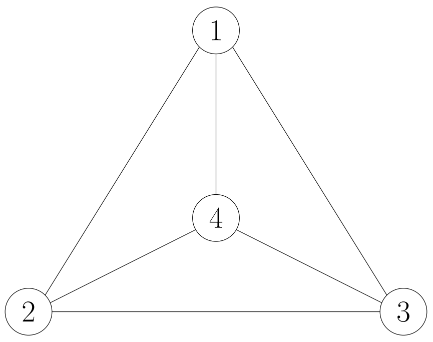


Figure 2.8: Only good drawing of the incomplete rotation system

rotation system we have the problem to decide whether or not there exists a good drawing that has this rotation system. If this is the case, we say that a rotation system is realizable.

Definition 2.10 (Realizability) *[1] Let a rotation system $\mathcal{R}$ be given. $\mathcal{R}$ is said to be realizable if it is a rotation system of a good drawing of a complete graph.*

It is clear that if a rotation system $\mathcal{R}$ is realizable, then all its equivalent rotation systems are realizable too.

2.3 Semi-Good Drawings

In this work we will also deal with rotation systems that are not drawable as a good drawing, but as a *semi-good drawing*.

Definition 2.11 (Semi-Good Drawing) *A semi-good drawing of a graph is a drawing with the property that every two edges that share an incident vertex do not cross.*

Definition 2.12 (Semi-Realizable) *We call a rotation system $\mathcal{R}$ that is drawable as a semi-good drawing semi-realizable.*

In a semi-good drawing, like in a good drawing, two edges that share a vertex are not allowed to cross. In contrast, two distinct edges are now allowed to cross not only once, but multiple times.

We now state a lemma that links the position of a vertex and the rotation system. This lemma holds for good drawings as well as for semi-good drawings. Here it is stated for semi-good drawings.

Definition 2.13 ((Empty) Triangle) *Any sub-drawing of three vertices a, b, c of a good or semi-good drawing of the complete graph forms a closed curve connecting the three vertices. This sub-drawing is called the triangle $\triangle abc$. This triangle decomposes the plane into two connected regions, one is unbounded and one is bounded. If one of these components does not contain any of the remaining vertices of the drawing, then the triangle is called empty.*

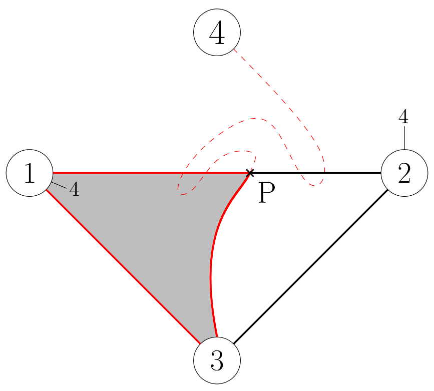


Figure 2.9: When two edges of a triangle start inside the triangle and the vertex is outside

Lemma 2.2 *Given is a rotation system of a semi-good drawing of the complete graph K_n and a triangle of this drawing. This triangle decomposes the plane into* *two regions. For any arbitrary vertex v that is not a vertex of the triangle, the rotation system tells us in which of these two regions v is contained. It is the region in which at least two of the three edges of the sub-drawing to vertex v start.*

Proof: Assume for the sake of contradiction that there exists a triangle $\triangle 123$ and a vertex 4 for which the statement does not hold. Without loss of generality we assume that vertex 4 lies in the unbounded cell of the triangle $\triangle 123$ and that two edges $(1,4)$ and $(3,4)$ start inside $\triangle 123$. This configuration is shown in Figure 2.9. We show that this configuration is not drawable as a semi-good drawing.

Since the edge $(3,4)$ has to leave the triangle $\triangle 123$ and must not cross any edge incident to 3, it has to cross the edge $(1,2)$. We call the first crossing point P. Then the curves $(1,3)$, $(3,P)$, and $(1,P)$ bound a cell inside which the edge $(1,4)$ starts at vertex 1. This cell lies completely inside the triangle $\triangle 123$. Therefore vertex 4 does not lie in this cell. Since the edge $(1,4)$ must not cross any of the edges incident to 1 and 4, it is not allowed to cross any of the curves $(1,3)$, $(3,P)$, and $(1,P)$. Thus, this configuration is not drawable as a semi-good drawing, which completes the proof. $\square$

3 Generating Random Realizable Rotation Systems

If we want to analyze good drawings of K_n with respect to their crossing properties we can instead analyze realizable rotation systems. We know that the number of equivalence classes of realizable rotation systems of n vertices is between $2^{\Omega(n^2)}$ and $2^{n^2\alpha(n)^{\mathcal{O}(1)}}$, where α is the inverse of the Ackermann function [15, 20]. These numbers grow too fast to analyze all realizable rotation systems with more than 9 vertices. In fact, for $n = 8$ there are already 5 370 725 different realizable rotation systems and there is a database of all these systems with up to nine vertices [1]. To support conjectures about good drawings of K_n or find contradictions to them, we aim for an algorithm to generate a big number of random realizable rotation systems.

In this chapter we present different methods to randomly generate realizable rotation systems of good drawings of K_n. It turns out that realizable rotation systems are very rare in the set of all possible rotation systems, even for small values of n. So it is very difficult to find a big amount of different realizable rotation systems with more than seven vertices by chance.

We need an efficient method to decide whether a given rotation system is realizable. In 2011 Kynčl provided an algorithm that proves that this problem is solvable in polynomial time [16]. Later he found out that a rotation system is realizable if and only if all sub-rotation systems that consist of six vertices are realizable [17]. From the database we know that every sub-rotation system of six vertices is realizable if and only if every sub-rotation system of four and five vertices is realizable. To simplify the terminus in this thesis we call such sub-rotation systems *4-tuples* and *5-tuples*, respectively. So we only have to analyze all 4-tuples and 5-tuples to decide whether a rotation system is realizable and can decide this in $\mathcal{O}(n^5)$ time.

To create random rotation systems we considered three different approaches.

- Generating Rotation Systems from Scratch
- Generating Rotation Systems by Extension
- Generating Rotation Systems using Good Double-Flips

3.1 Generating Rotation Systems from Scratch

In this section we bound the number of equivalence classes of rotation systems and show how rare realizable rotation systems are already for small numbers of vertices. At first we concentrate on the total number of rotation systems. Then we approximate the number of equivalence classes. We consider the complete graph $K_n = (V, E)$ with vertices $V = \{1, 2, \ldots, n\}$. We have already seen that every rotation of a vertex v can be represented as a cyclic permutation of $n - 1$ vertices without fixed points. The number of such cyclic permutations is $(n-2)!$. This can easily be seen when we start every rotation with the smallest vertex. For the remaining $n - 2$ vertices we have $(n-2)!$ permutations left. Therefore the number of different rotation systems with n vertices is

$$(n-2)!^n.$$

That means that already for $n = 6$ there are more than 191 million systems. Of course a lot of them are in the same equivalence class. So it would be useful to consider less different rotation systems. The important requirement for the generation is that from every weak isomorphism class of good drawings at least one representative rotation system can be generated.

We know that for every rotation system there exists a relabeling, such that the rotation of vertex 1 is in the canonical order, that is, the rotation of vertex 1 is fixed to $(2\ 3\ 4\ \dots\ n)$. This reduces the number of possible rotation systems that we have to consider to

$$(n-2)!^{n-1}.$$

Also this number is already about 8 million for $n = 6$ and it is rapidly growing with n. Also this is still not the number of equivalence classes of rotation systems. The reason is that a rotation system that has the stated properties that the rotation of vertex 1 is $(2\ 3\ 4\ \dots\ n)$ and every other rotation starts with 1, might not be a fingerprint. As an example we consider the two rotation systems of four vertices in Example (3.1). The left system is lexicographically minimal. By relabeling the vertices, such that the vertices 1 and 2 change their labels, we obtain the right system, which is different but also fulfills the requirements from above. The two rotation systems are equivalent.

$$\begin{array}{lll} 1:\ 2\,3\,4 & 2 \rightarrow\ 1 & 1:\ 2\,3\,4 \\ 2:\ 1\,3\,4 & 1 \rightarrow\ 2 & 2:\ 1\,3\,4 \\ 3:\ 1\,2\,4 & 3 \rightarrow\ 3 & 3:\ 1\,4\,2 \\ 4:\ 1\,2\,3 & 4 \rightarrow\ 4 & 4:\ 1\,3\,2 \end{array} \qquad (3.1)$$

Note that for calculating the fingerprint we only have to consider all choices of the vertices 1 and 2 plus the two directions, clockwise and counterclockwise, in which we can read the emanating edges. The labels of the remaining vertices result from the fact that the rotation of vertex 1 is $(2\ 3\ 4 \ldots\ n)$. If a rotation system is given, we choose which vertex is labeled 1, at which vertex we start enumerating the emanating edges (this is then vertex 2), and in which direction we enumerate. Then all other vertices get their labels according to the rotation of vertex 1. This means we can relabel every rotation system in at most $2n\,(n-1)$ different ways. From this it follows that the number of equivalence classes $|\mathfrak{R}_n|$ is bounded by

$$\frac{(n-2)!^{n-1}}{2n\,(n-1)} \leq |\mathfrak{R}_n| \leq (n-2)!^{n-1}.$$

We can write this in an asymptotic notation an get the following Lemma.

Lemma 3.1 (Compare [20]) *The number of equivalence classes of rotation systems is* $2^{\Theta\left(n^2 \log n\right)}$.

We cannot say that $|\mathfrak{R}_n|$ is equal to $\frac{(n-2)!^{n-1}}{2n(n-1)}$, because it could happen that a relabeling of a rotation system leads to the same system. We see this case in the following Example (3.2). The same rotation system as in Example (3.1) is given. By relabeling it like stated, the same rotation system occurs.

$$\begin{array}{lll} 1:\ 2\ 3\ 4 & 4 \rightarrow\ 1 & 1:\ 2\ 3\ 4 \\ 2:\ 1\ 3\ 4 & 1 \rightarrow\ 2 & 2:\ 1\ 3\ 4 \\ 3:\ 1\ 2\ 4 & 2 \rightarrow\ 3 & 3:\ 1\ 2\ 4 \\ 4:\ 1\ 2\ 3 & 3 \rightarrow\ 4 & 4:\ 1\ 2\ 3 \end{array} \tag{3.2}$$

When taking realizability into account, the following property strongly reduces the number of rotation systems that could be realizable. We know that in a drawing of K_4, the rotation of the fourth vertex is determined by the rotations of the first three vertices. Hence it follows that the rotation of the n-th vertex is determined by the rotations of the vertices 1 to $n-1$.

n	lower bound for $\lvert\mathfrak{R}_n\rvert$ $\frac{(n-2)!^{n-2}}{2n(n-1)}$	exact number of equivalence classes $\lvert\mathfrak{R}_n\rvert$	upper bound for $\lvert\mathfrak{R}_n\rvert$ $(n-2)!^{n-2}$	exact number of realizable equivalence classes
3	< 1	1	1	1
4	< 1	3	4	2
5	5.4	50	216	5
6	5529.6	134 180	331 776	102
7	$\sim 2.96 \times 10^8$		$\sim 2.4 \times 10^{10}$	11 556
8	$\sim 1.24 \times 10^{15}$		$\sim 1.3 \times 10^{17}$	5 370 725

Table 3.1: Comparing the numbers of rotation systems and realizable rotation systems (missing entries are too large to be computed in reasonable time)

Therefore the number of different rotation systems the algorithm has to consider is reduced by a factor $(n-2)!$ and we get the following lemma.

Lemma 3.2 *The number of realizable rotation systems of the complete graph with n vertices is bounded from above by* $(n-2)!^{n-2}$.

For $n \leq 6$ we were able to determine the exact number of equivalence classes of rotation systems with an algorithm that computes all possible rotation systems and determines their fingerprints. The results are shown in Table 3.1 together with the upper and lower bound for $\lvert\mathfrak{R}_n\rvert$, as well as the number of equivalence classes of realizable rotation systems.

When comparing the upper and lower bound of the number of equivalence classes, we see how large the gap between this numbers is. We can also compare the number of realizable equivalence classes and all equivalence classes of rotation systems. Already for $n = 6$ roughly every thousandth

Number of considered different rotation systems	$4!^4 = 331\,776$
Number of considered realizable rotation systems	4472
Number of weakly non-isomorphic good drawings	102
Probability to generate a realizable rotation system	$\frac{331\,776}{4472} = 0.0135$
Average number of considered good drawings per class	$\frac{4472}{102} = 44$

Table 3.2: Numbers of rotation systems and good drawings for $n = 6$

class is realizable.

To estimate the probability to generate a realizable rotation system, we also need to bound the number of rotation systems we consider in each equivalence class. This will be of importance for generating random realizable rotation systems. Since we know from above that every rotation system has $2n(n-1)$ labelings that match our requirements, the number of considered realizable rotation systems of one equivalence class is bounded by $2n(n-1)$. The number of considered rotation systems that are realizable is then bounded by the number of equivalence classes of realizable rotation systems times $2n(n-1)$. For $n = 7$, this upper bound is already 970 704. Therefore the probability that a randomly picked rotation system is realizable is $< \frac{970\,704}{2.4 \times 10^{10}} = 0.000\,040\,4$. This is the quotient of the upper bound of realizable rotation systems and the number of considered different rotation systems. This means in the expected case we have to check about 24 thousand random rotation systems to find a realizable one.

In fact, the exact probability is smaller, because there are in general less than $2n(n-1)$ different rotation systems in an equivalence class. This follows from the fact that relabeling might lead to the same rotation system. (See Example 3.2). For $n = 6$, it was possible to compute some exact numbers, which are listed in Table 3.2.

Therefore, in average we need about $\frac{1}{0.0135} \approx 74$ tries to find a realizable rotation system for 6 vertices by chance. This result was confirmed by simulations. Since we know that the number of realizable rotation systems is $2^{\Omega\left(n^2\right)}$ and the number considered rotation systems is $2^{\Theta\left(n^2 \log n\right)}$, the fraction of realizable systems goes to zero as n grows. Thus it is nearly impossible to find a realizable rotation with more than ten vertices by chance. Therefore we need another approach.

3.2 Generating Rotation Systems by Extension

This approach uses the fact that every realizable rotation system of n vertices contains n realizable sub-rotation systems of $n-1$ vertices.

The idea now is to choose a random realizable rotation system with $n-1$ vertices, extend it randomly to a rotation system of n vertices and check if it is realizable. For the extension we have to insert a new vertex in every rotation and add the rotation of the new vertex to the rotation system. In the rotation of vertex 1 we add the new vertex at the last position. We have already seen that this is sufficient to generate all non-equivalent realizable rotation systems. For the rotations of the vertices 2 to $n-1$ we can include the new vertex randomly at every position except at the first position, because the rotations should start with 1. As seen before, the rotation of the new vertex results from the rotations of all other vertices. So we do not have to generate this rotation randomly.

This leads to a total number of $(n-2)^{n-2}$ possible extensions from $n-1$ to n vertices. Of course, not every extension is a realizable rotation system. But the probability to gain a realizable rotation system with this approach is much larger than with the first approach, since we start with a realizable rotation system with $n-1$ vertices. Another advantage is that it simplifies the realizability test. We know that for the realizability test we have to check all 4-tuples and 5-tuples. Since we started with a realizable rotation system with $n-1$ vertices, we know that all 4-tuples and 5-tuples without the new vertex are already realizable. So we only have to check the 4-tuples and 5-tuples that include the new vertex.

To generate a realizable rotation system of a desired size of n vertices, we start with the only rotation system with three vertices and extend it randomly. After each extension step we have to check if the new system is realizable.

Unfortunately, for large n the number of possible extensions becomes very large and the probability to find a realizable rotation system gets very small.

In practice, this approach works very well for rotation systems up to nine vertices. For the extension from nine to ten vertices usually a few thousand extension tries were needed to get a realizable rotation system. Therefore, we were not able to create a big amount of random realizable rotation systems with more than nine vertices in an acceptable time with this approach.

3.2.1 Improvement of the Extension-Step

As noted above, in a realizable rotation system, the rotation of the n-th vertex is determined by the rotations of the other $n-1$ vertices. To get the missing rotation we have to analyze all 4-tuples that contain the new vertex n. By analyzing a 4-tuple, we can determine the partial rotation of the n-th vertex including the three other vertices of this 4-tuple. By doing this with all 4-tuples, we get the whole rotation of vertex n. During this fixing of the rotation of the n-th vertex, contradictions in the partial rotations of the n-th vertex can arise. If such a contradiction arises, we have a non-realizable sub-rotation system. Such contradictions could be used to check the realizability in an early state.

To make use of this, we try to fix the rotation of the new vertex as early as possible. Therefore, for the extension, we choose the positions of the new vertex in the rotations of the vertices 2 and 3 randomly. Then we can determine the order of the vertices 1 to 3 in the rotation of vertex n, using the incomplete sub-rotation system of the 4 vertices $\{1, 2, 3, n\}$. After that we continue with vertex 4. We choose the position of the new vertex in the rotation of vertex 4 randomly and try to determine the position of vertex 4 in the incomplete rotation of vertex 4. This can be done by analyzing all 4-tuples $\{(x, y, 4, n) : x \neq y < 4\}$ and check if all can be completed to a realizable rotation system. In the same way we continue with all remaining vertices up to $n - 1$. If in one step a contradiction arises, we know that the last chosen position gives us a non-realizable rotation system and so we choose a different position.

With this method we obtain a rotation system in which every 4-tuple is realizable. It remains to check all 5-tuples that include the n-th vertex to make sure this rotation system is realizable.

With this improvement we were able to generate realizable rotation systems with up to 13 vertices in reasonable time. Comparing the improved and the not improved algorithm for $n = 9$ we get a boost-factor of approximately 10.

3.3 Generating Rotation Systems using Good Double-Flips

The third approach is based on the idea that we start with a realizable rotation system and modify this system randomly in a way that we get another realizable rotation system. For this approach we need an operation that modifies a rotation system, but does not change its realizability. Up to now we did not find such an operation, but we have found one that leaves at least all 4-tuples realizable. We call this operation a *good double-flip*.

3.3.1 Good Double-Flips

Looking at a realizable rotation system of four vertices, we see that modifying a single rotation always results in a non-realizable rotation system. When we modify two rotations in a realizable rotation system of four vertices, we again get a realizable rotation system. This can be used in rotation systems of arbitrary size. We choose an appropriate pair of modifications such that every 4-tuple is affected either by both or by none of these changes. Then every 4-tuple stays realizable, if it was realizable before.

Definition 3.1 (Good Double-Flip) *Let $\mathcal{R}$ be a rotation system of size n. Let v_i and v_j be two neighbored vertices in the rotation of vertex v_k. If in the rotation of v_i the vertices v_j and v_k are also neighbored, then we call the exchange of v_i and v_j in the rotation of v_k together with the exchange of v_j and v_k in the rotation of v_i a good double-flip and denote it by the vector $(v_k, v_i, v_j, v_i, v_j, v_k)$.*

Lets look at the good double-flips with the Example (3.3). There we see two different realizable rotation systems with five vertices. We marked the vertices that change their positions in the rotations bold. This marks a good double-flip, which modifies the left rotation system to the right one. If we have a close look at all five 4-tuples of this rotation systems, we can see that all stay realizable. Now lets have a look at the good drawings of these rotation systems. We can see them in Figure 3.1. When we apply the marked modifications in the rotations of the vertices 1 and 3 in the left drawing, we obtain the right drawing.

$$\begin{array}{rl} 1: & \mathbf{2}\,\mathbf{3}\,4\,5 \\ 2: & 1\,3\,4\,5 \\ 3: & \mathbf{1}\,\mathbf{2}\,4\,5 \\ 4: & 1\,2\,3\,5 \\ 5: & 1\,2\,3\,4 \end{array} \qquad \begin{array}{rl} 1: & \mathbf{2}\,4\,5\,\mathbf{3} \\ 2: & 1\,3\,4\,5 \\ 3: & \mathbf{1}\,4\,5\,\mathbf{2} \\ 4: & 1\,2\,3\,5 \\ 5: & 1\,2\,3\,4 \end{array} \tag{3.3}$$

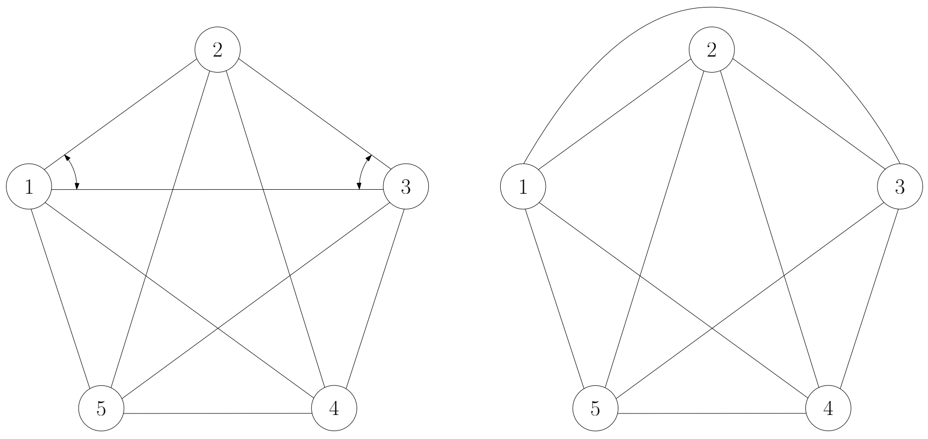


Figure 3.1: Example of a good double-flip in K_5

Next we show that like in Example (3.3) good double-flips never change the realizability of any 4-tuples.

Proposition 3.1 *Let $\mathcal{R}$ be a rotation system wherein all 4-tuples are realizable. A good double-flip of $\mathcal{R}$ leaves all 4-tuples realizable.*

Proof: In a good double-flip $(v_k, v_i, v_j, v_i, v_j, v_k)$, three different vertices v_i, v_j, and v_k are involved. In all sub-rotation systems of four vertices that contain v_i, v_j, and v_k both modifications of the double-flip will be applied. Since we know that two modification in the rotation system of K_4 do not change the realizability, these 4-tuples stay realizable. All sub-rotation systems of four vertices that contain only one or none of the three vertices will not be affected by the double-flip at all.

The sub-rotation systems of four vertices that contain two of the three vertices will also not be affected. We see this by looking at their according rotations. Assume a 4-tuple contains the vertices v_i and v_j. We only have to inspect the rotation of v_i, because the other rotations of this 4-tuple do not change. In the rotation of v_i, the vertices v_j and v_k are neighbored and switch their positions. Since v_k does not show up in our sub-rotation system, no modification will be recognized in this 4-tuple. A similar argument works for 4-tuples that contain v_j and v_k. Now assume the 4-tuple contains the vertices v_k and v_i. Since v_j is not part of the 4-tuple, in none of the rotations a change will be recognized. Therefore no 4-tuple will become non-realizable. □

To make use of good double-flips for the generation of random realizable rotation systems, we have to start with a rotation system with realizable 4-tuples and make random good double-flips. For this purpose, we start with the realizable rotation system of the geometric drawing of the complete graph whose vertices are in convex position. If we label the vertices clockwise from 1 to n we get the minimal possible realizable rotation system. To generate a new random realizable rotation system, we compute all possible good double-flips of this rotation system. Then we choose one of them randomly and apply it to the rotation system. We know that all 4-tuples stay realizable. We still have to check all 5-tuples, to make sure that the new rotation system is again realizable. If it is not realizable, we undo the double-flip and randomly choose a different one. It is easy to see that 5-tuples that do not contain all three involved vertices of the double-flip are not affected by the good double-flip (Compare the proof of Proposition 3.1). Therefore it suffices to only check the $\mathcal{O}(n^2)$ 5-tuples that contain all three vertices. After a long enough sequence of random good double-flips we get a more or less random realizable rotation system.

In comparison to the other two approaches, this one is very fast, but it is not known whether it is possible to generate a representation of every equivalence class of realizable rotation systems in this way. In other words, it is not known whether the graph of all equivalence classes of realizable rotation systems with n points is connected by good double-flips. We analyzed this graph for $n \leq 8$. The result is that for these drawings the graph is connected. (Compare Chapter 4.1)

3.4 Checking Realizability

As stated at the beginning of Chapter 3, for the realizability of rotation systems it suffices that all 4-tuples and 5-tuples are realizable. In the next section we discuss how we check this property.

3.4.1 Checking 4-tuples

In Figure 3.2 we can see the two possible good drawings of K_4. These drawings have the following rotation systems.

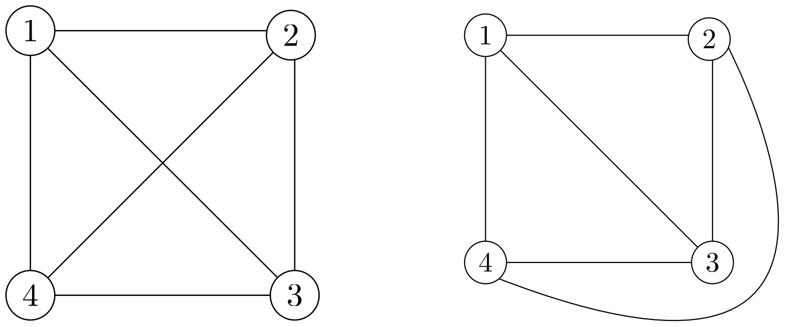


Figure 3.2: The two possible drawings of K_4

$$\begin{array}{ll} 1:\ 2\,3\,4 & \qquad 1:\ 2\,3\,4 \\ 2:\ 1\,3\,4 & \qquad 2:\ 1\,4\,3 \\ 3:\ 1\,2\,4 & \qquad 3:\ 1\,2\,4 \\ 4:\ 1\,2\,3 & \qquad 4:\ 1\,3\,2 \end{array} \tag{3.4}$$

To check all 4-tuples of a rotation system, we have to determine all sub-rotation systems with four vertices. Then we have to compute the fingerprint of each sub-rotation system and check whether it is equal to one of the two given rotation systems of Example (3.4). Every 4-tuple can be relabeled in at most twelve different ways. This means we can compute the fingerprint of a 4-tuple in constant time. So the runtime of this is $\mathcal{O}(n^4)$, because there are $\binom{n}{4}$ 4-tuples in a rotation system with n vertices. Actually we can check all 4-tuples in $\mathcal{O}(n^3)$ time using the Propositions 3.2 and 3.3 [8].

Proposition 3.2 *Let $\mathcal{R}$ be rotation system of five vertices. The number of non-realizable 4-tuples in $\mathcal{R}$ is always even.*

Proof: A single flip (x, y, z) in a rotation system is the exchange of the positions of two neighbored vertices y and z in the rotation of vertex x. Using the argument of various sorting algorithms, we know that by multiple single flips we can obtain every possible rotation system. Let $\mathcal{R}$ be a rotation system of the vertices $\{a, b, c, d, e\}$ and let (a, b, c) be an arbitrary single flip, in which the vertices b and c in the rotation of vertex a are neighbored. We know that this flip only affects the two 4-tuples (a, b, c, d) and (a, b, c, e). Since in each of these two 4-tuples, only one rotation changes, the realizability of the two 4-tuples switches. So by one single flip, the number of non-realizable 4-tuples either changes by two or stays the same. Since we know that a 5-tuple without a non-realizable 4-tuple exists, the number of non-realizable 4-tuples is always even. $\square$

Proposition 3.3 *Let $\mathcal{R}$ be a rotation system of n vertices, labeled 1 to n. If $\mathcal{R}$ contains a non-realizable 4-tuple, then it also contains a non-realizable 4-tuple that includes vertex 1.*

Proof: Assume that there are four vertices $v, x, y, z \neq 1$ such that the sub-rotation system of these vertices is non-realizable. Then it follows that the sub-rotation system consisting of the five vertices $\{1, v, x, y, z\}$ is non-realizable too. From Proposition 3.2 we know that there exists a second non-realizable 4-tuple in this 5-tuple. This 4-tuple must include vertex 1, which completes the proof. $\square$

From Proposition 3.3 it follows that it suffices to check all sub-rotation systems with four vertices containing the vertex 1 to decide whether all

4-tuples are realizable. Therefore the check for the realizability of all 4-tuples is doable in $\mathcal{O}(n^3)$.

3.4.2 Checking 5-tuples

It remains to show how we can check all sub-rotation systems of five vertices. To do this we introduce the *crossing vector*.

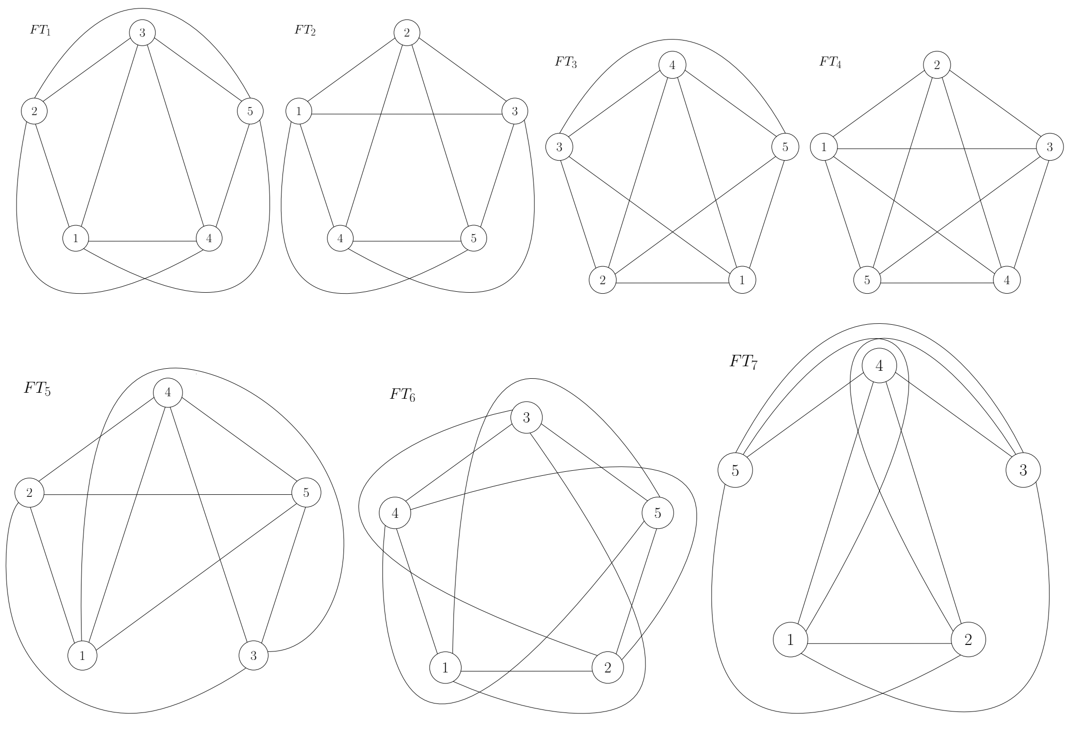


Figure 3.3: Drawings FT_1 to FT_7

Definition 3.2 (Crossing Vector) *The crossing vector $v = (c_0, c_1, c_2, c_3, \ldots)$ of a rotation system $\mathcal{R}$ is a vector of non-negative integers. The number c_i gives the* *number of edges in the associated good drawing that are crossed exactly i times. In the crossing vector, only crossings are counted that result from the rotation system.*

If every 4-tuple in the rotation system is realizable, the crossing vector can be computed from the rotation system. Then all necessary crossings of a potentially good drawing with this rotation system are determined.

In the Figure 3.3 we can see drawings of the seven different rotation systems of K_5, in which all 4-tuples are realizable. Note that up to weak isomorphism, those are the only possible drawings of K_5. The drawings FT_1 to FT_5 are good drawings. The drawings FT_6 and FT_7 are not good drawings, because they contain edges that cross twice, but they are semi-good drawings. In Table 3.3 we present the crossing vectors of these seven realizations of K_5. These crossing vectors only count crossings, which follow from the 4-tuples. Especially, for every pair of edges at most one crossing is counted.

From Table 3.3 we can see that the crossing vector is sufficient to distinguish between the different rotation systems with five vertices and realizable 4-tuples. Further, it is sufficient to compute the number c_0 of every 5-tuple to decide whether it is realizable as a good drawing.

In practice we make use of this and compute the number of edges in a 5-tuple without non-realizable 4-tuples that are non-crossed. If this number is 2 or 0 we know that this sub-rotation system is not realizable.

3.4.3 Implementation Details and Runtime

For the realizability test for the approach by extension we only have to check all 5-tuples that include the new vertex after each extension step. To the contrary, when using the approach with good double-flips, it makes sense

5-tuple	crossing vector
FT_1	$(8, 2, 0, 0)$
FT_2	$(5, 4, 1, 0)$
FT_3	$(6, 2, 2, 0)$
FT_4	$(5, 0, 5, 0)$
FT_5	$(4, 3, 2, 1)$
FT_6	$(0, 10, 0, 0)$
FT_7	$(2, 6, 2, 0)$

Table 3.3: Crossing vectors of the different realizations of K_5

to store the crossed edges of every 5-tuple and update them after each good double-flip.

To maintain the number of edges in every 5-tuple that are non-crossed, we use two data-structures. The first one is a simple set that saves every pair of crossing edges. From this set of crossing edge-pairs we can compute the crossings in every 5-tuple. We save this information in a map where the key-value is a vector of five vertices. In this map we store the number of crossings per edge of the associated 5-tuple, as well as the number of edges that are not crossed. This set has size $\mathcal{O}(n^5)$. The size of the set containing the crossings is bounded by the maximal number of crossings of a good drawing, which is $\mathcal{O}(n^4)$, because every 4-tuple has at most one crossing. The algorithm therefore has a space complexity of $\mathcal{O}(n^5)$. The runtime is determined by the time needed to build up the initial data structure, which is $\mathcal{O}(n^5)$. After this, every update of the data structure takes at most $\mathcal{O}(n^2)$ time, since we only have to update $\mathcal{O}(n^2)$ 5-tuples.

3.5 Results

With the approach of using good double-flips, we were able to generate more or less random realizable rotation systems with up to 30 vertices in reasonable time. We also used the approach of extension, because we do not know if the flipgraph is connected. With this approach we were able to analyze rotation systems with up to 13 vertices. In the following we describe some problems for which we used the generated rotation systems.

3.5.1 Plane Hamiltonian Cycles

We used our implementation to check if every randomly created realizable rotation system contains a plane Hamiltonian cycle. This is true for all good drawings with up to nine vertices, which was shown by exhaustive search using the database of good drawings [1, 22].

Definition 3.3 (Plane Hamiltonian Cycle) *Let $G = (V, E)$ be a graph. A Hamiltonian cycle of G is a cycle that contains every vertex $v \in V$ exactly once. A plane Hamiltonian cycle of a drawing $D(G)$ is a Hamiltonian cycle that does not cross itself.*

Of course every complete graph contains a Hamiltonian cycle. Also the number of Hamiltonian cycles in a complete graph is well known. It is easy to see that the following proposition is true.

Proposition 3.4 *The number of distinct Hamiltonian cycles in a complete graph with n vertices is $(n-1)!/2$.*

In contrast, the question whether every good drawing of the complete graph contains a plane Hamiltonian cycle is still unsolved. It is conjectured to be true, but up to now it is not proven.

Conjecture 3.1 *[25] Every good drawing of a complete graph contains at least one plane Hamiltonian cycle.*

With the use of rotation systems, it is possible to check whether a Hamiltonian cycle exists or not, because by Proposition 2.1 rotation systems determine the crossing edge-pairs. To get a better runtime, we used a simple heuristic to find a Hamiltonian cycle. This is useful, because by Proposition 3.4 the number of different Hamiltonian cycles is very large. Only if the heuristic did not find a cycle, we checked all possible cycles.

We tested about 15 000 random realizable rotation systems with 15 to 30 vertices that were generated using good double-flips. We also tested about 3000 random realizable rotation systems with 10 to 12 vertices that were generated using the approach of random extension. We did not find any realizable rotation system that does not contain a plane Hamiltonian cycle. So we can support the conjecture that every good drawing contains a plane Hamiltonian cycle.

We remark that the conjecture is not true for semi-good drawings. In Chapter 6 we present a semi-good drawing without a plane Hamiltonian cycle.

3.5.2 Crossing Families

The second property that we checked was the size of the biggest crossing family in a good drawing.

Definition 3.4 (Crossing Family) *Let D be a drawing of the graph $G = (V, E)$. A crossing family of D is a subset $E' \subset E$ of edges with pairwise different endpoints, where each edge $e \in E'$ crosses each other edge $f \in E' \backslash \{e\}$ in the drawing D. The size of a crossing family E' is the cardinality $|E'|$ of E'.*

The interesting question here is, what is the minimum number of vertices $n(k)$, such that every good drawing of the complete graph with $n(k)$ vertices contains a crossing family of size greater or equal to k. For example, there exists a good drawing of K_4 without a crossing. This implies that $n(2) > 4$. We already know that K_5 is not planar. Therefore every drawing of K_5 has at least one crossing and we get $n(2) = 5$.

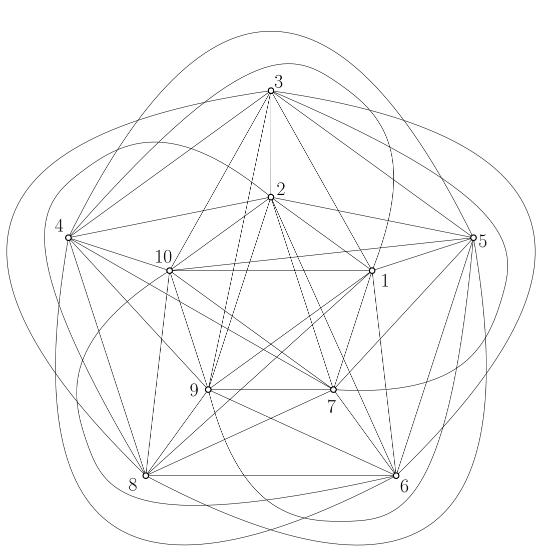
It is conjectured that the number of edges of a good drawing with n vertices without a crossing family of size k is $\mathcal{O}(n)$ [4]. This conjecture was proven for $k = 2, 3$, and 4 [3, 5]. In [4] Ackerman and Tardos give a concrete value for the maximum number of edges of a good drawing without a crossing family of size 3, which is $8n - 20$. By the fact that the complete graph with n vertices has $\frac{n(n-1)}{2}$ edges, this leads to the following corollary.

Corollary 3.1 *[4] Every good drawing of the complete graph with $n \geq 15$ vertices contains a crossing family of size 3.*

For $k = 4$ there exists also a concrete value of at most $36(n - 2)$ edges [3]. Therefore we can also state a corollary for $k = 4$.

Corollary 3.2 *[3] Every good drawing of a complete graph with $n \geq 71$ vertices contains a crossing family of size 4.*

These two corollaries for the complete graph do not give a tight bound for the number of vertices. With the use of our implementation we were able to



Credit: Oswin Aichholzer

Figure 3.4: Drawing of the complete graph with ten vertices without a crossing family of size 3

check a big amount of random good drawings. The result is that already for ten vertices, we did not find a good drawing by chance that does not contain a crossing family of size 3. It is known that for geometric drawings of the complete graph, already ten vertices are sufficient for every drawing to contain a crossing family of size 3 [6]. Since every geometric drawing is a good drawing, we know that for good drawings this number has to be between 10 and 15 due to Corollary 3.1.

With the use of the database of rotation systems, we were able to compute a tight bound for the number of vertices such that every good drawing of the complete graph contains a crossing family of size 3. To get the exact value we extended every rotation system with nine vertices that does not contain a crossing family of size 3 to ten vertices. Then we checked all these extensions. From these extensions, only nine rotation systems do not contain such a family. One of these examples is given in Figure 3.4. We extended these systems to eleven vertices and found out that all of them contain a crossing family of size 3. Therefore we can state the following theorem.

Theorem 3.1 *Every good drawing of the complete graph with $n \geq 11$ vertices contains a crossing family of size* 3*. This bound is tight.*

We also analyzed crossing families of size 4. We know that for straight line drawings, 15 vertices are sufficient for every drawing to contain a crossing family of size 4. So for good drawings this number has to be at least 15. By extending rotation systems without a crossing family of size 4 we were able to find examples of good drawings with 15 vertices that do not contain a crossing family of size 4. Since there is an extremely large number of different good drawings with $n \geq 9$ vertices without a crossing family of size 4, we were not able to compute the exact number for $k = 4$, but we know that $n(4) \geq 16$.

4 Good Double-Flips and the Flipgraph

We have already seen that a good double-flip leaves all 4-tuples realizable if they were realizable before. In this chapter we will analyze good double-flips and the flipgraph of all realizable rotation systems with up to nine vertices.

At first we will define the flipgraph. In this definition the term rotation system refers to the whole class of equivalent rotation systems.

Definition 4.1 (Flipgraph of Realizable/Semi-Realizable Rotation Systems) *The flipgraph of size n is a graph $G = (V, E)$, where V is the set of realizable or semi-realizable rotation systems with n points, respectively. For two non-equivalent rotation systems $\mathcal{R}$ and $\mathcal{P}$ of V, the edge $(\mathcal{R}, \mathcal{P})$ is contained in E if there exists a good double-flip that transforms $\mathcal{R}$ to $\mathcal{P}$.*

Although we defined the flipgraph on rotation systems, it is also reasonable to speak about good double-flips between good drawings and semi-good drawings, respectively, and the flipgraph of good and semi-good drawings. We say a good double-flip of a rotation system is *realizable*, if the flip transforms a realizable rotation system into another realizable rotation system.

As already pointed out, we do not know whether all possible realizable or semi-realizable rotation systems can be generated by good double-flips starting with an arbitrary realizable rotation system. In other words, it is not known if the respective flipgraph is connected.

4.1 Connectivity of the Flipgraph

With our implementation we found out that the flipgraph of all realizable rotation systems with $n \leq 8$ vertices is connected. We found these results by randomly generating realizable rotation systems using good double-flips and storing the fingerprints of the encountered systems. For $n \leq 7$ it was possible to find all realizable rotation systems in reasonable time by chance. For $n = 8$ we were able to find all but a few thousand. By using the database of realizable rotation systems we were able to check the remaining systems and figured out that they are also connected to the rest. This means that our approach to generate random realizable rotation systems by good double-flips works for $n \leq 8$. Unfortunately, not all rotation systems will be generated with the same probability by this approach. It might happen that a rotation system is connected to the other systems just by one good double-flip. Then it is very unlikely that this rotation system will be generated. To illustrate this in an example, we present the flipgraph of the semi-good drawings of size 4 and 5 in Figure 4.1 and Figure 4.2, respectively. The numbers beneath the arrows give the numbers of good double-flips that lead to the neighbored drawing.

It is easy to see that every realizable rotation system of K_4 has twelve good double-flips. It is clear that every good double-flip is realizable, because good double-flips leave 4-tuples realizable and we only have four vertices. From the good drawing of K_4 where two edges cross, four of the twelve flips lead to the drawing without a crossing edge-pair. The remaining eight flips lead to the same drawing of K_4 with one crossing, but relabel the vertices. From the drawing of K_4 without a crossing edge-pair, nine flips lead to the other good drawing. The remaining three flips perform a relabeling of the vertices.

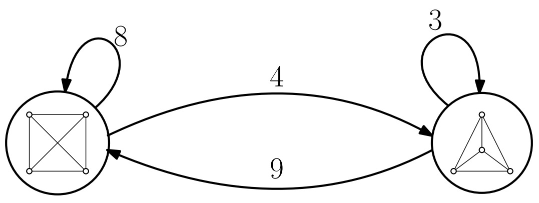


Figure 4.1: Flipgraph of size 4

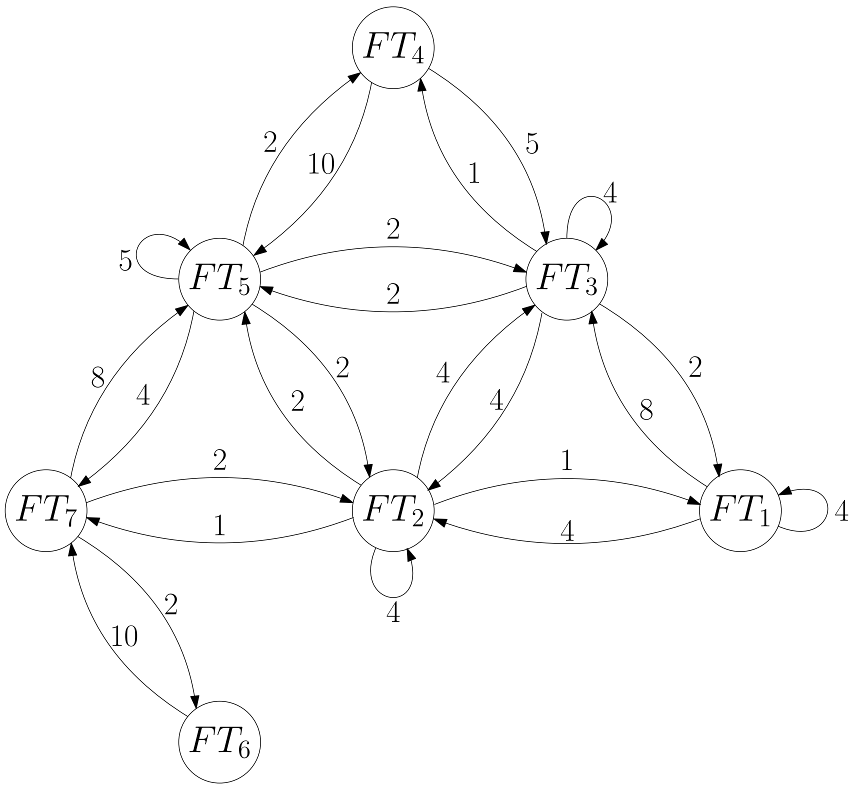


Figure 4.2: Flipgraph of size 5

Good Drawing	FT_1	FT_2	FT_3	FT_4	FT_5
Probability	0.095	0.262	0.310	0.071	0.262

Table 4.1: Probability to reach a good drawing with 5 vertices after a sufficient large amount of random good double-flips within the set of good drawings (stationary distribution of the Markov chain)

In Figure 4.2 we can see the flipgraph of all semi-good drawings of size 5. The vertices represent all semi-good drawings of K_5 and are named like in Figure 3.3. Note that the drawings FT_6 and FT_7 are not good drawings, but only semi-good drawings.

For example we see in this figure that the rotation system of FT_1 has 16 good double-flips. Eight double-flips lead to FT_3, four lead to FT_2 and the remaining four flips lead to good drawings that are weakly isomorphic to FT_1.

The flipgraph of all good drawings of size 5 forms a Markov chain. We can therefore compute the stationary distribution. This can be seen as the probability that after m random good double-flips we result at FT_x, for any $x \in \{1, 2, 3, 4, 5\}$ and m sufficient large. Thereby it does not matter at which drawing we start. When we assume that we stay in the set of good drawings we get for a sufficient large m the probability-distribution given in Table 4.1. We can see that it is very unlikely to get FT_4 by doing multiple random good double-flips. This illustrates that the probability of generating a specific good drawing of size 5 is distributed very unequally.

From this graph we can also see that with good double-flips we do not always stay at realizable rotation systems. The non-realizable rotation system of the drawing FT_7 is connected to realizable rotation systems. But only 5 flips lead from some realizable rotation system to this non-realizable rotation system, four flips from FT_5 to FT_7 and one flip from FT_2 to FT_7.

Therefore the chance to end up in a non-realizable rotation system is small. This also means that by starting at a realizable rotation system and doing one double-flip we never end up in FT_6. For testing realizability of 5-tuples we do not have to check if FT_6 is part of our rotation system when we create random rotation systems by good double-flips.

4.1.1 Flipgraph of Good Drawings with 9 Vertices

We know that the number of realizable rotation systems with nine vertices is 7 198 391 729 [1]. This number is too large to analyze the complete flipgraph of size 9 in reasonable time. To find an argument that the flipgraph for good drawings with nine vertices is not connected, we searched for realizable rotation systems with a small number of good double-flips. The idea was that a small number of good double-flips would also lead to a small number of realizable good double-flips. If we are able to find a realizable rotation system without any realizable good double-flip, we know that the flipgraph is not connected. The computation of good double-flips of rotation systems is very fast. So we were able to check all realizable rotation systems with nine vertices and found out that every system has at least 9 good double-flips and that 149 have exactly 9 flips. We then checked these 149 systems, but none of them has no realizable good double-flip. Every checked system has at least 8 realizable flips.

This investigation does not prove that the flipgraph is connected, unfortunately it also does not prove that it is not connected.

4.1.2 Rotation Systems without any Good Double-Flips

Up to now we did not encounter a rotation system without a good double-flip. So we implemented a deterministic backtracking algorithm that creates all rotation systems that do not contain any good double-flips. We found out that there is no rotation system with less than 8 vertices without any good double-flips. For $n = 8$ there are rotation systems without any good double-flips, but of course all of them are non-realizable.

4.2 Minimum Number of Good Double-Flips

To get a better insight into the number of realizable good double-flips of good drawings, we analyzed all good drawings with a small number of vertices and searched for the drawings with the minimum number of realizable good double-flips. For $n = 5$ and $n = 6$ these drawings were from a special class of drawings. At first we will introduce this new class of good drawings, namely the class of 2-page book drawings.

Definition 4.2 (2-page Book Drawing) *[13] A 2-page book drawing of a graph is a drawing where the vertices are drawn along a line and each edge lies completely on one of the two sides of this line.*

We can see two examples of a 2-page book drawing in Figure 4.3.

We were able to analyze all good drawings with up to eight vertices with respect to their number of realizable good double-flips. So we know which drawings of K_n have the minimum number of realizable good double-flips for each size of $n \leq 8$.

We know that every drawing of K_4 has exactly twelve realizable good double-flips.

We also have already seen that the good drawing of K_5 with the minimal number of realizable good double-flips is the drawing FT_2 with 12 flips. The number of realizable good double-flips is between 12 and 16 for all drawings of K_5. For $n = 6$ this number is between 6 and 21. It is interesting to have a look at the realizable rotation systems with 5 and 6 vertices that have the minimum number of realizable flips. We can see them in Figure 4.3. Both rotation systems can be realized as 2-page book drawings.

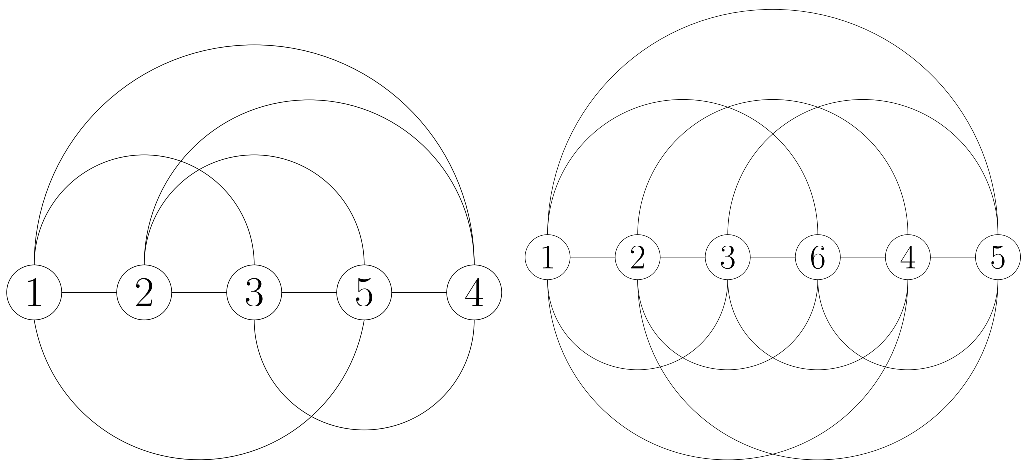


Figure 4.3: Good drawings with the minimum number of realizable flips with 5 and 6 vertices

For $n = 7$, there are two drawings with the minimum number of realizable flips. One of them is also drawable as a 2-page book drawing. The number of realizable good double-flips in this rotation systems is 7. For drawings with 8 and 9 vertices, we found examples that have only 7 and 8 realizable good double-flips, respectively.

In Table 4.2 we can see the minimum number of realizable good double-flips

Vertices	4	5	6	7	8	9
Min. number of flips	12	12	6	7	7	≤ 8

Table 4.2: Minimum number of realizable good double-flips in good drawings with n points

for good drawings with up to 9 vertices. For $n = 9$ we do not know whether there exists a drawing with less than 8 realizable good double-flips, but we found an example with only 8 flips.

4.3 Good Double-Flips in Good Drawings

Up to now we looked at good double-flips from the rotation system point of view. Now we look at good double-flips from a drawing point of view and show how some good double-flips change the drawing and under which conditions they lead to another good drawing. In the following we will often speak of the good double-flip of a good drawing and mean the double-flip in the associated rotation system. In a good double-flip there are three vertices involved. In a complete graph these three vertices are all connected. Thus the subgraph consisting of these three vertices forms a triangle. Moreover in the rotation of two of the involved vertices the other two vertices have to be consecutive in the cyclic order. This follows directly from the definition of the good double-flip. An easy example of a good double-flip is presented in Figure 4.4. We can see that here the edges (a, b) and (a, c) at vertex a are consecutive without an edge between them inside the given triangle. The same is true for the rotation of vertex b and the edges (b, a) and (b, c). Due to this we will call such a good double-flip an inner-inner flip.

This easy configuration is not the only one that leads to a good double-flip.

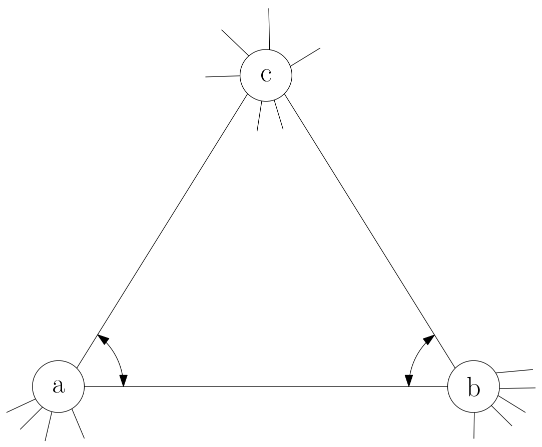


Figure 4.4: Inner-inner flip

It can also happen that the involved edges of the vertices a and b are in consecutive order outside the triangle. We call this double-flip an outer-outer flip. Since the definition of inside and outside only depends on the definition of the unbounded cell, we will not further distinguish between inner-inner and outer-outer flips.

The most complicated case shows up if the consecutive edges are inside the triangle for one vertex and outside for the other. We will call this good double-flip an inner-outer flip. Figure 4.5 shows the configuration of an outer-outer and of an inner-outer flip, respectively.

Here we will only analyze inner-inner flips. We start with a simple proposition.

Proposition 4.1 *Given is a good double-flip (a, b, c, b, a, c) in a good drawing of the complete graph K_n, for $n \geq 3$. We define the inside of the triangle as the cell that is to the left of the closed curve formed by the edges (a, b), (b, c), and (c, a). If the flip is an inner-inner flip then the triangle $\triangle abc$ is interior-empty.*

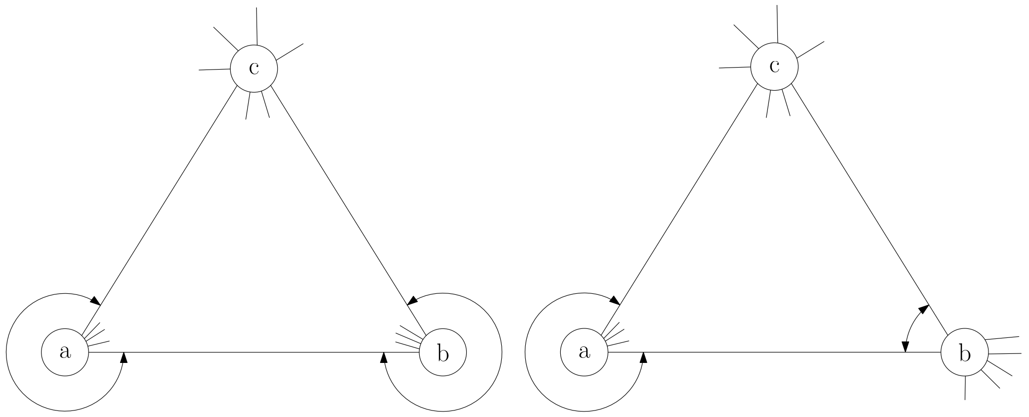


Figure 4.5: Outer-outer flip and inner-outer flip

Proof: We assume that (a, b, c, b, a, c) is an inner-inner flip in a good drawing of the complete graph K_n. The given configuration is shown in Figure 4.4. Since the vertices b and c are consecutive in the rotation of vertex a, no edge could leave vertex a in the interior of the triangle. The same is true for the rotation of vertex b. Therefore, for any arbitrary vertex $x \neq a, b, c$ the edges (a, x) and (b, x) start in the exterior of the triangle $\triangle abc$. By Lemma 2.2 it follows that x lies in the exterior of the triangle $\triangle abc$. $\square$

Proposition 4.1 helps us to characterize when an inner-inner flip leads to another realizable rotation system. Figure 4.6 shows how we can locally redraw the good drawing after applying an inner-inner and an outer-outer flip to it. For an inner-inner flip we know that the interior of the triangle is empty. For an outer-outer flip we know that the exterior is empty. The red dashed lines in the drawings are the old edges. We can reroute these edges like the green lines, after applying the good double-flips. This re-routing will not always lead to a good drawing, but we can characterize when such a flip leads to another good drawing of the complete graph. For an inner-inner flip (a, b, c, b, a, c) this is the case if and only if no edge crosses both edges (a, c) and (b, c), see Figure 4.7. Otherwise the re-routing leads to a double-crossing and we only get a semi-good drawing. We now state this fact as a lemma and proof it formally.

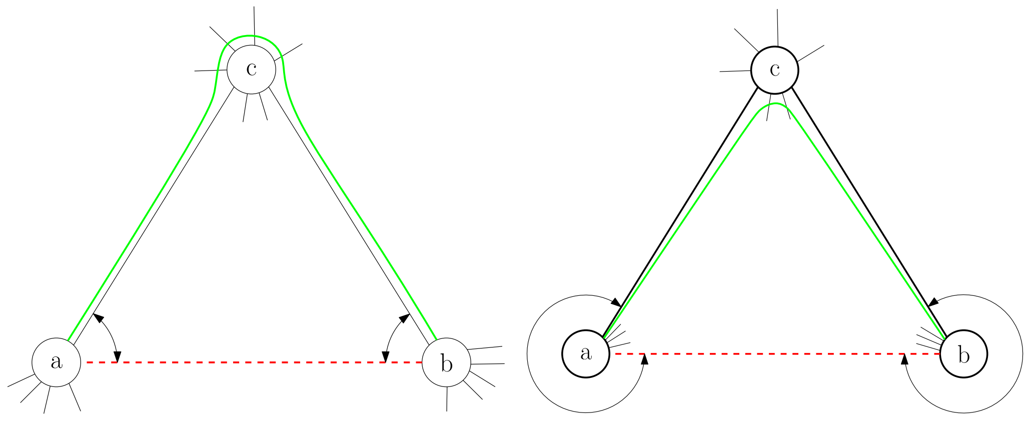


Figure 4.6: Locally applying an inner-inner and an outer-outer flip to the drawing

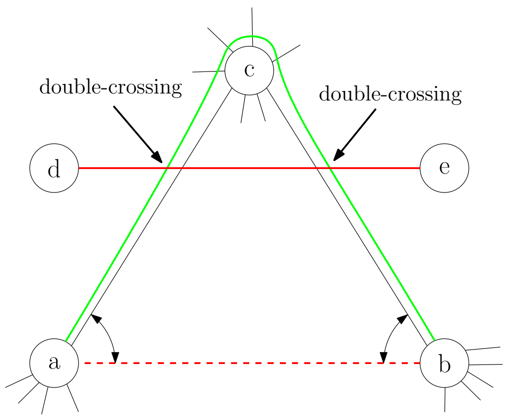


Figure 4.7: Edge (d, e) prohibits a realizable inner-inner flip

Lemma 4.1 *Given is an inner-inner flip (a, b, c, b, a, c) of a good drawing D of the complete graph K_n. The flip leads to another good drawing if and only if there is no edge crossing both edges (a, c) and (b, c).*

Proof: We assume that (a, b, c, b, a, c) is an inner-inner flip of the good drawing D of the complete graph K_n. At first we show that if no edge exists that crosses both (a, c) and (b, c) we can redraw the good drawing like in Figure 4.6 and get a new good drawing. Due to Proposition 4.1 the triangle $\triangle abc$ is interior-empty. Therefore no edge that is incident to a can cross (b, c). Otherwise the edge has to leave the triangle $\triangle abc$ again. To do this it has to cross one of the two other bounding edges. Both are incident to a and therefore the edge is not allowed to cross them. The same is true for vertex b and the edge (a, c). It is also clear that the edge (a, b) is allowed to cross the edges that leave vertex c on the outside of the triangle $\triangle abc$. So it follows that we can redraw the edge like in Figure 4.6, without generating a double-crossing with the edge (a, b) and we get a new good drawing.

Now we show that if there exists an edge that crosses (a, c) and (b, c), the flip can never lead to a good drawing. We look at the configuration in Figure 4.7 before applying the double-flip. From the crossing of the edges (a, c) and (d, e) and their crossing rotation we get the rotation system of the 4-tuple (a, c, d, e) by Lemma 2.1. In the same manner we get the rotation system of the 4-tuple (b, c, d, e). Since the red line crosses through the triangle $\triangle abc$, we know that the edges (c, d) and (c, e) start in the exterior of the triangle $\triangle abc$. Otherwise the edge (c, d) (resp. (c, e)) starts in a cell at vertex c that is bounded by three edges that all share a vertex with (c, d) (resp. (c, e)). Since we know three rotations of the 4-tuples (a, b, c, d) and (a, b, c, e), their rotation system is determined. From the rotation systems of these 4-tuples we can generate the rotation system of the full good drawing of the 5-tuple before applying the flip. We can see the 5-tuple before and after the flip in Figure 4.8. The rotation systems before and after the flip are shown in Table 4.3. The rotation system before the flip is equivalent to the one of FT_2, by relabeling the vertices with 1 to 5. The given flip in this rotation system leads to a rotation system that is equivalent to the one of

a: **b** e d **c**	a: **b** **c** e d
b: **a** **c** e d	b: **a** e d **c**
c: a d e b	c: a d e b
d: a b c e	d: a b c e
e: a b d c	e: a b d c

Table 4.3: Rotation systems of the drawings in Figure 4.8

the drawing FT_7. We know that this rotation system is only realizable as a semi-good drawing. Therefore this flip will never lead to realizable rotation system. This concludes the proof. □

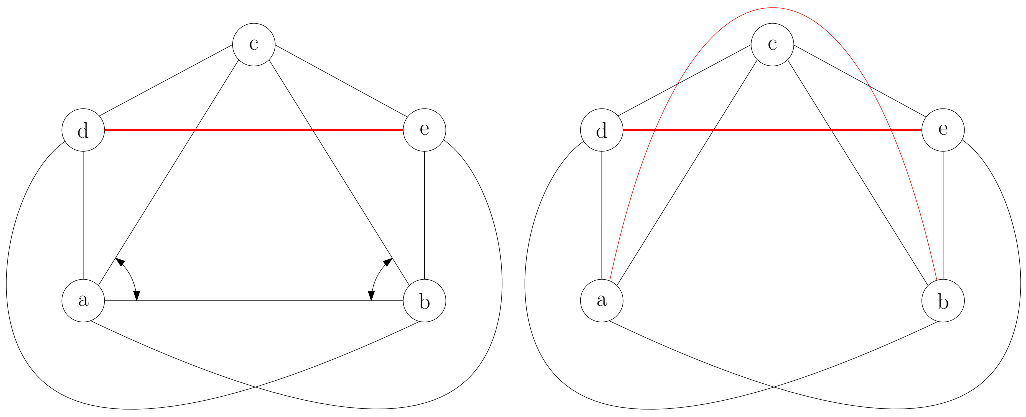


Figure 4.8: Configuration before and after the double-flip

4.4 Relabeling of the Convex Geometric Good Drawing

By looking at good double-flips in good drawings of K_4, we see that there are flips that do not change the equivalence-class of the rotation system, but only exchange the labels of two vertices. This leads to the question whether it is possible to relabel any good drawing of the complete graph with an arbitrary number of vertices by good double-flips arbitrarily. We show that this is possible for the convex geometric good drawing. This is the drawing where all vertices are arranged in convex position and the edges are straight lines. In the following we show how it is possible to exchange the labels of two neighbored vertices by a series of inner-inner flips. From this we can conclude that we can produce any labeling of the convex geometric good drawing with good double-flips.

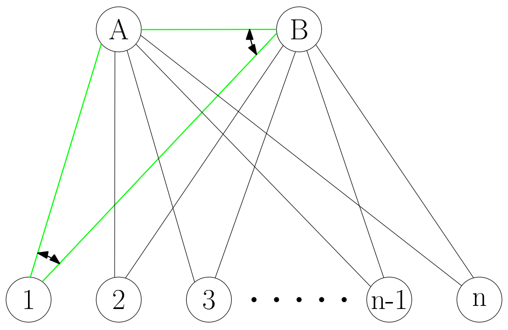


Figure 4.9: Basic configuration before the relabeling

In Figure 4.9 we see our basic configuration before the exchange of the labels of vertex A and vertex B. For simplicity the vertices 1 to n are drawn on a line and not in general position. When the vertices A and B switch their positions in the drawing, the rotations of these two vertices stays the same. The only difference between the old and the relabeled rotation system is that in the rotations of the vertices 1 to n, the vertices A and B switch their positions. By doing the marked flip in the green triangle in Figure 4.9 we force this exchange of A and B in the rotation of vertex 1. It is easy to see that this flip leads to another good drawing. In Figure 4.10 we can see that we can go on and apply more inner-inner flips, which switch the vertices A and B in all rotations of the vertices 1 to n. The rotation of vertex

A does not change at all. By this series of flips A exchanges its position with every vertex in the rotation of B. In the end it ends up at the same position as it started. Therefore no change is made in the rotation of vertex B. By moving vertex B to the left of vertex A, we can see that this drawing is weakly isomorphic to the old drawing with exchanged labels A and B.

It is known from several sorting algorithms that, if we are able to exchange two arbitrary vertices, we can relabel the drawing arbitrarily.

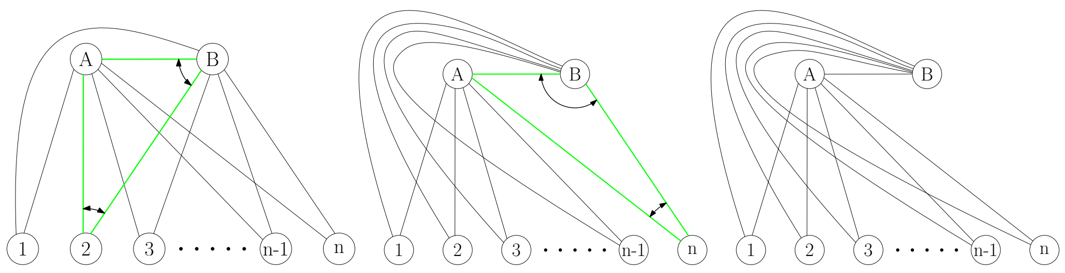


Figure 4.10: Doing inner-inner flips to exchange two labels of the convex geometric drawing

5 Semi-Realizable Rotation Systems

Additionally to good drawings we considered semi-good drawings. When we look at non-realizable rotation systems with four vertices, we see that the reason why they are not drawable as good drawings is the rule that two incident edges must not cross. We have seen that a rotation system with five vertices that consists of realizable 4-tuples is at least semi-realizable. The rule that two edges are not allowed to cross multiple times prohibits two of this systems to be realizable. For being realizable it suffices that all 4-tuples and 5-tuples are realizable. For four and five vertices it seems like the 4-tuples are responsible for the adherence of the first rule and the 5-tuples are responsible for the adherence of the second rule. This leads to the conjecture that a rotation system wherein all 4-tuples are realizable can be drawn without violating the first rule, that is, as a semi-good drawing.

Conjecture 5.1 *Every rotation system $\mathcal{R}$ wherein every 4-tuple is realizable is semi-realizable.*

To verify this conjecture, we analyzed all rotation systems with six and seven vertices that consist of realizable 4-tuples and checked if they are drawable as semi-good drawings. Surprisingly, it turned out that already for six vertices there exists a counterexample to Conjecture 5.1.

Before we show the results of our analysis we state some facts about semi-good drawings and describe how our algorithm works.

In a semi-good drawing, two edges are allowed to cross multiple times. Therefore we can have cells that are bounded only by two edges. We show that every semi-realizable rotation system can be realized without such cells. This has been shown in [19]. We state this result in Theorem 5.1 and show the proof for self-containment.

Definition 5.1 (Lens) *[19] Assume that two edges e_1 and e_2 cross at least twice. A region enclosed by two segments of these two edges is called a lens. A lens is called empty if it does not contain a vertex.*

Remark 5.1 *It does not play a role whether a lens is bounded or unbounded. Also an unbounded region can be an empty lens. See Figure 5.1. The proofs in this chapter all consider bounded lenses. They all work analogously for the unbounded case.*

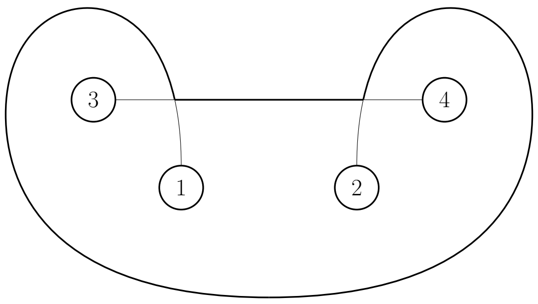


Figure 5.1: Example of an unbounded empty lens

Theorem 5.1 *[19] Every semi-realizable rotation system can be realized without empty lenses.*

Proof: We show that in a semi-good drawing we can redraw an empty lens locally, such that the two involved crossings of the lens are removed and no new crossings are introduced. Assume that our drawing contains an empty lens. Choose any empty lens that does not contain any other empty lens. Such a lens exists, since we can choose the smallest one. In the left two drawings of Figure 5.2 we can see how we can redraw this lens to resolve it locally. We have to show that the new drawing is also a semi-good drawing.

Since our lens was empty and does not contain another empty lens, every edge that crosses e_1 also has to cross e_2. This is clear, because any edge that enters the lens at e_1 has to leave it at e_2. Otherwise a new empty lens inside our empty lens exists. Therefore by our redrawing no new crossings occur and the new drawing is also semi-good.

With this redrawing we get rid of two crossings and we do not create any new crossings. Every empty lens can be redrawn in this way, which reduces the number of crossings. This implies that this process is finite and results in a drawing without empty lenses.

$\square$

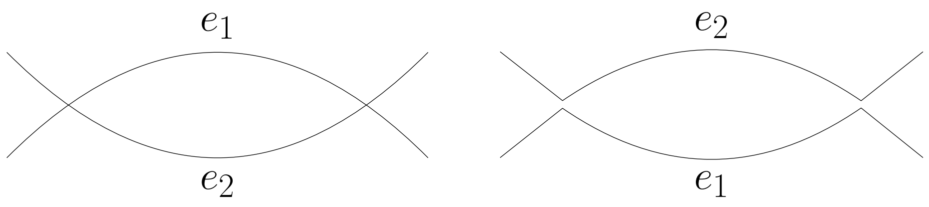


Figure 5.2: How to redraw an empty lens

5.1 The Algorithm

To decide whether a given rotation system is semi-realizable, we used a backtracking approach based on the algorithm used in [22]. It uses the *half-edge data structure* to represent a drawing [27]. In this data structure the edges consist of several pairs of directed segments in both directions. The edges are fragmented by crossings into segments. Each segment has a link to its successor, predecessor, and neighbor. It also stores to which edge it belongs to. Then every cell in the drawing is bounded by a set of segments, which are connected with the successor- and predecessor-pointer. With this data structure we can create a drawing of the graph incrementally that fulfills a given rotation system if it is semi-realizable. We start with a star-graph, where every vertex is connected with a straight line to vertex 1 according to its rotation. The next edges will be added by recursively checking every combinatorially possible way the edge can go. These possibilities depend on the rotation system, since it defines in which cell an edge starts. In this manner every edge connected with vertex 2 will be added. This will be done recursively in all possible ways. Then it continues with the remaining edges of vertex 3 and so on, until a complete semi-good drawing was found or all possibilities were checked. In a list we store how often two edges are allowed to cross and reduce this number, if we add a crossing. This ensures that the algorithm terminates. In Figure 5.3 we can see the representation of a drawing of K_4 with one crossing in a half-edge data structure. Here every edge consists of at least two directed segments. Crossings split these segments and every cell is bordered by a connected set of segments. A detailed description of the algorithm for good drawings can be found in [22].

The original algorithm only works for the search of good drawings. To adapt it for the search of semi-good drawings, we had to implement that a

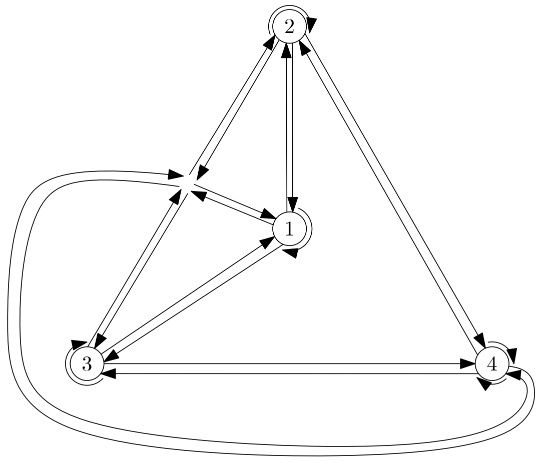


Figure 5.3: Half-edge data structure of a drawing of K_4

pair of edges is allowed to cross multiple times. It is not possible to allow that two edges cross an arbitrary number of times, because this would lead to an infinite loop in the algorithm. To avoid this we set the number of crossings for each pair of edges to a maximum number, which is set by a parameter. This allows us to get a more detailed and faster analysis of the rotation systems. By studying various drawings, we recognized that in many semi-good drawings only a few edge-pairs need the maximum number of crossings. All the other edge-pairs cross a very small number of times. This led us to implement a more detailed search. We added two new parameters to our algorithm. This allowed us to define how many edge-pairs are allowed to cross the maximum number of times and how often the remaining edge-pairs are allowed to cross.

5.1.1 Avoidance of Empty Lenses

As we have seen, every semi-realizable rotation systems is drawable without empty lenses. It was necessary to implement methods in the algorithm that avoid the generation of empty lenses. Without such methods, a very big number of superfluous sub-drawings will be checked in the generation of a semi-good drawing and this leads to a very bad runtime.

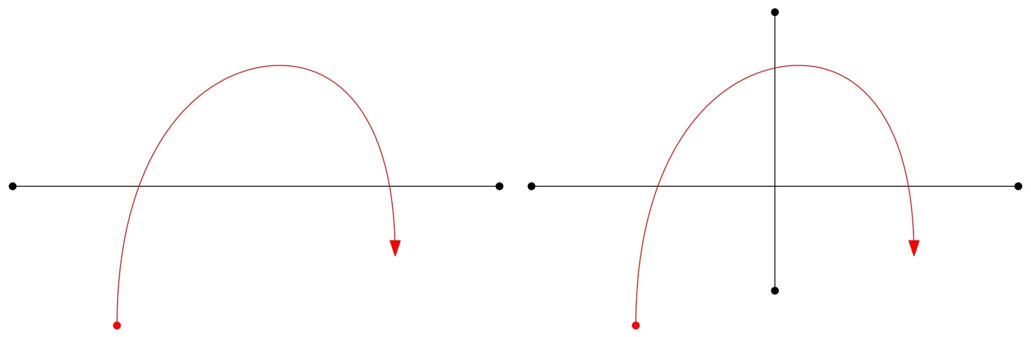


Figure 5.4: How the algorithm might create empty lenses

In Figure 5.4 we see two basic ways the algorithm might create an empty lens. Here the red edge is the new edge, which crosses the horizontal edge twice. The left one shows up if the algorithm generates an edge and crosses the same edge twice consecutively in different directions. This can be avoided easily by storing the last generated crossing. However, we have to show that this generated lens is indeed empty in the complete drawing and therefore its avoidance will not change the outcome of the algorithm.

The right figure illustrates the generation of an empty lens, where the horizontal edge is not crossed consecutively. In this case we have to check if the now generated lens is empty. By doing this check in advance we can avoid the generation of an empty lens. The second case indeed includes the first one. Since the check of the second case is more time consuming, it makes sense to distinguish these two cases.

Lemma 5.1 *If the described algorithm generates an edge that crosses a second edge twice with two different crossing rotations, and without another crossing in between* *on the newly inserted edge, then this will generate an empty lens in the complete drawing.*

Proof: We proof this lemma by contradiction. We assume that one edge crosses another edge twice consecutively, i.e. the edge crosses no other edge in between, with different crossing rotations and this creates a lens that is not empty in the complete drawing. We see this configuration in Figure 5.5. The red edge (x, y) is the one that is actually drawn. It crosses the edge (a, b) twice consecutively. We assume that a vertex c lies inside this lens. Here the edge (a, b) exists before the edge (x, y) is generated. Due to the fact that the algorithm creates all edges incident to one vertex before it continues with another vertex, all edges incident to vertex a or b are already drawn. Without loss of generality we assume all edges incident to a are already drawn. Then especially the edge (a, c) is already drawn. Since we assumed that c lies in the lens bounded by (a, b) and (x, y) and (a, b) is crossed twice by (x, y) with different crossing rotations, the vertices c and a lie in distinct cells. So the edge (a, c) has to cross the boundary of the lens. It can only cross the boundary of the lens at the edge (x, y) (see the green dotted line in Figure 5.5). Therefore (x, y) does not cross (a, b) consecutively, which is a contradiction and proves the lemma. □

Now we show that we also can prohibit the generation of two consecutive crossings with the same crossing rotation, because this would lead to a non-completable drawing. The following lemma characterizes a non-completable sub-drawing.

Lemma 5.2 *If an edge e_1 crosses an edge e_2 twice with the same crossing rotation consecutively, i.e. without crossing e_2 with a different crossing rotation in between, then this will lead to a non-completable drawing.*

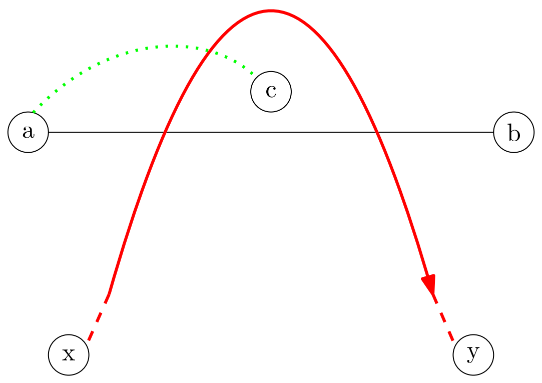


Figure 5.5: Generation of a non-empty lens by the algorithm

Proof: We assume the edge (x, y) crosses the horizontal edge (a, b) twice with the same crossing rotation consecutively. To do so the edge (x, y) has to round either the vertex a or b. Without loss of generality the edge rounds the vertex a. Then the edge crosses the edge (a, b) a second time. To do so it can cross this edge on the left or on the right side of the first crossing. We see these two possibilities in Figure 5.6. In both cases it splits the plane into two cells, which are bounded by the edges (x, y) and (a, b). Therefore the vertices a and b lie in distinct cells. This prohibits the edge (x, a) or (x, b) and therefore the drawing is not completable. □

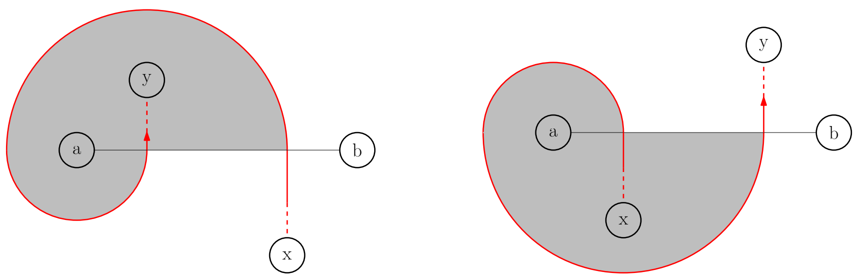


Figure 5.6: Crossing the same edge twice with the same orientation

So we have seen that prohibiting two consecutive crossings between the same edge-pair does not affect the possible rotation systems the algorithm can realize. From now on we restrict to the assumption that two edges do not cross twice with the same crossing rotation consecutively, because this would lead to a non-completable drawing, which our algorithm detects anyhow.

Now we show how we prohibit empty lenses where one edge does not cross the second one consecutively, like in the right drawing in Figure 5.4. This drawing is of course only a symbolic drawing. There might be more edges crossing the lens.

It is clear that a double crossing of an edge-pair always generates a lens. So every time an edge-pair crosses for the i-th time, with $i \geq 2$, we have to check if the now generated lens is empty. Therefore we inspect the boundary of the now generated lens. If there exists an edge that crosses the boundary of the lens an odd number of times, then we know that the lens is not empty. This is true in particular for a drawing of the complete graph. We will show that this is also true at any time during the generation of the drawing by our algorithm.

Lemma 5.3 *We assume that we have a lens that was generated by our algorithm by two edges that cross twice in different orientations. This lens is empty if and only if every edge crosses the boundary of the lens an even number of times.*

Proof: We consider the case of Figure 5.5, where the edge (a, b) exists and we generate a lens when drawing the edge (x, y). At first we show that if a vertex exists inside the lens, then there has to be an edge that crosses the lens an odd number of times. Like in the proof of Lemma 5.1 we assume without loss of generality that the edge (a, c) is already drawn. It is clear that this edge has to cross the boundary of the lens an odd number of times to connect a vertex on the inside of the lens with a vertex on the outside. Therefore for every non-empty lens there is an edge, which crosses the boundary of the lens an odd number of times.

The other direction follows immediately. If there is an edge that crosses the lens an odd number of times, this means the edge connects a vertex on the outside of the lens with one on the inside. Therefore, there exists a vertex inside this lens. □

In our algorithm we can use Lemma 5.3 and count only how often the edges emanating from *a* and *b* cross the boundary of the lens. If this number is odd for one edge, we know that the lens is not empty. Otherwise it is empty.

With the use of these observations we were able to implement the avoidance of empty lenses in our algorithm. This results in a better runtime, especially, when we allow many crossings per edge pair.

5.2 Results for Semi-Good Drawings with Six Vertices

We have to inspect only rotation systems where all 4-tuples are realizable, as otherwise, the rotation system cannot be semi-realizable. To get these rotation systems, we extended every rotation system with four vertices in all possible ways. We did this in a way that we did not generate systems with non-realizable 4-tuples. After the extension we removed all equivalent rotation systems. By repeating these steps we computed all rotation systems with up to seven vertices that do not contain any non-realizable 4-tuples.

We started with the analysis of all rotation systems with five vertices that do not contain a non-realizable 4-tuple. We already know that there are

7	total number of rotation systems with realizable 4-tuples
5	good drawings
2	semi-good drawings with at most two crossings per edge-pair

Table 5.1: Rotation systems for $n = 5$ and their realizability

173	total number of rotation systems with realizable 4-tuples
102	good drawings
62	semi-good drawings with at most two crossings per edge-pair
5	semi-good drawings with at most three crossings per edge-pair
3	semi-good drawings with at most four crossings per edge-pair
1	not semi-good drawable rotation system

Table 5.2: Rotation systems for $n = 6$ and their realizability

seven different rotation systems with five vertices wherein all 4-tuples are realizable. We can see these realizations in Figure 3.3. Five drawings are good drawings. For the remaining two drawings only two crossings are needed for every edge-pair. These results are listed in Table 5.1. With the use of our implemented algorithm we checked, whether these semi-good drawings are unique for the given rotation systems. The result is that all drawings are unique up to relabelings. This means that they do not have a different realization without empty lenses, even if we allow edge-pairs to cross more than twice.

For the drawings of the complete graph with six vertices we have 173 rotation systems wherein all 4-tuples are realizable. We already know that 102 of them are realizable as a good drawing [1]. We checked the remaining 71 with different numbers of allowed crossings per edge-pair. The results are listed in Table 5.2.

There is one rotation system that does not have a realization as a semi-good

1:	2	3	4	5	6
2:	1	3	4	6	5
3:	1	4	5	2	6
4:	1	6	3	5	2
5:	1	3	6	4	2
6:	1	4	2	3	5

Table 5.3: Rotation system with six vertices that is not semi-realizable

drawing. The rotation system is given in Table 5.3. This rotation system contains six sub-rotation systems with five vertices. The sub-rotation system of the 5-tuple $(1, 2, 3, 4, 5)$ is equivalent to the one of FT_4. All other 5-tuples are equivalent to the rotation system of FT_7. In Figure 5.7 we can see a partial drawing of the given example. The drawing consists of the 5-tuple $(1, 2, 3, 4, 5)$ and an indication where the edge to the sixth vertex emanate from the vertices 1 to 5.

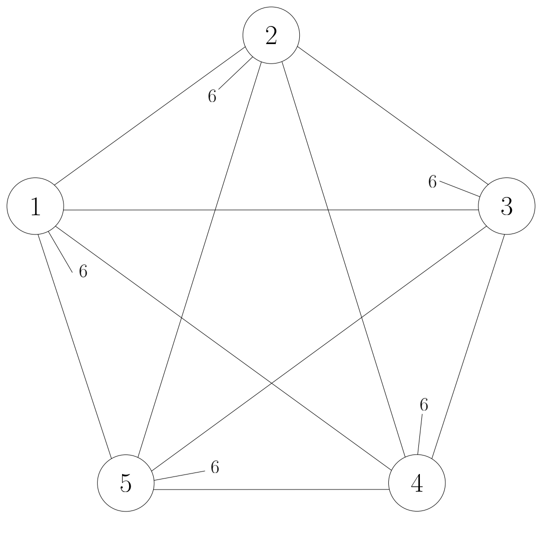


Figure 5.7: Partial drawing of the non-semi-realizable rotation system

5.2.1 Proof of the Non-Semi-Realizability

The given example in Table 5.3 is a counterexample to Conjecture 5.1 that the semi-realizability of the rotation system follows from the realizability of all 4-tuples. We now present a proof that this example is indeed not drawable as a semi-good drawing. For the proof we make use of the fact that the rotation systems of the 4-tuples give us the information in which triangles the sixth vertex has to lie (see Lemma 2.2). Then we conclude that there is no valid position for the sixth vertex in this drawing.

Theorem 5.2 *Given is a rotation system $\mathcal{R}$. The semi-realizability of $\mathcal{R}$ does not follow from the realizability of all 4-tuples of $\mathcal{R}$.*

Proof: We show that the rotation system of Table 5.3 is not semi-realizable. The first important observation is that the drawing of FT_4 in Figure 5.7 is the only possible semi-good drawing without empty lenses. Therefore the given drawing FT_4 has to be a sub-drawing of our example with six vertices.

Now we consider the triangles $\triangle 145$, $\triangle 135$, and $\triangle 245$. For any triangle $\triangle xyz$ we refer to the region bounded by $\triangle xyz$ and that is to the right when traversing the boundary of $\triangle xyz$ from x to y then to z and back to x as the *interior of the triangle* $\triangle xyz$. With this definition, the interiors of our stated triangles in Figure 5.7 are the bounded regions defined by the triangles.

Now we look at the rotation system of our example. We consider the triangle $\triangle 145$ together with vertex 6. From Lemma 2.2 it follows that the sixth vertex has to lie in the interior of the triangle $\triangle 145$. With the same argument we can conclude that vertex 6 has to lie in the exterior of the triangles $\triangle 135$ and $\triangle 245$.

If we can prove that the triangle $\triangle 145$ is always covered by the triangles $\triangle 135$ and $\triangle 245$, then it follows that the sixth vertex cannot be placed in this drawing. We see this configuration in Figure 5.8.

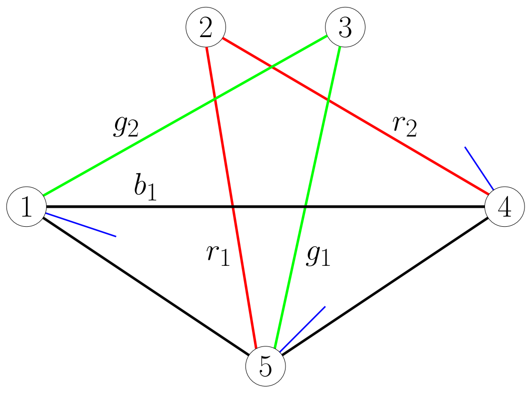


Figure 5.8: Basic configuration of the non-semi-realizable rotation system

To prove this we give names to the edges like in Figure 5.8: we denote by r_1 the edge from 5 to 2, r_2 the edge from 2 to 4, and analogously g_1 the edge from 5 to 3, g_2 the edge from 3 to 1 and b_1 the edge from 1 to 4. The crossings shown in Figure 5.8 follow from the rotation system.

Notice that with the definition of the inside and outside of a triangle it does not make a difference which region is the unbounded one in the drawing. See Figure 5.9 for an example. There, by our definition, the interior of the green triangle $\triangle 135$ is the unbounded region. For the proof we stick to our basic configuration, where the interior of the triangles is bounded. We remark that the proof also works if some of the interiors are unbounded.

We will show that the green triangle always covers at least the part of the black triangle that is not covered by the red triangle. First we notice that

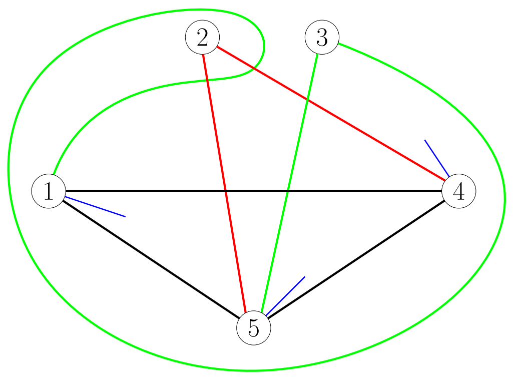


Figure 5.9: Green triangle closed in the other direction

if g_2 (resp. r_2) intersects the boundary of the black triangle $\triangle 145$, it can only do this at the edge form 4 to 5 (resp. 1 to 5). Since the edge has to leave triangle $\triangle 145$ again, it then creates an empty lens and the area of the black triangle covered by the green (resp. red) one gets bigger. That is, we can always redraw g_2 and r_2 in a way that they do not intersect the black triangle without increasing the area of the black triangle covered by the green and the red ones. Therefore we can assume in the proof that g_2 and r_2 do not intersect the black triangle.

We now look at edge r_1 (see Figure 5.10). When two edges have to cross once by the rotation system, in a semi-good drawing they may cross an arbitrary odd number of times. By Lemma 2.2, vertex 2 lies outside of the black triangle and hence the edge r_1 has to cross the edge b_1 an odd number of times. By multiple crossings of r_1 with b_1 it can form regions bounded only by those two edges. Vertex 3 cannot lie in any of these regions since they lie either inside the red or the black triangle, and vertex 3 lies in the exterior of both (by Lemma 2.2).

Therefore, starting from vertex 5, the edge g_1 crosses an odd number of

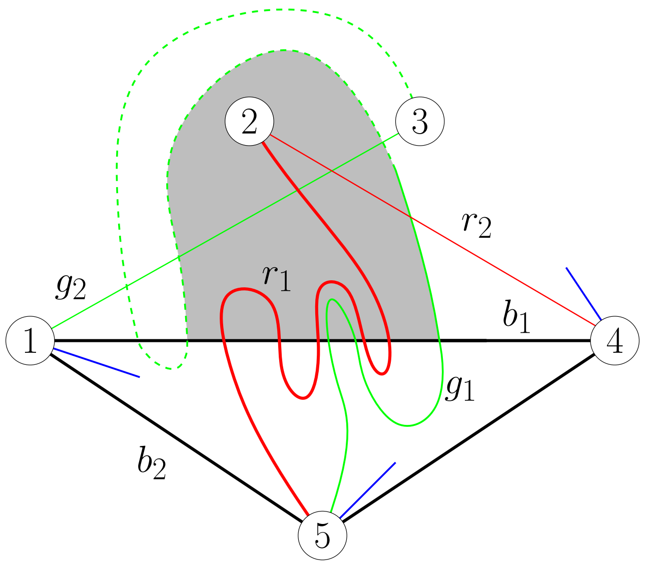


Figure 5.10: Illustration of the proof of Theorem 5.2

times the edge b_1 inside the red triangle. Since the edges r_1 and g_1 are not allowed to cross, it follows that g_1 stays inside the red triangle until it crosses r_2. It remains to show that g_1 cannot cross b_1 outside the red triangle. If g_1 crosses b_1 outside the red triangle, then it has to surround vertex 2 or surround the whole black triangle. In both cases it creates a lens formed by g_1 and b_1 including vertex 2 and not vertex 1 and 4. Vertex 3 has to lie outside this lens, because otherwise edge g_2 is not possible. This follows from the fact that g_2 is not allowed to cross any of b_1 and g_1, because g_2 shares a vertex with each of b_1 and g_1.

Since g_2 is not allowed to cross the lens formed by g_1 and b_1 and the lens is to the right of g_1, it follows that the lens lies completely within the green triangle and therefore vertex 2 lies also inside this triangle. This is a contradiction to the fact that vertex 2 lies outside the green triangle (which follows from the rotation system, by Lemma 2.2).

Recalling that we were assuming without loss of generality that g_2 and r_2 do not intersect the black triangle, it follows that the green and the red triangles cover the black triangle completely. This contradicts the assumption that we are able to place a sixth vertex inside the black triangle but outside the green and the red ones. □

The next question is how we can decide whether a rotation system is semi-realizable. As we have seen, for realizability it suffices that all 4-tuples and 5-tuples are realizable. For semi-realizability it does not suffice to analyze only the 4-tuples, but one could think we only have to analyze all 6-tuples. For that reason we checked all rotation systems with seven vertices and realizable 4-tuples that do not include the non-semi-realizable rotation system with six vertices as a sub-rotation system. As we will see, the results obtained show that it does not suffice to check all 6-tuples to guarantee semi-realizability.

5.3 Results for Semi-Good Drawings with Seven Vertices

We know that there is a 6-tuple that prohibits the semi-realizability of a rotation system. We wanted to know if there are also new configurations that show up with seven vertices for the first time or if the semi-realizability of all 6-tuples suffices for the semi-realizability of rotation systems with seven vertices. To do this we checked all rotation systems with seven vertices and realizable 4-tuples.

There are 39 349 rotation systems with seven vertices where all 4-tuples are realizable. We already know from [1] that 11 556 rotation systems are

39 349	total number of rotation systems with realizable 4-tuples
11 556	good drawings
20 634	semi-good drawings with at most 2 crossings per edge-pair
3379	semi-good drawings with at most 3 crossings per edge-pair
2152	semi-good drawings with at most 4 crossings per edge-pair
568	semi-good drawings with at most 5 crossings per edge-pair
154	semi-good drawings with at most 6 crossings per edge-pair
27	semi-good drawings with at most 7 crossings per edge-pair
34	semi-good drawings with at most 8 crossings per edge-pair
14	semi-good drawings with at most 9 crossings per edge-pair
11	semi-good drawings with at most 10 crossings per edge-pair
340	rotation systems include the not realizable 6-tuple
480	not semi-good drawable rotation systems

Table 5.4: Rotation systems for $n = 7$ and their realizability

drawable as a good drawing. This means we have 27 793 rotation systems left. From this number we can remove the rotation systems that have the non-semi-realizable rotation system with six vertices as a sub-rotation system. These are 340. The remaining 27 453 rotation systems have to be tested whether they are realizable as a semi-good drawing. In Table 5.4 we present the number of rotation systems and how they are realizable as semi-good drawings.

We can see in Table 5.4 that we have 480 rotation systems left for which all 4-tuples, 5-tuples, and 6-tuples are semi-realizable, but the whole rotation system is not semi-realizable. The proof for this non-realizability follows from the observations in Section 5.4. So checking all 4-tuples and 6-tuples does not suffice to decide semi-realizability. There are structures that show up in rotation systems with seven vertices for the first time that can prevent the rotation systems from being semi-realizable.

5.4 Maximum Number of Crossings per Edge-Pair

To check the rotation systems for their realizability as a semi-good drawing, we had to limit the number of crossings per edge-pair we allow. To get a reliable result we need to know how large the maximum number of crossings for an edge-pair can get. When we allow empty lenses, this number is infinite. Since we know that every semi-realizable rotation system can be realized without empty lenses, we only consider such semi-good drawings. We have seen that for $n = 6$ we only need four crossings per edge-pair to realize all rotation systems that are semi-good drawable. In contrast to the uniqueness of the drawings for $n = 5$, there exists a rotation system with six vertices that can be drawn with five crossings for one edge-pair without empty lenses. For an example see Figure 5.11. The red marked edges cross five times. Nevertheless, the underlying rotation system can also be drawn as a semi-good drawing with only three crossings per edge-pair. This drawing is shown in Figure 5.12. In this drawing the edge pair $\{(1,6),(4,5)\}$ crosses only once and the edge-pair $\{(1,6),(3,4)\}$ crosses three times. This shows that we might not need the maximum number of crossings that is possible to realize all semi-realizable rotation systems.

To determine the maximum number of crossings per edge-pair for $n = 6$ and 7 we enumerate all different ways how two edges can cross multiple times. This is an iterative process starting with one crossing. To do this we first make some observations, which will make our enumeration much easier and more efficient.

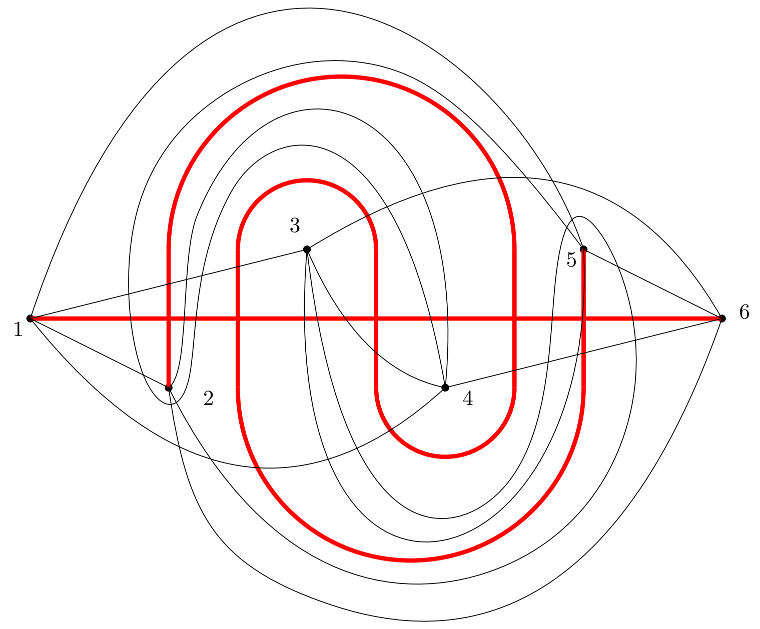


Figure 5.11: Example of a drawing with six vertices and five crossings of one edge-pair

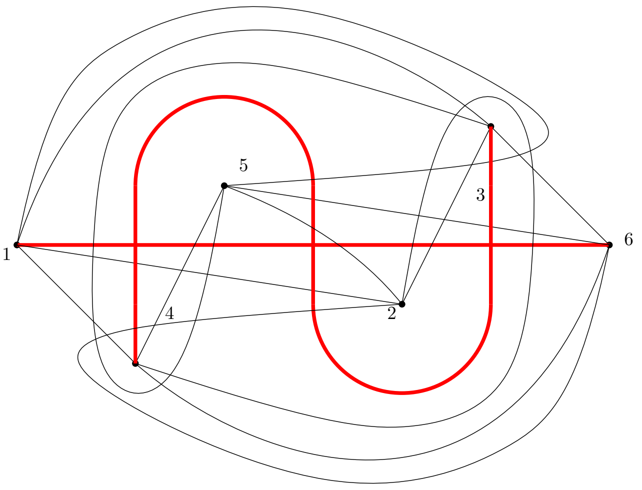


Figure 5.12: Rotation system of Figure 5.11 realized with only three crossings of one edge-pair

5.4.1 Basic Generalizations

The first very basic observation is that we can draw one edge as a finite horizontal line connecting its endpoints. In every semi-good drawing we can stretch out an arbitrary edge and draw it as a straight line. This can be realized with a homeomorphism on the sphere and therefore it does not change the strong isomorphism class of the drawing. We will call this edge the *horizontal edge* and the edge crossing it multiple times the *second edge*. We also know from Lemma 5.2 that the two edges have to cross in different directions alternately.

Our enumerated configurations should be sub-drawings of a semi-good drawing of the complete graph with n vertices. Therefore we have to make sure that the considered drawings of two edges are completable as semi-good drawings. So it is necessary that all endpoints of the two edges lie in the same cell. This is another restriction our considered sub-drawings must fulfill.

Lemma 5.4 *Let $D(K_n)$ be a semi-good drawing of the complete graph K_n, with $n \geq 4$ and $e_1 = (v_1, v_2)$, $e_2 = (w_1, w_2)$ are two edges. In the subdrawing D' that only consists of the two edges e_1 and e_2, the four vertices v_1, v_2, w_1, and w_2 lie in the same cell.*

Proof: All cells in D' can only be bounded by e_1 and e_2. Assume that v_1 lies in one of these cells. In the complete drawing, v_1 is connected to both w_1 and w_2 by an edge. None of these two edges is allowed to cross e_1 or e_2, because they share an endpoint. Therefore, w_1 and w_2 have to lie in the same cell as v_1. The same is true for v_2. It follows that all four vertices have to lie in the same cell. $\square$

The next configuration that we prohibit is that the second edge rounds both endpoints of the horizontal edge. This case is shown in the left drawing in Figure 5.13. Since isomorphism is defined on the sphere, this drawing is isomorphic to the right drawing in Figure 5.13. Hence it is sufficient to consider only the right one.

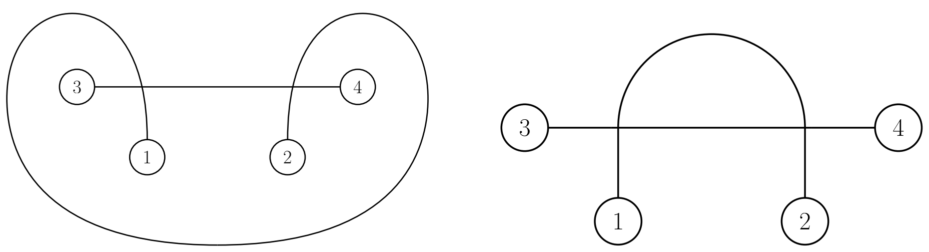


Figure 5.13: Isomorphic drawings when the edge rounds both endpoints

Any other loop between two crossings of the horizontal line can be redrawn as a simple half circle as in Figure 5.14 by a homeomorphism. So we draw our configuration as simple as possible to cover all sub-drawings of two edges we have to consider. By this restrictions we can assume that our second edge does not cross the dashed lines in the left drawing of Figure 5.14.

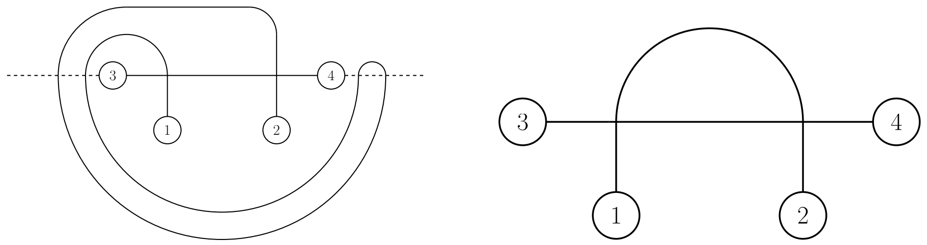


Figure 5.14: Complicated loops can be redrawn to simply half-circles on one side of the edge

5.4.2 Prohibited Sub-Drawings

As already stated, it is necessary that the drawing of the two edges and their vertices is completable. We do not know how to decide whether a sub-drawing of a graph is completable to a semi-good drawing efficiently, but we know a few configurations that cannot appear in a semi-good drawing of the complete graph. This is sufficient to get an upper bound for the number of crossings of an edge-pair in a semi-good drawing without empty lenses, with a small number of vertices in total.

The first configuration is that we have to avoid empty lenses. We are allowed to create empty lenses as long as we have points left that we could place in the lens. If all remaining points are placed in a lens, then we are not allowed to create new empty lenses.

As already pointed out in Lemma 5.4 we have to make sure that we do not lock any of the four endpoints of our two edges in a lens. Therefore we are not allowed to loop around any of our fixed points of the two edges. Otherwise we lock this point in a lens and the drawing is not completable.

The third configuration that we have to avoid is the configuration of a *spiral* (see Figure 5.15). In this configuration the edge from the vertex x, which is placed in the lens, to the vertex y has to cross the horizontal edge (a, b) at least three times. The second crossings is to the left and the third one is to the right of x (green line in Figure 5.15). This is the only way how vertex x can be connected to vertex y. We can see in Figure 5.15 that x is then locked in a lens bounded by the edges (a, b) and (x, y) (gray region in Figure 5.15). Since the vertices a and b are on the outside of this lens, they can not be connected to vertex x. So having a spiral in a drawing means that the drawing is non-completable.

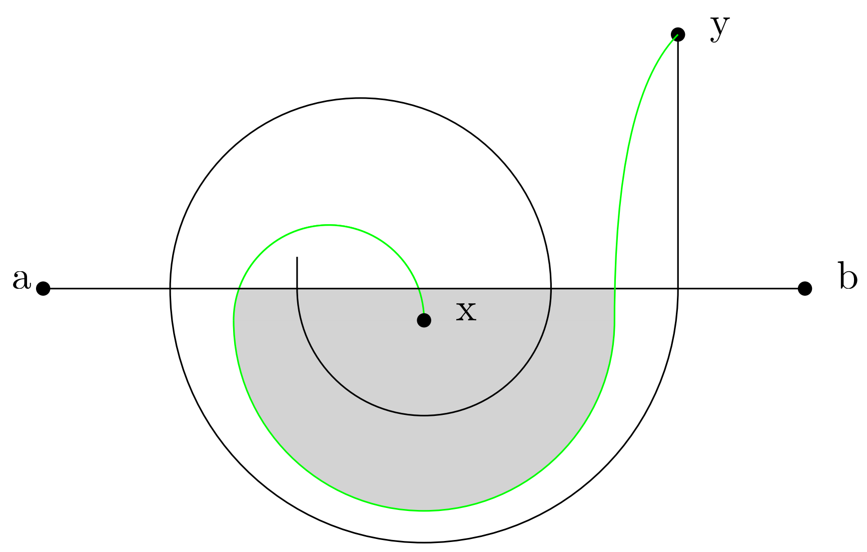


Figure 5.15: Spiral of two edges

5.4.3 Enumeration of all Allowed Crossing-Configurations

Now we show how we can compute the maximum number of crossings per edge-pair in a semi-good drawing of the complete graph K_n without empty lenses. To do this we generate all different configurations how two edges can cross that do not contain our forbidden sub-drawings. We start with three fixed vertices, which are the two endpoints of the horizontal edge and our starting vertex of the second edge. We draw a vertical line from our starting point crossing the horizontal line. Then in every step we extend this drawing at the end of this line in all possible ways. That is, we avoid spirals, empty lenses and locking in a fixed vertex. In every extension step we have to cross the horizontal line in the inverse direction of the last crossing. So we have to cross back in all ways and check if this does not violate our regulations. In the first step we just cross back to the left of the first crossing. We do not have to consider the case crossing back to the right, because all drawings arising from that are only the mirrored drawings of the others. During this generation we do not care whether the end of the line lies in a lens. Since this is the spot where we extend the edge, as it could happen that a few steps further the end of the line is again in the cell where all other vertices lie. We can see this in Figure 5.16 in the extension from the bottom most drawing of column $c = 3$ to the bottom most drawing of column $c = 4$. In this extension step we can also see that we surround the existing point in the lens. It makes no sense to create another empty lens and place another point there, because then the former lens would also be blocked by the newly placed point. So we can use only one point to make sure that both lenses are non-empty.

In Figure 5.16 we can see how we extend the drawings of two edges to the maximum number of crossings for six vertices. There we always extend the drawings at the red cross, which marks the end of the line. We do not create loops around any of our three fixed vertices. We extend the drawings in all possible ways and check if they contain empty lenses or spirals. If we have vertices left, we place a vertex in the empty lens. If we create a new empty lens inside another lens we can reuse the vertex to make the new lens non-empty. Therefore we surround existing vertices in lenses whenever it is possible, to reduce to number of used vertices. Otherwise these drawings are marked and will not be extended. Note that it happens in intermediate steps that the extension point lies in a lens. We also extend these drawings, because thereby the extension point can leave the lens again and reach the cell where the other three end-vertices lie.

From this extension of drawings we get that for six points we can have at most five crossings per edge-pair. We also know that four crossings are sufficient to draw all semi-realizable rotation systems. We also did this analysis by hand for seven vertices. So we know that for seven vertices we can get at most ten crossings per edge-pair. Actually there are rotation

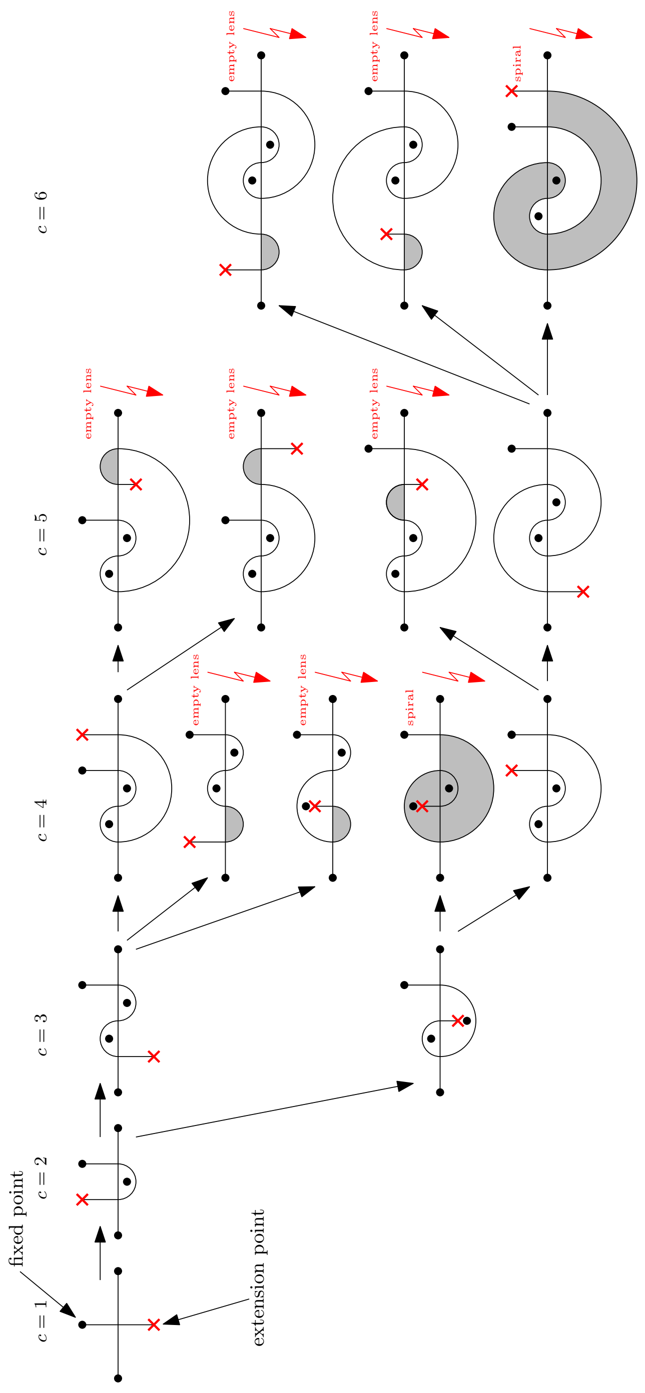


Figure 5.16: All possible crossing-configurations of two edges with 6 vertices (c represents the number of crossings of the two edges)

systems that are only semi-realizable with ten crossings for at least one edge-pair.

5.4.4 Algorithm to Compute the Maximum Number of Crossings per Edge-Pair

We implemented the described extension progress. The data-structure for this is very easy. We use two arrays. In the first array we store the positions where we cross the horizontal line ordered by the second line starting at the fixed vertex. In the second array we store the positions where the horizontal edge is crossed and where the points in the lenses are placed ordered by the x-coordinate. There we can also store whether the point lies above or below the line. With this data-structure it is possible to compute the positions where we can cross the horizontal line in every step. With a recursive algorithm we check all potential crossing positions. As before, we prohibit the generation of empty lenses and spirals. Whenever we enter a lens where already a point is placed, we surround this point and reuse it to make sure that we need as few extra points as possible.

The detection of an empty lens is easily possible. We only have to check whether there lies a point on the appropriate side of the horizontal edge between the last two crossing positions. If this is not the case we put a point on the appropriate side between the two last crossing positions if we have one left. If there is no point left we stop this recursion branch.

The detection of a spiral is a little bit more sophisticated. There we make use of the following lemma.

Lemma 5.5 *Consider a horizontal edge e_1 and an edge e_2 that crosses e_1 multiple times. Further, consider a lens in this drawing of the two edges with a vertex v in it. Then e_2 crosses e_1 on the left and on the right side of the vertex. These two crossings cut e_2 into three sub-curves. We call them the left curve, the right curve and the middle curve (see Figure 5.17). We define that the points where the left and the right curve meets the middle curve is part of the middle curve.*

The vertex v lies in a spiral in the sense of Figure 5.15 if one of the following two symmetric conditions is fulfilled:

1. *The left curve crosses e_1 on the left side an even number of times, then on the right side an odd number of times and then again on the left side.*
2. *The right curve crosses e_1 on the right side an even number of times, then on the left side an odd number of times and then again on the right side.*

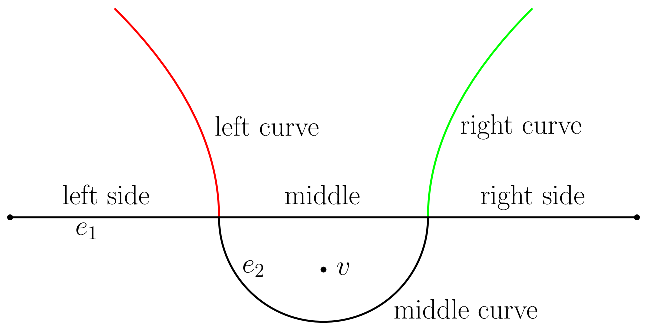


Figure 5.17: Configuration for Lemma 5.5

Proof: We show that if the first condition is fulfilled, then the vertex lies in a spiral. Consider the case of Figure 5.17, where the lens is below the horizontal edge. Since we assume that the left curve crosses e_1 on the left side an even number of times and then on the right side, we know that there exists a curve above e_1 from the left to the right side. From this it follows that the edge from v to the the end-vertex of the left curve has to cross e_1 to the right of the right curve, because the left curve locks this edge above e_1 in a lens and the endpoint of the left curve has to be outside this lens by Lemma 5.4. Now the left curve crosses e_1 on the right side an odd number of times and then again on the left side. This means there also exists a curve below the lens that goes from right to left. Therefore the point lies in a spiral. We can see this in Figure 5.18. The proof for the second condition runs analogously. $\square$

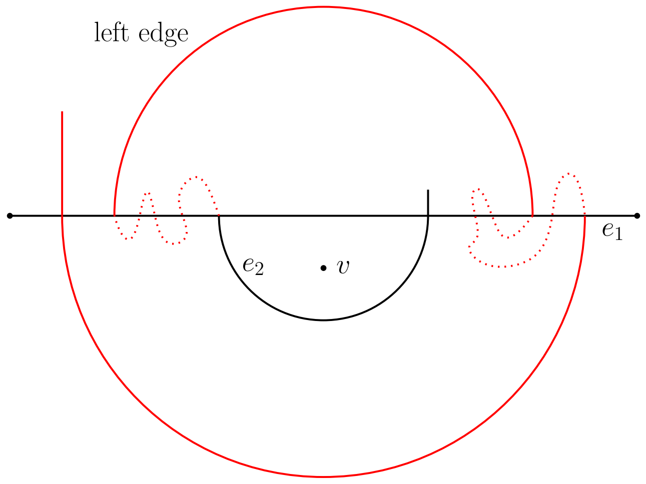


Figure 5.18: Characterization of a spiral

With our algorithm we can determine the left and the right curve to every vertex. Then we can follow these sub-curves and determine whether they form a spiral.

So we can compute the maximal number of crossings for an edge-pair in a semi-good drawing of the complete graph K_n with our algorithm. In Table 5.5 we display the results of our algorithm for up to 13 vertices. The first row contains the number of vertices in total. The number in the second row is the upper bound for the number of crossings that two edges can have without empty lenses and spirals computed by our algorithm. In the third

Vertices	4	5	6	7	8	9	10	11	12	13
Max. crossings per edge-pair	1	2	5	10	27	35	59	83	143	197
Really needed crossings	1	2	4	10						

Table 5.5: Upper bounds and exact values for the number of crossings per edge-pair

row we show the number of crossings per edge-pair that we really needed to draw all semi-realizable rotation systems up to seven vertices. It is very interesting that we do not need the upper bound for six vertices, but for seven we do. Up to now, we do not know whether the number of crossings for an edge-pair that fulfills our regulations or even in a semi-good drawing is finite for a finite total number of vertices.

For a given n there exists a construction with two edges, without empty lenses and spirals that has $\Theta(n^2)$ crossings. So we know that the lower bound for this number is $\Theta(n^2)$ [24].

6 Related Topics and Summary

6.1 Conway's Thrackle Conjecture

Conway's Thrackle conjecture deals with good drawings where all edges share an endpoint or have a common crossing, called *thrackles*. Conway conjectured that there exists no thrackle with more than n edges of a graph with n vertices.

Definition 6.1 (Thrackle) *[18] A thrackle is a good drawing of a graph where each pair of edges either share an endpoint or cross exactly once.*

Conjecture 6.1 (Conway's Thrackle Conjecture) *[28] The number of edges of a thrackle cannot exceed the number of its vertices.*

This conjecture implies that every good drawing of the complete graph with n vertices cannot contain a thrackle with $n+1$ edges. We generalize this and consider now semi-good drawings. We were able to find a semi-good drawing with seven vertices that contains a subgraph of eight edges that pairwise are either incident to a common vertex or cross an odd number of times (Figure 6.1). This can be seen from the rotation system and its crossing properties, which are determined by the 4-tuples. By the definition above, this is no thrackle and therefore no counterexample to the conjecture, because it has edge-pairs that cross multiple times. Nevertheless, it shows that the condition of a thrackle being a good drawing is necessary and must be considered in a possible proof of the thrackle conjecture. Thinking about rotation systems, it is possible to determine the crossing edges by analyzing the 4-tuples. When we try to prove the thrackle conjecture with the use of rotation systems, it is necessary to take the property of realizability into account.

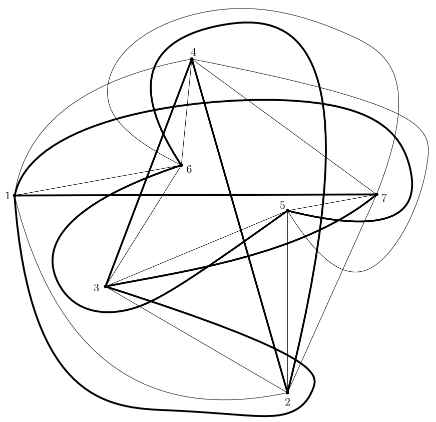


Figure 6.1: A "thrackle" in a semi-good drawing

6.2 Plane Hamilton Cycles in Semi-Good Drawings

Up to now there is no example of a good drawing without a plane Hamiltonian cycle. It is conjectured that every good drawing contains such a cycle. We know that the conjecture is not true for semi-good drawings. There exists an example with six vertices that does not contain a plane Hamiltonian cycle. The drawing is depicted in Figure 6.2.

All the information that we need to decide whether a good drawing contains a plane Hamiltonian cycle is contained in the rotation system of it. For a proof of the Hamiltonian cycle conjecture using rotation systems, we have to consider the property of realizability of the rotation system.

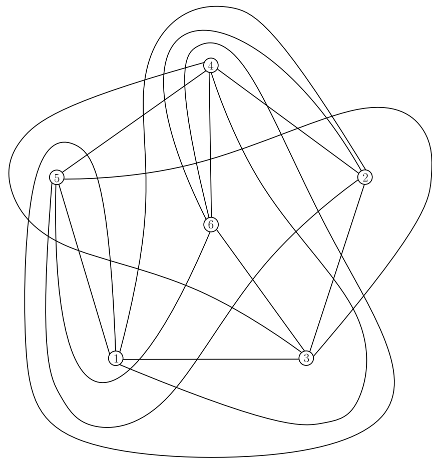


Figure 6.2: A semi-good drawing without a plane Hamiltonian Cycle

6.3 Summary and Open Problems

We motivated that it might be difficult to generate a realizable rotation system randomly. For a small number of vertices, it is doable within reasonable time, but already when the number of vertices exceeds eight, it gets very hard to generate a single random realizable rotation system from scratch. We found out that with the help of good double-flips we can generate larger more or less random realizable rotation systems. There we have to start with a realizable rotation system and modify it in a good way. An open problem in this area is whether the flipgraph of good drawings is connected by good double-flips. We have shown that this is true for $n \leq 8$.

With the use of our algorithm to generate random realizable rotation system we can support the conjecture that every good drawing contains a plane Hamiltonian cycle. We have seen that for semi-good drawings this conjecture is not true.

We stated the theoretical bounds for crossing families with three and four edges. For complete graphs, these bounds leave a big gap to be closed. We did a full analysis of good drawings without a crossing family of size three and showed that every good drawing with at least eleven vertices contains a crossing family size three.

The assumption that every rotation system that consists of realizable 4-tuples is semi-good drawable could be rejected. We showed that there exists a rotation system with six vertices where every 4-tuple is realizable that is not drawable as a semi-good drawing. We proved by hand that this rotation system is indeed not semi-realizable. With our analysis of semi-good drawings with seven vertices, we showed that there are also examples that are not semi-realizable, although all 4-tuples and 6-tuples are semi-realizable. This means there are structures that prevent semi-realizability that show up in rotation systems with more than six vertices for the first time. The proof that the examples with seven vertices are non-realizable is computer-assisted. It would be interesting to see a direct proof by hand of the non-realizability of one of these examples with seven vertices. Such a proof might give us a better insight in the structures that prevent semi-realizability.

We were surprised how often two edges need to cross in some semi-good drawings. We computed an upper bound for the number of crossing of an edge-pair in semi-good drawings with up to 13 vertices.

A lot of questions regarding the realizability of rotation systems and the properties of good and semi-good drawings are still unsolved. In the following we summarize a few of them.

- We have presented the good double-flip, which is an operation that modifies a semi-realizable rotation system and leaves all 4-tuples realizable. Does there exist a similar operation for realizable rotation systems that also leaves the 5-tuples realizable?
- We have seen that we do not have to check all 4-tuples of a rotation system to make sure that all 4-tuples are realizable. Is there a similar property that holds for 5-tuples? How fast can we check the realizability of all 5-tuples? Can the bound of $\mathcal{O}(n^5)$ time for deciding the realizability of rotation systems be improved?
- It is still unsolved if every good drawing contains a Hamiltonian cycle. Is it possible to prove this conjecture with the use of realizable rotation systems?
- We know that there are good drawings with 15 vertices that do not contain a crossing family of size 4. What is the tight lower bound for the number of vertices such that every good drawing contains a crossing family of size 4? How can we find the answer to this question? It seems like an entire extension of all rotation systems is not doable in appropriate time.
- Up to now we do not know how we can decide semi-realizability of rotation systems efficiently. We know that checking the semi-realizability of all 6-tuples is not sufficient. What is the complexity of deciding semi-realizability?
- We found structures that bound the number of times two edges can cross in a semi-good drawing of the complete graph with at most 13 vertices and without empty lenses. Is this number finite for every finite number of vertices? If yes, what is an upper bound in dependance of n?

List of Figures

Bibliography

[1] Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Thomas Hackl, Jürgen Pammer, Alexander Pilz, Pedro Ramos, Gelasio Salazar, and Birgit Vogtenhuber. "All Good Drawings of Small Complete Graphs." In: *Proc. 31st European Workshop on Computational Geometry EuroCG '15*. Ljubljana, Slovenia, 2015, pp. 57–60 (cit. on pp. 4, 14, 15, 18, 21, 39, 48, 69, 75).

[2] Bernardo M. Ábrego, Oswin Aichholzer, Silvia Fernández-Merchant, Pedro Ramos, and Gelasio Salazar. "Shellable drawings and the cylindrical crossing number of *K_n*." In: *ArXiv e-prints* (2013). arXiv: `1309.3665 [math.CO]` (cit. on p. 4).

[3] Eyal Ackerman. "On the Maximum Number of Edges in Topological Graphs with no Four Pairwise Crossing Edges." In: *Discrete & Computational Geometry* 41.3 (2009), pp. 365–375. ISSN: 1432-0444. DOI: `10.1007/s00454-009-9143-9`. URL: `http://dx.doi.org/10.1007/s00454-009-9143-9` (cit. on p. 41).

[4] Eyal Ackerman and Gábor Tardos. "On the maximum number of edges in quasi-planar graphs." In: *Journal of Combinatorial Theory, Series A* 114.3 (2007), pp. 563–571. ISSN: 0097-3165. DOI: `http://dx.doi.org/10.1016/j.jcta.2006.08.002`. URL: `http://www.sciencedirect.com/science/article/pii/S0097316506001397` (cit. on p. 41).

[5] Pankaj K. Agarwal, Boris Aronov, János Pach, Richard Pollack, and Micha Sharir. "Quasi-planar graphs have a linear number of edges." In: *Combinatorica* 17.1 (1997), pp. 1–9. ISSN: 1439-6912. DOI: 10.1007/BF01196127. URL: http://dx.doi.org/10.1007/BF01196127 (cit. on p. 41).

[6] Oswin Aichholzer, Franz Aurenhammer, and Hannes Krasser. "Enumerating Order Types for Small Point Sets with Applications." In: *Order* 19.3 (2002), pp. 265–281. ISSN: 1572-9273. DOI: 10.1023/A:1021231927255. URL: http://dx.doi.org/10.1023/A:1021231927255 (cit. on p. 42).

[7] Oswin Aichholzer, Thomas Hackl, Alexander Pilz, Pedro A. Ramos, Vera Sacristán, and Birgit Vogtenhuber. "Empty triangles in good drawings of the complete graph." In: *CoRR* abs/1306.5081 (2013). URL: http://arxiv.org/abs/1306.5081 (cit. on p. 7).

[8] Oswin Aichholzer and Alexander Pilz. "Personal communication." Nov. 2016 (cit. on p. 34).

[9] J. Blažek and M. Koman. "A minimal problem concerning complete plane graphs." In: *Theory of Graphs and Its Applications (ed. M. Fiedler)* (1964), pp. 113–117 (cit. on p. 4).

[10] Stefan Felsner. *Geometric Graphs and Arrangements*. Vieweg+Teubner Verlag, 2004. ISBN: 978-3-528-06972-8. DOI: 10.1007/978-3-322-80303-0 (cit. on pp. 1–3).

[11] Richard K. Guy. "A combinatorial problem." In: *Nabla (Bull. Malayan Math. Soc.)* 7 (1960), pp. 68–72 (cit. on p. 4).

[12] Frank Harary and Anthony Hill. "On the Number of Crossings in a Complete Graph." In: *Proceedings of the Edinburgh Mathematical Society* 13.4 (1963), pp. 333–338. DOI: 10.1017/S0013091500025645. URL: https://www.cambridge.org/core/journals/proceedings-of-the-

edinburgh-mathematical-society/article/div-classtitleon-the-number-of-crossings-in-a-complete-graphdiv/A38E01D40F9382988F187EA9DEB39DC9 (cit. on pp. 3, 4).

[13] Hongmei He, Ondrej Sýkora, and Erkki Mäkinen. "Genetic algorithms for the 2-page book drawing problem of graphs." In: *Journal of Heuristics* 13.1 (2007), pp. 77–93. ISSN: 1572-9397. DOI: 10.1007/s10732-006-9000-4. URL: http://dx.doi.org/10.1007/s10732-006-9000-4 (cit. on p. 49).

[14] Jan Kynčl. "Enumeration of simple complete topological graphs." In: *European Journal of Combinatorics* 30.7 (2009), pp. 1676–1685. ISSN: 0195-6698. DOI: http://dx.doi.org/10.1016/j.ejc.2009.03.005. URL: http://www.sciencedirect.com/science/article/pii/S0195669809000523 (cit. on p. 13).

[15] Jan Kynčl. "Improved Enumeration of Simple Topological Graphs." In: *Discrete & Computational Geometry* 50.3 (2013), pp. 727–770. ISSN: 1432-0444. DOI: 10.1007/s00454-013-9535-8. URL: http://dx.doi.org/10.1007/s00454-013-9535-8 (cit. on p. 21).

[16] Jan Kynčl. "Simple Realizability of Complete Abstract Topological Graphs in P." In: *Discrete & Computational Geometry* 45.3 (2011), pp. 383–399. ISSN: 1432-0444. DOI: 10.1007/s00454-010-9320-x. URL: http://dx.doi.org/10.1007/s00454-010-9320-x (cit. on pp. 9, 12, 22).

[17] Jan Kynčl. "Simple Realizability of Complete Abstract Topological Graphs Simplified." In: *Graph Drawing and Network Visualization: 23rd International Symposium, GD 2015, Los Angeles, CA, USA, September 24-26, 2015, Revised Selected Papers*. Ed. by Emilio Di Giacomo and Anna Lubiw. Springer International Publishing, 2015, pp. 309–320. ISBN: 978-3-319-27261-0. DOI: 10.1007/978-3-319-27261-0_26. URL: http://dx.doi.org/10.1007/978-3-319-27261-0_26 (cit. on p. 22).

[18] László Lovász, János Pach, and Mario Szegedy. "On Conway's Thrackle Conjecture." In: *Discrete & Computational Geometry* 18.4 (1997), pp. 369–376. ISSN: 1432-0444. DOI: 10.1007/PL00009322. URL: http://dx.doi.org/10.1007/PL00009322 (cit. on p. 89).

[19] János Pach, Radoš Radoičić, and Géza Tóth. "Relaxing Planarity for Topological Graphs." In: *Discrete and Computational Geometry: Japanese Conference, JCDCG 2002, Tokyo, Japan, December 6-9, 2002. Revised Papers*. Ed. by Jin Akiyama and Mikio Kano. Springer Berlin Heidelberg, 2003, pp. 221–232. ISBN: 978-3-540-44400-8. DOI: 10.1007/978-3-540-44400-8_24. URL: http://dx.doi.org/10.1007/978-3-540-44400-8_24 (cit. on pp. 60, 61).

[20] János Pach and Géza Tóth. "How Many Ways Can One Draw A Graph?" In: *Combinatorica* 26.5 (2006), pp. 559–576. ISSN: 1439-6912. DOI: 10.1007/s00493-006-0032-z. URL: http://dx.doi.org/10.1007/s00493-006-0032-z (cit. on pp. 6, 12, 21, 24).

[21] János Pach and Géza Tóth. "Which Crossing Number Is It Anyway?" In: *Journal of Combinatorial Theory, Series B* 80.2 (2000), pp. 225–246. ISSN: 0095-8956. DOI: http://dx.doi.org/10.1006/jctb.2000.1978. URL: http://www.sciencedirect.com/science/article/pii/S0095895600919786 (cit. on p. 3).

[22] Jürgen Pammer. "Rotation Systems and Good Drawings." MA thesis. Institute for Software Technology, Graz University of Technology, Austria, 2014 (cit. on pp. 16, 39, 62).

[23] Shengjun Pan and R. Bruce Richter. "The crossing number of K11 is 100." In: *Journal of Graph Theory* 56.2 (2007), pp. 128–134. ISSN: 1097-0118. DOI: 10.1002/jgt.20249. URL: http://dx.doi.org/10.1002/jgt.20249 (cit. on p. 4).

[24] Irene Parada. "Personal communication." Jan. 2017 (cit. on p. 88).

[25] Nabil H. Rafla. "The good drawings D_n of the complete graph K_n." PhD thesis. McGill University, Montreal, 1988 (cit. on p. 40).

[26] Rob Shields. "Cultural Topology: The Seven Bridges of Königsburg, 1736." In: *Theory, Culture & Society* 29.4-5 (2012), pp. 43–57. DOI: `10.1177/0263276412451161`. eprint: `http://dx.doi.org/10.1177/0263276412451161`. URL: `http://dx.doi.org/10.1177/0263276412451161` (cit. on p. 1).

[27] Kevin Weiler. "Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments." In: *IEEE Comput. Graph. Appl.* 5.1 (1985), pp. 21–40. ISSN: 0272-1716. DOI: `10.1109/MCG.1985.276271`. URL: `http://dx.doi.org/10.1109/MCG.1985.276271` (cit. on p. 62).

[28] Douglas R. Woodall. "Thrackles and deadlock." In: *Combinatorial Mathematics and Its Applications* (1971), pp. 335–348 (cit. on p. 89).