DIPLOMARBEIT

Perfect Pseudo Matchings on Snarks

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Technische Mathematik

eingereicht von

Benjamin Schwendinger, BSc
Matrikelnummer: 01225371

ausgeführt am Institut für Logic and Computation
der Fakultät für Informatik der Technischen Universität Wien

Betreuung
Betreuer: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Mitwirkung: Projektass. Dipl.-Ing. Benedikt Klocker

Wien, am 09.04.2019

Benjamin Schwendinger

Günther Raidl

Kurzfassung

Das Problem des CYCLE DOUBLE COVER (CDC) wird nun seit über 45 Jahren von Graphentheoretikern betrachtet. Obwohl dieses Problem für viele Familien von Graphen bereits gelöst werden konnte, bleibt es für den allgemeinen Fall weiterhin ungelöst ([31]). Die Aufgabenstellung ist simpel: Gegeben sei ein brückenloser Graph G. Gibt es eine Familie von Zyklen, bestehend aus Kanten von G, sodass jede Kante von G von dieser Familie genau doppelt überdeckt wird?
In dieser Diplomarbeit versuchen wir einen neuen, bisher nicht untersuchten Lösungsansatz für das CDC zu entwickeln. Dabei greifen wir auf das exakte Verfahren der ganzzahligen linearen Programmierung zurück. Wir beginnen zuerst mit dem Begriff des Pseudo Matchings. Dieses stellt eine Generaliserung des graphentheoretischen Matchings dar. Analog zu gewöhnlichen Matchings definieren wir weiters den Begriff des perfect pseudo matching (PPM). In weiterer Folge betrachten wir den Kontraktionsgaphen, der durch die Kontraktion eines Graphen mit einem zugehörigen PPM entsteht. Sollte der Kontraktionsgraph planar bzw. zumindest ohne K_5 Minor sein, so beweisen wir fußfassend auf der Vorarbeit von Fan und Zhang ([11]), dass der ursprüngliche Graph dann ein CDC besitzen muss. Für planare Graphen wurde dies bereits durch Fleischner bewiesen ([13]). Nachdem Jaeger ([21]) bewies, dass ein Gegenbeispiel mit minimaler Kantenanzahl für das CDC ein Snark (ein zyklisch 4-Kanten zusammenhängender, brückenloser kubischer Graph mit chromatischem Index 4) sein muss, betrachten wir in Folge Snarks bis zu einer Ordnung von 52 Knoten. Dabei können wir einerseits das CDC für Graphen bis zu einer Knotenanzahl von 26 verifizieren, anderseits werden auch die Limitationen unseres neuen Ansatzes aufgezeigt. So finden wir Snarks mit 26 (bzw. 28) Knoten, die kein planares (bzw. K_5 Minor freies) PPM besitzen und für welche sich somit mittels unseres entwickelten Ansatzes keine Aussage treffen lässt, ob sie ein CDC besitzen. Um die Effizienz unseres entwickelten Ansatzes und des dahinter liegenden Algorithmus aufzuzeigen, werden zuletzt auch weitere zufällige kubische Graphen bis zu einem Knotengrad von 100 betrachtet.

Abstract

The graph theoretic problem of the Cycle Double Cover (CDC) has been around for over 45 years. It still remains to be an open problem, although specilizations for many families of graphs have been proven in this time period ([31]). The question is easy to state: Given a bridgeless graph G, does a collection of cycles of G exist, such that every edge of G appears in exactly two of these cycles?

In this thesis we try to develop a new approach for the CDC, which has not been investigated so far. There we will make use of integer linear programming as exact solution method. First, we start with the definition of a pseudo matching which is a generalization of the graph theoretic matching. Analogous to matchings we further define the term of a perfect pseudo matching (PPM). We continue with the examination of the contraction graph, which arises through the contraction of a graph and an according PPM of this graph. If the contraction graph is planar (or at least has no K_5 minor) then we will prove, based on the work of Fan and Zhang ([11]), that the original graph has to have a CDC. Fleischner proved this for planar graphs ([13]). Since Jaeger proved in ([21]) that a counterexample with a minimum number of edges to the CDC has to be a snark (a cyclically 4-edge connected, bridgeless, cubic graph with edge chromatic number 4), we will further examine snarks up to an order of 52 vertices. There we can verify the CDC for graphs up to a size of 26 vertices, but our experiments also show the limitations of our new developed approach. So we will find snarks with 26 (resp. 28) vertices for which no planarizing (resp. K_5 minor free) PPM exists and therefore our approach cannot decide, whether there exists a CDC for them or not. Last but not least to demonstrate the efficient running time of our approach we will test it with cubic random graphs with up to 100 vertices.

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Wien, am 09.04.2019

Benjamin Schwendinger

Contents

1 Introduction

The graph theoretic problem of the Cycle Double Cover (CDC) has been around for over 45 years. The question is easy to state: Given a bridgeless graph G, does a collection of cycles of G exists, such that every edge of G appears in exactly two of these cycles? The CDC has already been proven for many different classes of graphs ([31]). Moreover, it also has been established that a minimum counterexample to the CDC has to be a snark and therefore this class of graphs represents the bottleneck of the CDC conjecture ([21] or see also Theorem 4.1.7).

We elaborate a new approach to the CDC by using the definition of the perfect pseudo matching (PPM), which is a generalization of a matching. See Figure 1.1 for a representation of the problem reductions we want to use. In Theorem 4.2.3 we will show that if the contraction graph G/M of a cubic graph G and a PPM M of G, has a compatible cycle decomposition (CCD) that this implies the existence of a CDC for the original graph G. Hence instead of searching for a CDC in the original graph G, we just have to find a CCD for a much smaller minor of G. We further define the terms of planarizing perfect pseudo matchings (PPPMs) (resp. K_5-minor free perfect pseudo matchings (K5PPMs)). Now instead of trying to find a CCD explicitly we base our approach on the work of Fan and Zhang. With Theorem 4.2.2 they showed in ([11]) that for a K_5-minor free graph G, the transitioned graph (G,$\mathcal{T}$) has a compatible cycle decomposition for every admissible transition system $\mathcal{T}$ of G. Hence we can reduce our problem further to the search of a K5PPM on the much smaller graph G/M instead of searching for a CDC on the original graph G. Therefore, if we can prove the existence of K5PPMs for snarks up to a certain size, we can also validate the CDC for the same class of graphs.

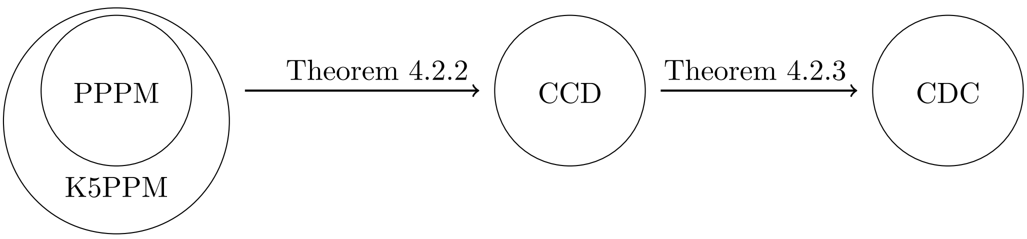

Figure 1.1: Reduction chain of our approach

Since we want to solve the problem of finding a PPPM exactly, we want to develop algorithms which use exact solution methods. Hence we will develop an enumeration approach as well as an integer linear programming approach to compare their strengths resp. weaknesses.

1.1 Aim of the Thesis

This thesis aims to develop a new approach for finding CDCs for snarks. Hereby, the focus does not lie on improving the bounds for a minimum counterexample as in [4] or [19], but on the approach and its practicability itself. Moreover should the new found process outrun an enumeration approach so that its usage is preferable for bigger instances.

1.2 Contribution

This thesis contributes to the research and understanding of PPMs, which are a generalization of matchings. Furthermore a connection between PPMs and the CDC is established. Hereby, Theorem 4.2.3 creates the basis for our new approach for solving the CDC. We elaborate this approach further to achieve appropriate running times for bigger instances.

1.3 Structure of the Work

We present a short overview over the individual chapters and the content of this thesis:

In Chapter 1 we give a short introduction of the CDC which is the underlying problem of this thesis and also introduce the aims of this thesis. Moreover, we give a brief scheme of the problem reductions we want to use.

In Chapter 2 we will discuss the work related to our problem.

In Chapter 3 we introduce the basic definitions and concepts which later will be needed. This covers basic introductions to the topics of graph theory, in particular planarity and K_5 minor testing, as well as integer linear programming.

In Chapter 4 we give a motivation for our solving approach. Here we justify why we concentrate on snarks for a potential solution to the CDC. Moreover we establish our relation between CCD and CDC via PPPMs resp. K5PPMs. This builds the graph theoretic basis for the following development of a solving algorithm.

In Chapter 5 we develop different algorithmic approaches. Here an enumeration approach as well as an integer linear programming approach is developed.

In Chapter 6 we give an overview over our test instances. Furthermore we examine the computational results of the previously developed approaches.

In Chapter 7 we draw conclusions and outline further possible work.

2 Related Work

In this chapter we will discuss problems which are related to our problem of finding a PPPM (resp. a K5PPM) for a given graph. We will mainly focus on work related to the core elements of our approach.

2.1 Graph Theory

2.1.1 Planarity Testing

The graph theoretic problem of planarity testing is the problem of determining whether a given graph is planar or not. This is a well studied problem in computer sciences and several algorithms solving this problem in linear time (linear in the number of edges), have already been found. The first linear time algorithm for solving this problem is due to Hopcroft and Tarjan ([18]) and was published in 1974. In ([8]) a characterization of planar graphs based on Trémaux trees is presented. This leads to a rather simple linear time algorithm for planarity testing.

2.1.2 K_5 Minor Testing

The graph theoretic problem of K_5-minor testing is the problem of determining whether a given graph contains a K_5-minor or not. This can be seen as a generalization of the planarity testing problem since due to Kuratowski's theorem, every planar graph is also K_5-minor free but not vice versa. In ([22]) a quadratic (quadratic in the number of edges) algorithm for testing whether a given graph is K_5-minor free is presented. Moreover it is shown how to extend this algorithm in such a way that it does not only report whether a graph contains a K_5-minor but if so, also returns a model of the found minor. In ([29]) a linear time (linear in the number of edges plus the number of vertices) for the K_5-minor testing problem is announced.

2.1.3 The Cycle Double Cover Conjecture

The CDC conjecture is an unsolved graph theoretic problem. It asserts that in each bridgeless graph G, a collection of cycles of G exist, such that every edge of G appears in two of these cycles? According to ([31]) it is unclear who stated the CDC first. In ([21]) it is shown that a minimum counterexample to the CDC has to be a snark. Since then the bounds for a minimum counterexample have been tightened. In ([19]) it is shown that a minimum counterexample to the CDC has to be a snark with girth at least 12.

2.1.4 Compatible Cycle Decomposition

In ([11]) Fan and Zhang proved that for a K_5-minor free graph G, the transitioned graph $(G, \mathcal{T})$ has a CCD for every admissible transition system $\mathcal{T}$ of G. Fleischner proved this for planar graphs already in 1980 ([13]). In ([15]) this result is generalized for eulerian graphs which do not contain a special type of K_5-minor.

2.1.5 Snarks

A snark is a cyclically 4-edge connected, bridgeless, cubic graph with edge chromatic number 4. The study of snarks already began in the 19th century when Tait showed in ([30]) that the four colour theorem is equivalent to the statement that no snark is planar. A planar snark is called a **boojum** and the existence of such would therefore refute the four colour theorem. The term snark itself goes back to Lewis Carroll ([5]). According to ([17]) this is because at the first appearance of snarks they seemed to be "very rare and unusual creatures". The first found snark is the Petersen graph which is widely used in graph theory as example and counterexample for various graph properties ([17]). Since then some infinite families of snarks have been found. In ([20]) methods for creating the two infinite families of Flower and Blanuša–Descartes–Szekeres snarks are presented. In ([4]) an algorithm for the generation of all non-isomorphic snarks of a given order, is presented. There they also generated all non-isomorphic snarks up to an order of 36. Moreover it is shown that there does not exist a counterexample to the CDC of order $n \leq 36$.

3 Preliminaries

Most parts of this thesis are based on graph theory, in particular planarity and minor containment of graphs, and integer linear programming. We begin with basic introductions to all of these topics. The advanced reader however might skip this chapter. Most of the definitions, notations, lemmas and theorems established in this chapter can be found in introductional books regarding these topics. We can particularly suggest the following: for the graph theory part ([9]), for the planarity part ([28]) and for the integer linear programming part ([7]).

3.1 Graph Theory

Most of the following notations and definitions are based on Diestel ([9]), but can also be found in most introduction books for graph theory.

A **directed graph** is a pair $G = (V, E)$ from a finite set V and a set E of ordered pairs (a, b) with $a, b \in V$ (see Figure 3.1). We call the elements of V **vertices** (or sometimes nodes) and the elements of E **edges**.

In the case of an edge $e = (a, b)$ with $a = b$ we call e a **loop** (see Figure 3.3). If two or more edges connect the same two vertices, we call them **multiple edges** (see Figure 3.2). Instead of $e = (a, b)$ we sometimes write just $e = ab$, where a is the start vertex and b is the end vertex of e. If the elements of E are not ordered, but only unordered pairs, we call G an **undirected** graph.

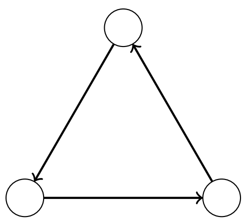

Figure 3.1: Directed graph

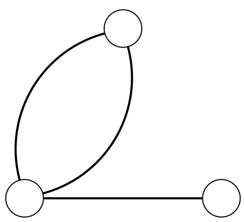

Figure 3.2: Undirected graph with multiple edges

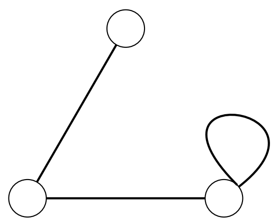

Figure 3.3: Undirected graph with a loop

The graph $G = (V, E)$, where V is the empty set, is called the **null graph**. Since we don't want to consider this graph, we set up the precondition that V is non-empty from now on.

A graph is **simple**, if it doesn't contain loops or multiple edges. For the rest of this thesis, we will only consider undirected simple graphs. Therefore whenever we talk about a graph, we always imply these conditions, unless other stated.

Two vertices $a, b \in V$ of a graph $G = (V, E)$ are **adjacent**, if (a, b) is an edge of E. If a and b are adjacent we also call them **neighbors** and we denote the set of neighbors for a certain vertex v in a certain graph G by $N_G(v)$.
Moreover, for an edge $e = (a, b)$ of a graph we say that e is **incident** to a respectively b. If two edges e, f are incident to the same vertex, then we also say that they are **adjacent**. This will not lead to confusions since we only defined adjacency for vertices so far. Furthermore, if all the vertices of G are pairwise adjacent, then G is called **complete**. The complete graph on n vertices is denoted by K_n (see Figure 3.4).

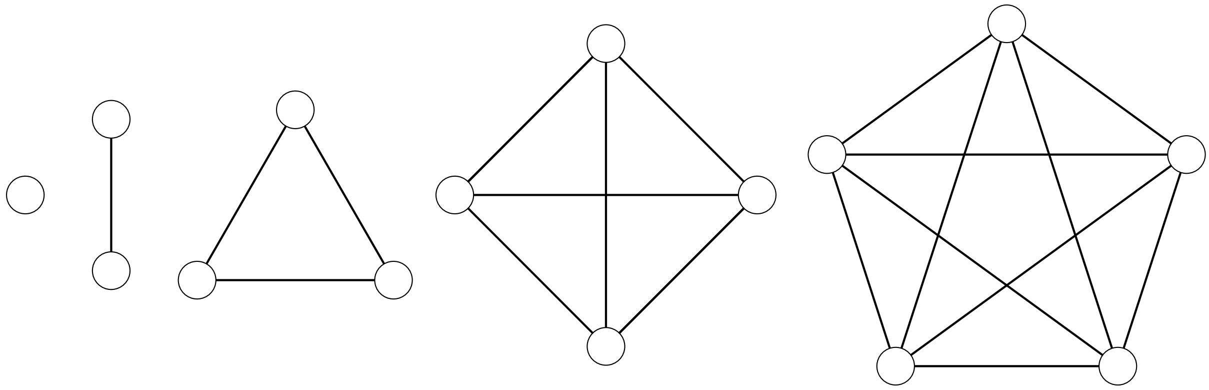

Figure 3.4: Complete graphs K_1 to K_5

Let $r \geq 2$ be an integer. A graph $G = (V, E)$ is **r-partite**, if V admits a partition into r classes such that the start vertex and the end vertex of each edge are in different classes. Vertices in the same partition class must not be adjacent.

Instead of 2-partite, we usually say **bipartite**. An r-partite graph in which every two vertices from different partition classes are pairwise adjacent is also called **complete** (see Figure 3.5).

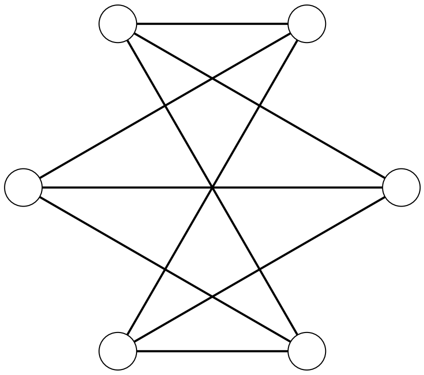

Figure 3.5: Complete bipartite graph with 6 vertices, $K_{3,3}$

The number of incident edges to a certain vertex v is called the **degree** of v and is denoted by $d(v)$. Furthermore, we denote the minimum degree of G by $\delta(G) := \min\{d(v)|v \in V\}$. If all the vertices of a graph G have the same degree k, then G is called **k-regular**. Moreover a 3-regular graph is called **cubic** (see Figure 3.6).

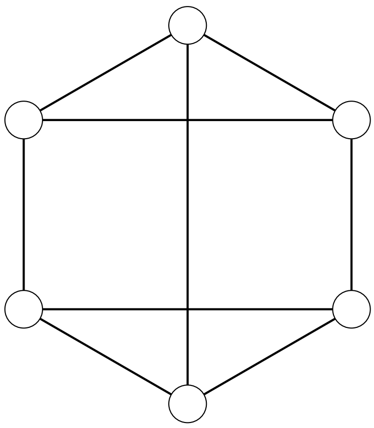

Figure 3.6: Cubic graph with 6 nodes

Lemma 3.1.1 (Handshaking lemma ([9])). *Let $G = (V, E)$ be a graph. Then*

$$\sum_{v \in V} d(v) = 2|E|.$$

Proof: Since each edge is incident to two vertices, it counts as 2 in the sum of the degrees. Hence, if we do this for all edges we see that the sum of the degree has to be 2 times the number of edges. □

From the handshaking lemma it follows directly that in a graph, the number of vertices with odd degree is even. This can be compared to the number of people in party, who have shaken an odd number of other people's hand, has to be even, hence the name handshaking lemma.

Let $G = (V, E)$ and $G' = (V', E')$ be graphs. If $V' \subseteq V$ and $E' \subseteq E$, then G' is a **subgraph** of G (and G a **supergraph** of G'), denoted as $G' \subseteq G$. We call G' a spanning subgraph of G, if $V = V'$. If G' contains all the edges of G that connect two vertices in V', then G' is said to be **induced** by V', which we denote by $G[V'] = G'$.

Let G be a graph and let $G' = G[C]$ be the graph induced by C. If G' is a complete graph, then we call C a **clique** in G.

Now we can also represent the **deletion of vertices and edges** (see Figure 3.7). Let $G = (V, E)$ be a graph. For the deletion of a set of vertices $W \subseteq V$ from G, the graph we obtain is $G' = G[V \backslash W]$, which we denote by $G' = G - W$. If W only consists of a single vertex v, we will also use $G - v$ instead of $G - W$. For the deletion of an edge set F from G, the graph we obtain is $G' = (V, E \backslash F)$, which we denote by $G' = G - F$. If E only consists of a single edge e, we will also use $G - e$ instead of $G - F$.

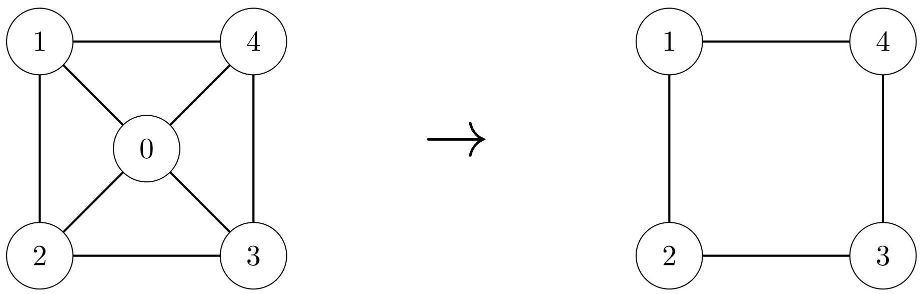

Figure 3.7: Deletion of the vertex 0

Edges can not only be deleted, but can also be contracted. Let $G = (V, E)$ be a graph and $e = (a, b)$ be an edge of G. Hereby, a new vertex v is inserted into G and new edges are inserted such that v is adjacent to all neighbors of a and b. Afterwards the vertices a and b are deleted from the graph. We denote the **contraction** of the edge e in G by G/e (see Figure 3.8). A **claw** is basically one vertex v with 3 edges to 3 different vertices. We

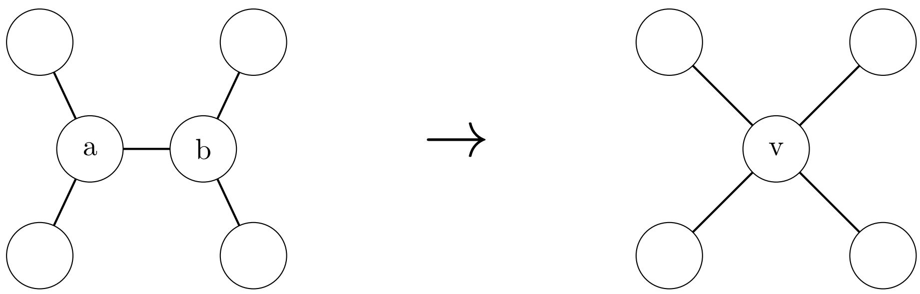

Figure 3.8: Edge contraction of the edge (a, b)

define the **claw contraction** of v as the consecutive edge contraction of all edges which are incident to v. We also call v the center of the claw and denote the graph obtained by the contraction of v in G by G/v. We can generalize this even a step further. Let $G = (V, E)$ be a graph and let $G' = (V', E')$ be a subgraph of G. Then we call the successively contraction of all e of E' in G a **graph contraction**. We denote this by G/G'.

Let G be a graph. The operation of deleting an edge (a, b) and instead inserting a new vertex v together with the edges (a, v) and (v, b) is called a **subdivision on the edge** (a, b). Moreover the graph H which results after a series of subdivisions on various edges of G is called a **subdivision** of G. We will further define graph minors.
A graph G contains a graph H as a **minor** if H can be obtained from G by the deletion of vertices and edges and by the contraction of edges.

Lemma 3.1.2. *Let G and H be graphs. If G is a subdivision of H, then H is a minor of G.*

Proof: Let G and H be a graphs and let G be a subdivision of H. By the definition of a subdivision we know that we can obtain G by a series of subdivisions of edges on H. Therefore if we go the reverse way and start with G, but now contract in each step one of the two edges, which were created by the subdivision process, we see that H can be obtained by a series of edge contractions on G. $\square$

Furthermore, we say that two graphs G and H are **homeomorphic** if one subdivision of G is isomorphic to a subdivision of H.

A **path** is a non-empty graph $P = (V, E)$ of the form

$$V = \{x_0, x_1, ..., x_k\} \quad E = \{x_0x_1, x_1x_2, ..., x_{k-1}x_k\}$$

where the x_i are all distinct.

A graph G is called **connected**, if there is a path between any two vertices of G. A graph $G = (V, E)$ with $|V| > k$ is said to be ***k*−vertex connected** if $G - X$ is connected for every set $X \subseteq V$ with $|X| < k$. In a similar manner a graph $G = (V, E)$ with $|E| > k$ is said to be ***k*−edge connected**, if $G - X$ is connected for every set $X \subseteq E$ with $|X| < k$. The vertex- respectively edge-connectivity is hereby the largest k for which G is still $k-$vertex respectively $k-$edge connected. To get back to our definition of connected graphs via paths we can equivalently say that a graph is $k-$vertex connected if any two of its vertices are joined by k disjoint paths. That these definitions are indeed equivalent can be seen from the following theorem that was first proven by Menger ([26]).

Theorem 3.1.3 (Menger 1927 [9])**.**
Let $G = (V, E)$ be a graph and $A, B \subseteq V$. Then the minimum number of vertices separating A from B in G is equal to the maximum number of disjoint $A - B$ paths in G.

A **cycle** is a connected $2 -$ *regular* graph (see Figure 3.9). The cycle graph on n nodes is denoted by C_n. Moreover the cycle on 3,4,5 nodes is called a triangle, quadrilateral, pentagon.

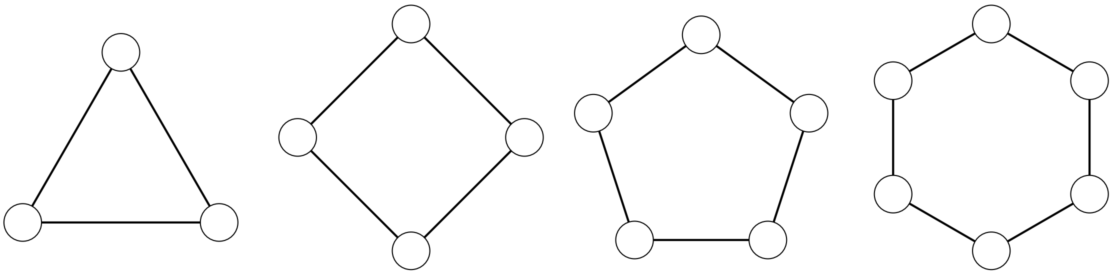

Figure 3.9: Cycle graphs C_3 to C_6

A graph which does not contain any cycles is called a **forest**. Moreover, we call a connected forest a **tree**.

A maximal connected subgraph of a graph G is a called a **component** of G.

Let G be a graph. We say that G is **cyclically $k-$edge-connected**, if at least k edges have to be removed from G to disconnect it into multiple components for which at least two contain cycles.

A **cut-vertex** of a graph G is a vertex whose deletion increases the number of components of G (see Figure 3.10). We can further extend this concept for edges. A **bridge** (or cut-edge) is an edge whose deletion increases the number of components of G (see Figure 3.11). Equivalently, a bridge is an edge that is not contained in any cycle of G. To make it even more general, we call a vertex set $W \subseteq V$ a **vertex cut**, if $G - W$ has more components than G. In a similar manner we call an edge set $F \subseteq E$ an **edge cut**, if $G - F$ has more components than G. A cut with a set of cardinality n is called a **n-cut** (see Figure 3.12). Moreover we call a n-cut of G which divides G into m or more components a **(n,m)-cut**.

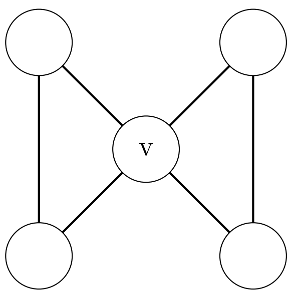

Figure 3.10: Graph with cut-vertex v

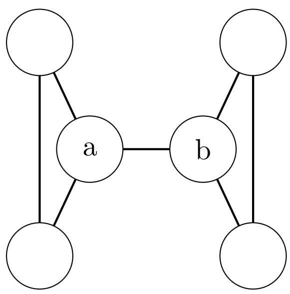

Figure 3.11: Graph with bridge (a, b)

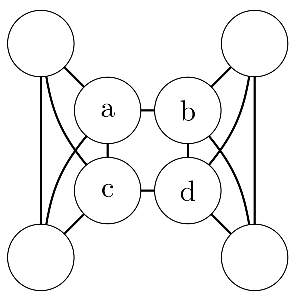

Figure 3.12: Graph with 2-cut $[(a, b), (c, d)]$

A proper **edge coloring** of a graph $G = (V, E)$ is an assignment of colors to the elements of

E such that no two adjacent edges have the same color. If the number of needed colors for such a coloring is minimal then it is a **minimum edge coloring**. The **edge chromatic number** (or chromatic index) of a graph G is hereby the minimum number of colors needed for an minimum edge coloring.

The **wheel graph** W_n is constructed by adding a single vertex to the cycle graph C_{n-1} and connecting all vertices of C_{n-1} to the newly added vertex.

If we look at the wheel graph W_n, we can see that its edge chromatic number is $n-1$. The bottleneck here is clearly the vertex in the middle of the graph (see Figure 3.13). Since this vertex has degree $n-1$, we need at least $n-1$ colors for a proper edge coloring. The edges of the outer cycle can be colored in such a way that we color each edge with the same color as the edge from the opposite vertex to the middle and therefore $n-1$ colors are also sufficient for a proper edge coloring.

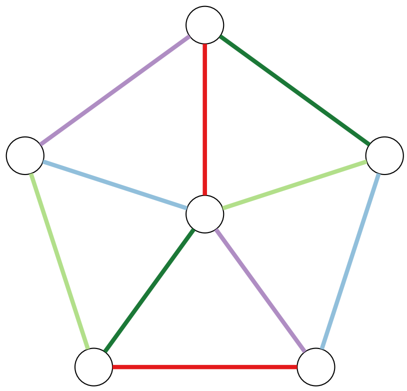

Figure 3.13: Minimum edge coloring of wheel graph W_6

The **girth** of a graph is the length of its shortest cycle.

Now we gathered all the needed definitions to define what a snark is.

A **snark** is a cyclically 4-edge connected, bridgeless, cubic graph with edge chromatic number 4. (see Figure 3.14)

Sometimes there is also the additional requirement that a snark has at least girth 5. Whenever we are not fulfilling this requirement, we will from now on call it a **weak snark**.

Furthermore we have to establish what we mean by the term matching. A set of vertices or edges is **independent** (or stable), if no two of its elements are adjacent. A set M of independent edges in a graph $G = (V, E)$ is called a **matching**. This is equivalent to that a matching M is a subgraph of a graph G and each connected component of M is a K_2. We say a matching M in a graph $G = (V, E)$ is **maximal**, if there is no other independent edge in E.

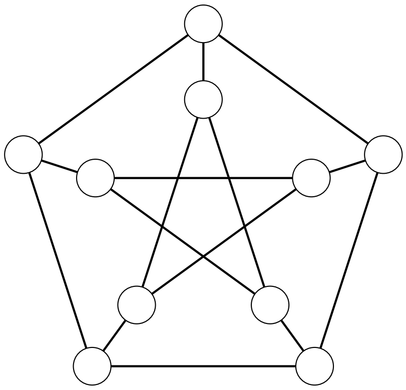

Figure 3.14: Petersen graph, which is the smallest snark.

Moreover we say a matching M in a graph $G = (V, E)$ is **perfect**, if M contains all vertices of G.

Consider the complete bipartite graph $K_{1,3}$. This graph is also called "claw". (see Figure 3.15)

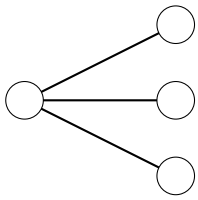

Figure 3.15: $K_{1,3}$, also known as claw.

The following definition is probably the most important and one of the main topics of this thesis.

Let M be a subgraph of a graph G. We say that M is a **pseudo matching** of G if each connected component of M is either a K_2 or a $K_{1,3}$. (see Figure 3.16)

Furthermore we say a pseudo matching $M = \{C_1, C_2, ..., C_m\}$ in a graph $G = (V, E)$ is **perfect**, if each vertex of V is in exactly one component of M.

3.2 Planarity

A **planar** graph is a graph which can be drawn onto the plane without any edges crossing each other. Such a drawing of a graph is called a **planar embedding**. Therefore if we can find such a drawing we know that the graph has to be planar. However, if we find a non planar drawing that does not mean the graph is not planar as Figure 3.17 shows below.

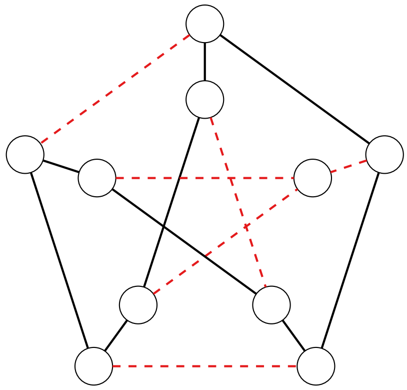

Figure 3.16: A perfect pseudo matching (PPM) for the petersen graph.

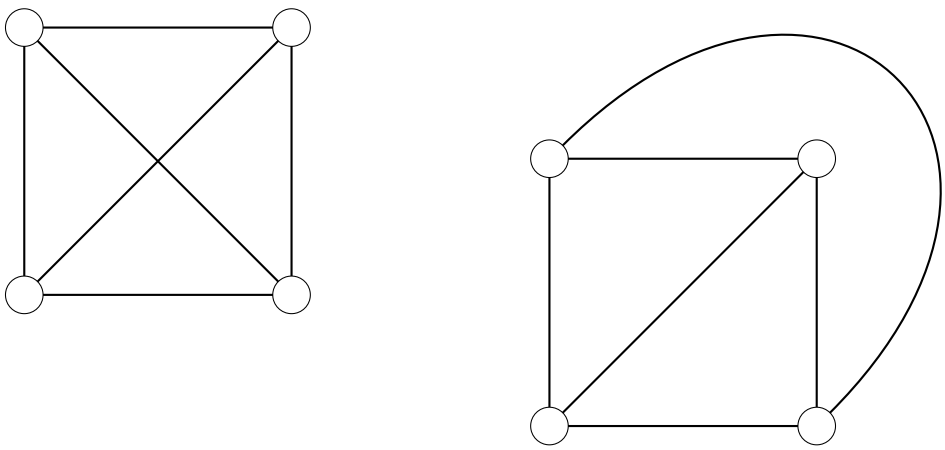

Figure 3.17: Two different embeddings of K_4

The edges of a planar graph divide the plane into regions, which are called **faces**.

We begin now with some observations to see how planarity is preserved under several operations.

Lemma 3.2.1. *Let G be a planar graph. Then every subgraph of G is also planar.*

Proof: Let H be subgraph of G. Since we know that G is planar, we also know that G has a planar embedding P. From P we can now remove all vertices resp. all edges which are not part of H. This results into a plane embedding of H and hence H also has to be planar. □

Lemma 3.2.2. *Let G be a planar graph. Then G/e is also planar for every edge e of G.*

Proof: We look again at the planar embedding P of G. If we contract now the edge $e = (a, b)$ in P then P is still a plane embedding and hence G/e is planar. □

The last two lemmas can be subsumed to the following corollary.

Corollary 3.2.3. *Let G be a planar graph. Then every minor of G is also planar.*

Theorem 3.2.4 (Euler's formula)**.**
Let G be a connected planar graph with n vertices, m edges and f faces. Then

$$n - m + f = 2$$

Proof: We apply induction on m.

The formula is trivially true for the base cases of $m = 0$ or $m = 1$.

Assume that the formula is true for all connected plane graphs having fewer than m edges with $m \geq 2$.

Case 1: Let $m \leq n - 1$. Since G is a connected planar graph, G is a tree and $m = n - 1$. Therefore G has to have a vertex v with degree one. The connected plane graph $G - v$ has $n - 1$ vertices, $m - 1$ edges and f faces, and therefore, by the induction hypothesis it holds that $(n - 1) - (m - 1) + f = 2$. Therefore, it follows that $n - m + f = 2$.

Case 2: Let $m \geq n$. Again since G is a connected planar graph, G cannot be a tree and therefore has to have a cycle. Let e be an edge on this cycle. The connected plane graph $G - e$ has n vertices, $m - 1$ edges and $f - 1$ face, and again by the induction hypothesis it holds that $n - (m - 1) + (f - 1) = 2$. Therefore, Euler's formula holds. □

In a similar manner as for vertices, we define the degree of a face. The number of edges on the boundary of a face f, where bridges are being counted twice, is called **degree** of f and denoted by $d(f)$.

Lemma 3.2.5 (Handshaking lemma for faces)**.** *Let G be a planar graph with m edges. Then*

$$\sum_{f \in F} d(f) = 2m.$$

Proof: The proof works similar as the proof for 3.1.1. Since each edge is incident to two faces (or are bridges), it counts as 2 in the sum of the degrees. Hence, if we do this for all edges, we see that the sum of the degree has to be 2 times the number of edges. □

Corollary 3.2.6. *Let G be a connected planar graph with $n \geq 3$ vertices and m edges. Then*

$$m \leq 3n - 6$$

Proof: From 3.2.5 we know that $2m = \sum_{f \in F} d(v)$ holds.
Moreover $n \geq 3$ and therefore it holds that $d(f) \geq 3$ for all f in F. Hence, we get the inequality chain

$$2m = \sum_{f \in F} d(f) \geq \sum_{f \in F} 3 = 3f$$

Thus $f \leq \frac{2}{3}m$. If we use this together with Euler's formula (3.2.4), we get that

$$n - m + \frac{2}{3}m \geq 2$$

which can finally be rewritten into the claimed

$$m \leq 3n - 6.$$

□

Corollary 3.2.7. *Let G be a connected planar bipartite graph with $n \geq 3$ vertices and m edges. Then*

$$m \leq 2n - 4$$

Proof: We can use similar idea as previously in Corollary (3.2.8). Since our graph now is bipartite, we know that it doesn't contain any triangles (in fact it doesn't contain any cycles of odd length). Therefore, we know that $d(f) \geq 4$ for all f in F. Thus,

$$2m = \sum_{f \in F} d(v) \geq \sum_{f \in F} 4 = 4f$$

which is equal to

$$f \leq \frac{2}{4}m.$$

Together with Euler's formula (3.2.4), we get that

$$n - m + \frac{2}{4}m \geq 2$$

which can finally be rewritten into the claimed

$$m \leq 2n - 4.$$

□

Corollary 3.2.8. *The complete graph K_5 is not planar.*

Proof: Assume that K_5 is planar. From Corollary 3.2.6 we know that a planar graph with $n \geq 3$ vertices and m edges must satisfy $m \leq 3n - 6$ and with K_5 we would therefore get $10 \leq 9$. Hence K_5 cannot be planar. □

Corollary 3.2.9. *The complete graph $K_{3,3}$ is not planar.*

Proof: Assume that $K_{3,3}$ is planar. From Corollary 3.2.7 we know that for a planar bipartite graph with $n \geq 3$ vertices and m edges it must hold that $m \leq 2n - 4$ and with $K_{3,3}$ we would therefore get $9 \leq 8$. Hence $K_{3,3}$ cannot be planar. □

The last two corollaries showed us that it is not that hard to find some non planar graphs, but we would actually be more interested in finding a criteria to test for planarity and not only to test for non-planarity.

Corollary 3.2.10. *Let G be a connected planar graph. Then G contains a vertex of degree of at most 5.*

Proof: Suppose that $G = (V, E)$, with $|E| = m$, does not contain such a vertex. Therefore, $d(v) \geq 6$ for all $v \in V$. Hence, from this and 3.1.1 we know that

$$2m = \sum_{v \in V} d(v) \geq 6n$$

holds. Thus we get that $m \geq$ 3n which is in direct contradiction to 3.2.8 and therefore G has to contain at least one vertex with a degree less than 6. □

We already saw from the Corollaries 3.2.8 and 3.2.9 that K_5 and $K_{3,3}$ are not planar. Therefore we know that every planar graph does not contain a subdivision of K_5 or $K_{3,3}$. What is quite surprising is that the opposite also holds which is stated by the following theorem.

Theorem 3.2.11 (Kuratowski's theorem)**.**
A graph is planar if and only if it does not contain a subdivision of K_5 or $K_{3,3}$.

A proof of Kuratowski's theorem can be found in [9].

Let G be a graph and let H be a subgraph of G. If H is subdivision of K_5 or $K_{3,3}$, then we call H a **Kuratowski subgraph** of G.

Theorem 3.2.12 (Wagner's theorem)**.**
A graph is planar if and only if it contains neither K_5 nor $K_{3,3}$ as minor.

Proof: Suppose that G is a non-planar graph. Then by Theorem 3.2.11 it contains at least a subdivision of K_5 or $K_{3,3}$. This subdivision can be contracted into K_5 resp. $K_{3,3}$. Hence G also contains at least one of them as minor.
Suppose G is a planar graph. Therefore by Corollary 3.2.3 we know that all its minors are also planar. Hence G can not contain K_5 nor $K_{3,3}$ as minor. □

Let $G = (V, E)$ be a graph and let $M = \{C_1, ..., C_n\}$ be a PPM of G. We call the graph which results after we carry out all the contractions of M on G the **contraction graph**

G/M.

Let $M = \{C_1, ..., C_n\}$ be a PPM of the graph G. We say that M is **planarizing** with respect to G if the contraction graph $G_M(G)$ is planar. If the contraction graph does not contain a minor of K_5, we say that M is a $\mathbf{K_5}$ **minor free** PPM with respect to G.

3.3 K_5 Minor Testing

H-Minor Containment is an important problem in many graph theoretic algorithms. The problem can be stated as follows: Given two graphs G, H determine if H is a minor of G. Although this problem sounds quite simple, it is actually quite hard because of the high number of different minors a graph contains. Note that if H is not fixed, the problem if G contains H as a minor is also NP-complete ([25]). Fast algorithms for this problem have been developed for graphs with special properties, like for planar graphs or graphs of bounded branchwidth ([1]).

The following lemmas and their proofs are based on [22]. There Kézdy and McGuiness developed an $\mathcal{O}(n^2)$ algorithm which determines if a given input graph has a K_5 minor. In this part we will take a closer look at this algorithm which still depends on a fast algorithm for planarity testing like [2] (edge addition method), [18] (path addition method) or [8] (Left-Right Planarity Test). It should also be noted that Reed and Li already proposed a linear K_5-minor testing algorithm ([29]). However the implementation of this would go beyond the scope of this diploma thesis, why we settled with a quadratic algorithm.

Let G be a graph containing a $H-$minor with $H = (V, E)$. Each vertex v of H is now associated with a set of vertices of G, called the **branch set** of v. The branch set of v consists hereby of all vertices of G that have been contracted to form the vertex v. For describing a minor it is sufficient to define the branch set of each vertex $v \in V$. We call such a presentation a **model** of the minor H in G.

Let G be a non-planar graph and let K be a Kuratowski subgraph of G. Then we call all vertices of K, which have a degree of at least 3 the **branch vertices** of K. The other nodes of K will be called **path vertices**, since they will lie on paths between our branch vertices. In Figure 3.18 the vertices $1-6$ are branch vertices and $7-9$ are path vertices.

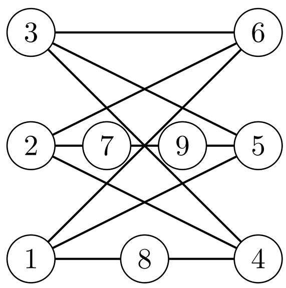

Figure 3.18: Subdivision of $K_{3,3}$

Let $G = (V, E)$ be a connected graph and let $X \subseteq V$ be a vertex cut of G which divides G into the components $G_1, ..., G_n$. The graphs $G_i \cup C(X)$ obtained from $G[V(G_i) \cup X]$ with $i \in \{1, ..., n\}$ by adding a clique on X are called the **augmented components** induced by X.

Theorem 3.3.1.
Let $G = (V, E)$, with $|V| = n$, be a graph with more than 4 vertices. If G has at least $3n-5$ edges, then G contains a K_5 minor.

Proof: We prove this by induction.

Base case $n = 5$: The only graph on 5 vertices with at least $3n - 5$ edges is the complete graph K_5.

Inductive step: $|V(G)| = n$ and $|E(G)| = m \geq 3n - 5$. Let v be an arbitrary vertex of G and let $H = G[N(v)]$ be the graph induced by $N(v)$.

If $\delta(H) \geq 3$ then it is due to Dirac [10] that H has a subgraph which is homeomorphic to K_4. Hence together with v we get a subgraph which is homeomorphic to K_5.

On the other hand if $d(u) < 3$ for a vertex u in H, then we can contract the edge (u, v) and get the graph $G' = G/(u, v)$. On this reduced graph G' we get that $|V(G')| = n - 1$ and $|E(G')| = m' \geq m - 3$. Since the inductive hypothesis holds for G', the minor relation is transitive and G' is a minor of G, we get that G contains a K_5 minor. $\square$

We move on with some observations about minor containment. Let G be a graph. Then G can only contain a K_5 minor if some connected component of G contains it. Therefore, for finding a K_5 minor of G we can simply look at all the connected components of G one after one. Hence, let G be a connected graph. Furthermore, let G contain a cut-vertex x which divides G into the components $G_1, ..., G_n$. Then G will only contain a K_5 minor if one of its augmented components $G_1 \cup \{x\}, ..., G_n \cup \{x\}$ contains a K_5 minor. Now we want to generalize this idea.

Theorem 3.3.2.
Let G be a 2-connected graph and let X be a 2-cut of G. Then G contains a K_5 minor if and only if some augmented component of G induced by X contains a K_5 minor.

This idea can be generalized one step further into the following theorem.

Theorem 3.3.3.
Let G be a 3-connected graph and let X be a (3,3)-cut of G. Then G contains a K_5 minor if and only if some augmented component of G induced by X contains a K_5 minor.

Theorem 3.3.4.
Let G be a 3-connected graph containing a subdivision S of the $K_{3,3}$ with red branch vertices $\{r_1, r_2, r_3\}$ and blue branch vertices $\{b_1, b_2, b_3\}$. Then at least one of the following must hold:

1. *G contains a K_5 minor*
2. *$\{r_1, r_2, r_3\}$ divides G into components such that $\{b_1, b_2, b_3\}$ are in different components*
3. *$\{b_1, b_2, b_3\}$ divides G into components such that $\{r_1, r_2, r_3\}$ are in different components*
4. *G is isomorphic to W, an 8-cycle with cross edges (see Figure 3.19)*

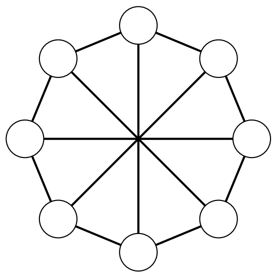

Figure 3.19: W graph

Taking this all together leads to the K_5 minor testing algorithm described in Algorithm 1.

Algorithm 1: K_5 minor containment

```
Input: A graph G with n = |V| vertices and m = |E| edges
Output: Boolean value whether G contains a K
Function has_K5_minor(G)
    if n ≤ 4 then
        return False
    end
    if m ≥ 3n − 5 then
        return True
    end
    if G contains a cut vertex x then
        let C1, .., Cn be the components of G − {x}
        has_minor = False
        for C in {C1, ..., Cn} do
            Gn = G[C ∪ x]
            has_minor = has_minor or has_K5_minor(Gn)
        end
        return has_minor
    end
    if G contains 2-cut X then
        let C1, .., Cn be the components of G − X
        has_minor = False
        for C in {C1, ..., Cn} do
            Gn = G[C ∪ X]
            has_minor = has_minor or has_K5_minor(Gn)
        end
        return has_minor
    end
    if G is planar then
        return False
    else
        let S be the Kuratowski subgraph of G
        if S is a K5 subdivision then
            return True
        else
            if G is isomorphic to W then
                return False
            end
            let R = {r1, r2, r3} be the red branch vertices of S and B = {b1, b2, b3} be the blue branch
              vertices of S
            if b1, b2, b3 lie in pairwise different components of G − R then
                let C1, ..., Ck be the components of G − R
                has_minor = False
                for C in {C1, ..., Ck} do
                    Gn = G[C ∪ R]
                    add a clique on R to Gn
                    has_minor = has_minor or has_K5_minor(Gn)
                end
                return has_minor
            end
            if G − B has not 3 components then
                return True
            end
            if r1, r2, r3 lie in pairwise different components of G − B then
                let C1, ..., Ck be the components of G − B
                has_minor = False
                for C in {C1, ..., Ck} do
                    Gn = G[C ∪ B]
                    add a clique on B to Gn
                    has_minor = has_minor or has_K5_minor(Gn)
                end
                return has_minor
            end
        end
    end
    return True
```

3.4 Integer Linear Programming

The following chapter is based on [7]. Here we will give a very short introduction to integer linear programming and an even shorter introduction to linear programming. Furthermore we give some examples of integer linear programs.

3.4.1 Linear Programming

A **linear program** is a problem of the form

$$\begin{aligned} \text{maximize} \quad & cx \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{3.4.1}$$

where the row vector $c = (c_1, ..., c_n)$, the $m \times n$ matrix $A = (a_{ij})$ and the column vector $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ contain the known input values. The column vector $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ contains the values which are optimized. We call the expression, which is maximized, the **objective function**. The set $P := \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ is the set of feasible solutions.

Simplex Method

The simplex method is one of the most used algorithms for solving LPs. We present here just the key idea of the simplex algorithm. For a detailed description we refer the reader to [27].

Given the LP

$$\begin{aligned} \text{maximize} \quad & cx \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0, \end{aligned} \tag{3.4.2}$$

with the set of feasible solutions $P := \{x \in \mathbb{R}^n_+ : Ax \leq b\}$.

Geometrically we see that the set of all points $x \in R^n$, which fulfill the equation

$$a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_i,$$

defines a hyerplane. Hence the set of all points $x \in R^n$ which fulfill the equation $a_i x \leq b_i$ builds a half-space. Thus each line of the equation system $Ax \leq b$ defines a half-space and the intersection of these is P. Therefore is P a convex polyhedron.

The key idea of the simplex method is now to trace along the edges of P from one corner of P to another with non-decreasing values of the objective function. If the tracing procedure to another corner is not possible anymore then a local optimum is reached. Since our linear program is a convex optimization problem this local optimum is also a global one.

Because of its good average performance in practice, the simplex method is one of the leading algorithms for solving linear programs. Klee and Minty proved in ([24]) that the simplex has an exponential running time as a worst case, but speculate that this bad cases appear rarely in practice. However linear programs can also be solved in polynomial time as Khachiyan proved with the ellipsoid method in ([23]).

3.4.2 Basic Definitions

A **(pure) integer linear program** is a problem of the form

$$\begin{array}{lrl} \text{maximize} & cx & \\ \text{subject to} & Ax & \leq b \\ & x & \geq 0 \ \text{ integral,} \end{array} \tag{3.4.3}$$

where the row vector $c = (c_1, ..., c_n)$, the $m \times n$ matrix $A = (a_{ij})$ and the column vector $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ contain the known input values. The column vector $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ contains the values which are optimized. We say that a $n-$vector x is **integral**, if $x \in \mathbb{Z}^n_+$. The set $S := \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$ of feasible solutions to 3.4.3 is called a **pure integer linear set**.

We will mainly focus on the following generalization.

A **Mixed Integer Linear Program (MILP)** is a problem of the form

$$\begin{array}{lrcl} \text{maximize} & cx + hy & & \\ \text{subject to} & Ax + Gy & \leq & b \\ & x & \geq & 0 \ \text{ integral} \\ & y & \geq & 0 \end{array} \tag{3.4.4}$$

where the row vectors $c = (c_1, ..., c_n)$, $h = (h_1, ..., h_p)$, a $m \times n$ matrix $A = (a_{ij})$, a $m \times p$ matrix $G = (g_{ij})$ and the column vector $b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ contain the known input values. The column vectors $x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ and $y = \begin{pmatrix} y_1 \\ \vdots \\ y_p \end{pmatrix}$ contain the values which are optimized. The set $S := \{(x, y) \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : Ax + Gy \leq b\}$ of feasible solutions to 3.4.4 is called a **mixed integer linear set**.

Let $S \subset \mathbb{Z}^n \times R^p$ be a mixed integer linear set. Then we call a set $P := \{(x, y) \in \mathbb{R}^n \times \mathbb{R}^p : Ax + Gy \leq b\}$ which contains S a **linear relaxation** of S. Moreover, we call the linear program $\max\{cx + hy : (x, y) \in P\}$ the **natural linear programming relaxation** of (3.4.4).

3.4.3 Solving Methods

Example

Given the IP

$$\begin{array}{rrcl} \text{maximize} & x_1 + x_2 & & \\ \text{subject to} & -x_1 + x_2 & \leq & 2 \\ & 4x_1 + x_2 & \leq & 12 \\ & x_1, x_2 & \geq & 0 \\ & x_1, x_2 & & \text{integer} \end{array} \tag{3.4.5}$$

By looking at the natural linear programming relaxation of (3.4.5) we can draw the feasible region of the relaxed problem (see Figure 3.20). We can see that the relaxed problem has the optimal solution of $x_1 = 2$, $x_2 = 4$ with an objective value of 6. Since this solution is also an integer solution, it is also the optimal solution of our original problem.

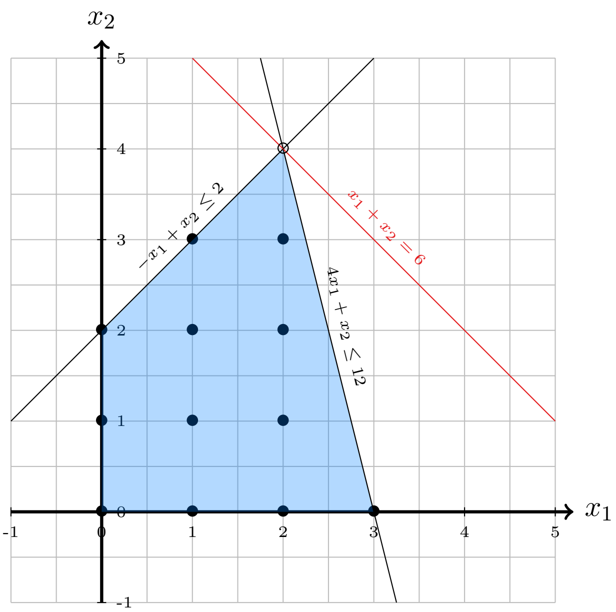

Figure 3.20: Feasible region and solution to IP

The Branch-and-Bound Method

We give here an informal description of the Branch-and-Bound Method. For a formal description we refer the reader to [7].

Given the MILP

$$\max\{cx + hy : (x, y) \in S\}$$

with $S := \{(x,y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^p : Ax + Gy \leq b\}$. Let j be an index of x such that x_j^0 is fractional. Then we can define the sets

$$S_1 := S \cap \{(x,y) : x_j \leq \lfloor x_j^0 \rfloor\}, \qquad S_2 := S \cap \{(x,y) : x_j \geq \lceil x_j^0 \rceil\}$$

where $\lfloor\ \rfloor$ ($\lceil\ \rceil$) denotes the floor (ceiling) function. Now S_1 and S_2 are a partition of S and we can look on the MILPs based on this partition

$$\text{MILP}_1 : \max\{cx + hy : (x,y) \in S_1\}, \qquad \text{MILP}_2 : \max\{cx + hy : (x,y) \in S_2\}.$$

Since S_1 and S_2 are a partition of S we know that an optimal solution of our original problem is the best solution of MILP_1 and MILP_2. Hence we reduced our original problem to two subproblems. We call this process step **branching**.
Let P_1, P_2 be the natural relaxations of S_1, S_2,

$$P_1 := P \cap \{(x,y) : x_j \leq \lfloor x_j^0 \rfloor\}, \qquad P_2 := P \cap \{(x,y) : x_j \geq \lceil x_j^0 \rceil\}$$

and let LP_1, LP_2 be their natural relaxed programs

$$\text{LP}_1 := \max\{cx + hy : (x,y) \in P_1\}, \qquad \text{LP}_2 := \max\{cx + hy : (x,y) \in P_2\}.$$

We can make now the following conclusions

- If one of the linear programs LP_i is infeasible then the corresponding MILP_i is also infeasible since it holds that $S_i \subseteq P_i$. Hence MILP_i does not have to explored any further. We say that this problem is **pruned by infeasibility**.
- Let (x^i, y^i) be an optimal solution of LP_i and let z_i be its objective value. Then we have to consider 3 cases
  1. x^i is an integral vector:
     Then (x^i, y^i) is also an optimal solution of MILP_i and a feasible solution for our original problem. Moreover since we know that $S_i \subseteq S$ it holds that z_i is a lower bound on the objective value of our original problem. We say that this problem is **pruned by integrality**.
  2. x^i is not an integral vector and z_i is smaller or equal to the best already known lower bound on the objective value of our original problem:
     Since $S_i \subseteq S$ it holds that S_i cannot contain a better solution. We say that this problem is **pruned by bound**.
  3. x^i is not an integral vector and z_i is greater than the best already known lower bound on the objective value of our original problem:
     Hence S_i might still contain an optimal solution to our original problem. Now let $x_{j'}^i$ be a fractional component of x^i. Then we can repeat the branching by defining the sets $S_{i_1} := S_i \cap \{(x,y) : x_j \leq \lfloor x_{j'}^i \rfloor\}$ and $S_{i_2} := S_i \cap \{(x,y) : x_j \geq \lceil x_{j'}^i \rceil\}$ and repeat the steps from above.

The Cutting Planes Method ([7])

Given the MILP

$$\max\{cx + hy : (x, y) \in S\}$$

with $S := \{(x, y) \in \mathbb{Z}_+^n \times \mathbb{R}_+^p : Ax + Gy \leq b\}$ and let P_0 be the natural linear relaxation of S. Now let z_0 be the optimal value and (x^0, y^0) an optimal solution of our relaxed problem. We have to consider two cases:

1. If (x^0, y^0) is in S, then it also an optimal solution for our original integer linear program and we are done.
2. If (x^0, y^0) is not in S, then we try to find an inequality $\alpha x + \beta y \leq \gamma$ that is satisfied by every point in S such that $\alpha x^0 + \beta y^0 > \gamma$.

   We call such an inequality $\alpha x + \beta y \leq \gamma$ that is satisfied by every point in S and violated by (x^0, y^0) a **cutting plane** separating (x^0, y^0) from S.

   Now let $\alpha x + \beta y \leq \gamma$ be a cutting plane. Then we define

   $$P_1 := P_0 \cap \{(x, y) : \alpha x + \beta y \leq \gamma\}$$

   We see that now the linear programming relaxation based on P_1 is stronger than the natural linear programming relaxation, in the sense that the optimal solution of

   $$\max\{cx + hy : (x, y) \in P_1\}$$

   is an upper bound for the optimal solution of our original integer linear program, while the optimal solution of the natural linear programming relaxation does not belong to P_1 by definition of P_1.

The recursive application of this procedure is called the **Cutting Planes Method**. The step where a separating cutting plane needs to be found, is called the **separation process**.

Combining the Branch-and-Bound Method with the Cutting Planes Method leads to the **Branch-and-Cut Method**. Here tight upper bounds for the pruning of the enumeration tree are calculated by applying the Cutting Planes Method.

For our purposes we will use a variation of the Branch-and-Cut Method. Here we also allow that the relaxed LP program does not contain all constraints of our MILP, but these constraints are still added if needed. We call this kind of constraints **lazy constraints**, since we gonna add them in a lazy manner. Whenever a lazy constraint is violated in the separation process, we add it to our set of active constraints. Hence, our lazy constraints can also cut off invalid integer solutions which were still valid in the relaxed program.

3.4.4 Examples

We will provide some examples of MILPs which will later help us to tackle our initial problem. First we will look at Maximal Matching Problem.

Maximal Matching

Instance: A graph $G = (V, E)$.
Problem: Find a maximum matching M of G which is maximal regarding cardinality.

Here $e \sim u$ denotes that e is incident on u. Hence we want to find a maximal set of independent edges of a given graph. This problem can be formulated as an integer linear program with binary variables x_e for $e \in E$. Here $x_e = 1$ if and only if e is part of our matching M. Furthermore, we know that each vertex G can be covered by at most one edge of M, which can be modeled by the *degree constraint* $\sum_{e \sim v} x_e \leq 1,\ v \in V$. Now we can formulate our whole problem by

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in E} x_e \\ \text{subject to} \quad & \sum_{e \sim v} x_e \leq 1,\ v \in V \\ & x_e \in \{0,1\}^E. \end{aligned}$$

Figure 3.21 displays here all the possible maximal matchings for the Petersen graph.

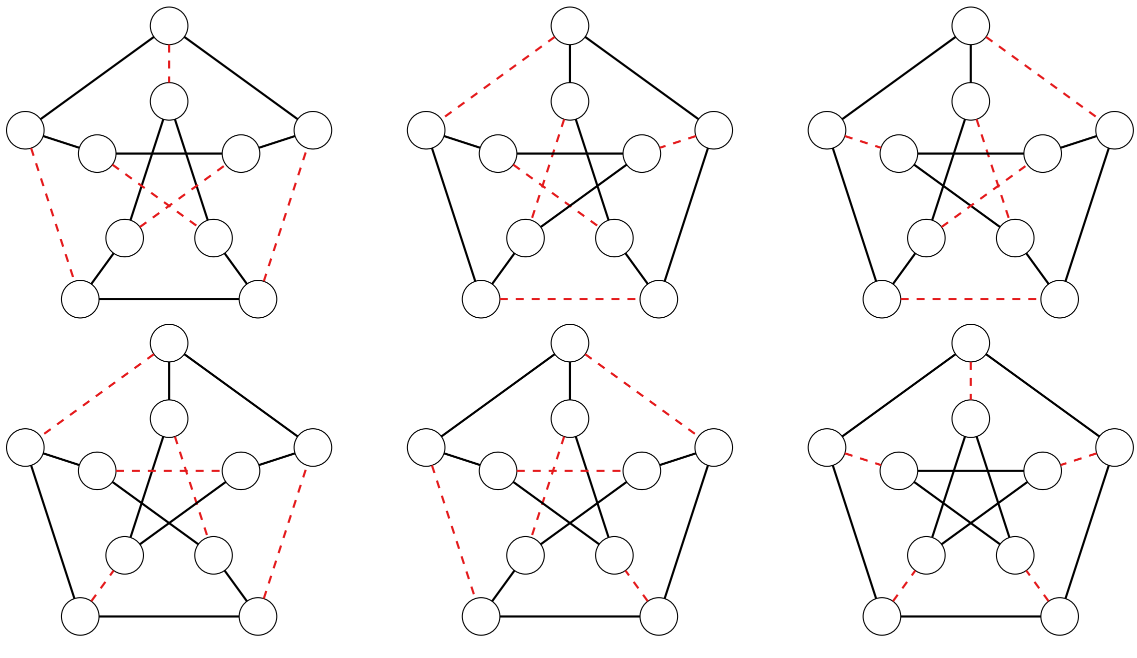

Figure 3.21: All 6 maximal matchings of the Peterson graph.

Now we will look at the Maximal Independent set problem. An **independent set** of a graph is a set of vertices of the graph, where no two vertices in the set are adjacent.

Maximal Independent Set

Instance: A graph $G = (V, E)$.
Problem: Find an independent set I of G which is maximal regarding cardinality. .

This problem can be formulated as an integer linear program with binary variables y_v for $v \in V$. Here $y_v = 1$ if and only if v is part of our independent set I. Furthermore we know that only either a vertex itself or its neighbor can be part of our set, which can be modeled by the *adjacency constraint* $y_u + y_v \leq 1,\ \{u, v\} \in E$. Now we can formulate our whole problem by

$$\begin{aligned} \text{maximize} \quad & \sum_{v \in v} y_v \\ \text{subject to} \quad & y_u + y_v \leq 1,\ \{u, v\} \in E \\ & y \in \{0, 1\}^V. \end{aligned}$$

Figure 3.22 displays one of the maximal independent sets for the Petersen graph.

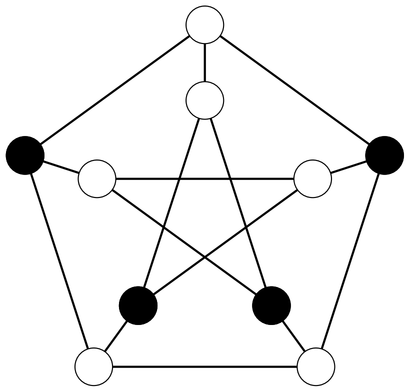

Figure 3.22: A Maximal Independent Set of the Peterson graph.

Maximal Pseudo Matching

Instance: A graph $G = (V, E)$.
Problem: Find a maximum pseudo matching M of G which is maximal regarding the number of vertices it matches.

We can see now that for finding maximal pseudo matchings, we have to maximize the number of covered vertices, where a vertex can either be covered by being part of a K_2 or by being part of a $K_{1,3}$, hence maximizing $\sum_{e \in E} 2x_e + \sum_{v \in V} 4y_v$. Combining now the constraints of our two previous results, we can establish an IP for finding maximal pseudo matchings:

$$\begin{aligned}
\text{maximize} \quad & \sum_{e\in E} 2x_e + \sum_{v\in V} 4y_v \\
\text{subject to} \quad & \Big(y_u + \sum_{e\sim u} x_e\Big) \leq 1,\ \ u \in V \\
& y_a + y_b \leq 1,\ \ \{a,b\} \in E \\
& x \in \{0,1\}^E \\
& y \in \{0,1\}^V.
\end{aligned}$$

4 Perfect Pseudo Matchings

To recall Definition 3.1, a pseudo matching of a graph is a subgraph whose connected components are either a K_2 or a $K_{1,3}$. A pseudo matching is therefore a generalization of a matching (since every matching is a pseudo matching but not vice versa), because not only independent edges are allowed to be part of our matching set, but also claws can be in it. We call a pseudo matching M of a graph G perfect, if each vertex of G is in exactly one component of M. Clearly we can encode a pseudo matching M of a graph $G = (V, E)$ not only as a subgraph of G, but also as a set of claws C and a set of edges EM. For (C, EM) being a pseudo matching it has to hold that $C = \{C_1, C_2, ..., C_n\}$, with $C_i = \{v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$ and $EM = \{EM_1, EM_2, ..., EM_m\}$ with $EM_j = \{v_{j1}, v_{j2}\}$ where $C_i, EM_j \subseteq V\ \forall i, j$. Moreover it has to hold that the edges $(v_{i1}, v_{i2}), (v_{i1}, v_{i3}), (v_{i1}, v_{i4}), (v_{j1}, v_{j1})$ are in $E\ \forall i, j$ and are not adjacent.

4.1 Motivation

The following conjecture is one of the most well-known and studied problems in graph theory. Although its statement is fairly simple, it still remains an open problem as of today.

Conjecture 4.1.1 (Cycle Double Cover (CDC) conjecture)**.** *Every bridgeless graph has a collection of cycles such that every edge of the graph is contained in exactly two of the cycles. (see Figure 4.1)*

The CDC has been proven for many different classes of graphs (see ([31]) for a list). The reason why snarks are of such importance for the CDC is due to [21]. The following lemmas up to the next corollary are from ([21]).

Lemma 4.1.2. *Let G be a minimum counterexample to the CDC regarding the number of edges of G. Then G is 3-edge connected.*

Proof: Suppose that G is a minimum counterexample to the CDC. Furthermore we conclude that G has to be connected because otherwise a component of G would already be a minimum counterexample. If G has an edge cut of size 2, then the graph H, which is obtained by contracting one of these edges, is a bridgeless graph which has fewer edges than G. Hence H has a CDC, but such a CDC can be extended to a CDC of G. Therefore, it holds that G cannot have an edge cut of size 2, which proves that G has to be 3-edge connected. □

This directly implies the following.

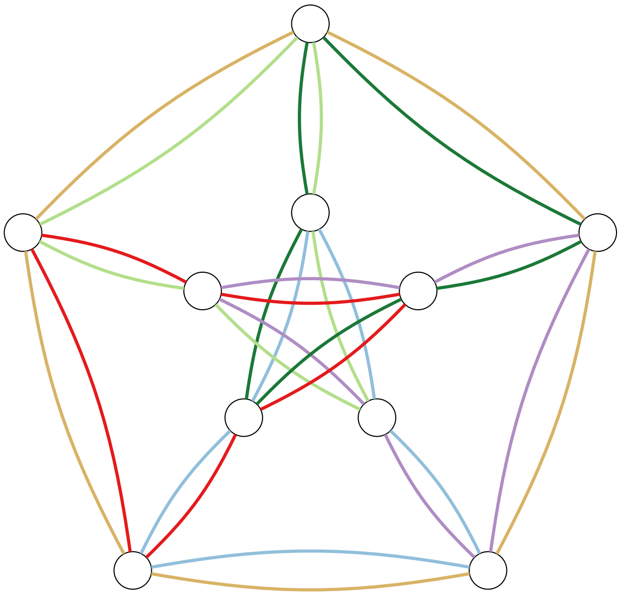

Figure 4.1: CDC of the Petersen graph

Corollary 4.1.3. *Let G be a minimum counterexample to the CDC regarding the number of edges of G. Then G has no vertices of degree smaller than 3.*

We further want to conclude that a minimum counterexample has to be cubic.

Lemma 4.1.4. *Let G be a minimum counterexample to the CDC regarding the number of edges of G. Then G has no vertices of degree greater than 3.*

Proof: Let G have the same properties as in our previous proof. If G has a vertex v with degree at least 4, then it is due to Fleischner ([12]) that one can find two vertices a and b which are adjacent to v and the graph $H := G \backslash \{(a,v) \cup (b,v)\} \cup (a,b)$ is a bridgeless graph with fewer edges than G. Hence H has a CDC, but such a CDC can be extended to a CDC of G. Therefore holds that G has to be cubic. □

Lemma 4.1.5. *Let G be a minimum counterexample to the CDC regarding the number of edges of G. Then G has to be cyclically-4-edge-connected.*

Proof: Now assume that G has an edge cut of size 3 such that the vertices of G can be biparted into two sets of size greater than 1. We can identify this two sets by a single vertex and therefore obtain two cubic bridgeless graphs G' and G'' such that G' and G'' are smaller than G. Therefore, G' and G'' have to have CDCs and we can piece by piece extend such covers to a CDC of G. This is a contradiction to G being a counterexample. □

Lemma 4.1.6. *Let G be a counterexample to the CDC. Then G cannot be 3-edge-colorable.*

Proof: Suppose G has an edge coloring with the colors red, blue and green. We can look at the subgraphs induced by the red and blue edges, by the red and green edges respectively by the blue and green edges. These subgraphs form disjoint cycles since G is a cubic graph and together they form a CDC, hence G cannot be a counterexample. $\square$

Summing up the previous lemmas implies the following theorem.

Theorem 4.1.7.
A minimum counterexample regarding the number of edges to the cycle double cover must be a cyclically 4-edge connected, bridgeless, cubic graph with chromatic index 4. Hence a snark.

This started the search for snarks without a CDC. Since then, the requirements for a minimum counterexample have been tightened as the following theorem shows.

Theorem 4.1.8 ([19])**.**
A minimum counterexample to the CDC must be a snark with girth at least 12.

We will prove now the CDC for two simple types of graphs.

Lemma 4.1.9 ([31])**.** *Let G be a 2-connected planar graph. Then G has a cycle double cover.*

Proof: Since the graph is planar and 2-connected, every face is bounded by a cycle. Therefore if we take the collection of these cycles as our cover, we get a CDC. $\square$

Lemma 4.1.10. *Let G be a 2-regular graph. Then G has a cycle double cover.*

Proof: A 2-regular graph is already a collection of cycles. Hence G is the cycle cover of itself and taking every cycle of G twice gives a double cycle cover of G. $\square$

4.2 Connection to the CDC

In this chapter we establish our connection between PPMs and the CDC. For the following theorems we have to make some further definitions first ([15]).

Let G be an Eulerian graph. Let v be a vertex of G with degree at least 4. A **transition set** of v (denoted by $\mathcal{T}(v)$) is a non-empty subset of a partition into 2-subsets of $E(v)$. A member of $\mathcal{T}(v)$ is called a **transition** at v.

A cycle C is **compatible**, if $|E(C) \cap T| \leq 1$ for every $T \in \mathcal{T}$. A **cycle decomposition** of G is a set of edge-disjoint cycles of G whose union is G. We say that $(G, \mathcal{T})$ has a compatible cycle decomposition (CCD), if G has a cycle decomposition $\mathcal{F}$ such that every member $C \in \mathcal{F}$ is a compatible cycle.

Let $(G, \mathcal{T})$ be a transitioned eulerian graph of order at least 2. We call a cut-vertex v of G a **bad cut-vertex**, if $\{uv, vw\}$ is an edge cut and $\{uv, vw\} \in \mathcal{T}$ for some neighbors

u and w of v. Moreover, we say that $\mathcal{T}$ is **admissible** of G, if $(G, \mathcal{T})$ has no bad cut-vertex.

Let G be a cubic graph, M a PPM of G and let G_M be the contraction graph G/M. Then we can define a transition set on G/M in the following way. Two edges in G/M form a transition if and only if their corresponding edges in the original graph G are adjacent. See Figure 4.2 for a visualization of this idea.

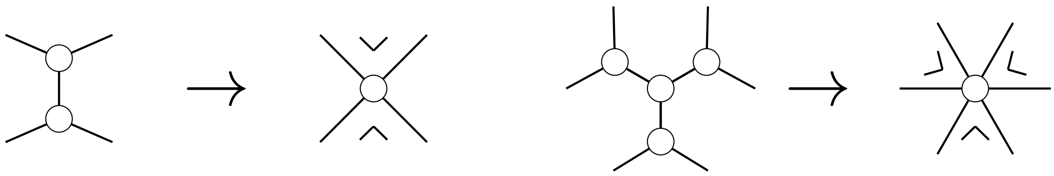

Figure 4.2: Example contraction to form a transition

The following theorem is due to Fleischner and Frank ([14]).

Theorem 4.2.1 ([14]).
Let G be a planar graph. Then for every admissible transition system $\mathcal{T}$ of G, (G,$\mathcal{T}$) has a compatible cycle decomposition.

This result was later generalized by G. Fan and C.-Q. Zhang ([11]).

Theorem 4.2.2 ([11]).
Let G be a K_5 minor free graph. Then for every admissible transition system $\mathcal{T}$ of G, (G,$\mathcal{T}$) has a compatible cycle decomposition.

The following theorem establishes the essential bridge for our chosen approach between a CCD and a CDC via PPMs.

Theorem 4.2.3.
Let G be a cubic graph and let M be a perfect pseudo matching (PPM) of G. If G/M has a CCD, then G has a CDC.

Proof: We take now a closer look at the contraction graph $G_M = G/M$.

For a vertex v of G_M we have to consider two different cases:

Case 1: vertex v was a K_2 in the original graph.
Now we can cover the edge (u, v) with two cycles by connecting the original cycles of u with the original cycles of v. Therefore, our new extended cycle cover covers every edge of G/M once and the edge (u, v) twice. This idea is visualized in Figure 4.3, where the dashed line represents not drawn vertices of a cycle and this extension also works for the symmetric case (see Figure 4.4) where the vertices 3 and 4 are exchanged.

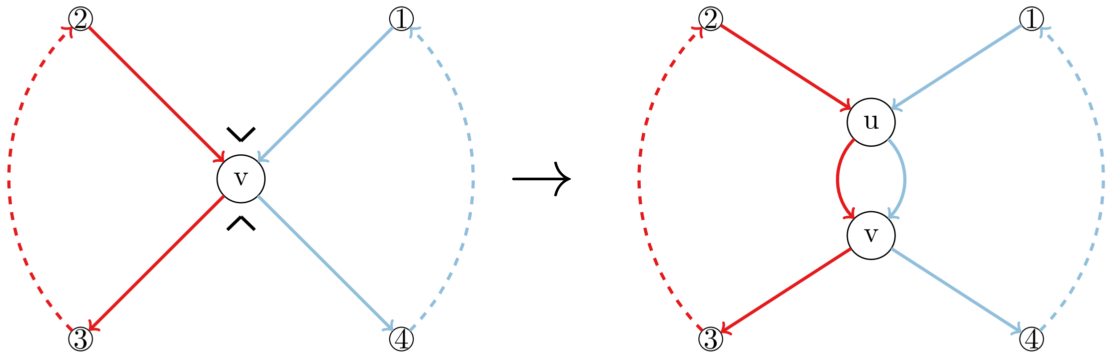

Figure 4.3: Cycle cover extension case 1

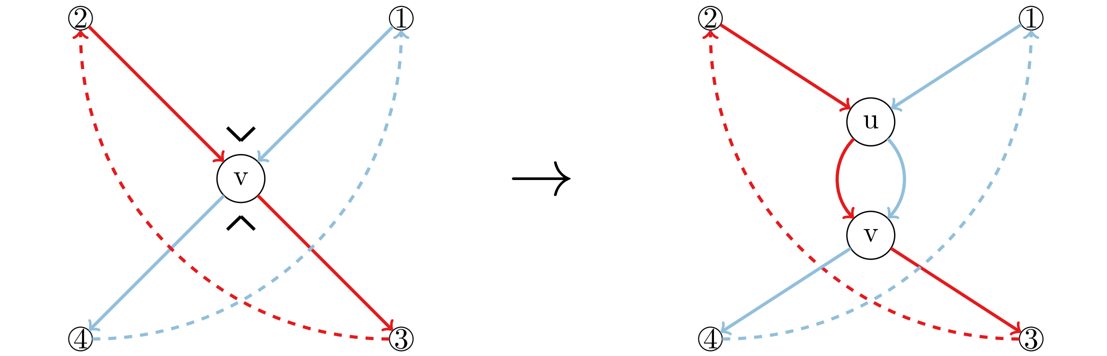

Figure 4.4: Cycle cover extension case 1 for the symmetric case

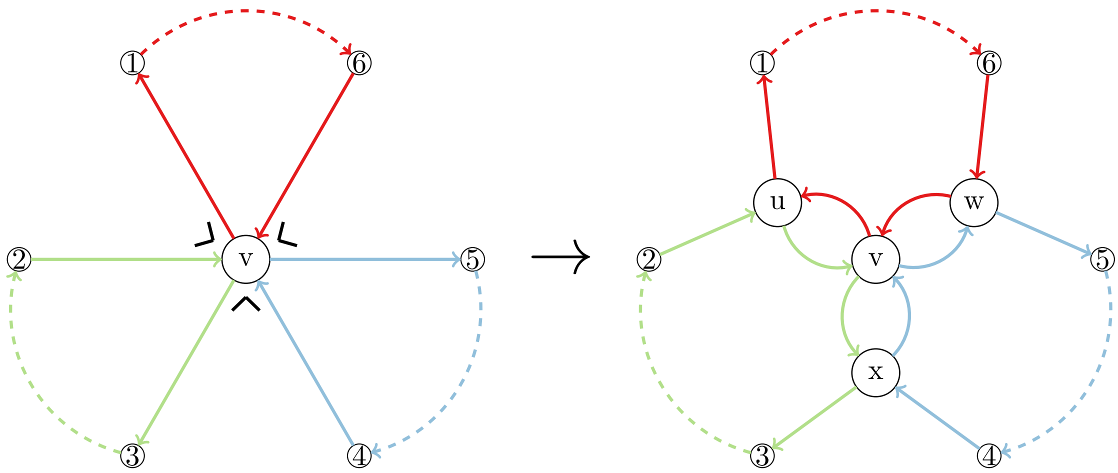

Figure 4.5: Cycle cover extension case 2

Case 2: vertex v was a $K_{1,3}$ in the original graph.

We can use a similar approach as in the first case. We can cover the edges (u, v), (v, w), (v, x) by connecting the original cycles of u with original cycles x and the original cycles of w. Therefore, our new extended cycle cover covers every edge of G/M once and the edges (u, v), (v, w) and (v, x) twice. This again works for symmetric cases and is visualized in Figure 4.5.

Furthermore, we know that $G - M$ is a 2-regular graph since M is a PPM, what means that $G - M$ is a collection of cycles. Hence, if we take an extended cycle cover as in our case distinction together with $G - M$, we end up with a CDC of the original graph G. □

Theorem 4.2.3 lays the foundation for an approach to prove the CDC. Hence, our interest lies in the following two conjectures.

Conjecture 4.2.4 (Weak PPM snark conjecture). *Every snark has a K_5-minor free perfect pseudo matching (K5PPM).*

Conjecture 4.2.5 (Strong PPM snark conjecture). *Every snark has a planarizing perfect pseudo matching (PPPM). (see Figure 4.6)*

One can directly see that Conjecture 4.2.5 would imply Conjecture 4.2.4 since every PPPM is also a K5PPM due to Wagner's theorem (3.2.12).

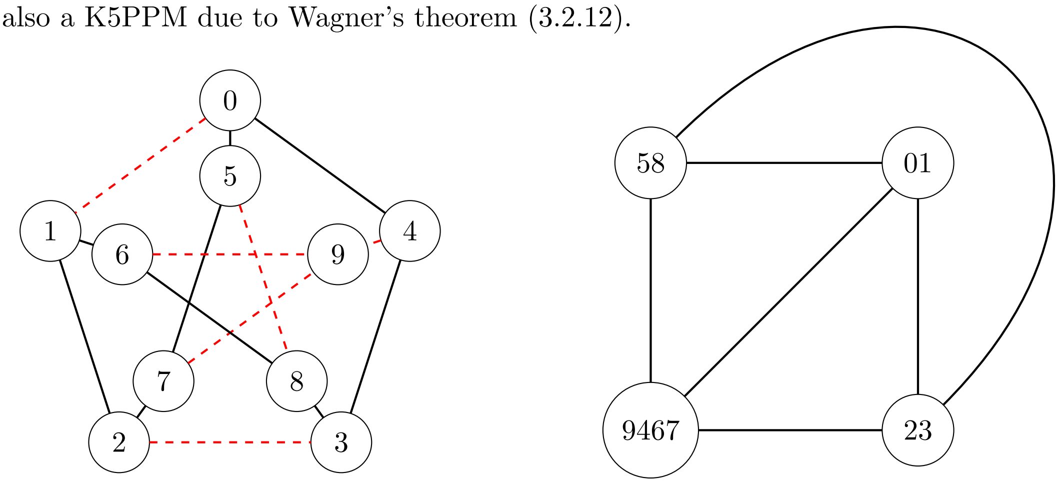

Figure 4.6: A PPPM M of the Petersen graph P and the contraction graph P/M

Flower Snarks

The family of flower snarks is an infinite family of snarks (see Figure 4.7). Flower snarks have been invented by Isaacs in 1975 ([20]). They exist for all odd $n \geq 3$ and can be constructed the following way:

- Build n copies of the star graph on 4 vertices. Hereby the center of the i-th star is denoted by A_i and the outer vertices are denoted by B_i, C_i resp. D_i.
- Add the edges of the cycle $B_1 ... B_n$.

- Add the edges of the cycle $C_1 ... C_n D_1 ... D_n$.

The resulting graph is therefore a cyclically 4-edge connected, bridgeless, cubic graph with edge chromatic number 4 and $4n$ vertices and $6n$ edges. Hence a snark.

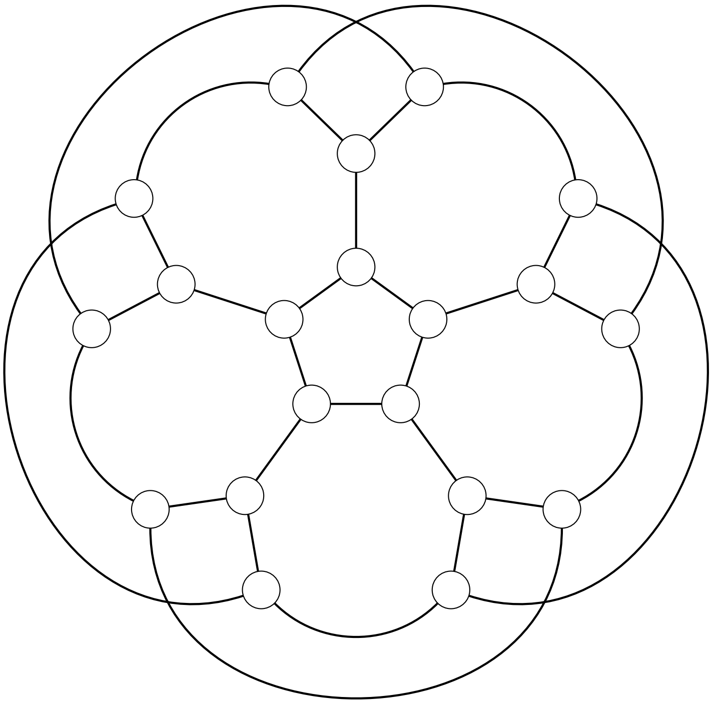

Figure 4.7: Flower snark J_5

Corollary 4.2.6. *Every flower snark has a perfect planarizing pseudo matching.*

Proof: By the way flower snarks are constructed we can clearly see that if we choose $B_1 ... B_n$ as our claws then the contraction graph will be a circle and circles are trivially planar (see Figure 4.8). $\square$

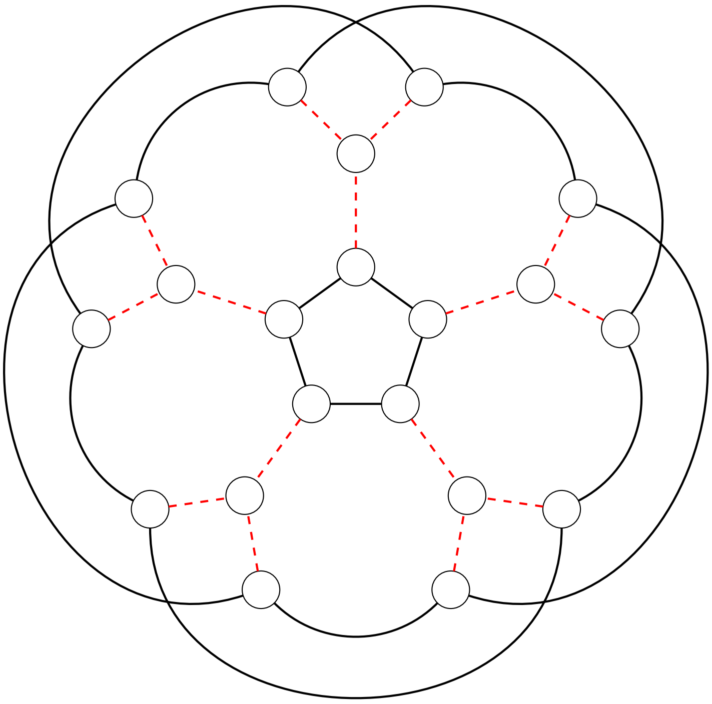

Figure 4.8: PPPM for flower snark J_5

5 Algorithmic Approach

We begin with the formulation of the PPPM and the K5PPM as problems.

Planarizing Perfect Pseudo Matching Problem

Instance: A cubic graph G.
Problem: Does G have a perfect pseudo matching (PPM) M such that G/M is planar?

K_5 **Minor Free Perfect Pseudo Matching Problem**

Instance: A cubic graph G.
Problem: Does G have a perfect pseudo matching (PPM) M such that G/M is K_5 minor free?

5.1 Enumeration

5.1.1 Symmetry Free Enumeration

A method which is feasible for smaller instances of snarks is the enumeration method. Here we simply generate all possible PPMs for a corresponding snark and afterwards check which of the corresponding contracted graphs are planar.

All possible pseudo matchings of a graph can be recursively generated in the way as described in Algorithm 2. To avoid symmetric solutions which represent the same pseudo matching, we can use the property that our pseudo matching has to be perfect. This means that for each vertex of the input graph exactly one of the following must hold:

- The vertex is the center of a claw.
- One of the neighbors of the vertex is the center of a claw.
- The vertex together with one of its neighbors is part of an edge of the matching.

Algorithm 2: Enumerate Perfect Pseudo Matchings

```
Input: A graph G = (V, E), a list of all so far found PPMs, the current pseudo
       matching, list of visited nodes and the current node v
Output: A list of all PPMs of the input graph
Function generatePPM(G, allPPM, currentPM, visited, v)
    while v is not next of last node of V and v is labeled as visited do
        set v to next v
    end
    if v is next of last node of V then
        add currentPM to allPPM
        Return
    end
    label v as visited
    if all neighbors of v are labeled as not visited then
        label all neighbors of v as visited
        add the claw with center v to currentPM
        generatePPM(G, allPPM, currentPM, visited, next v)
        remove the claw with center v from currentPM
        label all neighbors of v as not visited
    end
    for i in neighbors of v do
        if i is not labeled as visited then
            label i as visited
            if all neighbors of i except v are labeled as not visited then
                label all neighbors of i as visited
                add the claw with center i to currentPM
                generatePPM(G, allPPM, currentPM, visited, next v)
                remove the claw with center i from currentPM
                label all neighbors of i except v as not visited
            end
            add the edge (v,i) to currentPM
            generatePPM(G, allPPM, currentPM, visited, next v)
            remove the edge (v,i) from currentPM
            label i as not visited
        end
    end
    label v as not visited
```

For a recursive version of this algorithm we end up with the following recurrence relation for an upper bound of the number of recursion steps

$$T(n) \leq 4T(n-4) + 3T(n-2) \quad \forall n \geq 4$$

where n is the number of vertices in G.

Theorem 5.1.1.
If $f(n) \leq 4f(n-4) + 3f(n-2)$ for $n \geq 4$ and $f(n) \geq 0$, then $f(n) \leq c \cdot 2^n$ with $c := \max(f(0), \frac{1}{2}f(1), \frac{1}{4}f(2), \frac{1}{8}f(3))$.

Proof: **Base Case:**

$$\begin{array}{rcl} f(0) & \leq & c \\ f(1) & \leq & c \cdot 2^1 \\ f(2) & \leq & c \cdot 2^2 \\ f(3) & \leq & c \cdot 2^3 \end{array}$$

This is true for our chosen c.

Recursive Case:

$$\begin{array}{rcl} f(n) & \leq & 4f(n-4) + 3f(n-2) \\ & \leq & 4 \cdot c \cdot 2^{n-4} + 3 \cdot c \cdot 2^{2n-2} \\ & = & c \cdot (1+3)2^{n-2} \\ & = & c \cdot 2^n \end{array}$$

□

Hence we see that our enumeration approach is bounded by exponential running time. This is also due to the fact that the number of PPMs is increasing exponentially as we will see in Chapter 6. Moreover, we notice that for a graph the more claws we have in our PPM the fewer number of vertices will our contraction graph have. This is favorable since every edge contraction might raise the chance of the graph to be planar.

5.2 Integer Linear Programming

5.2.1 Naive IP

Next to the enumeration approach we also developed an integer linear programming approach. From 3.4.4 we already know a possible formulation for a maximal pseudo matching. However, since we not only want a maximal but also a perfect pseudo matching (PPM) we can turn the objective function into a constraint:

$$\sum_{e \in E} 2x_e + \sum_{v \in V} 4y_v = |V|.$$

Now we know that every vertex of our input graph will get covered and therefore we will actually receive not only a maximal pseudo matching but even a PPM as result.

Putting this to together leads us to the following IP:

$$
\begin{aligned}
\text{maximize} \quad & 0 \\
\text{subject to} \quad & \sum_{e \in E} 2x_e + \sum_{v \in V} 4y_v = |V| \\
& \Big(y_u + \sum_{e \sim u} x_e\Big) \leq 1, \ u \in V \\
& y_a + y_b \leq 1, \ \{a, b\} \in E \\
& x \in \{0,1\}^E \\
& y \in \{0,1\}^V
\end{aligned}
$$

However, this problem can be further rewritten and specified if we think about the fact that either the vertex itself, one of its neighbors or an edge which is incident to this vertex has to be part of our pseudo matching. This can be expressed by the *perfect pseudo matching constraint* $\big(\sum_{v \in N(u)} y_v\big) + y_u + \sum_{e \sim u} x_e = 1$ which is a combination of our previous constraints. Furthermore to reach a PPM the perfect pseudo matching constraint has to hold for all vertices of our graph. Hence we can combine this together to the following IP.

$$
\begin{aligned}
\text{maximize} \quad & 0 \\
\text{subject to} \quad & \big(\sum_{v \in N(u)} y_v\big) + y_u + \sum_{e \sim u} x_e = 1, \ u \in V \\
& x \in \{0,1\}^E \\
& y \in \{0,1\}^V
\end{aligned}
$$

Therefore our IP provides us with solution sets Y and X such that Y contains all the centers of our claws and X contains all the edges of our current matching. Unfortunately this is only a solution for PPM and therefore only a **pseudo solution**. All the following constraints are part of our model, but since we follow a Branch-and-Cut approach we only add them, in a lazy manner, if they are violated.

$$\sum_{v \in V'} y_v + \sum_{e \in E'} x_e \leq |V'| + |E'| - 1 \ \ \forall\, V' \subseteq V, E' \subseteq E \text{ s.t. } (V', E') \text{ is not a PPPM (or K5PPM)}$$

Hence in the separation process, after receiving a pseudo solution we simply test if the contraction graph of the current solution is planar respective K_5 minor free. If it is, we return the current solution. Otherwise we add our lazy constraint. It should also be noted that our separation process is only executed on integer solutions in contrary to standard Branch-and-Cut approaches. This is due to the fact that we need an integer solution to create the contraction graph and check its planarity status.

In practice, our integer linear programming approach so far, is just sugar-coating of an enumeration approach until a PPPM is found. The advantage over an enumeration approach here lies in the fact that we do not have to write a generator for PPMs. However the big disadvantage is clearly the running time since an IP has to be solved in every step and the generation of all PPMs of a given graph is rather fast for small graphs.

5.2.2 Pursuit of Smart Cuts

Right now our IP is already able to produce valid solutions, but is actually just imitating our enumeration approach. Therefore we want to further improve our lazy constraints. Hence, instead of only cutting off the current pseudo solution we want to cut off all pseudo solutions, where the contraction graph also contains the same Kuratowski subgraph as the contraction graph of the current solution.

Let G_M be the contraction graph of our current pseudo solution and let K be a Kuratowski subgraph of G_M.

With $c(u)$ we denote the vertex of G_M to which u is mapped by the contraction of G/M for a vertex u of G. Moreover we denote the branch set of v with $c^{-1}(v)$ for a vertex v of G_M.

For a vertex v of K we will further define the **branch component set** of v denoted by $bc(v)$. For a branch vertex x, its branch component set is the set of the vertex itself ($bc(x) := \{x\}$). Since we know that a path vertex lies on a path between two branch vertices, the branch component set of a path vertex are these two branch vertices.

In Figure 5.1 the vertices $1-5$ are branch vertices and $6-10$ are path vertices. In Table 5.1 we can see the branch component set of each vertex.

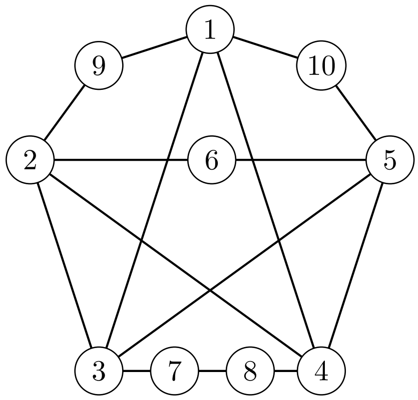

Figure 5.1: Branch component set example

Now we can define our relation $\approx$. We say that $u \approx v$ iff $bc(c(u)) \not\subseteq bc(c(v))$ and $bc(c(v)) \not\subseteq bc(c(u))$. We will say that u and v have **completely different branch component sets** if $u \approx v$.

Regarding our example of branch component sets we see in Figure 5.1 or Table 5.1 that

Vertex	Branch Vertex	Path Vertex	Branch component set
1	×		{1}
2	×		{2}
3	×		{3}
4	×		{4}
5	×		{5}
6		×	{2,5}
7		×	{3,4}
8		×	{3,4}
9		×	{1,2}
10		×	{1,5}

Table 5.1: Branch component set example

the vertex 1 is in relation to $\approx$ with 2,3,4,5,6,7 and 8 but not to 9 or 10.

Together with the relation $\approx$ we can now formulate our constraints for smart cuts in the following way:

$$\sum_{\substack{(u,v)\in E(G)\\ c(u),c(v)\in V(K)\\ u\approx v \ in \ G_M(G)}} (x_{(u,v)} + y_u + y_v) + \sum_{\substack{w\in V(G)\\ u\in N_G(w)\\ v\in N_G(w)\\ u\neq v\\ c(u),c(v)\in V(K)\\ u\approx v \ in \ G_M(G)}} y_w \geq 1 \tag{5.2.1}$$

$\forall\ K, M$ s.t. M is a PPM of G and K is a Kuratowski subgraph in $G_M(G)$.

Hence these constraints forbid a subdivision of K_5 respectively $K_{3,3}$ in the contraction graph by enforcing that at least two vertices with completely different branch component sets of this subdivision will be contracted together.

Theorem 5.2.1.
The lazy constraints described by Equation 5.2.1 eliminate all non-planarizing PPMs.

Proof: Suppose that the above cuts do not eliminate all non-planarizing PPM. Therefore, a PPM M_0 exists which fulfills the constraint in Equation 5.2.1 and which is not planarizing. Hence the contraction graph G_{M_0} is not planar and must therefore contain a Kuratowski subgraph K_0. Since (X, Y) fulfill the constraint in Equation 5.2.1 for $M = M_0$ and $K = K_0$ we have to consider two cases:

1. $(x_{(u,v)}+y_u+y_v) \geq 1$ for $(u,v) \in E(G)$, $c(u), c(v) \in V(K)$ and $u \approx v$ in $G_M(G)$. Hence one of $x_{(u,v)} = 1$, $y_u = 1$ or $y_v = 1$ must hold. Therefore the edge (u,v) becomes

contracted in M. This implies that $c_{M_0}(u) = c_{M_0}(v)$, which is a contradiction to $u \approx v$ in G_M, since they have the same branch component set. ↯

2. $y_w \geq 1$ for $w \in V(G)$, $u \in N_G(w)$, $v \in N_G(w)$, $u \neq v$, $c(u), c(v) \in V(K)$ and $u \approx v$ in $G_M(G)$. Therefore the vertex w is contracted in M. This implies that $c_{M_0}(u) = c_{M_0}(v)$, which is a contradiction to $u \approx v$ in G_M, since they have the same branch component set. ↯

□

Theorem 5.2.2.
The lazy constraints described by Equation 5.2.1 do not eliminate planarizing PPMs.

Proof: Suppose there exists a planarizing PPM M_1 whose variable encoding (X, Y) does not fulfill at least one constraint in Equation 5.2.1. Let M_2 and K be the matching and the Kuratowski subgraph corresponding to the constraint not satisfied by (X, Y). See Figure 5.2 for a representation of the different graphs occurring in this proof. Now we define

$$U_I := \{u \in V(G) : c_{M_2}(u) \in V(K), bc_{M_2}(c_{M_2}(u)) = I\} = c_{M_2}^{-1}(bc_{M_2}^{-1}(I)), \quad I \subseteq V(K)$$

$$W_I := \{w \in V(G_{M_1}) : c_{M_1}^{-1}(w) \cap U_I \neq \emptyset\}, \quad I \subseteq V(K)$$

Note that the sets U_I are pairwise disjoint, if $I \not\subseteq J$ and $J \not\subseteq I$, by definition.

We claim now that $W_I \cap W_J = \emptyset$ if $I \not\subseteq J$ and $J \not\subseteq I$.

Suppose there is a $w \in V(G_{M_1})$ s.t. $w \in W_I$ and $w \in W_J$. By definition this is equivalent to $c_{M_1}^{-1}(w) \cap U_I \neq \emptyset$ and $c_{M_1}^{-1}(w) \cap U_J \neq \emptyset$. Remember that U_I and U_J are disjoint for our I and J. This would imply that for two different vertices $u_I \in U_I$ and $u_J \in U_J$, $c_{M_1}(u_I) = c_{M_1}(u_J) = w$, hence u_I and u_J are contracted together into w. This would further imply that either $x_{(u_I,u_J)} + y_{u_I} + y_{u_J} > 0$ or $y_{u_k} > 0$ for $u_I, u_J \in V(G)$ and $(u_I, u_J) \in E(G)$ or $u_I, u_J, u_k \in V(G)$, $u_I \in N_G(u_k)$ and $u_J \in N_G(u_k)$ holds, since $I \not\subseteq J$ and $J \not\subseteq I$. Since $u_I \in U_I$ and $u_J \in U_J$ together with $I \not\subseteq J$ and $J \not\subseteq I$, one can see that $bc_{M_2}(c_{M_2}(u_I)) = I$ and $bc_{M_2}(c_{M_2}(u_J)) = J$, which implies that $u_I \approx_{M_2} u_J$. Hence (X, Y) would satisfy the corresponding constraint (of Equation 5.2.1) of M_2 which is a contradiction to our precondition. Thus such a w can not exist.

For each branch vertex b_i of K we know that $bc_{M_2}(b_i) = \{b_i\}$ by the definition of the branch component set and it holds that $bc_{M_2}(b_i) \not\subseteq bc_{M_2}(b_j)$ and $bc_{M_2}(b_j) \not\subseteq bc_{M_2}(b_i)$ for $b_i \neq b_j$. Hence $c_{M_2}^{-1}(b_i) \subseteq U_{\{b_i\}}$ for each branch vertex b_i of K and the sets $W_{\{b_1\}}, W_{\{b_2\}}, ..., W_{\{b_n\}}$ are pairwise disjoint.
Moreover we know that there exist paths out of path vertices between the branch vertices (or they are directly connected) in K, because K is a Kuratowski subgraph. This implies that there exists a path

$$u_{b_i}, u_1, u_2, ..., u_m, u_{b_j}$$

for $u_{b_i} \in c_{M_2}^{-1}(b_i)$ and $u_{b_j} \in c_{M_2}^{-1}(b_j)$ in G with $bc_{M_2}(c_{M_2}(u_k)) = \{b_i, b_j\}$, hence $u_k \in U_{\{b_i, b_j\}}$.

From this we can conclude that there exists a trail

$$c_{M_1}(u_{b_i}), c_{M_1}(u_1), c_{M_1}(u_2), ..., c_{M_1}(u_m), c_{M_1}(u_{b_j}) \tag{5.2.2}$$

in G_{M_1} with $c_{M_1}(u_{b_i}) \in W_{\{b_i\}}$ and $c_{M_1}(u_{b_j}) \in W_{\{b_j\}}$.
Now we know that $u_k \in U_{\{b_i,b_j\}}$ and $c_{M_1}(u_k) \in W_{\{b_i,b_j\}}$ by definition. This implies that for $b_o \in \{b_1, ..., b_n\} \setminus \{b_i, b_j\}$ the set $W_{\{b_o\}}$ is disjoint to $W_{\{b_i,b_j\}}$. Hence $c_{M_1}(u_k) \notin W_{\{b_o\}}$. Furthermore, we see that $c_{M_1}(u_k)$ can not be an element of $W_{\{b_x,b_y\}}$ for $\{b_x, b_y\} \neq \{b_i, b_j\}$. Now for each branch vertex b_i which has a path out of path vertices to another branch vertex b_j (or is directly connected to it) in $V(K)$, we have identified a trail from $W_{\{b_i\}}$ to $W_{\{b_j\}}$ in G_{M_1}, which only includes vertices from $W_{\{b_i\}}$, $W_{\{b_j\}}$ or $W_{\{b_i,b_j\}}$ and does not include vertices from other $W_{\{b_x\}}$. Therefore, there exists a path $P_{\{b_i,b_j\}}$ between $W_{\{b_i\}}$ and $W_{\{b_j\}}$ which consists only of vertices in $W_{\{b_i,b_j\}}$ which are not in any other W_I for $I \neq \{b_i, b_j\}$.

By definition of the branch component set $bc_{M_2}(b_i) = \{b_i\}$, hence $bc^{-1}_{M_2}(\{b_i\}) = \{b_i\}$. This implies that $U_{\{b_i\}} = c^{-1}_{M_2}(b_i)$ which is either a claw or a K_2 and therefore connected. Hence, the $U_{\{b_i\}}$ are connected. Thus, the $W_{\{b_i\}}$ are connected and we already know that the $W_{\{b\}}$ are pairwise disjoint for b being a branch vertex of $V(K)$.

Now for each branch vertex b_i of $V(K)$ we can contract $W_{\{b_i\}}$ to the single vertex b_i and moreover we can contract the paths $P_{\{b_i,b_j\}}$ to edges. Hence we found a minor of G_{M_1}, which is a subdivision of K_5 or $K_{3,3}$. This minor can be contracted to a K_5 or a $K_{3,3}$, which is a contradiction to that M_1 was planarizing in the first place. □

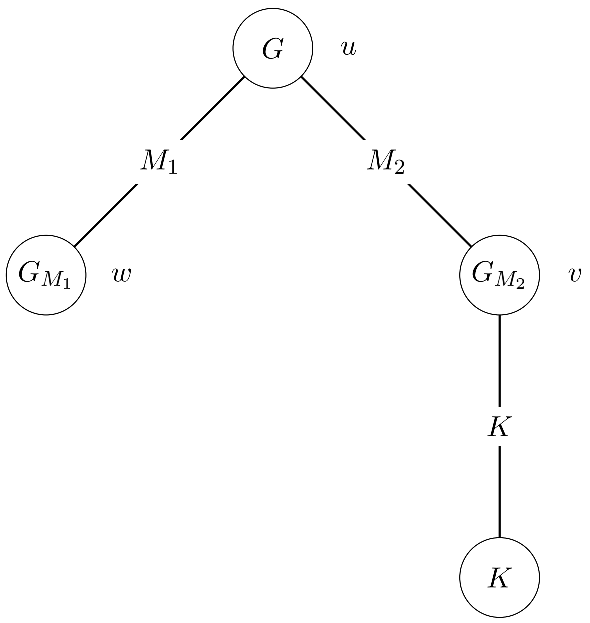

Figure 5.2: Connection between PPMs, original graph and Kuratowski subgraph.

It should be noted that the constraint of Equation 5.2.1 can also be formulated for finding K5PPMs instead of PPPMs. For this we just have to replace the Kuratowski subgraph K in Equation 5.2.1 by a K_5-minor model $K5M$. Accordingly the branch vertices, the branch set and the branch component set work also for a K_5-minor model instead of a Kuratowski subgraph. Furthermore the proofs for our Theorem 5.2.1 and Theorem 5.2.2 can be modified to work with K_5 minors with little work. Hence we can state our constraint of Equation 5.2.1 also for the search of K5PPMs:

$$\sum_{\substack{(u,v)\in E(G)\\ c(u),c(v)\in V(K5M)\\ u\approx v \text{ in } G_M(G)}} (x_{(u,v)} + y_u + y_v) + \sum_{\substack{w\in V(G)\\ u\in N_G(w)\\ v\in N_G(w)\\ u\neq v\\ c(u),c(v)\in V(K5M)\\ u\approx v \text{ in } G_M(G)}} y_w \geq 1 \tag{5.2.3}$$

$$\forall\ K, M \text{ s.t. } M \text{ is a PPM of } G \text{ and } K5M \text{ is a } K_5\text{-minor model of } G_M(G).$$

Theorem 5.2.3.
The lazy constraints described by Equation 5.2.3 eliminate all non K_5-minor free PPMs.

Theorem 5.2.4.
The lazy constraints described by Equation 5.2.3 do not eliminate K_5-minor free PPMs.

Although a quadratic algorithm for finding a K_5-minor model is described in [22], the implementation of such is out of scope for thesis, why we restricted ourself to an implementation of smart cuts for finding PPPMs.

5.2.3 Separation Process

Suppose we got an input graph G and a PPM M. For the separation process we first take a look at the contraction graph G_M. If G_M is planar then is our PPM planarizing and we are done. If G_M is not planar, then the planarity search will output a Kuratowski subgraph. Moreover, we start an empty sum of additions S. For each each edge (u, v) of our original graph G we now take a look at the branch sets of u and v. If it holds that $u \approx v$, then we add $x_{(u,v)}$, y_u and y_v to S. Furthermore for all paths (u, w, v) in our original graph with $u \neq v$, $u \neq w$ and $v \neq w$ we also check if $u \approx v$. If this holds, then we add y_w to S. Now we can add $S \geq 1$ as a new constraint.

Regarding the runtime, we see that the separation process runs in linear time regarding the number of vertices of the input graph. For a separation step, we begin with a planarity test, which runs in linear time. Furthermore, we have to iterate through the edges and paths of length 2, which is also in $\mathcal{O}(n)$ where n is the number of vertices in G.

6 Computational Results

In this chapter we will evaluate and compare the computation times of our different algorithmic approaches.
All tests were run on a single thread of an Intel Xeon E5540 with 2.53 GHz and 3 GB RAM available. The code was written in Python3. The general purpose solver Gurobi Optimizer version 8.0.0 ([16]) was used.

6.1 Test Instances

6.1.1 Snarks

As test instances for our algorithms we used snarks from the website House of Graphs ([3]). Moreover we also allowed weak snarks with girth at least 4 to be part of our testing set. For the snarks of order 30 up to order 36 we only used the first 100 instances as test sample for the benchmarking purposes. For the snarks of order 38 and 40 we used only snarks with girth at least 6. The snarks of higher order than 40 were found through the database search of the House of Graphs site.

6.1.2 Non Snarks

For benchmarking our algorithms with bigger graphs we created some additional test instances. These do not have to be snarks but still fulfill the requirement to be cubic graphs. For the creation we used the NetworkX generator for d-regular graphs. Hence we created cubic random graphs for even n from 4 to 100. These instances will be called random cubic graphs for the rest of the thesis.

6.2 Observational Results

6.2.1 Planarizing Perfect Pseudo Matchings

As previously pointed out, we will take a look at the PPPMs first, because being a PPPM also implies finding a K5PPM.

Observation 6.2.1. *Every weak snark of order 24 or less has at least one PPPM.*

Theorem 6.2.2.
The smallest weak snark without a PPPM has 26 vertices. Moreover there only exist 2 different weak snarks with 26 or less vertices without a PPPM.

# Nodes	# snarks	# w/o PPPM	# w/o K5PPM	% w/o PPPM	% w/o K5PPM
10	1	0	0	0.00	0.00
18	2	0	0	0.00	0.00
20	6	0	0	0.00	0.00
22	20	0	0	0.00	0.00
24	38	0	0	0.00	0.00
26	280	2	0	0.01	0.00
28	2 900	30	14	0.01	0.00

Table 6.1: Snarks without PPPM or K5PPM

Proof: We found PPPMs for all snarks with order 26 or less except for the snarks of order 26 and the indices (according to the line number, starting with 0, of the House of graphs file for snarks of girth at least 4 and order 26) 82 and 124. A graph6 representation of these can be found in the appendix. For these two snarks our approach checked all the according perfect pseudo matchings, but none of their contraction graphs is planar. □

This disproves our strong conjecture (Conjecture 4.2.5). Now we will take a look if at least the weaker form (Conjecture 4.2.4) holds.

6.2.2 K_5 Minor Free Perfect Pseudo Matchings

Observation 6.2.3. *Every weak snark of order 26 or less has at least one K5PPM.*

Theorem 6.2.4.
The smallest weak snark without a K5PPM has 28 vertices. Moreover there only exist 15 different weak snarks with 28 or less vertices without a K5PPM.

Proof: We found K5PPMs for all snarks with order 28 or less except for the snarks of order 28 and the indices (according to the line number, starting with 0, of the House of graphs file for snarks of girth at least 4 and order 28) 1616, 2640, 3465, 3563, 3565, 4445, 4998, 5911, 6751, 6886, 8253, 8889, 8895, 11300 and 12499. A graph6 representation of these can be found in the appendix as well. For these snarks, our approach looked at all the according PPMs, but none of their contraction graphs is K_5 minor free. □

This disproves also our weak conjecture (Conjecture 4.2.4). The results of Theorem 6.2.2 and Theorem 6.2.4 can also be found in Table 6.1 respectively Table 6.2. There, the first column represents the tested class. The second column of the table contains the number of graphs in this class. The columns 3 to 6 contain the number of graphs in the tested class with a PPPM resp. K5PPM resp. their percentages. Regarding the instances of our cubic random graphs, we see in Table 6.3 that at least 32 vertices are needed for the appearances of graphs with K5PPMs which are not PPPMs.
The following observation gives a first impression why we aimed for creating an IP with better lazy constraints.

Snark class	# tested weak snarks	# w/o PPPM	# w/o K5PPM	% w/o PPPM	% w/o K5PPM
10	1	0	0	0.00	0.00
18	2	0	0	0.00	0.00
20	6	0	0	0.00	0.00
22	31	0	0	0.00	0.00
24	155	0	0	0.00	0.00
26	1 297	2	0	0.15	0.00
28	12 517	45	15	0.36	0.12
30	100	2	2	2.00	2.00
32	100	0	0	0.00	0.00
34	100	5	5	5.00	5.00
36	100	2	2	2.00	2.00
38	39	25	25	64.10	64.10
40	25	11	25	44.00	44.00
44	31	3		9.68	
50	2	2		100.00	

Table 6.2: Weak Snarks without PPPM resp. K5PPM

Observation 6.2.5. *The number of PPMs for cubic graphs of a certain order is exponentially growing with higher orders. (see Table 6.4, Table 6.5 and Figure 6.1)*

The first column represents the tested class of graphs and the second column contains the number of graphs in this class. The third column contains the average number of PPMs for graphs of this class.
Hence, we see that our enumeration approach has to execute a high number of planarity tests and that this number is increasing exponentially. This also has a major negative impact on the running time of our enumeration approach. Moreover, we also see that the average number is of PPMs is increasing exponentially not only for snarks but for cubic graphs in general. The difference on the right of Figure 6.1 can be explained due to the small sample size of 1 for snarks with degree 10.

6.3 Benchmark Results

Table 6.6, Table 6.7 and Table 6.8 give an overview over the average running time in seconds as well as over the average number of planarity tests which have to be executed to find a PPPM regarding the chosen approach and the chosen set of instances. The first column represents the tested class. The second column contains the average running time in seconds for graphs of this class. The third column contains the average number of planarity tests which had to be executed during our tests. Together the second and the third column represent the data for our enumeration approach. Column 4 and 5 represent the results for

# Nodes	# tested graphs	# w/o PPPM	# w/o K5PPM	% w/o PPPM	% w/o K5PPM
04	100	0	0	0.00	0.00
06	100	0	0	0.00	0.00
08	100	0	0	0.00	0.00
10	100	0	0	0.00	0.00
12	100	0	0	0.00	0.00
14	100	0	0	0.00	0.00
16	100	0	0	0.00	0.00
18	100	0	0	0.00	0.00
20	100	0	0	0.00	0.00
22	100	0	0	0.00	0.00
24	100	0	0	0.00	0.00
26	100	0	0	0.00	0.00
28	100	3	3	3.00	3.00
30	100	12	12	12.00	12.00
32	100	22	19	22.00	19.00
34	100	47	44	47.00	44.00
36	100	52	50	52.00	50.00
38	100	53	52	53.00	52.00
40	100	75		75.00	
42	100	87		87.00	
44	100	91		91.00	
46	100	95		95.00	
48	100	97		97.00	
50	100	98		98.00	
52	100	100		100.00	
54	100	100		100.00	
56	100	99		99.00	
58	100	100		100.00	
60	100	100		100.00	
62	100	100		100.00	
64	100	100		100.00	
66	100	100		100.00	
68	100	100		100.00	
70	100	100		100.00	
72	100	100		100.00	
74	100	100		100.00	
76	100	100		100.00	
78	100	100		100.00	
80	100	100		100.00	
82	100	100		100.00	
84	100	100		100.00	
86	100	100		100.00	
88	100	100		100.00	
90	100	100		100.00	
92	100	100		100.00	
94	100	100		100.00	
96	100	100		100.00	
98	100	100		100.00	
100	100	100		100.00	

Table 6.3: Random cubic graphs without PPPM resp. K5PPM

# Vertices	# tested snarks	ø PPMs
10	1	13
18	2	221
20	6	353
22	20	590
24	38	990
26	280	1 661
28	2 900	2 782
30	100	4 673
32	100	7 827
34	100	13 179
36	100	21 858
38	38	35 772
40	24	59 112
44	31	176 287
50	2	829 836
56	1	3 822 358

Table 6.4: Average number of PPMs for snarks

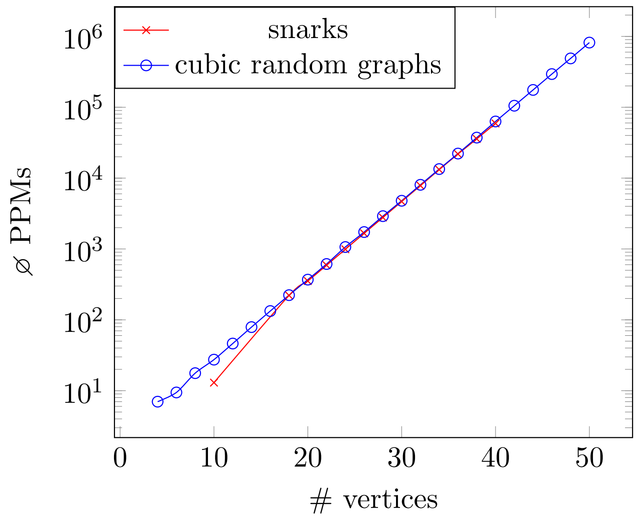

Figure 6.1: Growth of average number of PPMs

our naive IP approach and column 6 and 7 the results for our smart cuts approach.
If the input graph does not have a PPPM, then this number represents the number of planarity tests which have to be executed until to the point where it can be concluded that no PPPM exists. Here we also see that our enumeration approach sometimes (especially for snarks) has to execute more planarity tests than our naive IP approach. This is due to the order in which the set of all PPMs of the input graph is traversed in the search process

# Vertices	# tested graphs	ø PPMs
04	100	7
06	100	9
08	100	18
10	100	27
12	100	46
14	100	79
16	100	133
18	100	223
20	100	369
22	100	613
24	100	1 065
26	100	1 733
28	100	2 906
30	100	4 791
32	100	8 045
34	100	13 421
36	100	22 201
38	100	37 302
40	100	63 000
42	100	105 123
44	100	175 630
46	100	294 055
48	100	489 094
50	100	817 967

Table 6.5: Average number of PPMs for cubic random graphs

of finding a PPPM. For graphs without a PPPM the number of planarity tests for our enumeration and for our naive IP approach coincide of course. Moreover we can see that the number of planarity tests is the lowest for our smart cuts. (see also Figure 6.2 and Figure 6.3)

Regarding the running times of our algorithms we see that the enumeration approach is favorable for smaller graphs. Moreover we can also see in Table 6.6 and Figure 6.4 that our smart cut approach begins to outrun our enumeration for snarks of an order $n \geq 38$. This can be explained due to the tremendous raise on the average number of planarity tests which have to be executed. Table 6.2 also explains this sudden raise. For order 38 nearly $\frac{2}{3}$ of the examined snarks do not have a PPPM (resp. K5PPM). This means that our enumeration approach has to look at all according PPMs to conclude that no PPPM exists. Now we remind the reader of Observation 6.2.5 which states the average number of PPMs is increasing exponentially. Regarding the cubic random graphs, our smart cut approach also starts to outperform our enumeration approach for graphs with 38 vertices or more (see Table 6.7 and Figure 6.5). Table 6.3 explains this due to the fact that from this graph size onwards less than half of all of our tested graphs contain a PPPM. For cubic random graphs of order 40 or higher, this number goes down to $\frac{1}{4}$ and is decreasing rapidly.

For the search of K5PPMs, we can only compare our enumeration approach to our naive IP approach, since the implementation of an algorithm, for finding K_5-minor models and not only reporting that such exist was out of scope for this thesis. However the results from Table 6.9 and Table 6.10 show similar results as for the approaches for finding PPPMs. Here the columns 3 and 5 contain the average number of K_5-minor tests. We see that our enumeration outperforms our naive IP approach and that their running times are increasing exponentially due to the exponential growth of the PPMs. This can also be seen in Figure 6.6 and Figure 6.7.

By comparing our enumeration approaches for PPPMs resp. K5PPMs, hence comparing the average number of planarity tests with the average number of K_5-minor tests, we see that up to 40% fewer tests have to be executed if we only search for a K5PPM instead of a PPPM, because all graphs which are planar are K_5-minor free but not vice versa. On the opposite it has to be mentioned that it is a lot faster to check for planarity than to check if the graph contains a K_5-minor. This is due to the fact that our algorithm for testing if the graph is planar is in $\mathcal{O}(n)$ and our algorithm for testing whether the graph has a K_5-minor is in $\mathcal{O}(n^2)$. Naturally the average number of planarity tests coincides with the average number of K_5-minor tests for graphs without a K5PPM. The complete data regarding the average number of planarity tests resp. K_5-minor tests can be found in Table 6.11. The first column represents the tested class of graphs. The second column contains the average number of planarity tests which had to be executed during our tests. The third column contains the average number of K_5-minor tests. Column 4 to 6 show our results for the classes of cubic random graphs.

Regarding the percentage of graphs without a K5PPM (resp. PPPM), we see that this number is increasing relative to the number of vertices. Whereas our small (regarding the number of vertices) instances all have a K5PPM (resp. PPPM), it becomes rather unlikely to find a cubic random graph which has a K5PPM (resp. PPPM). Furthermore we see in Table 6.2 and Table 6.3, that from 50 vertices onwards for snarks and from 58 vertices onwards for our cubic random graphs, none of our test instances got a PPPM. Unfortunately these instances were already too big to search for a K5PPM with our enumeration approach. However from the flower snarks in Chapter 5 and together with Corollary 4.2.6 we know that snarks with an arbitrary high number of vertex size for $4n$ and $n \in \mathbb{N}_+$ can be found, which also have a PPPM.

# Vertices	Enumeration		Naive IP		Smart Cuts	
	ø time(s)	ø planarity tests	ø time(s)	ø planarity tests	ø time(s)	ø planarity tests
10	0.001	1.000	0.078	1.000	0.077	1.000
18	0.002	1.000	0.007	1.000	0.007	1.000
20	0.016	12.000	0.829	18.167	0.359	8.333
22	0.045	20.065	1.267	23.968	0.539	9.226
24	0.086	48.632	1.604	31.826	0.832	14.987
26	0.171	93.726	3.796	62.035	1.626	25.038
28	0.378	186.242	10.477	130.655	3.381	40.288
30	1.591	463.190	38.330	326.420	8.762	68.120
32	2.645	802.860	72.208	554.710	11.107	85.580
34	5.873	1 799.190	239.592	1 807.680	31.108	184.480
36	7.248	2 260.310	502.242	3 367.250	34.466	201.590
38	92.326	25 692.590	4 048.677	23 775.308	68.613	362.513
40	131.942	35 573.840	5 567.872	31 734.680	89.744	393.360
44	117.348	27 453.774			58.283	248.000
50	4 385.518	829 836.000			419.996	1 567.000

Table 6.6: Comparison of running times of different approaches for snarks regarding PPPMs

# Vertices	Enumeration		Naive IP		Smart Cuts	
	ø time(s)	ø planarity tests	ø time(s)	ø planarity tests	ø time(s)	ø planarity tests
04	0.000	1.000	0.012	1.000	0.013	1.000
06	0.001	1.000	0.001	1.000	0.001	1.000
08	0.001	1.000	0.002	1.000	0.002	1.000
10	0.001	1.000	0.002	1.000	0.003	1.000
12	0.001	1.030	0.003	1.000	0.003	1.000
14	0.001	1.010	0.013	1.290	0.012	1.220
16	0.002	1.110	0.042	1.610	0.043	1.610
18	0.002	1.440	0.051	2.470	0.044	2.150
20	0.003	1.980	0.138	4.780	0.078	2.790
22	0.011	7.830	0.371	10.090	0.223	5.900
24	0.045	29.690	1.987	45.320	0.709	15.500
26	0.197	107.180	8.681	134.880	2.881	46.130
28	0.645	251.000	40.500	349.660	7.908	72.870
30	4.992	1 077.410	168.288	1 104.510	22.750	133.530
32	9.660	2 498.900	336.571	2 667.080	24.713	181.180
34	32.672	7 466.640	1 074.057	7 435.920	51.813	316.020
36	59.164	13 386.200	1 949.662	13 360.050	69.484	393.360
38	128.557	23 441.100	4 130.026	23 062.340	80.400	416.590
40	296.355	49 331.150	10 479.874	51 261.510	115.237	533.550
42	579.059	95 850.420			128.030	624.420
44	1 025.798	162 608.490			146.976	655.720
46	1 973.035	281 526.450			163.019	690.090
48	3 355.749	474 967.630			184.552	735.700
50	6 430.941	806 446.330			190.799	759.910
52	11 146.379	1 349 592.340			206.081	833.470
54	38 176.898	2 308 113.806			209.651	840.680
56					318.664	846.880
58					394.040	888.060
60					437.614	889.610
62					468.450	911.800
64					525.223	926.220
66					575.926	959.250
68					635.845	979.710
70					682.564	995.630

Table 6.7: Comparison of running times of different approaches for cubic random graphs regarding PPPMs

# Vertices	Smart Cuts	
	ø time(s)	ø planarity tests
72	800.673	1 016.490
74	769.025	987.960
76	877.274	1 062.870
78	886.559	1 043.400
80	1 056.926	1 051.390
82	1 104.285	1 093.410
84	1 326.849	1 084.430
86	1 276.483	1 085.490
88	1 323.017	1 063.470
90	1 637.787	1 114.230
92	1 859.897	1 118.900
94	1 824.786	1 106.700
96	2 141.258	1 120.400
98	2 312.978	1 103.490
100	2 404.138	1 099.130

Table 6.8: Comparison of running times of different approaches for bigger cubic random graphs regarding PPPMs

6.4 Used Packages, Libraries

6.4.1 House of Graphs

House of graphs [3] is an online searchable database for graphs. Since the creation of all snarks up to a certain order is computationally expensive, we used the snarks from this site as input for our testing. Moreover whenever we will refer to a snark with a certain girth and a certain index it is meant to be according to the ordering of this database.

6.4.2 Graph6

The graph6 data format was created by Brendan McKay. It is used for storing undirected simple graphs in a compact manner that uses only printable ASCII characters. We will now provide an example to understand this format better.

Suppose we got the graph of Figure 6.8. This graph can also be represented by the following upper triangle adjacency matrix (6.12). We can also represent this matrix by the bit-vector b 0010001000010010001111101001100100 if we traverse the matrix (column wise)

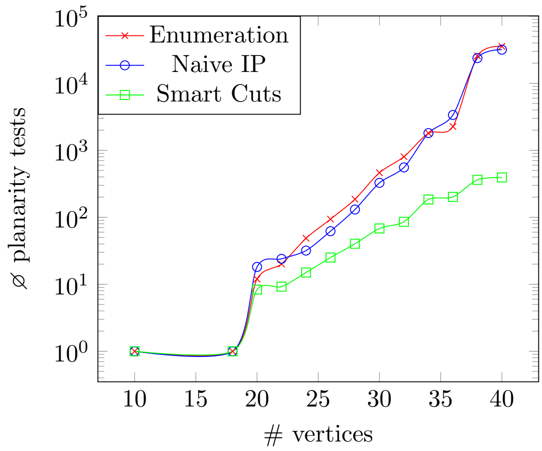

Figure 6.2: Avg. number of planarity tests for snarks

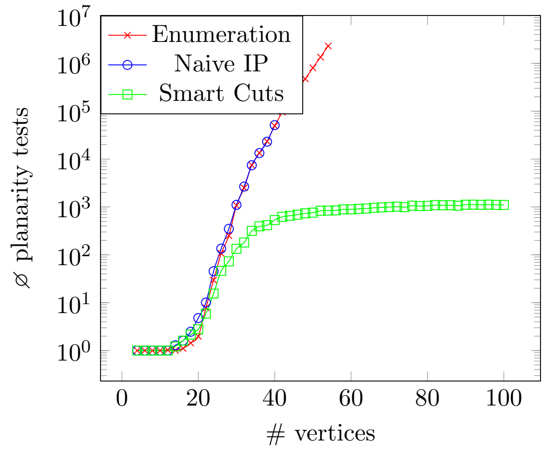

Figure 6.3: Avg. number of planarity tests for cubic random graphs

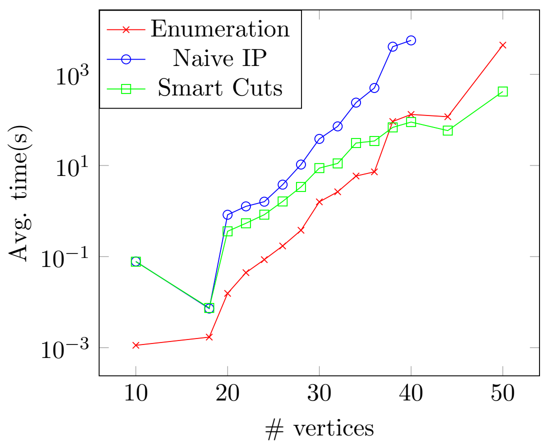

Figure 6.4: Running times for snarks

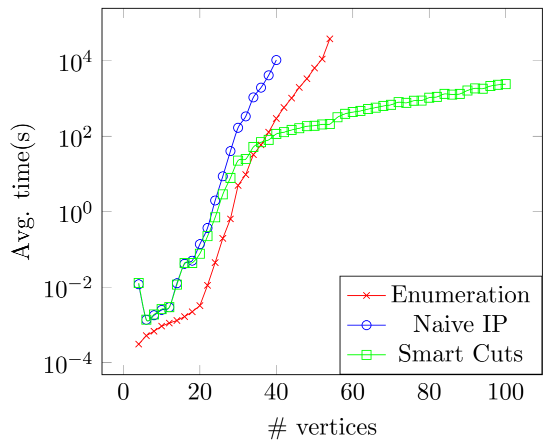

Figure 6.5: Running times for cubic random graphs

# Vertices	Enumeration		Naive IP	
	ø time(s)	ø K_5 minor tests	ø time(s)	ø K_5 minor tests
10	0.000	1.000	0.002	1.000
18	0.007	1.000	0.032	1.000
20	0.298	12.000	1.267	18.167
22	0.422	11.742	1.543	18.516
24	1.665	31.787	2.869	26.232
26	4.754	65.277	7.741	53.019
28	14.253	144.965	26.069	115.776
30	54.553	404.460	98.476	308.950
32	107.496	717.150	213.171	456.790
34	366.931	1 691.140	991.961	1 722.030
36	538.068	2 240.210		
38	26 211.580	25 692.590		
40	38 070.941	35 573.840		

Table 6.9: Comparison of running times of different approaches for snarks regarding K5PPMs

# Vertices	Enumeration		Naive IP	
	ø time(s)	ø K_5 minor tests	ø time(s)	ø K_5 minor tests
04	0.000	1.000	0.002	1.000
06	0.000	1.000	0.001	1.000
08	0.000	1.000	0.001	1.000
10	0.000	1.000	0.002	1.000
12	0.000	1.030	0.003	1.000
14	0.001	1.010	0.016	1.190
16	0.005	1.100	0.051	1.480
18	0.018	1.330	0.071	2.260
20	0.053	1.700	0.181	3.840
22	0.191	5.700	0.587	8.460
24	1.256	23.500	3.431	38.020
26	5.038	71.182	16.711	105.960
28	23.983	238.260	85.170	315.150
30	131.051	1 009.390	362.162	1 080.730
32	348.754	2 163.610	1 239.077	2 506.630
34	1 405.364	7 236.780	4 569.152	7 053.300
36	3 027.770	12 990.470	10 014.808	12 969.570
38	6 455.092	22 830.070		

Table 6.10: Comparison of running times of different approaches for cubic random graphs regarding K5PPMs

Snarks			Cubic random graphs		
# nodes	ø plan. tests	ø K_5 minor tests	# nodes	ø plan. tests	ø K_5 minor tests
10	1.000	1.000	4	1.000	1.000
18	1.000	1.000	6	1.000	1.000
20	12.000	12.000	8	1.000	1.000
22	20.065	11.742	10	1.000	1.000
24	48.632	31.787	12	1.030	1.030
26	93.726	65.277	14	1.010	1.010
28	186.242	144.965	16	1.110	1.100
30	463.190	404.460	18	1.440	1.330
32	802.860	717.150	20	1.980	1.700
34	1 799.190	1 691.140	22	7.830	5.700
36	2 260.310	2 240.210	24	29.690	23.500
38	25 692.590	25 692.590	26	107.180	71.182
40	35 573.840	35 573.840	28	251.000	238.260
44	27 453.774		30	1 077.410	1 009.390
50	829 836.000		32	2 498.900	2 163.610
			34	7 466.640	7 236.780
			36	13 386.200	12 990.470
			38	23 441.100	22 830.070
			40	49 331.150	
			42	95 850.420	
			44	162 608.490	
			46	281 526.450	
			48	474 967.630	
			50	806 446.330	
			52	1 349 592.340	
			54	2 308 113.806	

Table 6.11: Comparison of avg. number of planarity tests with avg. of K_5-minor tests

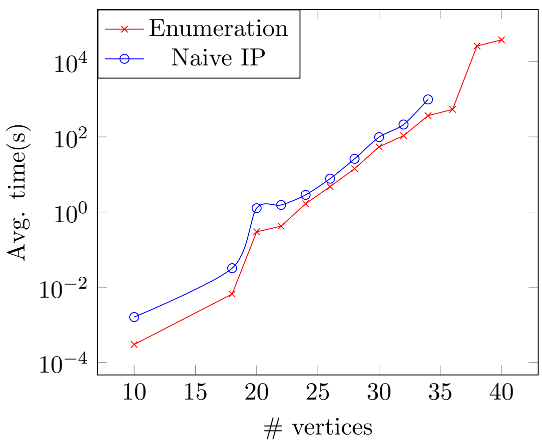

Figure 6.6: K_5 running times for snarks

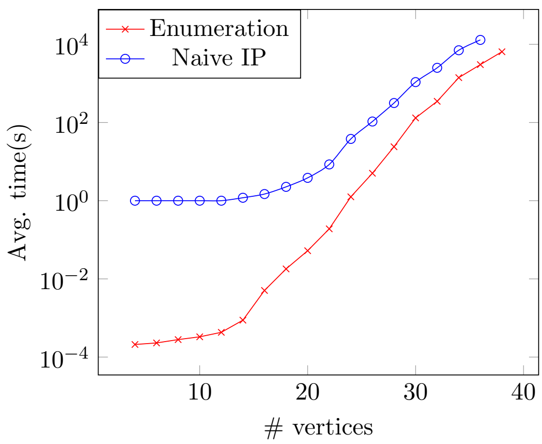

Figure 6.7: K_5 running times for cubic random graphs

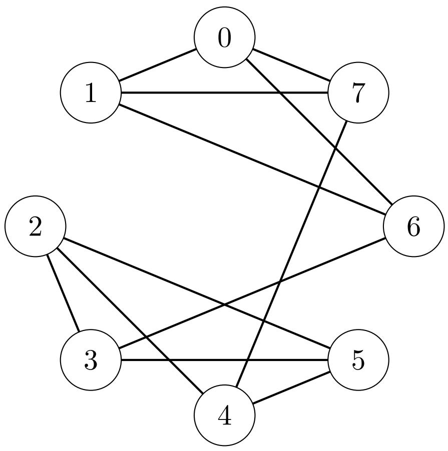

Figure 6.8: Graph6 example graph

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ & 0 & 0 & 0 & 0 & 1 & 1 \\ & & 1 & 1 & 1 & 0 & 0 \\ & & & 0 & 1 & 1 & 0 \\ & & & & 1 & 0 & 1 \\ & & & & & 0 & 0 \\ & & & & & & 0 \end{pmatrix}$$

Table 6.12: Graph6 example adjacency matrix

in the following order (0,1), (0,2), (1,2), (0,3), (1,3), ..., (n-1,n). Now we add a padding on the right side to get binary numbers of bit-length 6. These binary numbers are converted to decimal numbers and an offset of 63 is added to them. This leaves us with the decimal numbers 96 71 125 101 79. Furthermore, we prepend the number of vertices of our graph (also with an offset of 63) which gives us the numbers 71 96 71 125 101 79. Finally encoding this by the ASCII code gives us the string "G‘G}eO" which is the graph6 encoding of our original graph.

6.4.3 NetworkX

For representing our graphs we used NetworkX. NetworkX is a Python package for the representation and manipulations of graphs and networks. Moreover, it offers a lot of convenient functions for testing various graphs properties like in our case the connectivity of a graph resp. a version of the Left-Right Planarity Test ([8]) for testing if a graph is planar.

7 Conclusion

7.1 Summary

At the beginning of this thesis we started with the formulation of the CDC. To recall the problem: Given a bridgeless graph G, does a collection of cycles of G exist, such that every edge of G appears in exactly two of these cycles? We started off with retracing the result by Jaeger ([21]) that a minimum counterexample to the CDC has to be a snark. Hence we can reduce the CDC Conjecture to the class of snarks. Furthermore we elaborated the term of pseudo matchings which are a generalization of matchings. We proceeded by building the essential bridge between a CCD and a CDC via PPMs. To recall one of our main theorems:

Theorem 4.2.3
Let G be a cubic graph and let M be a perfect pseudo matching (PPM) of G. If G/M has a CCD, then G has a CDC.

Thus instead of searching for a CDC of a cubic graph, we can instead just search for a CCD. The thought behind step was that number of vertices in the contraction graph is at most half the number of vertices in original graph. Hence it might be faster to search for a CCD in the reduced graph than for a CDC of the original graph. Moreover we did not try to search for a CCD explicitly but built our approach on the work of Fan and Zhang. In [11] they stated the following theorem:

Theorem 4.2.2:
Let G be a K_5 minor free graph. Then for every admissible transition system $\mathcal{T}$ of G, $(G,\mathcal{T})$ has a compatible cycle decomposition.

Hence instead of searching for a CDC in the original graph G we just had a to find a PPM M of G and check whether G/M was K_5-minor free. Since implementing an algorithm which finds a K_5 minor and also returns a model of it, was out of scope for this thesis, we used the stronger check whether G/M was planar. To exactly solve this problem we designed three different algorithms. The first one is an enumeration approach. The second one is a Branch-and-Cut approach which merely imitates the enumeration and can be seen as intermediate step. The third one is also a Branch-and-Cut approach but here we improved the used cuts. The extended cuts are stronger which lead to the result that we had to execute a lot less planarity tests.

To test our algorithms we implemented our approaches in Python using Gurobi and NetworkX. We used two different classes of graphs as instances. As first class we used snarks, since these are the bottleneck of the CDC as mentioned earlier. As a second class we used cubic random graphs, because Theorem 4.2.3 still applies to this class. Another motivation for the second class was to test whether our integer linear programming approach achieves reasonable running times for instances with more vertices. Afterwards we compared the three programs by their running times and their number of executed planarity tests. Here we saw that for instances with fewer than 34 vertices, an enumeration approach is favorable. This is due the fact that the creation of all PPMs for a small given graph is fast. Additionally we saw that our integer linear programming approach with smart cuts is outrunning an enumeration by far for bigger instances, since the number of PPMs is increasing exponentially with an increasing number of vertices. This performance advantage is hence due to the lower number of planarity tests that have to be executed.

Our designed and implemented approach was able to verify the CDC for graphs up to a size of 26 nodes. Hereby also two conjectures about PPPMs and K5PPMs were stated. The conjectures are:

- Every snark has a planarizing perfect pseudo matching (PPPM).
- Every snark has a K_5-minor free perfect pseudo matching (K5PPM).

It's clear that the first conjecture implies the second one. Both conjectures were refuted by finding snarks (see Appendix) without a PPPM resp. without a K5PPM. This also shows that our new developed approach can, in the current state, not be used to prove the CDC for graphs with more vertices than 26.

7.2 Further Work

Since the extraction of Kuratowski subgraphs is a major part of our Integer Linear Program (IP) its running time might be improved by implementing an algorithm for finding multiple Kuratowski subgraphs at once like in [6]. Moreover, our approach for smart cuts can be extended for finding K5PPMs and not only for PPPMs by implementing an algorithm which finds a K_5-minor model for a given graph or reports that the graph is K_5-minor free.

Appendix

Graph6 Format

Snarks of order 26 without PPPM:

```
Y?gW@eOGGC?A???@__??T_?@??_?A???L??A??AIC?????J?B????a?_
Y?_W@c??G?GB?AO_g??CP_???OC@??G@C_????BH??GAC?@A??G??a?_
```

Snarks of order 28 without K5PPM:

```
[??G?EOG??GB_AO_g_?CPA??@?@???@?Cg???C?OA?C???F_?A?C?@?a_?????W@
[?GQ@eO?GC?AP??BO@@?GB????o?E???[G?????G??@G??@_??????D_A?A????T
[?`Q@?O??C?B_A?C@??A@@?G?BI?A@?GC@???G?K@??A??C?C?C???Bd????C?_@
[?HI?eOOGC?AD??B_@???_????g?@?O?U??A??DH?????A@??C??@?Ac???OA??H
[?hI?E??GCCA@??Bo@?A?OC???w??_`??_?A?O?G?C????HC??A???_a??A????J
[?HO@cO?KCCB????OC?W?_??A?I????aD?@???@G?E????R??a???C?`?O???@?B
[?GY@E???C?B_?O@g_??Ta??P?@?????[?_??C?__?@?C?????C???B_A??_??@D
[?gW?cO?G?CAP??@OC@A?OC??H??@_??K?G??_?I???_P??_??G???B_?W????Q@
[?GQ?eO?GC?B_?O??@??a_???@G?CG?@CGA???AX?G???E@P???G???`_???A??B
[?GQ@EOOGC?AP??BO@@?GB????o?E???[G?????K??????BC??@???G_G?A????R
[??W@C??G?GB_AO_g_?CP_??P?K??@O?C????G?G??C?A?DG??G?G?Aa??A?O?O@
[?H?@c??KC?BA?O?oC?O?D???OAC?@?GCO????PK????_A@?G?G??B?_?O@??@_@
[?GW?AOO?C?BA?O_CG??`?A?P?C?CA??E??G?O?H?G??CA@@??O??o?_O?A???W@
[?hW@aOGGC?A_???_???G?_??AG?OG?@OB?????y?A??@A?O?G????B_?g???@_@
[?GA@eOOGC?AP??BO@@?GB????o?E???[???C??G??@?_?@A??A???Co??C????J
```

List of Figures

List of Tables

List of Algorithms

ACRONYMS

CCD compatible cycle decomposition

CDC Cycle Double Cover

PPM perfect pseudo matching

PPPM planarizing perfect pseudo matching

K5PPM K_5-minor free perfect pseudo matching

LP Linear Program

IP Integer Linear Program

MILP Mixed Integer Linear Program

Bibliography

[1] Adler, I., Dorn, F., Fomin, F. V., Sau, I., and Thilikos, D. M. Faster parameterized algorithms for minor containment. In *Algorithm Theory - SWAT 2010* (Berlin, Heidelberg, 2010), H. Kaplan, Ed., Springer Berlin Heidelberg, pp. 322–333.

[2] Boyer, J. M., and Myrvold, W. J. On the cutting edge: simplified $O(n)$ planarity by edge addition. *J. Graph Algorithms Appl. 8*, 3 (2004), 241–273.

[3] Brinkmann, G., Coolsaet, K., Goedgebeur, J., and Mélot, H. House of graphs: A database of interesting graphs. *Discrete Applied Mathematics 161*, 1 (2013), 311 – 314.

[4] Brinkmann, G., Goedgebeur, J., Hägglund, J., and Markström, K. Generation and properties of snarks. *Journal of Combinatorial Theory, Series B 103*, 4 (2013), 468 – 488.

[5] Carroll, L. *The hunting of the snark*. Henry Altemus Company, 1909.

[6] Chimani, M., Mutzel, P., and Schmidt, J. M. Efficient extraction of multiple kuratowski subdivisions. In *Graph Drawing* (Berlin, Heidelberg, 2008), S.-H. Hong, T. Nishizeki, and W. Quan, Eds., Springer Berlin Heidelberg, pp. 159–170.

[7] Conforti, M., Cornuéjols, G. V., and Zambelli, G. V. *Integer programming.* Graduate texts in mathematics ; 271. Springer, Cham Heidelberg New York Dordrecht London.

[8] de Fraysseix, H., de Mendez, P. O., and Rosenstiehl, P. Trémaux trees and planarity. *Internat. J. Found. Comput. Sci. 17*, 5 (2006), 1017–1029.

[9] Diestel, R. *Graph theory*, fifth edition. ed. Graduate texts in mathematics. Springer, Berlin, 2017.

[10] Dirac, G. A. In abstrakten graphen vorhandene vollständige 4-graphen und ihre unterteilungen. *Mathematische Nachrichten 22*, 1-2 (1960), 61–85.

[11] Fan, G., and Zhang, C.-Q. Circuit decompositions of Eulerian graphs. *J. Combin. Theory Ser. B 78*, 1 (2000), 1–23.

[12] Fleischner, H. Eine gemeinsame Basis für die Theorie der Eulerschen Graphen und den Satz von Petersen. *Monatshefte für Mathematik 81*, 4 (Dec 1976), 267–278.

[13] Fleischner, H. Eulersche Linien und Kreisüberdeckungen, die vorgegebene Durchgänge in den Kanten vermeiden. *Journal of Combinatorial Theory, Series B 29*, 2 (1980), 145 – 167.

[14] Fleischner, H., and Frank, A. On circuit decomposition of planar eulerian graphs. *Journal of Combinatorial Theory, Series B 50*, 2 (1990), 245 – 253.

[15] Fleischner, H., Gh., B. B., Zhang, C.-Q., and Zhang, Z. Compatible cycle decomposition of bad k5-minor-free graphs. *Electronic Notes in Discrete Mathematics 61* (2017), 445 – 449. The European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB'17).

[16] Gurobi Optimization, L. Gurobi optimizer reference manual, 2018.

[17] Holton, D. A., and Sheehan, J. *The Petersen Graph*. Australian Mathematical Society Lecture Series. Cambridge University Press, 1993.

[18] Hopcroft, J., and Tarjan, R. Efficient planarity testing. *J. Assoc. Comput. Mach. 21* (1974), 549–568.

[19] Huck, A. Reducible configurations for the cycle double cover conjecture. *Discrete Applied Mathematics 99*, 1 (2000), 71 – 90.

[20] Isaacs, R. Infinite families of nontrivial trivalent graphs which are not tait colorable. *The American Mathematical Monthly 82*, 3 (1975), 221–239.

[21] Jaeger, F. A survey of the cycle double cover conjecture. In *Annals of Discrete Mathematics (27): Cycles in Graphs*, B. Alspach and C. Godsil, Eds., vol. 115 of *North-Holland Mathematics Studies*. North-Holland, 1985, pp. 1 – 12.

[22] Kézdy, A., and McGuinness, P. Sequential and parallel algorithms to find a K_5 minor. In *Proceedings of the Third Annual ACM-SIAM Symposium on Discrete Algorithms (Orlando, FL, 1992)* (1992), ACM, New York, pp. 345–356.

[23] Khachiyan, L. G. A polynomial algorithm in linear programming. *Yingyong Shuxue yu Jisuan Shuxue*, 2 (1980), 1–3. Translated from the Russian by Ke Gang Hao.

[24] Klee, V., and Minty, G. J. How good is the simplex algorithm? 159–175.

[25] Matoušek, J., and Thomas, R. On the complexity of finding iso- and other morphisms for partial k-trees. *Discrete Mathematics 108*, 1 (1992), 343 – 364.

[26] Menger, K. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae 10*, 1 (1927), 96–115.

[27] Nemhauser, G., and Wolsey, L. *Integer and combinatorial optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., New York, 1999. Reprint of the 1988 original, A Wiley-Interscience Publication.

[28] Nishizeki, T. T., and Chiba, N. N. *Planar graphs : theory and algorithms*. North-Holland mathematics studies ;. North-Holland ; Sole distributors for the U.S.A. and Canada, Elsevier Science Pub. Co., Amsterdam ; New York : New York, N.Y., 1988.

[29] Reed, B., and Li, Z. Optimization and recognition for k5-minor free graphs in linear time. In *LATIN 2008: Theoretical Informatics* (Berlin, Heidelberg, 2008), E. S. Laber, C. Bornstein, L. T. Nogueira, and L. Faria, Eds., Springer Berlin Heidelberg, pp. 206–215.

[30] Tait, P. G. Remarks on the colouring of maps. In *Proc. Roy. Soc. Edinburgh* (1880), vol. 10, pp. 501–503.

[31] Zhang, C.-Q. *Circuit Double Cover of Graphs*. London Mathematical Society Lecture Note Series. Cambridge University Press, 2012.