Masterarbeit

On Planar 3-SAT and its Variants

vorgelegt am
Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Simon Tippenhauer

Matr. 4372805

betreut von

Prof. Dr. Wolfgang Mulzer

2. November 2016

Eigenständigkeitserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, sind als solche gekennzeichnet. Die Zeichnungen oder Abbildungen sind von mir selbst erstellt worden oder mit entsprechenden Quellennachweisen versehen. Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner Prüfungsbehörde eingereicht worden.

Berlin, 2. November 2016

(Simon Tippenhauer)

Contents

List of Figures

1. Introduction

In 1977, David Lichtenstein introduced in his master thesis [Lic77] the concept of planar Boolean formulas. It was not until 1982 that his idea was first published in a journal [Lic82]. The motivation was to have an easier way to prove NP–completeness of properties for general graphs on their planar derivatives. Previously, the common approach was to first prove the NP–completeness for general graphs and then adjust the proof for planar graphs. This mainly involves the construction of complicated crossover boxes where two edges intersect. With Lichtenstein's definition of Planar 3–SAT and its proof of NP–completeness he accomplished a new and often easier way to show NP-completeness for properties of planar graph. It also opens the way for many variants of Planar 3–SAT and their usefulness for the study of computational complexity.

The idea of Planar 3–SAT is to restrict 3–SAT to Boolean formulas with some property that can be easily mapped to planar graphs. For that each Boolean formula ϕ in 3–CNF is associated with a graph. This graph consists of vertices for each variable and for each clause in ϕ. An edge is put between a variable-vertex and a clause-vertex if the clause contains a literal of the variable. Additionally the variable-vertices are connected with edges in a circular order. A formal definition is given in Section 4.1.

Lichtenstein's results are not just valid for Boolean formulas in 3-CNF but also for the generalization to quantified Boolean formulas. In the *Quantified Boolean Formula* problem (QBF) existential and universal quantifiers can be applied to each variable of the formula. Thus 3–SAT is just a special case of QBF where the Boolean formula is in 3–CNF and each variable is existentially quantified. Besides the NP–completeness of Planar 3–SAT Lichtenstein also proved that Planar QBF is PSPACE–complete [Lic82]. Lichtenstein also claims that it should be possible to develop easy NP–completeness proofs for Triangulation Existence

and Minimum Weight Triangulation. The NP–completeness of Minimum Weight Triangulation was shown in 2008 by Mulzer and Rote with a reduction from a variant of Planar 3–SAT, called Planar Positive exactly 1–in–3–SAT (Section 5.2). Already in 1977 the NP–completeness of Triangulation Existence was proved by Lloyd [Llo77]. A detailed comparison between this proof and a new proof with a reduction from Planar 3–SAT by Schulz [Sch06] in 2006 is left as an open problem (Section 9.2).

This thesis covers a wide range of variants of Planar 3–SAT and their various use in many fields of computer science. Based on planar versions of the well known Boolean satisfiability problems 3–SAT, 1–in–3–SAT, and not–all–equal 3-SAT, many restricted variants where developed over the years. During the study of these variants it turned out that the restrictions can be categorized into four basic groups, namely restrictions on

1. the planar Boolean formula,
2. the associated graph,
3. the planar embedding of the associated graph, and
4. the satisfying assignment.

Restrictions on a planar Boolean formula ϕ are for example that each clause of ϕ contains exactly three literals, that each variable appears a bounded number of times, or that ϕ is monotone. The property that the associated graph has no edges between the variable–vertices is an example for a restriction on the associated graph. The planar embedding of the associated graph can be restricted such that it is rectilinear (Section 4.1.1), or vertices are arranged in a special way. Well known problems with restriction on the satisfying assignment are 1–in–3–SAT and not–all–equal 3-SAT. A list of all restriction of each category is given in Appendix B. The goal of all these restrictions is mostly to define a variant of Planar 3–SAT that can be used to prove NP–completeness for other related problems. Another reason is to find the tightest possible restrictions under which a problem remains NP–complete. This can give some insights for the distinction of problems in P and NP [HS78]. A variant of Planar 3–SAT can be a combination of these restrictions because they are not necessarily mutually

exclusive.

This thesis has nine chapters and two appendices, including this introduction. A collection of definitions and terminology that is used is given in Chapter 2. An introduction to the theory of time complexity is the topic of Chapter 3 where some definitions and terminology is added also. With this preliminaries the detailed presentation of Planar 3–SAT and its variants begins. Each of the following chapters is dedicated to planar versions of 3–SAT (Chapter 4) and its main variants, namely 1–in–3–SAT (Chapter 5), and not–all–equal 3–SAT (Chapter 6). The planar version and variants of Monotone 3–SAT are described in the following chapter (Chapter 7). Restricted variants of Planar 3–SAT with bounds on the number of variable appearances are presented in Chapter 8. The proofs in these chapters are either made by the author of this thesis or they are shown according to the proofs of the mentioned authors. The thesis is concluded in Chapter 9, the last chapter, with some remarks and open problems for a preview of further work.

A List of all versions and variants of Planar 3–SAT in this thesis is given in Appendix A. The restrictions of the presented variants of Planar 3–SAT are adduced in Appendix B, together with a categorization of these variants based on their restrictions.

2. Preliminary

In this chapter definitions and terminology are given that are used throughout this thesis.

2.1. Boolean Algebra and Formulas

Boolean algebra is a branch of algebra which is based on the truth values *true* and *false*. The *Boolean values* are mostly represented by 1 and 0 respectively and can be manipulated by *Boolean operations*. The *negation* or *NOT* operation, denoted with $\neg$, performed on a Boolean value yields to the opposite value, i.e. $\neg 1 = 0$ and $\neg 0 = 1$. The *conjunction* or *AND* operation, denoted with $\wedge$, on two Boolean values evaluates to 1 if and only if both values are 1. The *disjunction* of two Boolean values is 1 if and only if either of both values is 1. The disjunction or *OR* operation is denoted with $\vee$. A Boolean expression is a combination of simple statements with Boolean operations.

A Boolean formula consists of variables for Boolean values, Boolean operations and parentheses. The use of parentheses determines the order of evaluation of the Boolean formula. To get more readable formulas with fewer parentheses the order of operations is defined such that negation precedes conjunction which precedes disjunction.

Example 2.1.1 (Boolean formula)**.**

$$\phi = (x_1 \vee x_2 \vee \neg x_2 \wedge x_3) \wedge (x_3 \vee \neg x_4) \vee x_4$$

A Boolean formula is in *conjunctive normal form*, or *CNF*, if it is a conjunction of clauses. A clause is a disjunction of *literals* which can be either a variable (*positive literal*) or a negated variable (*negative literal*). In general a clause

does not contain the same literal, only if it is explicitly mentioned. A common restriction on Boolean formula in CNF is that each clause contains at most k literals. This variant is called *k–CNF*. Sometimes k–CNF is denoted to Boolean formulas with *exactly* k literals per clause. In this case the formulas are explicitly denoted with *exactly k–CNF*.

Example 2.1.2 (Boolean formula in 2-CNF)**.**

$$\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge x_2$$

A clause of a Boolean formula in CNF is *monotone* if all its literals are positive or negative. Furthermore a monotone clause with only negative literals is called *negative*, and *positive* otherwise. Accordingly a *monotone Boolean formula in CNF* has only monotone clauses. A special case is when the formula consists only of positive or negative variables. Then this is called a *positive* respectively *negative Boolean formula*.

An *assignment* for a Boolean formula is a mapping of values to its variables. With such a mapping the formula can be evaluated according to the rules described above. If the formula is satisfied, i.e. evaluates to 1, the assignment is called *satisfying* and otherwise *unsatisfying*.

Example 2.1.3 (Satisfying and unsatisfying assignment of a Boolean formula)**.**

$$\phi = (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge x_2 \tag{2.1}$$

$x_1 \mapsto 1, x_2 \mapsto 1, x_3 \mapsto 1$	$x_1 \mapsto 0, x_2 \mapsto 1, x_3 \mapsto 0$
$\phi = (1 \vee \neg 1) \wedge (\neg 1 \vee 1) \wedge 1$	$\phi = (0 \vee \neg 1) \wedge (\neg 0 \vee 0) \wedge 1$
$\phi = (1 \vee 0) \wedge (0 \vee 1) \wedge 1$	$\phi = (0 \vee 0) \wedge (1 \vee 0) \wedge 1$
$\phi = 1 \wedge 1 \wedge 1$	$\phi = 0 \wedge 1 \wedge 1$
$\phi = 1$	$\phi = 0$

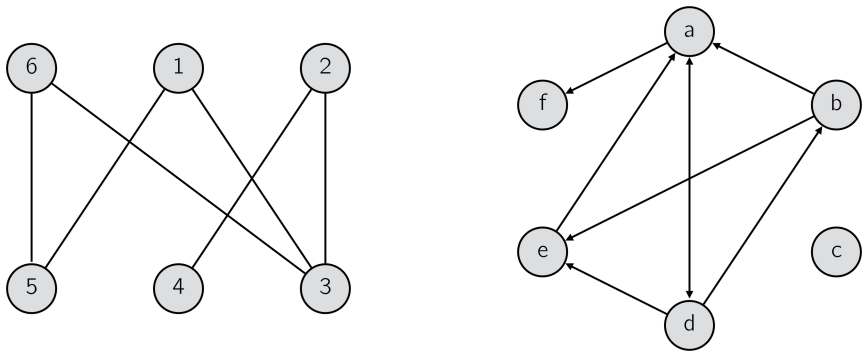
2.2. Graphs

A *graph* is a set of points, called *vertices* or *nodes*, with *edges* connecting some of the points. An edge between some vertices u and v in a graph is represented

by the pair (u, v). In an *undirected* graph the order of u and v is irrelevant, i.e. the pairs (u, v) and (v, u) represent the same edge. In contrast to that the order of u and v is important for a *directed* graph, where the pair (u, v) represents an edge from vertex u to v, which are said to be the *endpoints* of the edge. Between any pair of vertices can be at most one edge. Special variants of graphs where more than one edge between two vertices are allowed are not considered here.

Two vertices that are connected by an edge are called *adjacent*, and two edges that have the same vertex as an endpoint are called *incident*. An edge is also incident to a vertex if the vertex is an endpoint of that edge. The *degree* of a vertex is defined as the number of edges incident to this vertex. The *neighborhood* of a vertex is the set of adjacent vertices. The *maximum degree* and *minimum degree* of a graph are the maximum and minimum degree of its vertices.

If not specified whether a graph is directed or undirected it is assumed that the graph is undirected, which is true for the majority of cases analyzed in this thesis. A graph can be described with a diagram or more formally with a specification of V and E (Figure 2.1).



(a) A possible diagram of the undirected graph G_1.

(b) A possible diagram of the directed graph G_2

Figure 2.1.: Formal specification of an undirected graph $G_1 = (V_1, E_1)$ with $V_1 = \{1, 2, 3, 4, 5, 6\}$ and $E_1 = \{(1, 3), (1, 5), (2, 3), (2, 5), (3, 6), (5, 6)\}$, and a directed graph $G_2 = (V_2, E_2)$ with $V_2 = \{a, b, c, d, e\}$ and edges $E_2 = \{(a, d), (a, f), (b, e), (d, a), (d, b), (d, e), (e, a)\}$.

In a graph a *path* is a sequence of vertices which are connected by edges. A

path is called *simple* if no vertex is repeated. If a path starts and ends with the same vertex it is a *cycle*. A *simple cycle* has at least three vertices and repeats only the first and last vertex. A graph is *connected* if there exists for every pair of vertices a path connecting them. Furthermore a graph is *k–connected* if it is connected, has more than k vertices, and remains connected when less than k vertices are removed from the graph. *Biconnectivity* is equivalent to the property of a graph being 2–connected.

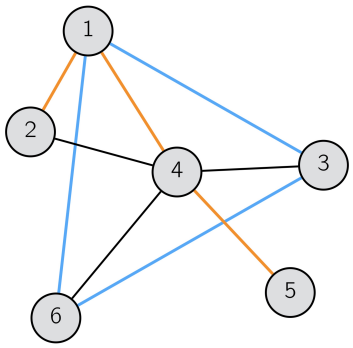


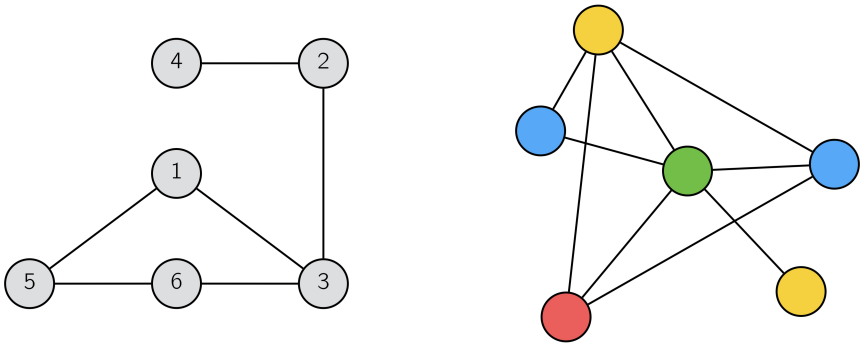
Figure 2.2.: A graph with a path $(2, 1, 4, 5)$ (orange edges) and a cycle $(1, 3, 6, 1)$ (blue edges).

With V the set of vertices and E the set of edges of a graph G, the graph is denoted with $G = (V, E)$. A *plane graph* is a drawing of a graph in the plane such that no edges cross each other. This is also called the *planar embedding of the graph*. A graph that has a planar embedding is called *planar* (Figure 2.3a). The embedding of a graph induces for each vertex a cyclic order of the incident edges. The set of all these cyclic orders is called a *combinatorial embedding* [Dji95; MW00].

A *proper vertex coloring*, or just *coloring*, of a graph is a labeling of the vertices with colors such that no adjacent vertices have the same color. A k–coloring is a coloring of a graph with at most k colors.

A graph $G = (V, E)$ is *bipartite* if it is possible to partition V into two subsets U_1 and U_2 such that each edge of E has one endpoint in U_1 and the other in U_2. In other words, there exist a 2–coloring for G. A *planar bipartite graph* is a graph that is planar and bipartite.

A graph is called a *split graph* if the vertices can be partitioned into a clique



(a) Planar embedding of graph G_1 in Figure 2.1a. Hence G_1 is planar.

(b) A 4–coloring of the graph in Figure 2.2.

Figure 2.3.: Example for (a) a planar graph and (b) a proper vertex coloring of a graph.

and an independent set. A *clique* is a subset of vertices of the graph such that each pair of distinct vertices in the clique are adjacent. An *independent set* is a subset of vertices of the graph such that any two distinct vertices from the independent set are not adjacent, and that every vertex not in the subset are adjacent to at least one vertex of the independent set.

A *chordal graph* is a graph in which every cycle of at least 4 vertices have a chord. A *chord* is an edge between two vertices of the cycle that are not adjacent to each other.

A *tree* is a graph that is connected and is without any cycle, i.e. every pair of nodes is connected by an unique path. It may have a designated *root* node. The nodes with degree 1, except the root, are called *leaves*. A *branch* is a path from the root to a leaf. The *depth* of a node is the number of edges from that node to the root. The *height of a tree* is the number of edges of a longest path from the root to a leaf. A *child* of a node is an adjacent node further away from the root. In a *binary tree* each node has at most two children.

2.3. Languages

A *language* is a set of strings. A *string over an alphabet* is a sequence, or concatenation, of symbols from an alphabet. The *alphabet* over which the string is defined is a non–empty finite set where the members are called *symbols*.

Example 2.3.1. The language L of binary numbers are all possible strings over the alphabet $\Sigma = \{0, 1\}$, e.g. $w = 0110001$ is a string over Σ and an element of L.

If w is a string over an alphabet Σ the length, denoted with $|w|$, is the number of symbols it contains. The so called *empty string* of length zero is written ϵ.

3. An Introduction to the Time Complexity Theory

In contrast to the computability theory, where the main question is which problems can be solved algorithmically in principle, is the motivation of the complexity theory to categorize the problems according to the necessary effort to solve them. In order to get a reasonable categorization some things and terminology have to be specified first.

The focus is mainly on decision problems of the form whether a given string w over an alphabet Σ is contained in a language $L \subseteq \Sigma^*$. For example if the language L is the set of primes encoded as binary numbers over the alphabet $\Sigma = \{0, 1\}$ the decision problem for a number $n \in \Sigma^*$ is whether n is contained in L or not. Algorithmically means that the problem can be solved based on a model of computation. There are many choices for a model of computation, e.g. random–access machine, Lambda calculus, or μ–recursive functions, but most commonly used is the Turing machine. The goal of all these models is to describe an abstract but also accurate model of a general purpose computer. The categorization depends mainly on some resource which is necessary to solve the problems with the chosen model of computation. However, which model is chosen is only of secondary importance as will be explained later. Some typical units of measurement is the consumed time or space (memory), or the use of randomness, non determinism or advice, or the number of designated operations, like the comparison of two numbers.

Further on an introduction to the time complexity of solving problems based on variants of Turing machines is given.

3.1. Turing Machine

The chosen model of computation is the Turing machine, proposed by Alan Turing in 1936. It has an infinite tape used as unlimited memory with a tape head that can read and write symbols while moving around on the tape. At the beginning of a computation the input is written on the tape with the head over the first symbol of the input. There is nothing else on the tape except the input. In one step of computation the Turing machine is in a certain state and reads the current symbol under the tape head. With this information it determines the next state, which symbol is written on the tape, and whether the head moves left or right to the next or previous symbol. These rules are set by the *transition function* $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ where Q is the set of states, Γ the tape alphabet, and L and R indicate the direction of movement for the tape head. The computation continues step by step until the Turing machine reaches a designated halting state for accepting or rejecting the input, or it never halts.

Definition 3.1.1. A Turing machine is a 7–tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$ where Q, Σ, Γ are all finite sets and

- Q is the set of states,
- Σ is the input alphabet,
- Γ is the tape alphabet, where $\Sigma \subset \Gamma$ and $\square \in \Gamma \setminus \Sigma$ is the designated blank symbol,
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function,
- $q_0 \in Q$ is the start state,
- $q_{accept} \in Q$ is the state of acceptance, and
- $q_{reject} \in Q$ is the state of rejection, where $q_{reject} \neq q_{accept}$.

The set of strings that are accepted by a Turing machine M is the *language of* M, denoted with $L(M)$. A Turing machine M *recognizes* a language L if $L(M) = L$, and it *decides* a language if it additionally halts on every input string. In the time complexity theory only decision problems that can be decided by a Turing machine are considered.

Remark. Some properties of Turing machines are of great importance. It is possible to encode every Turing machine M over $\{0, 1\}^*$ such that every string in $\{0, 1\}^*$ encodes a Turing machine, and for every Turing machine exists infinite

many equivalent encodings. The encoding of a Turing machine is denoted with $\langle M \rangle$. There also exists a universal Turing machine that can simulate an arbitrary Turing machine with an arbitrary input string.

Further more, the Church–Turing thesis states that the intuitive notion of algorithms is equal to algorithms for Turing machines. That means that every model of computation with the goal to describe the capabilities of computation is equivalent to Turing machines.

Multi-tape Turing Machine

A *multi-tape Turing machine* is similar to an ordinary Turing machine except that it has several tapes. Each tape has its own head which reads, writes, and moves simultaneously with other tape heads in each step of computation. Furthermore, it is possible that a head does not move in a step at all, indicated by S in the transition function. The transition function of a multi-tape Turing machine has the form $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, S, R\}^k$, where k is the number of tapes.

Nondeterministic Turing Machine

A *nondeterministic Turing machine* is defined like a deterministic Turing machine except that it has two transition functions δ_1 and δ_2. In each step a choice is made whether to use δ_1 or δ_2 for determining the next transition. All possible computations can be seen as a tree with each path from the root to a leave as one computation. A nondeterministic Turing machine accepts the input string if some branch leads to the state of acceptance.

3.2. Time Complexity

The time a Turing machine needs to decide an input string equals the number of steps for the computation. The number of necessary steps may depend on several parameters of the input, e.g. for graph algorithms the number of vertices and edges or the degree of the vertices. To simplify the analysis only the length of the string representing the input is considered.

Definition 3.2.1. Let $L \subseteq \Sigma^*$ be a language and let M be a deterministic Turing machine that decides L. Then the *running time* of M to decide L is the function $T : \mathbb{N} \to \mathbb{N}$, where $T(n)$ is the maximum number of steps necessary to decide any input string of length n.

Definition 3.2.2. Let $L \subseteq \Sigma^*$ be a language and let M be a nondeterministic Turing machine that decides L. Then the *running time* of M to decide L is the function $T : \mathbb{N} \to \mathbb{N}$, where $T(n)$ is the maximum number of steps necessary on any branch of the computation tree to decide any input string of length n.

Let L be a language and let M be a Turing machine that decides L in time $T(n)$. The linear speed-up theorem for Turing machines states that for any constant $c > 0$ there exists a Turing machine that decides L in time $c \cdot T(n) + 2n+2$. It follows that there is no difference between the running time of $c \cdot T(n)$ and $d \cdot T(n)$ for some Turing machines that decide the same language, where $c \neq d$, i.e. constants are of no importance.

This leads to a general definition of time complexity classes based on the running time of deterministic and nondeterministic Turing machines.

Definition 3.2.3. Let $T : \mathbb{N} \to \mathbb{N}$ be a function. The time complexity class *DTIME(T(n))* is the set of all languages that can be decided by a Turing machine with running time at most $c \cdot T(n)$, where $c > 0$ is a constant.

Definition 3.2.4. Let $T : \mathbb{N} \to \mathbb{N}$ be a function. The time complexity class *NTIME(T(n))* is the set of all languages that can be decided by a nondeterministic Turing machine with running time at most $c \cdot T(n)$, where $c > 0$ is a constant.

A *complexity class* groups problems with similar difficulty. In the time complexity theory it will be studied how classes are related to each other, or what consequences result in choosing a deterministic or nondeterministic Turing machine. An important fact is that with more time more problems can be decided as stated in the time hierarchy theorems for deterministic and nondeterministic Turing machines (Theorem 3.2.1 and 3.2.2).

Definition 3.2.5. A function $T : \mathbb{N} \to \mathbb{N}$ is called *time–constructible* if

1. $T(n) \geq n$, for all $n \in \mathbb{N}$, and
2. it exists a Turing machine M_T which computes the function $n \to T(n)$ in time $T(n)$, where n and $T(n)$ are encoded as binary numbers.

Theorem 3.2.1. Time Hierarchy Theorem [HS65; HS66]
Let $f, g : \mathbb{N} \to \mathbb{N}$ be time–constructible functions such that $f(n) \log f(n) = o(g(n))$. Then, DTIME$(f(n))$ *is a proper subset of* DTIME$(g(n))$*, i.e. it exists a problem that can be decided by a Turing machine in time $T(g(n))$ but not in time $T(f(n))$.*

Theorem 3.2.2. Nondeterministic Time Hierarchy Theorem [Coo72; SFM78; Žák83]
Let $f, g : \mathbb{N} \to \mathbb{N}$ be time–constructible functions such that $f(n+1) = o(g(n))$. Then, NTIME$(f(n))$ *is a proper subset of* NTIME$(g(n))$*.*

So, it does matter how much time is available to solve a problem. But how does the chosen computational model effect the time complexity of a problem?

3.2.1. Relationships Among Computational Models

In the following, the effect on the time complexity of a problem is examined when the computational model is a single-tape, a multi-tape, or a nondeterministic Turing machine.

Theorem 3.2.3. *Let $T(n) : \mathbb{N} \to \mathbb{N}$ be a function, where $T(n) \geq n$. Then every $T(n)$ time multi–tape Turing machine has an equivalent $O(T^2(n))$ time single–tape Turing machine* [Sip06]*.*

Proof idea. To show that every multi–tape Turing machine has an equivalent single–tape Turing machine it is only necessary to simulate the multi–tape Turing machine on a single tape. This single–tape Turing machine needs $O(T(n))$ time to simulate one step of the original multi–tape Turing machine. Hence, the time complexity of the simulation is $O(T^2(n))$. □

Theorem 3.2.4. *Let $T(n) : \mathbb{N} \to \mathbb{N}$ be a function, where $T(n) \geq n$. Then every $T(n)$ time nondeterministic single-tape Turing machine has an equivalent $2^{O(T(n))}$ time deterministic Turing machine* [Sip06]*.*

Proof idea. The proof is similar to the one of Theorem 3.2.3. All possible computations of the nondeterministic Turing machine are simulated on a deterministic Turing machine. For this, the deterministic Turing machine explores the computation tree by using a depth–first search. The input is accepted if the simulation reaches the state of acceptance on some branch, otherwise it is rejected. By definition of the nondeterministic Turing machine the computation tree is a binary tree of height $T(n)$ with at most $2^{T(n)}$ leafs. Each branch has a length of at most $T(n)$. Hence, the complexity for the simulation with a deterministic Turing machine is $O(T(n)2^{T(n)}) = 2^{O(T(n))}$. □

So, choosing a deterministic single-tape over a multi-tape Turing machine increases the time complexity of a problem of at most a square. This means that the difference between the time complexity of problems measured on these computational models is at most *polynomial*. In comparison to this, there is a *exponential* difference between the time complexity of problems measured on deterministic and nondeterministic Turing machines.

The deterministic single-tape and multi-tape Turing machine are *polynomially equivalent*, i.e. they can simulate each other with only a polynomial increase in running time. But this is not only the case for Turing machines. It is believed that all reasonable deterministic computational models are polynomially equivalent [Sip06; AB09]. This is one reason why polynomial differences in the time complexity of problems are considered to be insignificant and why it is considered to be irrelevant which deterministic computational model is used. Another reason is that the difference between the growth rate of a polynomial and an exponential running time in relation to the input size is much greater than between two polynomials.

Thus, one main distinction in the time complexity theory is the distinction between classes of Turing machines that decide problems in polynomial and exponential time.

3.2.2. Complexity Classes P and NP

The set of problems that are decided in polynomial time on a deterministic single–tape Turing machine is the time complexity class P.

Definition 3.2.6.

$$\mathrm{P} = \bigcup_{c \geq 1} \mathrm{DTIME}(n^c)$$

The class P is a mathematically robust class because it is invariant for all computational models that are polynomially equivalent to the deterministic single–tape Turing machine. It also corresponds to problems that are considered to be realistically solvable on a computer. Though a polynomial running time with a high exponent is not of any practical use, but problems with such running time are quite rare [Sip06]. The class was introduced in the mid–1960s independently by Cobham and Edmonds [Cob64; Edm65].

An exponential running time is often the result of a brute-force strategy where every potential solution is checked. For many interesting problems that have an exponential running time it is not known whether there exists a polynomial time algorithm at all. The time complexity class *NP* is the collection of problems that are decided in polynomial time on a nondeterministic Turing machine.

Definition 3.2.7.

$$\mathrm{NP} = \bigcup_{c \geq 1} \mathrm{NTIME}(n^c)$$

The time complexity class NP is, like the class P, a mathematically robust class because all reasonable nondeterministic computational models are polynomial equivalent [Sip06]. It was also introduced in 1965 by Edmonds [Edm65]. An interesting fact is that a given solution for each instance of every problem in NP can be verified in polynomial time. This fact can be used for an alternative but equivalent characterization of NP.

Definition 3.2.8. A *verifier* for a language $L \subseteq \Sigma^*$ is a deterministic Turing machine V, where

$$L = \{\, w \in \Sigma^* \mid V \text{ accepts } \langle w, c \rangle \text{ for some certificate } c \in \Sigma^*, |c| \leq |w|^k, k \geq 1 \,\}$$

So, a *polynomial time verifier* runs in polynomial time in the length of w. A language L is *polynomially verifiable* if it has a polynomial time verifier.

A verifier for a language has in addition to the input information to verify in polynomial time whether the input is a member of the language.

Definition 3.2.9. The complexity class *NP* is the set of languages that have polynomial time verifiers.

Theorem 3.2.5. *Let $L \subseteq \Sigma^*$ be a language. Then*

$$L \in \bigcup_{k \geq 1} NTIME(n^k) \iff L \text{ has a polynomial time verifier}$$

Proof idea. The idea is to convert the nondeterministic Turing machine M to a polynomial time verifier V and vice versa.

"$\Rightarrow$" The certificate is a branch of the computation tree of M that accepts the input. Then V simulates M with the certificate.

"$\Leftarrow$" M guesses the certificate and simulates the verifier V.

□

This alternative definition of NP is very useful to check whether a problem is in NP because it is often much easier to describe a polynomial time verifier than to define a polynomial nondeterministic Turing machine. It also leads to a vivid distinction of the time complexity classes P and NP:

- P is the class of languages that can be *decided* quickly.
- NP is the class of languages that can be *verified* quickly.

Where "quickly" means in polynomial time solvable on a deterministic Turing machine.

Even when it seems clear that nondeterministic Turing machines are much more powerful than deterministic Turing machines there is no such proof. It is one of the greatest unsolved problems in theoretical computer science whether P = NP [Sip06]. One approach to answer this question came in the early 1970s when Cook and Levin discovered independently that some problems in NP are representatives for the complexity of the entire class [Coo71; Lev73]. These representatives are called *NP–complete*.

3.2.3. NP–completeness and Polynomial Time Reducibility

The discovery of NP–complete problems is of great importance for theoretical and practical reasons. On the one hand it would be sufficient to show that one NP–complete problem is in polynomial time solvable to prove that P = NP. On the other hand it would also be enough if one of these problems cannot be solved in polynomial time to show that P $\neq$ NP. So computer scientists attempting to answer the NP vs. P question can focus on these two approaches. A practical reason of the concept of NP–completeness is that researches may stop trying to find a polynomial time algorithm for a NP–complete problem, even when the nonexistence is not proven. Instead, they can spend time to find an approximation of the optimal solution or a polynomial time algorithm for a specialized variant of the problem.

One important NP–complete problem is the *satisfiability problem* which is to test whether a Boolean formula is satisfiable. Hence, it is the language of all satisfiable Boolean formula. In the following, a problem, i.e. language, will be defined with a high level description of the input for the algorithm (*Instance*) and when the input is accepted (*Question*).

Definition 3.2.10. SAT (Satisfiablility Problem)
Instance: A Boolean formula ϕ.
Question: Is ϕ satisfiable?

This problem is a representative for the complexity of NP because a solution for SAT can be efficiently used to solve every other problem in NP. The concept of efficiently reducing a problem to another is called *polynomial time reducibility*. In 1972 Karp introduced this method and showed a great variety of NP–complete problems [Kar72].

Definition 3.2.11. Let L_1, L_2 be languages over the alphabets Σ^* and Γ^*. L_1 is *polynomial time reducible* to L_2, written $L_1 \leq_P L_2$, if a polynomial time computable function $f : \Sigma^* \to \Gamma^*$ exists, such that for every $w \in \Sigma^*$,

$$w \in L_1 \iff f(w) \in L_2.$$

A function f is a *polynomial time computable function* if a polynomial time

deterministic Turing machine exists that computes f.

So, if a language L is polynomial time reducible to a language in P then L is in P also.

Theorem 3.2.6. *Let L_1, L_2 be languages.*
If $L_1 \leq_P L_2$ and $L_2 \in P$, then $L_1 \in P$.

Proof. Let M_f be the deterministic Turing machine that computes the function f of the polynomial time reduction, and let M_2 be the deterministic polynomial time Turing machine deciding L_2.

Then the Turing machine M_1 decides L_1 on input w as follows:

1. Run M_f on input w to compute $f(w)$.
2. Run M_2 on input $f(w)$.

$$\begin{aligned} M_1 \text{ accepts } w &\iff M_2 \text{ accepts } f(w) \\ &\iff f(w) \in L_2 \\ &\iff w \in L_1 \end{aligned}$$

M_1 runs in polynomial time because the composition of polynomials is a polynomial. Hence, L_1 is in P. □

With the concept of polynomial time reducibility NP–complete languages are defined as follows.

Definition 3.2.12. A language L is *NP–complete* if

1. L is in NP, and
2. every language in NP is polynomial time reducible to L.

A problem is *NP–hard* if only the second property of Definition 3.2.12 is met. Previously it was already said that SAT is a NP–complete problem. This statement is known as the Cook–Levin theorem, with the consequence that if SAT is in deterministic polynomial time solvable then every problem in NP is in deterministic polynomial time solvable.

Theorem 3.2.7. *SAT is NP–complete [Coo71; Lev73].*

The proof of this theorem is a good example for how the NP–completeness of a problem is shown. In general, first it is shown that the problem is in NP by describing a polynomial time verifier for an instance of the problem with a suitable certificate. Second it is shown that every problem in NP is polynomial time reducible to the problem. For SAT this can be done with a construction of a polynomial time reduction for each problem L in NP. The reduction takes an instance w for L and computes a Boolean formula ϕ that simulates the computation on input w of the corresponding Turing machine of L. Hence w is in L if and only if ϕ is satisfiable. The technical details are spared at this point which can be found in great detail in many textbooks [Sip06; Cor+09]. Another way is to make a polynomial time reduction from a known NP–complete problem to the problem in question. This is sufficient as stated in the following Theorem 3.2.8.

Theorem 3.2.8. *If L_1 is NP–complete and $L_1 \leq_P L_2$ for some L_2 in NP, then L_2 is NP–complete.*

Proof. Let M_g be the deterministic Turing machine for $L_1 \leq_P L_2$ with the polynomial time computable function g. For every language L in NP let M_f be the deterministic Turing machine that computes the function f of the polynomial time reduction from L to L_1. Then $g \circ f$ is the polynomial time computable function of $L \leq_P L_2$, where the deterministic Turing machine $M_{g \circ f}$ computes the function for an instance $w \in \Sigma^*$ as follows:

1. Run M_f on input w to compute $f(w)$.
2. Run M_g on input $f(w)$ to compute $g(f(w))$.

The Turing machine $M_{g \circ f}$ runs in polynomial time because M_f and M_g compute $f(w)$ and $g(f(w))$ in polynomial time. Hence,

$$w \in L \iff f(w) \in L_1 \iff g(f(w)) = g \circ f(w) \in L_2.$$

□

So, to prove that a problem is NP–complete it is enough to show that the problem is in NP and that a known NP–complete problem is polynomial time reducible to it. Another good known and well studied computational decision

problems that is NP–complete is 3–SAT [Cor+09]. It is like SAT only that the Boolean formulas are in 3–CNF.

Definition 3.2.13. 3–SAT
Instance: A Boolean formula ϕ in 3–CNF.
Question: Is ϕ satisfiable?

This problem and its many variants are used as a starting point for many reductions to show NP–completeness of other problems.

4. Planar 3–SAT

Since Lichtenstein's description of Planar 3–SAT many properties on planar graphs were shown to be NP–complete by means of reductions from Planar 3–SAT itself or one of its many variants. Basic versions of Planar 3–SAT, where the definition of the associated graph is adjusted a little, are often used for restriction. The most used version for this is Simple Planar 3–SAT.

4.1. Planar 3–SAT

Let ϕ be a Boolean formula in 3–CNF with a set $\mathcal{C}$ of m clauses over the variables $\mathcal{X} = \{ x_1, \ldots, x_n \}$. The graph $G(\phi) = (V, E)$ is called the *associated graph* of ϕ, where

$$
\begin{aligned}
V &= \mathcal{X} \cup \mathcal{C}, \text{ and} \\
E &= E_1 \cup E_2 \text{ with} \\
E_1 &= \{ (x, C) \mid x \in C \text{ or } \neg x \in C, \text{ for } x \in \mathcal{X} \text{ and } C \in \mathcal{C} \} \text{ and} \\
E_2 &= \{ (x_1, x_2), (x_2, x_3), \ldots, (x_n, x_1) \}.
\end{aligned}
$$

The edges of E_2 describe a simple cycle with all variable–vertices. Given a planar embedding of the associated graph the inside of this cycle will also be denoted as the *left side* when describing on which side a clause lies according to a variable–vertex. Because there is no distinction between positive and negative literals it is not possible to derive the Boolean formula from a given associated graph. A *planar Boolean formula* is a Boolean formula in 3–CNF with an associated graph that is planar.

Definition 4.1.1. Planar 3–SAT [Lic82]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$.

Question: Is ϕ satisfiable?

Theorem 4.1.1. *Planar 3–SAT is NP–complete.*

Proof according to [Lic82]. Given an instance for Planar 3–SAT it is in polynomial time verifiable that the Boolean formula is in 3–CNF, the associated graph is planar [HT74; WW99; BM04; HT08], and the planar embedding matches the definition of the associated graph. The certificate is an assignment for the Boolean formula which can be easily used to evaluate the formula in linear time by checking that each clause evaluates to 1. Hence, Planar 3–SAT is in NP.

To show that Planar 3–SAT is NP–hard a reduction from 3–SAT is described. The main idea for the reduction is to transform nonplanar Boolean formulas in polynomial time into equisatisfiable planar Boolean formulas. Let ϕ be a Boolean formula with m clauses and n variables for the decision problem 3–SAT. We describe an algorithm f that transforms ϕ into ϕ' such that they are equisatisfiable and that the associated graph of ϕ' is planar.

Let G be a grid of size $3m \times 3m$. This is the maximal size necessary if each clause contains three literals. Each clause is aligned on the left border as one vertex covering a maximum of 3 vertical adjacent grid points. The number of grid points corresponds to the size of each clause. The vertices for the variables are aligned on the bottom.

If the variable x occurs n_x times in ϕ then the corresponding vertex covers n_x horizontal adjacent grid points. Every clause is then connected with the containing variables by a horizontal and vertical line on the grid, using every grid line just once. To fulfill the definition of the associated graph the variable–vertices are connected in a cyclic order.

In the next step every crossing (Figure 4.2a) is eliminated by adding new variables and clauses to build generic planar crossover boxes (Figure 4.2b) where two lines intersect. The crossover box is just a subgraph that carries the values of the two variables crossing free over the intersection. The underlying formula

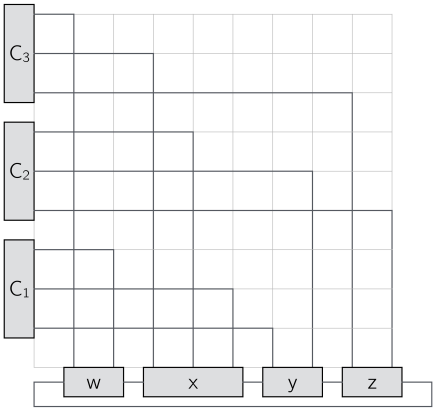


Figure 4.1.: Let $\phi = (w \vee \neg x \vee y) \wedge (\neg x \vee \neg y \vee z) \wedge (\neg w \vee x \vee y)$ be a Boolean formula in 3–CNF. Example for a nonplanar embedding of ϕ on a grid with clauses at the left side and vertices at the bottom.

of the subgraph is the following:

$$(\neg x_2 \vee \neg y_2 \vee \alpha) \wedge (x_2 \vee \neg \alpha) \wedge (x_2 \vee \neg \alpha) \tag{4.1}$$

$$(\neg x_2 \vee y_1 \vee \beta) \wedge (x_2 \vee \beta) \wedge (\neg y_1 \vee \neg \beta)) \tag{4.2}$$

$$(x_1 \vee y_1 \vee \gamma) \wedge (\neg x_1 \vee \neg \gamma) \wedge (\neg y_1 \vee \neg \gamma) \tag{4.3}$$

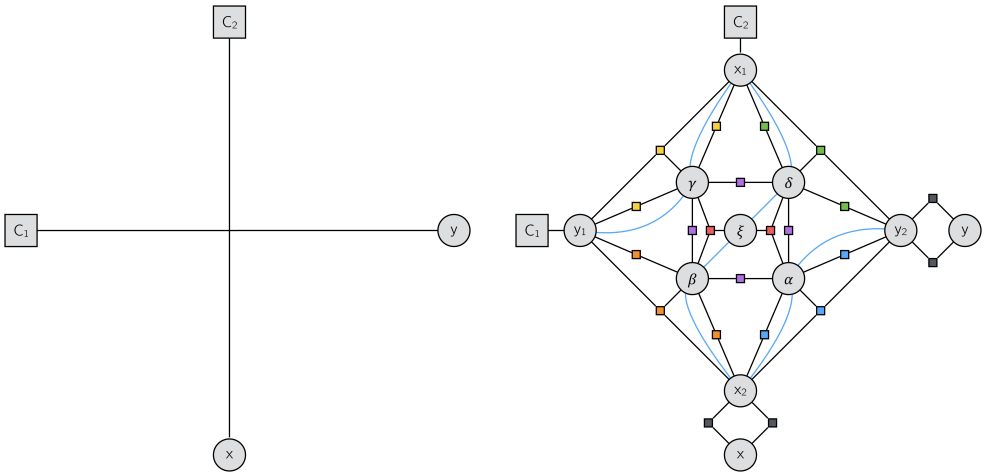
$$(x_1 \vee \neg y_2 \vee \delta) \wedge (\neg x_1 \vee \neg \delta) \wedge (y_2 \vee \neg \delta) \tag{4.4}$$

$$(\alpha \vee \delta \vee \xi) \wedge (\beta \vee \gamma \vee \neg \xi) \tag{4.5}$$

$$(\neg \alpha \vee \neg \beta) \wedge (\neg \beta \vee \neg \gamma) \wedge (\neg \gamma \vee \neg \delta) \wedge (\neg \delta \vee \neg \alpha) \tag{4.6}$$

$$(x_2 \vee \neg x) \wedge (x \vee \neg x_2) \wedge (y_2 \vee \neg y) \wedge (y \vee \neg y_2) \tag{4.7}$$

The following replacment is done for each crossing caused by edges from two variable x and y to the corresponding clauses. Every occurrence of y or $\neg y$ is replaced in the affected clauses with x_1 or $\neg x_1$, and analogues for y. It can be easily verified that the formula of the subgraph is satisfiable if and only if $x_1 \Leftrightarrow x$ and $y_1 \Leftrightarrow y$. Each crossing is eliminated with a constant number of new variables and clauses. It is also possible to connect all variables of the crossover box without adding an intersection (Figure 4.2b) to fulfill the definition of planar Boolean formulas. All new intersections that arise while connecting all the variable–vertices in a circular order can be eliminated with the previous



(a) Possible intersection.

(b) Crossover box eliminates the intersection. The variable–vertices can be connected in a cyclic order (blue edges).

Figure 4.2.: Example of the elimination of an intersection (Figure 4.2a) with a crossover box (Figure 4.2b). The intersection is caused due the edges connecting the variable–vertices of a and b with the corresponding clause–vertices of C_1 and C_2. The following color schema is used to simplify the identification of the clauses for the crossover box:
(4.1) blue ■, (4.2) green ■, (4.3) yellow ■, (4.4) orange ■, (4.5) red ■, (4.6) purple ■, (4.7) dark grey ■.

mentioned crossover box as well (Figure 4.3).

There only can be $O\left(m^2\right)$ intersections by performing these steps. Hence the algorithm runs in polynomial time and the size of the new but equisatisfiable formula is polynomial in the size of the original one. It follows that the Planar 3–SAT problem is NP–hard.

Together with the property that it is in NP it is shown that Planar 3–SAT is NP–complete. □

4.1.1. Rectilinear Planar 3–SAT

Knuth and Raghunathan use a different definition of Planar 3–SAT by replacing the definition of the associated graph [KR93]. A *rectilinear embedding* of a planar Boolean formula has all variable–vertices arranged on a straight line, and

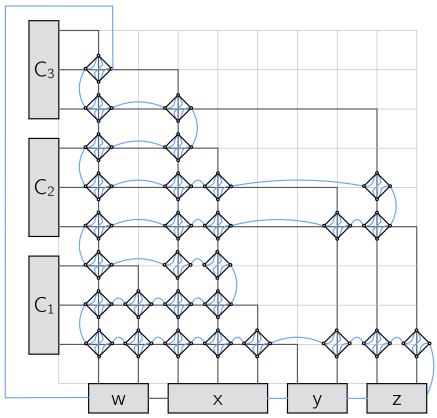


Figure 4.3.: Complete transformation of the embedding of a Boolean formula to a planar embedding.

the clauses are represented as horizontal lines with at most three vertical lines, also called *legs*, to there corresponding variable-vertices. The clauses can be nested such that the graph is still planar (e.g. Figure 4.4). The edges of the cycle through all variable-vertices is omitted but could be easily added.

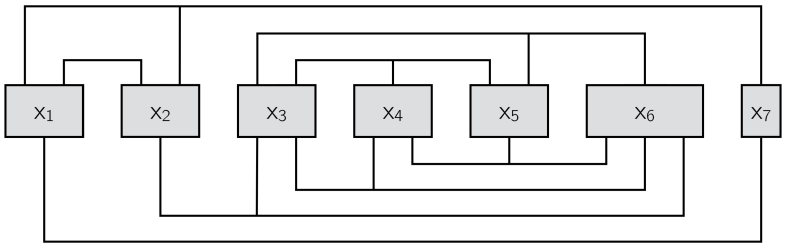


Figure 4.4.: Rectilinear embedding of a planar Boolean formula ϕ with clauses $(\neg x_1 \vee x_2)$, $(\neg x_1 \vee \neg x_2 \vee x_7)$, $(x_1 \vee \neg x_7)$, $(x_2 \vee x_3 \vee x_6)$, $(\neg x_3 \vee x_4 \vee \neg x_5)$, $(\neg x_3 \vee \neg x_4 \vee x_6)$, $(x_3 \vee x_5 \vee x_6)$, and $(x_4 \vee x_5 \vee \neg x_6)$.

Theorem 4.1.2. *Let ϕ be a planar Boolean formula. Given a combinatorial embedding of the associated graph $G(\phi)$ together with a cyclic order of the variable–vertices it is always possible to arrange a rectilinear embedding of $G(\phi)$.*

Proof. Let ϕ be a planar Boolean formula with n variables and m clauses. Let $G(\phi)$ be the associated graph of ϕ with a cyclic order of the variable-vertices $x_1, x_2, \ldots, x_n$.

To obtain the rectilinear embedding of the graph it is necessary to know how to nest the clauses with up to three legs without causing any crossing. Therefore a *level* $\mathcal{L} \in \mathbb{Z}$ is assigned to each clause which can then be associated with a distance to the variable-vertices. In the planar embedding of the associated graph clauses inside the cycle of the variable–vertices are assigned with positive integers, and clauses on the outside with negative integers. The variable–vertices are on level 0. A clause C_1 *encloses* another clause C_2 if they are on the same side and the leftmost and rightmost variable–vertices of C_1 are surrounding the variable–vertices of C_2 on the straight line of the variable–vertices. For instance, $C_7 = (\neg x_1 \vee \neg x_6)$ encloses $C_{10} = (x_3 \vee x_4 \vee \neg x_5)$ in example 4.1.1. If this is ambiguous, i.e. both clauses have the same leftmost and rightmost variables, then there are two possible cases.

The first case is that both clauses contain the same variables but different literals, e.g. $(x \vee \neg y \vee z)$ and $(\neg x \vee \neg y \vee z)$. This is possible because in the associated graph there is no distinction between positive and negative variables. The second case is when the number of containing literals differ. In both cases the cyclic ordering of the edges at the leftmost variable–vertex breaks the tie. That means that the clause with the leftmost edge incident to the variable–vertex encloses the other, e.g. clause C_2 encloses clause C_3 in example 4.1.1.

The level of each clause is defined recursively in dependence of the enclosing clauses. Each clause C that does not enclose any other clause is assigned with level $\mathcal{L}(C) = 1$. The level of clause C is then

$$\mathcal{L}(C) = 1 + max\,\{\,\mathcal{L}(D) \mid D \text{ clause of } \phi,\ C \text{ encloses } D\,\} \tag{4.8}$$

These assignments do not consider on which side the clauses are located. To take that into account levels of clauses outside the variable–vertices cycle are negated. With this information the levels for the horizontal lines representing the clauses in the embedding are given. The vertical legs are just straight lines from the variable–vertices in direction of the corresponding clauses until they touch, and mark the length of the horizontal lines. Vertical lines from the same variable–vertex are drawn side by side with a little gap between each other.

Lemma 4.1.3. *Horizontal lines on the same level cannot overlap each other.*

Proof. Suppose two horizontal lines h_1 and h_2 overlap each other, and let C_1 and C_2 be the corresponding clauses. Then there are three possibilities how they overlap:

1. They have the same leftmost and rightmost variable–vertex.
2. One horizontal line lies completely on the other.
3. The intersection of the horizontal lines is a proper subsection of both lines, i.e. there are points on both lines that are not contained in the intersection.

In the first case, the leftmost and rightmost edge in the combinatorial embedding of one corresponding clause–vertex, say C_1, have to be before and after the edges of the other clause–vertex, say C_2. Otherwise the graph cannot be planar. Though C_1 encloses C_2 and therefore have by definition a higher level. Hence, h_1 and h_2 are not on the same level which is a contradiction to the assumption.

For the second case assume without loss of generality that h_2 lies completely on h_1. It follows that the leftmost variable–vertex for C_1 is before the variable–vertices for C_2 on level 0. Analogous applies for the rightmost variable–vertex for C_1. Then by definition C_1 encloses C_2 and therefore $\mathcal{L}(C_1)$ is greater than $\mathcal{L}(C_2)$. Hence h_1 and h_2 are not on the same level and cannot overlay each other. This is a contradiction to the previous assumption.

Each clause is on a simple cycle with some edges of the variable cycle and the edges to the clause. For the third case there have to be at least one but not all variable-vertices of clause C_2 on the cycle of C_1 and vice versa. Hence, there is at least one variable–vertex of C_2 not on the cycle of C_1. Because the clause–vertex of C_2 have edges to all corresponding variable–vertices there is an intersection with the cycle of C_1. But this cannot be the case because the associated graph is planar. Thus, case three is not possible either. □

Lemma 4.1.4. *The vertical line from a variable–vertex to the corresponding horizontal line does not intersect with any other line.*

Proof. A vertical line cannot intersect with another vertical line because by construction they are set side by side at the variable–vertex. It cannot intersect with

a horizontal line either because then there would be a clause on a lower level that encloses the corresponding variable–vertex of the vertical line. But then the associated graph would be nonplanar as described in Lemma 4.1.3. Hence, no vertical line intersects with any other line except with the corresponding horizontal line. □

From lemmata 4.1.3 and 4.1.4 follows that in this way a rectilinear embedding of $G(\phi)$ can be easily drawn by giving each level an appropriate distance, and each variable–vertex a suitable width and distance between each other.

□

Example 4.1.1. Let ϕ be a planar Boolean formula and $G(\phi)$ the associated graph with a planar embedding as shown in Figure 4.5 of the form:

$$\phi = (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_6) \wedge (x_2 \vee x_3 \vee x_6) \wedge (x_4 \vee \neg x_6) \wedge (\neg x_4 \vee x_5) \\ \wedge (x_5 \vee \neg x_6) \wedge (\neg x_1 \vee x_6) \wedge (x_1 \vee \neg x_5 \vee x_6) \wedge (x_1 \vee \neg x_3) \wedge (x_3 \vee x_4 \vee x_5) \tag{4.9}$$

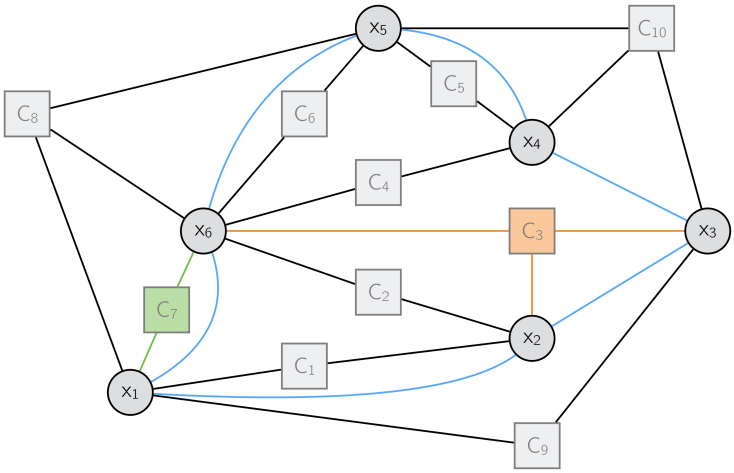


Figure 4.5.: Planar embedding of the associated graph of ϕ with the indices of the clauses according to their appearance in the formula. Clauses C_3 in orange and C_7 in green are colored for easier recognition in the rectilinear embedding of $G(\phi)$ (Figure 4.6). The blue edges represent the cycle through all variable–vertices.

To get a rectilinear embedding of $G(\phi)$ the variable–vertices are arranged on a straight line beginning with the x_1 and ending with x_6. The clauses

C_1, C_2, and C_3 on the left side have level 1, and the clauses C_9, and C_{10} on the other side of the variable–vertices have level -1 because they are not enclosing any other clause. From this on the levels of the other clauses are by definition

$$\begin{aligned}
l(C_4) &= 1 + max\{l(C_5), l(C_6)\} = 1 + max\{1, 1\} = 2 \\
l(C_3) &= 1 + max\{l(C_4), l(C_5), l(C_6)\} = 3 \\
l(C_2) &= 1 + max\{l(C_3), l(C_4), l(C_5), l(C_6)\} = 4 \\
l(C_8) &= min\{l(C_9), l(C_{10)}\} - 1 = -2 \\
l(C_7) &= min\{l(C_8), l(C_9), l(C_{10})\} - 1 = -3.
\end{aligned}$$

Given the level of each clause the rectilinear embedding of $G(\phi)$ can be drawn like in Figure 4.6.

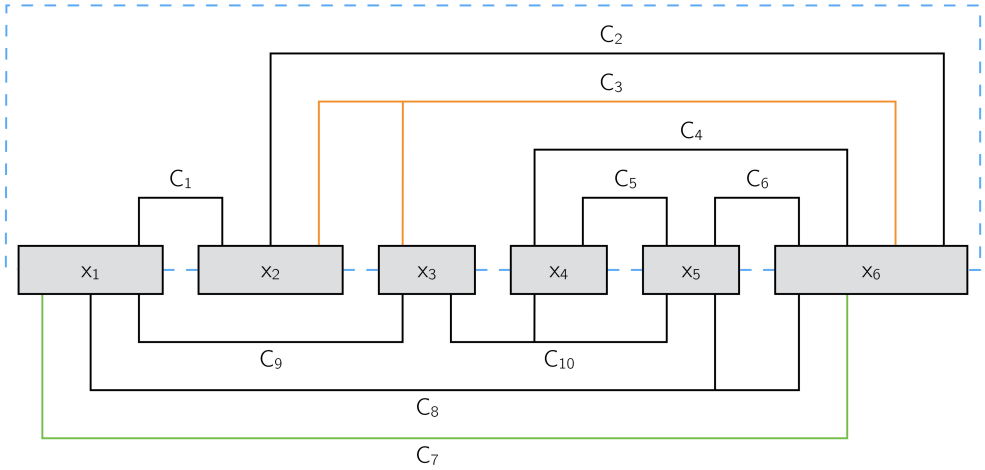


Figure 4.6.: Possible rectilinear embedding of $G(\phi)$. The dashed blue line for the cycle through all variable vertices are omitted by Knuth and Raghunathan but could be easily added. The colored lines represent the same clauses as in figure 4.5

.

From Theorem 4.1.2 follows that it can be assumed that the associated graph of a planar Boolean formula for Planar 3–SAT has a rectilinear embedding.

4.1.2. Application of Planar 3–SAT

To show NP-completeness for the Minimum Average Distance Triangulation decision problem Kozma performed a reduction form Planar 3–SAT [Koz11].

Remark. Kozma defines a restricted variant of Planar 3–SAT, where in the Boolean formula each variable occurs at most once in the same clause, and at least in two clauses. These restriction can be assumed without loss of generality [4.1.3].

Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of points in the plane, and let $w : P \times P \to \mathbb{R}$ be a semimetric weight function for edges between pairs of points in P. A *semimetric* on a set S is a function, called the *distance function*, $d : X \times X \to \mathbb{R}$ that satisfies the axioms $d(x, y) \geq 0$, $d(x, y) = 0$ if and only if $x = y$, and $d(x, y) = d(y, x)$. In contrast to a metric satisfies a semimetric not necessarily the triangle inequality $d(x, y) + d(y, z) \geq d(x, z)$. For a given weight W^* the *Minimum Average Distance Triangulation* decision problem is the question whether there exists a geometric, crossing-free graph $T = (V, E)$ embedded on P with weight

$$W(T) = \sum_{(p,q)\in E} w(p, q) \leq W * .$$

According to Kozma the problem is similar to several other problems, such as

- the Minimum Average Distance Spanning Subgraph [JLK78] problem,
- the Minimum Average Distance Spanning Tree [Hu74] problem,
- the Wiener-index [Wie47] in chemistry,
- the Minimum Weight Triangulation [MR08] problem, and
- the Minimum Dilation Triangulation [KM05] problem,

but he has not found any fundamental connections to the Minimum Average Distance Triangulation problem [Koz11]. The Minimum Weight Triangulation problem is presented in more detail in Section 5.2.

4.1.3. Restrictions on Planar 3–SAT

There are some variants of Planar 3–SAT which are mainly defined to just show the NP–completeness of other problems rather than focus on the study of Planar 3–SAT. Nevertheless, these variants are examples of the usefulness of Planar 3–SAT and its high adaptability to a wide range of computational problems with just a few restrictions on the definition of Planar 3–SAT or its variants. Some of these restrictions on Boolean formulas can be assumed without loss of generality others lead to new variants of Planar 3–SAT. The definition of a new variant comes in hand with a polynomial time reduction from or to a known problem.

Transformations of Boolean Formulas

To perform a reduction from one satisfiability problem to another it is a common proof technique to transform the given Boolean formula. These transformations are of the kind that new variables and clauses are added to the formula, or existing variables are replaced by new ones. Furthermore it is necessary that the given formula is satisfiable if and only if the new formula after the transformation is satisfiable, i.e. the formulas are *equisatisfiable*. For that a description or function is given to obtain from the satisfying assignment of one formula the satisfying assignment of the other, and vice versa.

Assumptions Without Loss of Generality

In the study of the satisfiability of Boolean formulas it is often helpful to make some assumptions to simplify the form of the formulas. For example with these assumptions the number of possible cases for a transformation to an equisatisfiable formula is reduced.

Both Literals of a Variable Occur Without loss of generality it can be assumed that both literals of every variable appear in the Boolean formula. Suppose that $\neg x$ does not occur in the Boolean formula. Then the variable x can be set to 1 in a possible satisfying assignment. It follows that every clause containing x is always satisfied and hence all these clauses can be deleted. The same

argumentation is valid if only $\neg x$ occurs. Then all clauses containing $\neg x$ can be deleted from the Boolean formula with the assignment $x = 0$.

No Clause Contains Both Literals of a Variable Without loss of generality it can be assumed that no clause contains both literals of a variable. Is this the case such clauses are always satisfied and can be deleted from the formula.

4.2. Planar exactly 3–SAT

In some cases it is easier to make a reduction from a variant of Planar 3–SAT where are in each clause exactly three literals, e.g. to prove NP–hardness for Simple Planar 1–in–3–SAT [5.1].

Definition 4.2.1. Planar exactly 3–SAT

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.

Theorem 4.2.1. *Planar exactly 3–SAT is NP–complete.*

Proof. The NP–completeness proof of Planar 3–SAT (Theorem 4.1.1) can be used to show that Planar exactly 3–SAT is also NP–complete. For this the reduction is not from 3–SAT but from 3–SAT restricted to Boolean formula in exactly 3–CNF, which is also NP–complete [GJ79]. No other adjustments are necessary and therefore Planar exactly 3–SAT is also NP–complete. □

4.3. Simple Planar 3–SAT

Another definition of planar 3–SAT is without the necessity that the variable-vertices are connected in a circular order.

Definition 4.3.1. Simple Planar 3–SAT

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:

1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Theorem 4.3.1. *Simple Planar 3–SAT is NP–complete.*

Proof. The NP-completeness of Simple Planar 3–SAT follows directly from the proof of the NP-completeness of Planar 3–SAT. The proof is the same with the exception that the edges between the variables are not necessary. □

4.4. Separable Planar 3–SAT

Lichtenstein also introduced a variant of Planar 3–SAT to present an alternative and simpler proof to show that the Geometric Connected Dominating Set problem is NP–complete.

Let $G = (V, E)$ be a graph and let S be a subset of V. S is called a *dominating set* if every vertex in $V \setminus S$ is adjacent to at least one member of S. Additionally S is a *connected dominating set* if S is connected and a dominating set. The *Geometric Connected Dominating Set* problem is given a graph G with vertices representing points in the Euclidean plane and with edges between these vertices if the distance between them is not greater than 1, and an integer k, does there exist a connected dominating set of size k. This graph is also called *unit disk graph*.

This problem is of great interest in the study of networking, e.g. finding a virtual wireless backbone for routing in an ad-hoc network [DB97; GMM13].

Definition 4.4.1. Separable Planar 3–SAT[1] [Lic82]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:
  1. At each variable–vertex all edges representing a positive literal are incident to one side of the vertex, and all edges representing a negative literal are incident to the other side. The separation is according to the edges of the cycle through all variable–vertices.

Question: Is ϕ satisfiable?

[1] In the paper of Lichtenstein the problem is mentioned in Lemma 1 with no specific name.

Another way to look at this definition is to have a separate vertex for each literal of a variable and an additional edge connecting the literal–vertex of this variable. Gibson et al. and Mitchell and Packer call this variant *Positive-Negative Planar 3–SAT* [Gib+08; MP09].

Theorem 4.4.1. *Separable Planar 3–SAT is NP–complete.*

Proof according to [Lic82]. A given Boolean formula and the embedding of the associated graph can be verified in polynomial time to be a valid instance for Separable Planar 3–SAT. To verify the embedding of the associated graph it is necessary to check whether at each variable–vertex all edges incident to the same side correspond to the same literal. In addition with a assignment of the variables it can also be verified without greater effort that the assignment is satisfiable. Thus a given certificate for the problem can be validated in polynomial time, which shows that Separable Planar 3–SAT is in NP.

With a reduction from Planar 3–SAT it is shown that the problem is NP–hard. Let $I = (\phi, G)$ be an instance for Planar 3–SAT, where ϕ is a planar Boolean formula and $G = (V, E)$ the planar embedding of the associated graph $G(\phi)$. For the following transformation to Separable Planar 3–SAT a variable x in ϕ is called *invalid* if the variable–vertex x has edges representing positive and negative variables on the same side. An instance of Planar 3–SAT with no invalid vertices in the associated graph is obviously a valid instance for Separable Planar 3–SAT.

Now, let x be an invalid variable in ϕ with a total of s occurrences in s clauses. The clause–vertices in $G(\phi)$ are placed in clockwise order around x, where the first t clause–vertices lie on the left side (e.g. Figure 4.7a).

The variable x is removed from ϕ and replaced by $2s - 2$ new variables $x_1, x_2, \ldots, x_{2s-2}$. Additionally ϕ is appended by the following clauses $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge \cdots \wedge (x_{2s-2} \vee \neg x_1)$ to force the new variables to have the same value in a satisfying assignment. The substitution of x in the affected clauses depends on the new embedding of $G(\phi)$.

This is done by arranging the new clause-vertices and variable-vertices as a cycle at the place of the previous variable–vertex x. The cycle has the form that clause C_i is associated with the clause–vertex of $(x_{2i-1} \vee x_{2i})$, for $i = 1, 2, \ldots, t$, and with the clause–vertex of $(x_{2i} \vee x_{2i+1})$, for $i = t, t+1, \ldots, s-1$ and $x_{2s-1} = x_1$. On the variable cycle x_1 is connected with the predecessor vertex of

the original vertex x, and x_{2t} has an edge to the successor of x. The previous edge form C_i to x is then replaced with an edge between the variable–vertex of either the positive or negative variable of the associated clause. This ensures that at each new variable–vertex all edges representing positive variables are on one side and edges indicating negative variables are on the other side (Figure 4.7b).

Remark. If the edges of a variable–vertex x to the corresponding clause–vertices are only on one side then the variable is replaced by $2s$ variables. Additionally $2s$ clauses $(x_i \vee x_{i+1})$, and $(x_{2s} \vee \neg x_1)$ are appended to the formula, for $i = 1, 2, \ldots, 2s-1$. The vertices are then aligned on a line except the last clause–vertex of $(x_{2s} \vee \neg x)$ which is put on the opposite side.

Transforming each invalid variable of ϕ in this way leads to a planar Boolean formula with an embedding according to Separable Planar 3–SAT. It is obvious that the replacement of the invalid variables ensures equisatisfiability with the original formula. This is because all new variables for one invalid variable have the same value in an satisfying assignment due to the additional clauses. The transformation can be done in polynomial time because each individual replacement of a variable can be done in polynomial time and the number of variables is linear with respect to the input size. Hence the problem is NP–hard, and therefore NP–complete.

□

4.5. Separable Simple Planar 3–SAT

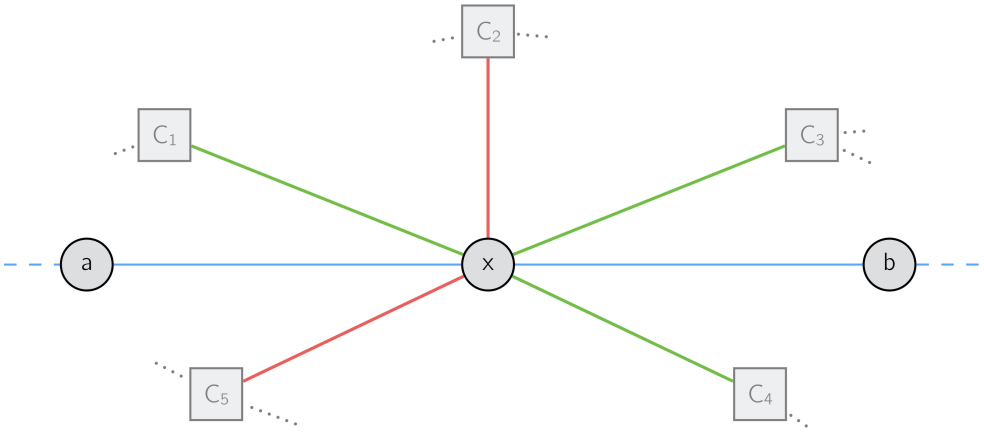
Separable Simple Planar 3–SAT[2] introduced by Du [DKW02] is a combination of Simple Planar 3–SAT and Separable Planar 3–SAT.

Definition 4.5.1. Separable Simple Planar 3–SAT [DKW02]

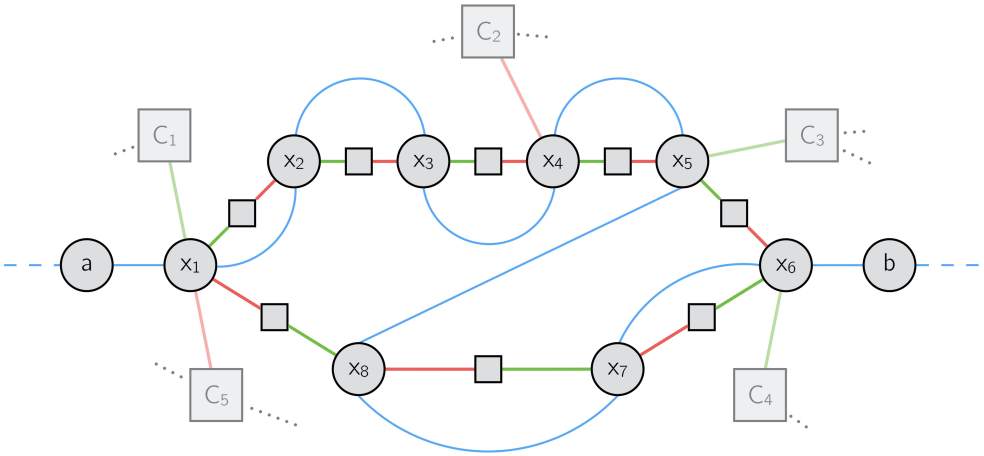
Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:

[2] In the paper of Du, Ko, and Wang the problem is called Strongly Planar 3–SAT.



(a) Variable–vertex x has connections to five clause–vertices $C_1, \ldots, C_5$. A green edge indicates a positive variable and a red edge a negative variable of x in the connected clause. The blue edges are part of the cycle of all variables in the formula.



(b) The variable–vertex x is replaced by a cycle of eight new variables–vertices $x_1, x_2, \ldots, x_8$ and eight new clause–vertices. The new clauses force all x_i, for $i = 1, 2, \ldots, 8$, to have the same value in a satisfying assignment. The clause–vertices for $C_1, C_2, \ldots, C_5$ are connected to the new variable–vertices such that for each variable–vertex all edges indicating positive variables (green edges) are on one side and edges indicating negative variables (red edges) are on the other side. The cycle through all variable-vertices (blue edges) connect the new variable–vertices in a zig-zag fashion.

Figure 4.7.: Example for a replacement of a variable x in a Boolean formula occurring in the clauses with literal x in C_1, C_3, C_4 and with literal $\neg x$ in C_2, C_5.

1. Each variable–vertex is replaced by two literal–vertices with an edge connecting them, and edges to the corresponding clause–vertices.
2. $G(\phi)$ has no other edges.

Question: Is ϕ satisfiable?

Theorem 4.5.1. *Separable Simple Planar 3–SAT is NP–complete.*

Proof. Separable Simple Planar 3–SAT is in NP because it is in linear time verifiable that the associated graph is planar, and the assignment of the certificate for the Boolean formula is satisfiable.

With a simple reduction from Separable Planar 3–SAT (Definition 4.4.1) it is shown that the problem is also NP–hard. Let ϕ be the planar Boolean formula of an instance for Separable Planar 3–SAT. Then ϕ is also an instance for Separable Simple Planar 3–SAT because the definition of the associated graph of ϕ fulfills the properties for Separable Simple Planar 3–SAT by deleting the edges between the variable–vertices, and separating them into literal–vertices with an connecting edge. If the instance do not meet the properties of Separable Planar 3–SAT, e.g. the variable–vertices are not on a cycle, then ϕ is expanded with a clause $(x \wedge y)$ in disjunctive normal form. In this way it is guaranteed that the instance is rejected because the Boolean formula is not in 3–CNF. Obviously the reduction needs only polynomial time. So, the problem is NP–hard and hence, NP–complete. □

Remark. With the reduction from Separable Planar 3–SAT the proof of theorem 4.5.1 is simplified in contrast to the proof of Du, Ko, and Wang [DKW02] described by Wu [Wu15]. There, a reduction from Planar 3–SAT is described where it is necessary to use a generic crossover box to eliminate the intersections caused by the separation of the variable–vertices.

4.6. Clause–Linked Planar 3–SAT

Definition 4.6.1. Clause–Linked Planar 3–SAT [KLN91a]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:

1. $G(\phi)$ has no edges between variable–vertices.
2. The set of clauses allows a linear ordering such that $G(\phi)$ is still planar when consecutive clause–vertices are connected with an edge.

Question: Is ϕ satisfiable?

Theorem 4.6.1. *Clause–Linked Planar 3–SAT is NP–complete* [KLN91a].

5. Planar 1–in–3–SAT

1–in–3–SAT is a main variant of the satisfiability problem 3–SAT. A restricted variant of the planar version of this problem is used to show NP–completeness for the 3–dimensional Matching problem [DF86]. A very important result based on a variant of 1–in–3–SAT is that the Minimum Weight Triangulation is NP–complete [MR08]. The NP–completeness of this problem is stated as an open problem in the influential textbook *Computers and Intractability: A Guide to the Theory of NP-completeness* by Garey and Johnson [GJ79].

5.1. Simple Planar 1–in–3–SAT

Dyer and Frieze use Simple Planar 1–in–3–SAT to show that the planar version of another well known problem [GJ79], namely the 3–dimensional Matching problem, is also NP–complete.

Let X, Y, and Z be finite, disjoint sets, and let T be a subset of all possible triples (x, y, z) such that $x \in X$, $y \in Y$, and $z \in Z$, and let $k \geq 0$ be an integer. A matching M is a subset of T with the property that for any two distinct members $m_1 = (x_1, y_1, z_1)$ and $m_2 = (x_2, y_2, z_2)$ the inequalities $x_1 \neq x_2$, $y_1 \neq y_2$, and $z_1 \neq z_2$ hold. The question of 3–dimensional Matching is whether there exists a matching of size k.

Their motivation was also to have an easier starting point to prove NP–completeness for planar variants of general problems.

Definition 5.1.1. Simple Planar 1–in–3–SAT[1] [DF86]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:

[1] In the paper of Dyer and Frieze the problem is called Planar 1–in–3–SAT.

1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is satisfied if exactly one literal is true.

Theorem 5.1.1. *Simple Planar 1–in–3–SAT is NP–complete.*

Proof. Simple Planar 1–in–3–SAT is in NP because the planar Boolean formula and the planar embedding of the associated graph, and the validity of an assignment can be verified in polynomial time.

With a quite simple reduction from Planar exactly 3–SAT (Definition 4.2.1) it can be shown that the problem is also NP–hard. Let ϕ be the planar Boolean formula of an instance for Planar exactly 3–SAT. First, eliminate the edges between the variable–vertices of the associated graph. Then each clause $C = (l_1 \vee l_2 \vee l_3)$ in ϕ is replaced by three clauses

$$(l_1 \vee w_C \vee x_C),\ (\neg l_2 \vee w_C \vee y_C), \text{ and } (\neg l_3 \vee x_C \vee z_C)$$

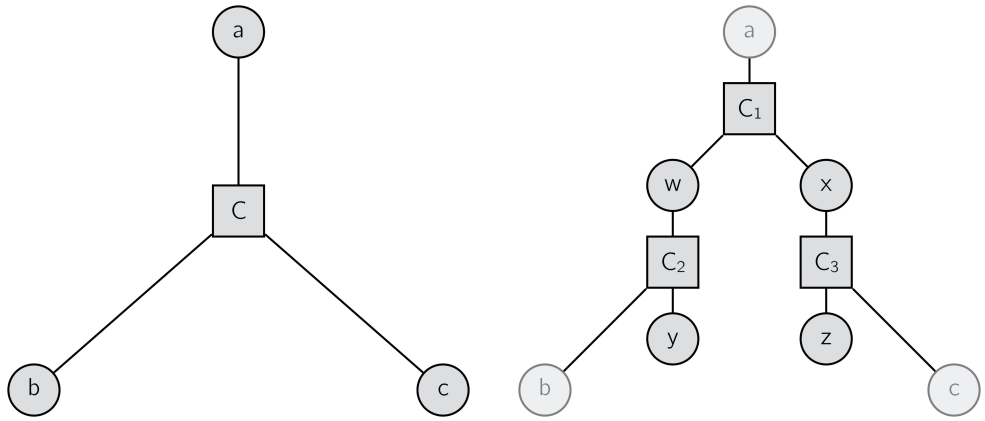
and new introduced variables w_C, x_C, y_C, and z_C. These new clauses are satisfied by exactly one literal each if and only if the original clause C is satisfiable. This can be verified by a simple truth table of the seven satisfying assignments for C and the resulting assignment of the new variables. It is also possible to draw the new clause–vertices at the place of the original one without breaking the planarity constraint of the Boolean formula.

For instances that do not meet the properties of Planar exactly 3–SAT, the Boolean formula is expanded with the clause $(x \wedge y)$ in disjunctive normal form. So the instance is not a member of Simple Planar 1–in–3–SAT either because the formula is not in 3–CNF.

Because these adjustments of ϕ to get a equisatisfiable formula for Simple Planar 1–in–3–SAT can be done in polynomial time (constant effort for each clause of ϕ) the problem is also NP–hard, and hence NP–complete. □

5.2. Planar Positive exactly 1–in–3–SAT

Let S be a point set in the plane. A *triangulation* of S is a maximal plane straight–line graph whose vertex set is S, i.e. adding an additional straight–



(a) Clause of an instance for Simple Planar 3–SAT.

(b) Equisatisfiable clauses for Simple Planar 1–in–3–SAT.

Figure 5.1.: Replacement of a clause $C = (a \vee b \vee c)$ for Simple Planar 3-SAT with three equisatisfiable clauses $C_1 = (a \vee w \vee x)$, $C_2 = (\neg b \vee w \vee y)$, and $C_3 = (\neg c \vee x \vee z)$ for Simple Planar 1–in–3–SAT.

line between two vertices makes the graph non–planar. For a triangulation T the *weight* of T is defined as the sum of the Euclidean length of all edges in T. Accordingly a *minimum-weight triangulation* of S is then a triangulation with the minimum weight of all possible triangulations for S. Computing such a minimum-weight triangulation of a given point set is shown to be NP–hard by Mulzer and Rote with a reduction from Planar Positive exactly 1–in–3–SAT [MR08].

Because of the open problem whether it is possible to compare the sums of radicals, e.g. Euclidean distances, in polynomial time [Blö91], it is not known whether the minimum-weight triangulation problem is in NP.

Definition 5.2.1. Planar Positive exactly 1–in–3–SAT[2] [MR08; Lar92]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. ϕ is positive.
- of the planar embedding of $G(\phi)$:

[2] In the paper of Mulzer and Rote the problem is called Positive Planar 1–in–3–SAT.

1. It is a rectilinear embedding.

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is only satisfied if exactly one literal is true.

Theorem 5.2.1. *Planar Positive exactly 1–in–3–SAT is NP–complete.*

Proof according to [MR08]. Planar Positive exactly 1–in–3–SAT is in NP because it is in polynomial time verifiable that an instance is a positive exactly 3–CNF formula and the given embedding is rectilinear, i.e. it is a valid instance, and that a given assignment is satisfying according to the definition of 1–in–3–SAT, i.e. it is a valid certificate.

With a reduction from Planar 3–SAT (Definition 4.1.1) it is shown that the problem is NP–hard. Let a planar Boolean formula ϕ and a rectilinear embedding of the associated graph $G(\phi)$ be an instance of Planar 3–SAT. Without loss of generality all clauses in ϕ are of size at least two [4.1.3].

In a first step, all clauses of size two are replaced in a way such that the formula contains only clauses with three literals. Let $C = (l_1 \vee l_2)$ be a clause with two literals. Then C is removed from the formula and replaced by two new clauses $(l_1 \vee l_2 \vee x)$ and $(l_1 \vee l_2 \vee \neg x)$. After this step it is easily seen that the old and new formula is equisatisfiable and can still be embedded in a rectilinear fashion.

For the next steps, namely the elimination of negated variables and the transformation of clauses to equisatisfiable 1–in–3–SAT clauses, two gadgets which enforce equality and inequality of two variables are used. The inequality gadget for two variables x and y is of the form

$$x \neq y \equiv (x \vee a \vee y) \wedge (a \vee b \vee c) \wedge (a \vee c \vee d) \wedge (b \vee c \vee d) \tag{5.1}$$

with new introduced variables a, b, c, d. The last three clauses forces the variable a to be set to 0 in an satisfying 1–in–3–SAT assignment. It follows that in an satisfying assignment for this gadget either x or y is set to 1. The equality gadget is then a combination of two inequality gadgets with an auxiliary variable a in between to enforce equality of two variables x and y, and has the form

$$x = y \equiv x \neq a \wedge a \neq y. \tag{5.2}$$

In a second step all negated variables are eliminated by an appropriate use of the inequality gadget. Let x be a variable with negated occurrences. Then x is replaced by a chain of variables $x_1.x_2, \ldots$ with alternating truth values according to the connections in the embedding of the associated graph. In this way the planarity of the drawing is kept and the formula is still equisatisfiable.

In the third and last step all clauses, except the clauses introduced with the gadgets, are transformed to an equisatisfiable representation according to 1–in–3–SAT. Let $C = (x \vee y \vee z)$ be a clause with three literals. Then C is replaced by

$$(x \vee u \vee a) \wedge (y \vee u \vee b) \wedge (a \vee b \vee q) \wedge (u = c) \wedge (d \neq z) \wedge (c \vee d \vee r). \tag{5.3}$$

The last three constraints can be reduced to the satisfiability of $u \Rightarrow z$. So, two cases have to be considered. The first case is if $u = 1$ and hence $z = 1$. To get a satisfying assignment for the other three clauses x, y, a, and b are set to 0 and q to 1. The second case is if $u = 0$ for which the first three clauses are reduced to the satisfiability of $\neg a \vee \neg b \Leftrightarrow x \vee y$ with the value of z arbitrarily chosen. Hence for all possible satisfying assignments of the original clause there exists a valid 1–in–3–SAT assignment of the replacement.

Because of the additional clauses it is necessary to add two auxiliary variables x' and z' together with equality gadgets for $x = x'$ and $z = z'$ to ensure that the value of x and z is reachable by other clauses in the rectilinear embedding. The variable x' is then placed to the left and z' to right of y. Then the clauses above the variables and between x and y can be nested between x' and y and clauses between y and z can correspondingly nested between y and z' (Figure 5.2).

All this transformations can be computed in polynomial time because there is only a constant number of alternations for each clause necessary. Furthermore all changes ensures a positive Boolean formula with a rectilinear embedding. Finally both formulas are equisatisfiable according to their problem definition. Hence Planar Positive exactly 1–in–3–SAT is NP–hard, and therefore NP–complete. □

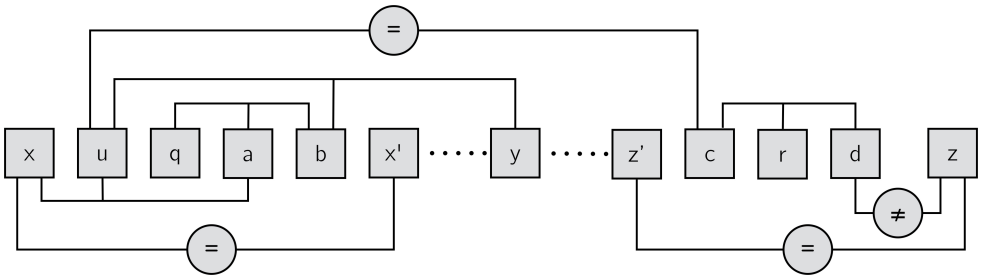


Figure 5.2.: Replacement of a clause $(x \vee y \vee z)$ by equisatisfiable clauses according to the restrctions of Planar Positive exactly 1–in–3–SAT [MR08].

5.3. Simple Planar Monotone exactly 3–bounded 1–in–3–SAT

Moore and Robson show that the problem of tiling a finite region with a given set of tiles is NP–complete even for right tromino and square tetromino on the square lattice, or for the right tromino alone [MR01]. The NP–completeness of the case for the right tromino alone is based on a reduction from this variant of Simple Planar Monotone 1–in–3–SAT.

Simple Planar Monotone 1–in–3–SAT

Definition 5.3.1. Simple Planar Monotone 1–in–3–SAT[3] [Lar92]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is monotone.
- of the associated graph:
  1. $G(\phi)$ has no edges between variable–vertices.

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is satisfied if exactly one literal is true.

Theorem 5.3.1. *Simple Planar Monotone 3–SAT is NP–complete* [Lar92].

[3] In the paper of Laroche the problem is called Planar Monotone 1-in-3 SAT.

Definition and Theorem

Definition 5.3.2. Simple Planar Monotone exactly 3–bounded 1–in–3–SAT[4] [MR01]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is monotone
  2. Each variable occurs in exactly three clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between variable–vertices.

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is satisfied if exactly on literal is true.

Theorem 5.3.2. *Simple Planar Monotone exactly 3–bounded 1–in–3–SAT is NP–complete* [MR01].

Proof sketch according to [MR01]. The problem is in NP because it is in polynomial time verifiable that the Boolean formula is monotone, each variable occurs in exactly three clauses, the planar embedding matches the definition of the associated graph, and the certificate (satisfying assignment for the Boolean formula) is valid according to the restriction of 1–in–3–SAT. The reduction is from Simple Planar Monotone 1–in–3–SAT (Definition 5.3.1) and proceeds in two steps. In the first step, the Boolean formula is transformed in an equisatisfiable and still planar formula where each variable appears at most three times. In the second step, the formula is expanded with new clauses and auxiliary variables to get a formula where each variable appears exactly three times. □

5.4. Separable Simple Planar 1–in–3–SAT

The paper "On strongly planar 3SAT" is dedicated to prove NP–completeness for some variants of Separable Simple Planar 3–SAT (Definition 4.5.1) [Wu15]. This variant has the restriction that a clause is only satisfied if exactly one literal is true.

[4] In the paper of Moore and Robson the problem is called Cubic Planar Monotone 1–in–3 SAT.

Definition 5.4.1. Separable Simple Planar 1–in–3–SAT[5] [Wu15]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the associated graph:
  1. Each variable–vertex is replaced by two literal–vertices with an edge connecting them, and edges to the corresponding clause–vertices.
  2. $G(\phi)$ has no other edges.

Question: Is ϕ with the following restrictions satifiable?

1. A clause is satisfied if exactly one literal is true.

Theorem 5.4.1. *Separable Simple Planar 1–in–3–SAT is NP–complete* [Wu15].

Proof according to [Wu15]. Separable Simple Planar 1–in–3–SAT is in NP because an instance and a certificate can be verified in polynomial time, i.e. the Boolean formula is planar, the planar embedding fulfills the restrictions on the associated graph, and the assignment is satisfying according to the 1–in–3–SAT restriction.

The NP–hardness of this variant is shown with a reduction from Separable Simple Planar 3–SAT (Definition 4.5.1). First, each clause is replaced in the same way like in the NP–hardness proof for Simple Planar 1–in–3–SAT [5.1.1]. Because in this variant every variable of the Boolean formula is represented in the associated graph with two literal–vertices, these replacements may cause some crossings. In the next step, these crossings are eliminated with a crossover box that fullfils the required definition for an instance of Separable Simple Planar 1–in–3–SAT (for details see [Wu15]) and leads to an equisatisfiable planar Boolean formula. Therefore Separable Simple Planar 3–SAT is NP–hard, and hence NP–complete. □

[5] In the paper of Wu the problem is called Strongly Planar 1–in–3–SAT.

6. Planar not–all–equal 3–SAT

The not–all-equal 3–SAT problem is one of the main variants of 3–SAT that is used for proofs of NP–completeness. It took some time after Lichtenstein's presentation of Planar 3–SAT to get results of the planar version of this problem. Variants of Planar not–all–equal 3–SAT are rarely present in the literature. This is maybe in consequence of Planar not–all–equal 3–SAT being not NP–complete but in P.

6.1. Planar not–all–equal 3–SAT

Surprisingly Moret has shown that Planar not–all–equal 3–SAT is in P [Mor88].

Definition 6.1.1. Planar not–all–equal 3–SAT [Mor88]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is satisfied if at least one literal is true and at least one literal is false.

Theorem 6.1.1. *Planar not–all–equal 3–SAT is in P.*

Moret made a reduction to the Planar Maximum Cut problem to prove that the problem is in P [Mor88]. Let G be a planar undirected graph, and let $k > 0$ be an integer. The question of the *Planar Maximum Cut* problem is whether there exists a partition of the vertices of G into two subsets such that the number of edges with endpoints in both subsets is at least k. Orlova and Dorfman, and Hadlock have shown independently [Kam12] that there exist a polynomial-time algorithm for the planar version of Maximum Cut [OD72; Had75].

Proof according to [Mor88]. Let a planar Boolean formula ϕ with k clauses and a planar embedding of the associated graph $G(\phi)$ be an instance for Planar not–all–equal 3–SAT. Without loss of generality it can be assumed that all clauses are of size two or three. Otherwise the formula cannot be satisfied by any assignment because clauses of size one are never satisfiable according to the not–all–equal restriction. To transform this instance to one for the Planar Maximum Cut problem the following steps are performed. First each variable–vertex x is replaced by a cycle of $2n_x$ vertices and correspondingly $2n_x$ edges, where n_x is the number of appearances of x in ϕ. If a variable appears only once then there is just one edge connecting the variable-vertices. In this cycle alternating vertices represent an assignment of 0 and 1 to the variable. Each clause–vertex is replaced either with a triangle if the represented clause is of size three or with two vertices and a connecting edge otherwise. The vertices of the clause components are connected by an edge to the corresponding variable–vertices such that an assignment with that value evaluates the literal to 0. The minimum number of cut edges are set to

$$7k + 4t - s, \tag{6.1}$$

where s is the number of variables with single occurrence, and t is the number of clauses with exactly three literals. This completes the transformation of an instance from Planar not–all-equal 3–SAT to Planar Maximum Cut, which can be done in polynomial time (example Figure 6.1).

For a given satisfying assignment all literal-vertices corresponding to the value of the assignment are put on one side of the partition together with the vertices of the clause triangle that are satisfied by it. It follows that all

$$2 \cdot 3t + 2 \cdot 2(k - t) - s = 4k + 2t - s \tag{6.2}$$

edges between a variable and its negation are cut. Furthermore all

$$3t + 2(k - t) = t + 2k \tag{6.3}$$

edges between the clause components and the variable cycles are cut. These are

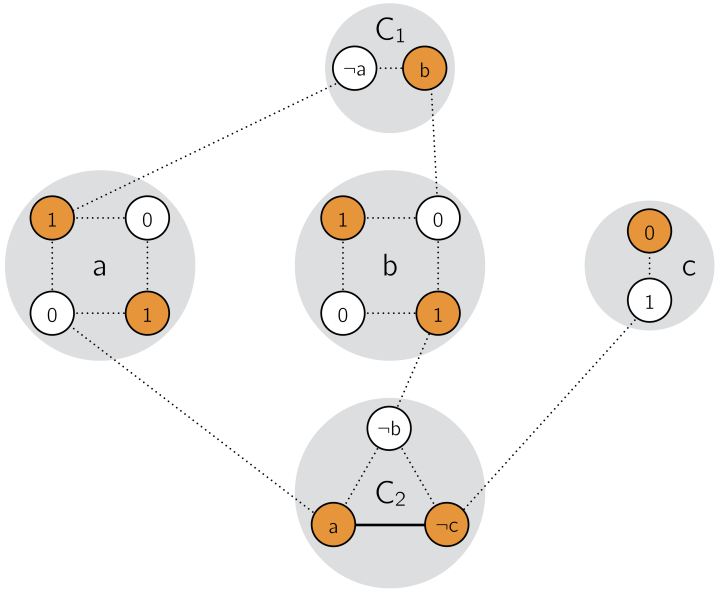


Figure 6.1.: Example of a Boolean formula $\phi = (\neg a \vee b) \wedge (a \vee \neg b \vee \neg c)$ with $k = 2, s = 1, t = 1$ clauses and a possible transformation for the Planar Maximum Cut problem. For a given satisfying assignment $a = 1, b = 1, c = 0$ for ϕ the colored vertices are put in one set of the partition and the dotted lines separate them from the other set. There are exactly $7k + 2t - s = 17$ edges in the cut. The gray cycles represent the original vertex in the associate graph. It is easily seen that the planarity constraint is still valid after the transformation.

cut because by construction the connections represent assignments such that the literal of the clause is not satisfied. But all clause vertices in the set are satisfied by the assignment. Then each clause triangle adds two more edges and the other clause components exactly one edge to the cut because it is a satisfying not–all–equal assignment, i.e. exactly one or two literals of each clause is satisfied. This adds

$$2t + (k - t) = t + k \tag{6.4}$$

edges to the cut which now has exactly

$$(4k + 2t - s) + (t + 2k) + (t + k) = 7k + 4t - s \tag{6.5}$$

edges as desired.

A solution for Planar Maximum Cut of size $11k - s - 2t$ yields to a satisfying assignment for Planar not–all-equal 3–SAT. This is because it is not possible to cut even more edges than all except one edge of each triangle component by putting one of these vertices on one side of the partition and the other two on the other side. Otherwise no edge of the triangle would be in the cut, hence the solution for Maximum Cut would be smaller. So a satisfying assignment for Planar not–all–equal 3–SAT can be computed from a solution for Planar Maximum Cut of size $11k - s - 2t$.

It follows that Planar not–all–equal 3–SAT is polynomial time reducible to Planar Maximum Cut, and is therefore in polynomial time solvable. □

Remark. Moret have made the reduction to Planar Maximum Cut only for Planar not–all–equal 3–SAT instances where the Boolean formula has exactly three literals per clause[Mor88].

6.2. Restricted Planar Positive not–all–equal 3–SAT

This variant of the Planar not–all–equal 3–SAT problem is not in P like the nonrestricted version but NP–complete. Dehghan used this fact to show that a modification of the Gap Vertex-Distinguishing Edge Coloring [TDK12] problem

is also NP–complete. The modified problem is defined as follows.

Let $G = (V, E)$ be a graph, k be a positive integer, and $f : V \to \{1, 2, \ldots, k\}$ be a mapping. Then the labeling $\mathcal{L}$ for every vertex in V induced by f is defined as

$$\mathcal{L}(v) = \begin{cases} 1 & \text{if the degree of } v \text{ equals 0,} \\ f(u)_{(u,v)\in E} & \text{if the degree of } v \text{ equals 1,} \\ \max_{(u,v)\in E} f(u) - \min_{(u,v)\in E} f(u) & \text{otherwise.} \end{cases}$$

The mapping f is called *vertex–labeling by gap* if no adjacent vertices have the same label. The problem in focus of Dehghan is to decide, given a planar bipartite graph G and an integer $k = 2$, whether there exists a vertex–labeling by gap such that the induced labeling $\mathcal{L}$ is a proper vertex 2–coloring of G. The NP–completeness of this problem is proved with a reduction from Restricted Planar Positive not–all–equal 3–SAT [Deh15].

Definition 6.2.1. Restricted Planar Positive not–all-equal 3–SAT[1] [Deh15]

Instance: A planar Boolean formula ϕ, a subset P of the variables in ϕ, and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is positive.

Question: Is ϕ with the following restrictions satisfiable?

1. A clause is satisfied if at least one literal is true and at least one literal is false.
2. Every variable in P is assigned with true.

Theorem 6.2.1. *Restricted Planar Positive not–all-equal 3–SAT is NP–complete.*

Proof according to [Deh15]. The problem is in NP because, given a certificate consisting of an assignment and the embedding of the associated graph, it can be verified in polynomial time that the graph is planar and it is a satisfying assignment according to the restrictions of this variant of Planar 3–SAT.

[1] In the paper of Dehghan the problem is called Restricted Planar Monotone Not–All-Equal 3SAT.

With a reduction from Separable Simple Planar 3–SAT (Definition 4.5.1) it is shown that the Restricted Planar Positive not–all–equal 3–SAT problem is NP–hard. Let ϕ be the Boolean formula of an instance for Separable Simple Planar 3–SAT over n variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ and with m clauses $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$. This formula will be converted into an instance for Restricted Planar Positive not–all–equal 3–SAT.

First, let $P = \emptyset$. Then ϕ, P and the planar embedding of the associated graph is an instance for Restricted Planar Positive not–all–equal 3–SAT, which is obviously equisatisfiable because P is the empty set.

Second, each clause $C = (l_1 \vee l_2 \vee l_3) \in \mathcal{C}$ is replaced by two clauses $(l_1 \vee l_2 \vee u)$ and $(l_3 \vee \neg u \vee \neg v)$, where u and v are new variables. Additionally v is added to P. With this conversation the new formula and P, and the original formula are equisatisfiable according to their problem definitions because of Lemma 6.2.2.

Lemma 6.2.2. *Each replacement of a clause $C = (l_1 \vee l_2 \vee l_3)$ in Step 2 yields a equisatisfiable formula according to the problem definitions, i.e. C is satisfied $\iff (l_1 \vee l_2 \vee u) \wedge (l_3 \vee \neg u \vee \neg v)$, with $v \in P$ is satisfied, where u, v are new variables.*

Proof. With a truth table it is easily seen that the formulas are equisatisfiable.

"⇒" Because v is a member of P the value of v in an satisfying assignment is always 1.

l_1	l_2	l_3	u	$(l_1 \vee l_2 \vee u) \wedge (l_3 \vee \neg u \vee \neg v)$
0	0	0	0 or 1	0
0	0	1	1	1
0	1	0	0	1
0	1	1	0 or 1	1
1	0	0	0	1
1	0	1	0 or 1	1
1	1	0	0	1
1	1	1	0	1

"⇐" A satisfying assignment for the new clauses yields a satisfying assignment for C.

Case $u = 0$**:** Six satisfying assignments with $u = 0$ are possible.

$(l_1 \vee l_2 \vee u) \wedge (l_3 \vee \neg u \vee \neg v)$	l_1	l_2	l_3	$(l_1 \vee l_2 \vee l_3)$
1	0	1	0 or 1	1
1	1	0	0 or 1	1
1	1	1	0 or 1	1

Case $u = 1$**:** Five satisfying assignments with $u = 1$ are possible.

$(l_1 \vee l_2 \vee u) \wedge (l_3 \vee \neg u \vee \neg v)$	l_1	l_2	l_3	$(l_1 \vee l_2 \vee l_3)$
1	0	0	1	1
1	0	1	0 or 1	1
1	1	0	0 or 1	1

Hence a satisfying assignment for the previous formula yields a satisfying assignment for the new formula, and vice versa, and therefore the formulas are equisatisfiable. □

The new formula is still planar because all replacements are just local adjustments around the old clauses which can be drawn independently of the other parts of the graph. With this conversation the new formula with the planar embedding of the associated graph and P is a valid instance for Restricted Planar Positive not–all–equal 3–SAT except that negations are still present.

In the final step, all negation are eliminated to get a final instance of Restricted Planar Positive not–all–equal 3–SAT. For each variable $x \in \mathcal{X}$ replace each occurrence of $\neg x$ in ϕ' with the new variable $y_{\neg x}$, add to the new formula the clauses $(x \vee y_{\neg x} \vee p_x)$, $(x \vee y_{\neg x} \vee q_x)$ and $(q_x \vee p'_x \vee p''_x)$, and add p_x, p'_x, and p''_x to the subset P. Because of the restrictions for a satisfying assignment of this problem this transformation leads to an assignment such that $\neg y_{\neg x}$ is always equivalent to x. Accordingly the formulas are still equisatisfiable. Furthermore the new formula is also still planar for the same reason as said in the previous step.

The Boolean formula of an instance that have not the properties of Separable Simple Planar 3–SAT, e.g. additional edges in the embedding of the associated graph, is expanded by the clause $(x \wedge y)$. This ensures that the instance is either

a member of Separable Simple Planar 3–SAT nor of this variant because the formula is not in 3–CNF.

It follows that an instance for Separable Simple Planar 3–SAT can be reduced in polynomial time to Restricted Positive Planar not–all-equal 3–SAT because for each clause can be replaced in polynomial time. Hence this problem is NP-hard and therefore NP–complete. □

7. Planar Monotone 3–SAT

Darmann, Döcker, and Dorn show NP–completeness for six variants of Planar Monotone 3–SAT with bounded variable appearances. They assume that this variants can be used as a starting point for reductions to prove that other decision problems are NP–hard [DDD16]. The NP–completeness of a restricted variant of Planar Monotone 3–SAT is shown by Berg and Khosravi [BK10]. A corollary of this proof is that Planar Monotone 3–SAT is also NP–complete.

7.1. Planar Monotone 3–SAT

Planar Monotone 3–SAT is a restricted version of Planar 3–SAT to monotone Boolean formulas. Because of the work of Berg and Khosravi [BK10] the NP–completeness of this variant is easily shown.

Definition 7.1.1. Planar Monotone 3–SAT

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is monotone.

Question: Is ϕ satisfiable?

Corollary 1 (of Theorem 7.2.1)**.** *Planar Monotone 3–SAT is NP–complete.*

Proof. Planar Monotone 3–SAT is in NP because it is in polynomial time verifiable that an instance satisfies all properties of the definition, and whether a certificate is valid.

Every instance of Restricted Planar Monotone 3–SAT (Definition 7.2.1) is without any effort reduced to Planar Monotone 3–SAT. To ensure equisatisfiability the Boolean formula of each instance that do not fulfill the properties

of the definition is expanded with a nonmonotone clause $(x \vee \neg y)$. Thus Restricted Planar Monotone 3–SAT is polynomial time reducible to Planar Monotone 3–SAT, and hence NP–hard. It follows that Planar Monotone 3–SAT is NP–complete. □

7.2. Restricted Planar Monotone 3–SAT

Restricted Planar Monotone 3–SAT is a variant of Planar 3–SAT with a restriction to planar monotone Boolean formulas. A *monotone rectilinear representation* of such a formula has all negative clauses on one side of the variables and all positive clauses on the other side.

Definition 7.2.1. Restricted Planar Monotone 3–SAT[1] [BK10]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is monotone.
- of the planar embedding of $G(\phi)$:
  1. It is a rectilinear embedding.
  2. All positive clauses are drawn on one side of the variables, and all negative clauses are drawn on the other side.

Question: Is ϕ satisfiable?

Theorem 7.2.1. *Restricted Planar Monotone 3–SAT is NP–complete.*

Proof according to [BK10]. It is in polynomial time verifiable that the formula is monotone, the embedding matches the definition of the associated graph, and all positive clauses are drawn on one side of the variables and all negative clauses are drawn on the other side. It is also in polynomial time verifiable whether a given assignment is satisfiable. Hence Restricted Planar Monotone 3–SAT is in NP.

To show the NP–hardness of the problem a reduction from Planar 3–SAT with a rectilinear embedding is described [BK10]. Let ϕ be a planar Boolean formula with m clauses over a set of n variables and with a rectilinear embedding. In the

[1] In the paper of Berg and Khosravi the problem is called Planar Monotone 3–SAT.

rectilinear embedding if a clause C on the positive side of the variables contains a negated variable then this pair of clause and variable is called *inconsistent*. A clause on the negative side of the variables with a positive literal is also called inconsistent. A representation without any inconsistent pairs is a rectilinear monotone representation. The goal is to transform ϕ into an equisatisfiable monotone Boolean formula without any inconsistent pairs.

To achieve this goal new variables and clauses are introduced. Let clause C and a negative literal $\neg x$ be an inconsistent pair of ϕ, i.e. C is drawn on the positive side of the variables but contains a negative literal. By introducing two new variables, a and b, and a modification of ϕ one inconsistent pair is eliminated as follows:

- Replace $\neg x$ by a in clause C.
- Add the clauses $(x \vee a) \wedge (\neg x \vee \neg a) \wedge (a \vee b) \wedge (\neg a \vee \neg b)$, hence $a = \neg x$ and $b = x$ to ϕ.
- Replace x by b in each clause containing x that is placed on the positive side of the variables and that connects x to the right of C (see Figure 7.1).

The new formula remains equisatisfiable to the original formula because the new clauses are only satisfiable when $a = \neg x$ and $b = x$, and the literal of the inconsistent pair is replaced accordingly.

The new clauses can be squeezed into the existing rectilinear representation without adding any crossing or violating any other constraints. Applying this transformation on every inconsistent pair of ϕ we get a monotone rectilinear representation for the Boolean formula. There are at most $3m$ inconsistent pairs which lead to a new formula with at most $13m$ clauses and at most $n+6m$ variables. So, the transformation can be done in polynomial time and hence, the problem is NP–hard.

It follows that Restricted Planar Monotone 3–SAT is NP–complete. □

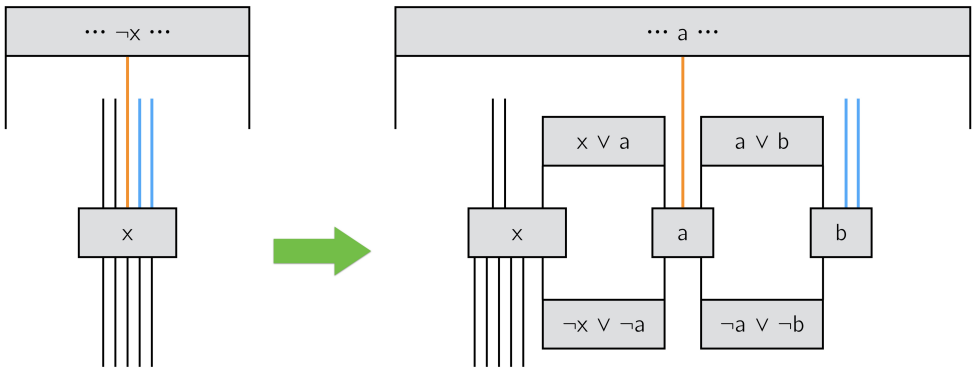


Figure 7.1.: Transformation of a clause C on the positive side that contains the negated variable x. Therefore $\neg x$ is replaced by the new variable a in C (see orange edge). The variable a is equivalent to $\neg x$ due the set of new clauses. This clauses can be squeezed into the embedding without adding any crossings. Furthermore the original value of x is still available through the new variable b for all clauses to the right (see blue edges).

7.3. Variable Bounded Variants of Simple Planar Monotone 3–SAT

This section is dedicated to the paper "On planar variants of the monotone satisfiability problem with bounded variable appearances" by Darmann, Döcker, and Dorn [DDD16] where several variants of Simple Planar Monotone 3–SAT are presented. All variants have in common that they are restricted to Boolean formulas with bounds on the number of variable appearance.

7.3.1. Simple Planar Monotone 3–bounded 3–SAT

This variant is the starting point to show NP–hardness for two additional variants of Simple Planar Monotone 3–SAT, namely Simple Planar Monotone exactly 3–bounded 3–SAT [7.3.2] and Simple Planar Monotone exactly 4–bounded exactly 3–SAT [7.3.5]. The NP–hardness of this variant is shown with a polynomial time reduction from Simple Planar exactly 3–bounded 3–SAT [8.4].

Definition 7.3.1. Simple Planar Monotone 3–bounded 3–SAT[2] [DDD16]

[2] In the paper of Darmann, Döcker, and Dorn the problem is called Planar Monotone (2,3)–SAT.

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula ϕ:
  1. ϕ is monotone.
  2. Each variable occurs in at most three clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Theorem 7.3.1. *Simple Planar Monotone 3–bounded 3–SAT is NP–complete.*

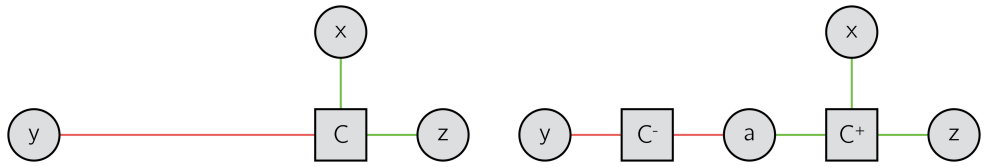
Proof according to [DDD16]. An instance of Simple Planar Monotone 3–bounded 3–SAT can be verfied in polynomial time, i.e. that the Boolean formula is monotone, each variable occurs in at most three clauses, and the embedding of the associated graph has no edges between the variable–vertices and corresponds to the definition of the graph. The verification of an assignment for the Boolean formula can also be done in polynomial time. Hence Simple Planar Monotone 3–bounded 3–SAT is in NP.

Simple Planar Monotone 3–bounded 3–SAT is NP–hard because a polynomial time reduction from Simple Planar exactly 3–bounded 3–SAT exists. If any variable of a Boolean formula of an instance for Simple Planar exactly 3–bounded 3–SAT does not appear exactly three times, then a nonmonotone clause $(x \vee \neg y)$ is added to the formula. This ensures that such instances are not accepted by the algorithm for Simple Planar Monotone 3–bounded 3–SAT either. Otherwise each mixed clause C is replaced by two clauses C^+ and C^-, where C^+ contains all positive literals of C and a new variable a_C, and C^- respectively all negative literals and $\neg a_C$. With this replacements equisatisfiablility is preserved and it is easy to see that new formula is still planar (See Figure 7.2).

Hence Simple Planar Monotone 3–bounded 3–SAT is NP–complete. □

7.3.2. Simple Planar Monotone exactly 3–bounded 3–SAT

This variant is a restriction of Simple Planar Monotone 3–bounded 3–SAT (Definition 7.3.1) such that each variable of the Boolean formula appears exactly



(a) Embedding of clause $C = (x \vee \neg y \vee z)$.

(b) Planarity preserving replacement of clause C by $C^- = (\neg y \vee \neg a)$ and $C^+ = (x \vee z \vee a)$.

Figure 7.2.: Equisatisfiable and planarity preserving replacement of a mixed clause.

three times. The NP–completeness of this variant is a corollary of the NP–completeness of Simple Planar Monotone 3–bounded 3–SAT (Theorem 7.3.1).

Definition 7.3.2. Simple Planar Monotone exactly 3–bounded 3–SAT[3] [DDD16]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula ϕ:
  1. ϕ is monotone.
  2. Each variable occurs in exactly three clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Corollary 2 (of Theorem 7.3.1)**.** *Simple Planar Monotone exactly 3–bounded 3–SAT is NP–complete.*

Proof according to [DDD16]. This variant is in NP for the same reasons Simple Planar Monotone 3–bounded 3–SAT [7.3.1] is in NP, and because it is in polynomial time verifiable that each variable appears exactly three times in the Boolean formula.

Simple Planar 3–bounded 3–SAT is polynomial time reducible to this varaint simply by adding the clauses $(x \vee a_x \vee b_x)$, $(a_x \vee b_x)$, and $(\neg a_x \vee \neg b_x)$ to the formula for each variable x that appears only two times (a_x, b_x are new auxiliary variables). It is obvious that the formula remains equisatisfiable because the

[3] In the paper of Darmann, Döcker, and Dorn the problem is called Planar Monotone (2,3)–SAT-E3.

clauses are always satisfiable by assigning $a_x = 1$ and $b_x = 0$, and planar because the new clauses and corresponding edges can be placed independently near the variable–vertex x. Hence Simple Planar Monotone exactly 3–bounded 3–SAT is NP–hard, and therefore NP–complete. □

7.3.3. Restricted Simple Planar Monotone [3,4]–bounded 3–SAT

Definition 7.3.3. Restricted Simple Planar Monotone [3,4]–bounded 3–SAT[4] [DDD16]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula ϕ:
  1. ϕ is monotone.
  2. Each variable occurs in at least three and at most four clauses.
  3. Each variable appears negated exactly once.
  4. Each clause containing three literals is positive.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Theorem 7.3.2. *Restricted Simple Planar Monotone [3,4]–bounded 3–SAT is NP–complete* [DDD16].

7.3.4. Restricted Simple Planar Monotone exactly 4–bounded 3–SAT

Definition 7.3.4. Restricted Simple Planar Monotone exactly 4–bounded 3–SAT[5] [DDD16]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula ϕ:
  1. ϕ is monotone.

[4] In the paper of Darmann, Döcker, and Dorn the problem is called Restricted Planar Monotone (2,3)–SAT–4.

[5] In the paper of Darmann, Döcker, and Dorn the problem is called Restricted Planar Monotone (2,3)–SAT–E4.

2. Each variable occurs in exactly four clauses.
3. Each variable appears negated exactly once.
4. Each clause containing three literals is positive.

- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Corollary 3 (of Theorem 7.3.2)**.** *Restricted Simple Planar Monotone exactly 4–bounded 3–SAT is NP–complete* [DDD16].

7.3.5. Simple Planar Monotone exactly 4–bounded exactly 3*–SAT

This variant is special because it explicitly allows that a literal can occur more than once in the same clause, i.e. clauses are multi-sets (denoted with 3*–SAT).

Definition 7.3.5. Simple Planar Monotone exactly 4–bounded exactly 3*–SAT[6] [DDD16]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. ϕ is monotone.
  3. A clause may contain the same literal more than once.
  4. Each variable occurs in exactly four clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: is ϕ satisfiable?

Theorem 7.3.3. *Simple Planar Monotone exactly 4–bounded exactly 3*–SAT is NP–complete.*

Proof according to [DDD16]. This variant is in NP because all properties of the planar Boolean formula, the associated graph and the planar embedding for an instance, and a assignment for the formula can be verified in polynomial time.

[6] In the paper of Darmann, Döcker, and Dorn the problem is called Planar Monotone 3–SAT*–E4.

Simple Planar Monotone exactly 4–bounded exactly 3*–SAT is NP–hard because Simple Planar Monotone exactly 3–bounded 3–SAT (Definition 7.3.2) is polynomial time reduciable to this variant. For the reduction it is only necessary to replace each clause $C = (x \vee y)$ with only two literals by two clauses $(x \vee y \vee a_C)$ and $(\neg a_C \vee \neg a_C \vee \neg a_C)$, where a_C is a new variable. Additonally, for each variable x in the original Boolean formula the clauses $(x \vee b_x \vee c_x)$, $(b_x \vee b_x \vee b_x)$, and $(c_x \vee c_x \vee c_x)$ is added to the set of clauses. This ensures that each clause contains three literals and each variable appears exactly four times in the formula. The new formula is obviously equisatisfiable because the new clauses are independently from the other clauses always satisfiable. An invalid instance of Simple Planar Monotone exactly 3–bounded 3–SAT remains after the reduction invalid for this variant of Simple Planar Monotone 3–SAT because if some variable appears not exactly three times, it also appears not exactly four times in the new instance. The other properties are not effected by this reduction. For each variable and clause is only constant effort necessary. Hence the reduction can be done in polynomial time with respect to the input size. □

7.3.6. Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT

Definition 7.3.6. Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT[7] [DDD16]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. ϕ is monotone.
  3. Each variable occurs in exactly five clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.
  2. $G(\phi)$ is biconnected.

[7] In the paper of Darmann, Döcker, and Dorn the problem is called Planar Monotone 3–SAT*–E5.

Question: Is ϕ satisfiable?

Theorem 7.3.4. *Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT is NP–complete*[DDD16].

8. Variable Bounded Variants of Planar 3–SAT

Restricted variants of Planar 3–SAT with bounds on the number of variable appearances are used many times. These variants are good examples for the high adaptability of Planar 3–SAT to the needs of proving NP–hardness of many other problems. This strategy "to massage Boolean formulas into forms more easily reducible to the problem at hand" [Lic82] often made the NP–hardness proof simpler.

8.1. Planar 3–bounded 3–SAT

This variant of Planar 3–SAT is the simplest version of bounding the appearance for each variable. If each variable appears at most two times in the Boolean formula then is the problem solvable in polynomial time [Tov84]. It is also not possible, without loosing the property of NP–hardness, to restrict this variant to Boolean formulas in exaclty 3–CNF because then every instance of Planar exactly 3–bounded 3–SAT is satisfiable [Tov84].

Definition 8.1.1. Planar 3–bounded 3–SAT

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. Each variable occurs in at most three clauses.

Question: Is ϕ satisfiable?

Corollary 4 (of Theorem 8.3.1)**.** *Planar 3–bounded 3–SAT is NP–complete.*

Proof. Planar 3–bounded 3–SAT is in NP because the validity of an instance (Boolean formula in 3–CNF, each variable occurs in exactly three clauses), and

a certificate (assignment of the Boolean formula) can be verified in polynomial time.

This variant is also NP–hard because every instance of Planar exactly 3–bounded 3–SAT (Definition 8.2.1) is with little work in polynomial time reducible to Planar 3–bounded 3–SAT. For instances that do not meet the properties of Planar exactly 3–bounded 3–SAT, the corresponding Boolean formula is expanded with the clause $(a \wedge b)$. So, the modified instance is not a member of Planar 3–bounded 3–SAT either because the Boolean formula is not in 3–CNF. Thus the variant is NP–hard, and hence NP–complete.

□

8.2. Planar exactly 3–bounded 3–SAT

Planar exactly 3–bounded 3–SAT is appended to the list of variants of Planar 3–SAT. The NP–completeness of this variant is a corollary of Theorem 8.3.1 which states that the restricted version is also NP–complete.

Definition 8.2.1. Planar exactly 3–bounded 3–SAT

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. Each variable occurs in exactly three clauses.

Question: Is ϕ satisfiable?

Corollary 5 (of Theorem 8.3.1)**.** *Planar exactly 3–bounded 3-SAT is NP–complete.*

Proof. Planar exactly 3–bounded 3–SAT is in NP because it is in polynomial time verifiable that an instance has the required properties, and whether a given assignment is satisfying. It is also NP–hard because Restricted Planar exactly 3–bounded 3–SAT (Definition 8.3.1) is with little effort polynomial time reducible to Planar Exactly 3–bounded 3–SAT. Only instances that have not the necessary properties of Restricted Planar 3–bounded 3–SAT, e.g. a variable appears not exactly three time, are modified. This is done by expanding the Boolean formula with the clause $(a \wedge b)$ so that the formula is not in 3–CNF. Hence this variant is NP–complete. □

8.3. Restricted Planar exactly 3–bounded 3–SAT

Maňuch and Gaur have shown the NP–completeness of a special case of the *protein chain lattice fitting* problem by performing a reduction from a restricted version of Planar 3–SAT. This variant is very similar to Simple Planar exactly 3–bounded 3–SAT [8.4]. The algorithm for solving the protein chain lattice fitting problem is used for computing solutions for other related problems, e.g. the genetic protein folding algorithm [RS96].

Definition 8.3.1. Restricted Planar exactly 3–bounded 3–SAT [MG08]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. Each variable occurs in exactly three clauses, once negated and twice not negated.

Question: Is ϕ satisfiable?

Theorem 8.3.1. *Restricted Planar exactly 3–bounded 3–SAT is NP–complete.*

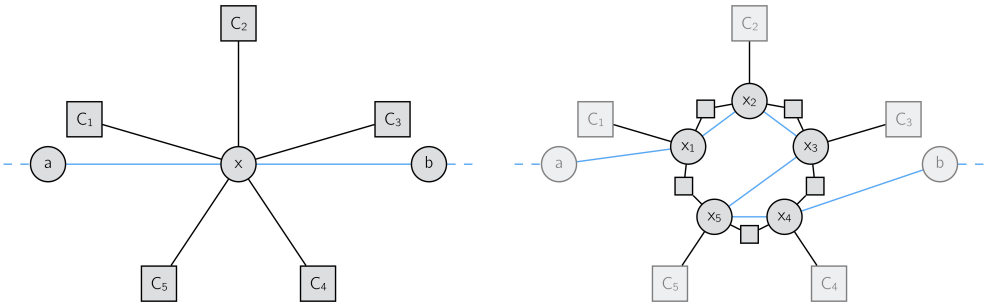
Proof. The validity of an instance can be verified in polynomial time by counting that each variable appears exactly three times (once negated, twice not negated), checking that the formula is in 3–CNF and that the embedding is planar and matches the definition of the associated graph. A certificate can also be verified in polynomial time, hence Restricted Planar exactly 3–bounded 3–SAT is in NP.

A polynomial time reduction from Planar 3–SAT (Definition 4.1.1) is described to show NP–hardness for this restricted version. Let a planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$ be an instance for Planar 3–SAT. The transformation of ϕ to an equisatisfiable formula for Restricted Planar exactly 3–bounded 3–SAT can be done in two steps.

First, each variable x of ϕ is replaced by k variables $x_1, \ldots, x_k$, where k is the number of appearances of x. To force that the new variables have all the same value in an satisfying assignment the following clauses are added:

$$(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge \cdots \wedge (x_k \vee \neg x_1)$$

Each appearance of x is replaced such that the associated graph is still planar



(a) Variable–vertex before the replacement.

(b) Variable–vertex is replaced by a set of new variables and clauses.

Figure 8.1.: Example of a possible planar embedding of a Boolean formula after replaceing a variable x of an instance for Planar 3–SAT. Equisatisfiablility before (Figure 8.1a) and after (Figure 8.1b) the replacement can be archived.

and the new variables appear exactly three times in ϕ (see Figure 3.2.5). This ensures that the new and original formula are equisatisfiable.

Second, each variable that occurs only once not negated and twice negated is replaced by its negation. This replacement has no effect on the satisfiability of the formula.

This transformations can be done in polynomial time because the individual transformation of each variable is done in polynomial time. So Planar 3–SAT is polynomial time reducible to Restricted Planar exactly 3–bounded 3–SAT. Thus this variant is NP–hard, and hence NP–complete. □

Remark. This proof is a little bit different in comparison to the proof of Maňuch and Gaur because there is no distinction between left and right clauses according to the cycle through all variable–vertices [MG08]. Though, it is actually the same reduction only with different labels, and a planar embedding, which make the reduction a little bit simpler.

Akl et al. performed a reduction from this restricted version to a variant of the uniform Covering with Variable Capacities problem with fixed facilities to prove its NP–hardness [Akl+15].

8.4. Simple Planar exactly 3–bounded 3–SAT

Middendorf and Pfeiffer use this variant of Simple Planar 3–SAT to prove that the *Planar Vertex-Disjoint Paths* problem is NP–complete. An instance of this problem is a planar graph $G = (V, E)$ and a set of pairs $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. The question is whether there exists a set of pairwise disjoint paths between each pair of s_i and t_i, for $i = 1, \ldots, k$. Two paths are called *disjoint* if they do not share a vertex.

Dahlhaus et al. have used the same variant to show that a version of the Multiterminal Cut problem is also NP–complete[Dah+94]. The *Multiterminal Cut* problem is defined as follows: Let $G = (V, E)$ be a graph with a positive weight $w(e)$ for each edge $e \in E$, and let $T = \{t_1, t_2, \ldots, t_k\}$ be a subset of V. The vertices in T are called *terminals*. The goal is to find a set of edges $E' \subset E$ with minimum weight such that with the removal of E' from the graph G all terminals are separated from each other. The weight of an edge set E is the sum of $w(e)$ of each edge $e \in E$. For the variant of this problem there is also given a limit B such that the weight of E' has to be at most B, and the graph has to be planar with a maximum vertex degree of three.

Definition 8.4.1. Simple Planar exactly 3–bounded 3–SAT [MP93; Dah+94]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph, with the following properties

- of the planar Boolean formula:
  1. Each variable occurs in exactly three clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

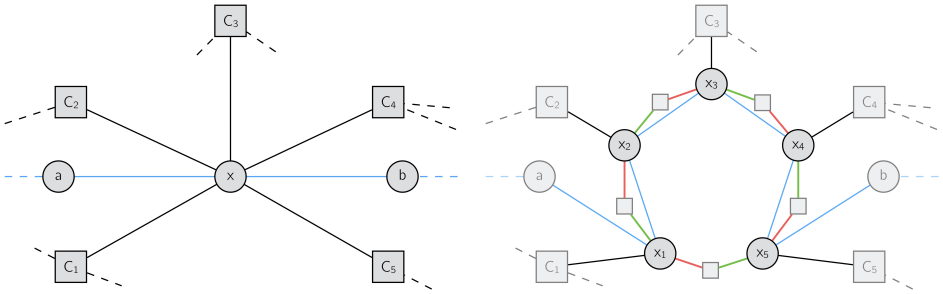
Question: Is ϕ satisfiable?

Remark. The only difference between the definition of Middendorf and Pfeiffer, and Dahlhaus et al. is that Dahlhaus et al. says precisely that for each variable one literal occurs once and the other twice. But this can be assumed without loss of generality [4.1.3].

Theorem 8.4.1. *Simple Planar exactly 3–bounded 3–SAT is NP–complete.*

Proof according to [MP93; Dah+94]. The validity of an instance can be verified in polynomial time, i.e. each variable appears in three clauses, each clause is of

size at most three, and the associated graph is planar. A given certificate, i.e. a satisfying assignment, can also be verified in polynomial time. Hence this variant is in NP.



(a) The variable-vertex x lies on the cycle through all variable–vertices (blue edges). It has edges to the corresponding clause–vertices of $C_1, \ldots, C_5$ where x appears in.

(b) The variable x is replaced by $x_1, \ldots, x_5$ in the according clauses. Additional clauses force all x_i to have the same value, for $i = 1, \ldots, 5$. In the associated graph the new clause–vertices and variable–vertices can be drawn at the previous place of x. Green edges represent a not negated occurrence of x in a clause, and red edges represent negated occurrences. This ensures planarity and equisatisfiability.

Figure 8.2.: Example for a transformation of a Boolean formula with a variable x that is contained in five clauses.

To show NP–hardness a reduction from Simple Planar 3–SAT (Definition 4.3.1) is described. Let ϕ be a planar Boolean formula and $G(\phi)$ the simple associated graph. For every variable x let $xC_1, xC_2, \ldots, xC_k$ be the edges between the variable–vertex x and the clauses containing x in $G(\phi)$, where k is the number of appearances of x. With the introduction of new variables $x_1, x_2, \ldots, x_k$ and clauses $(x_k \vee \neg x_1) \wedge (x_1 \vee \neg x_2) \wedge \cdots \wedge (x_{k-1} \vee \neg x_k)$ every occurrence of x or $\neg x$ in C_i can be replaced with x_i respectively $\neg x_i$ for $i = 1, 2, \ldots, k$. The additional clauses force every x_1 to x_k to have the same value, and to occur in exactly three clauses with both literals at least once. The planarity of this new formula ϕ' can be easily maintained (Figure 8.2) and it is clear that ϕ' is satisfiable if and only if ϕ is satisfiable which fulfills the reduction from Simple Planar 3–SAT to this variant.

It follows that Simple Planar exactly 3–bounded 3–SAT is NP–complete. □

8.5. Simple Planar [2,3]–bounded 3–SAT

Kobayashi, Miyamoto, and Tamaki show that the problem to decide whether a planar graph has a k–cyclic orientation is NP–complete for every fixed $k \geq 4$. For this, they make a reduction from this variant of Simple Planar 3–SAT which is only slightly different in comparison to Simple Planar exactly 3–bounded 3–SAT (Definition 8.4.1).

Let $G = (V, E)$ be an undirected graph, and let e be an edge between the vertices u and v of G. An *orientation* of the edge e is a directed edge (u, v) or (v, u). Accordingly an *orientation* of G is a directed graph with the vertices V and an orientation for each edge in E. The orientation of a graph is *k–cyclic*, for $k \geq 3$, if the orientation of every edge belongs to a cycle of length at most k.

Definition 8.5.1. Simple Planar [2,3]–bounded 3–SAT [KMT10]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. Each variable occurs in at least two and at most three clauses, once negated, and once or twice not negated.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Corollary 6 (of Theorem 8.4.1)**.** *Simple Planar [2,3]–bounded 3–SAT is NP–complete.*

Proof. It is easily seen that this variant is in NP because an instance of the problem and a certificate can be verified in polynomial time.

Because Planar 3–bounded 3–SAT (Definition 8.2.1) is NP–hard and can be reduced in polynomial time to this version without much effort, it is also NP–hard. This is done just by removing the edges between the variable–vertices of the associated graph. Additionally the Boolean formula of instances that do not have the properties of 3–bounded Planar 3–SAT, e.g. no edges between the variable–vertices, is expanded with the clause $(a \wedge b)$, where a, b are new

auxiliary variables. This guarantees that the instance is not a member Simple Planar [2,3]–bounded 3–SAT either. Hence, Simple Planar [2,3]–bounded 3–SAT is NP–complete. □

8.6. Simple Planar 1–negative [2,3]–bounded 3–SAT

Finding the metric dimension of a given graph is called the *Metric Dimension* problem. The *metric dimension* of a graph $G = (V, E)$ is the cardinality of a smallest subset L of V such that every pair of vertices from V is resolved by some vertex in L. A pair $u, v \in V$ is *resolved* by a vertex $l \in L$, also called *landmark*, if the shortest path from u to l and from v to l has not the same length. Such a set L of landmarks that resolves all pairs of vertices from V is called a *resolving set*. This problem is also known as *Harary's problem*, and as the *locating number* or *rigidity problem* which where defined independently [Dia+12] by the authors Harary and Melter [HM67], and Slater [Sla75].

The complexity of the Metric Dimension problem for a wide range of special graphs, e.g. general graphs [GJ79], sparse graphs[HSV10] or trees [KRR96; Sla75; HM67], where studied in the past. Diaz et al. have shown that the planar variant of this problem, i.e. finding the metric dimension of a given planar graph, is NP–hard by describing a reduction from the following variant of Planar 3–SAT [Dia+12].

Definition 8.6.1. Simple Planar 1–negative [2,3]–bounded 3–SAT[1] [Dia+12]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula ϕ:
  1. Each variable occurs in at least two and at most three clauses, once negated, and once or twice not negated.
  2. Each clause of size three contains at least one negated variable.
- of the associated graph $G(\phi)$:
  1. There are no edges between the variable–vertices.

Question: Is ϕ satisfiable?

[1] In the paper of Diaz et al. the problem is called 1–Negative Planar 3–SAT.

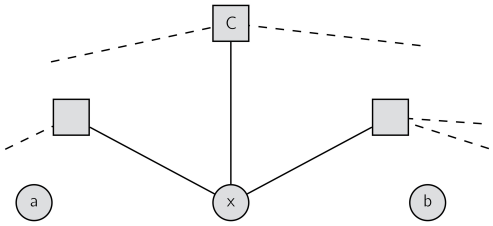
The original definition has an additional restriction so that each clause has two or three distinct variables. But this is not necessary because clauses are no multi sets, i.e. for every two literals of a clause they are not equal, and without loss of generality it can be assumed that not both literals of a variable are contained in the same clause [4.1.3]. Hence all clauses have only distinct variables.

Theorem 8.6.1. *Simple Planar 1–negative [2,3]–bounded 3–SAT is NP–complete.*

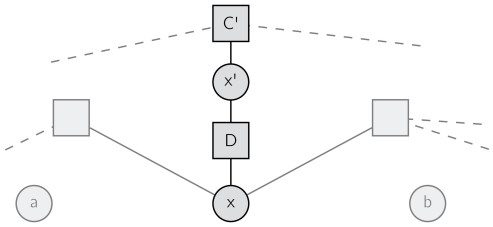
Proof according to [Dia+12]. For an instance of Simple Planar 1–negative [2,3]–bounded 3–SAT, consisting of a planar Boolean formula and an embedding of the associated graph, it can be verified in polynomial time that each variable appears three times (exactly once negated and at least once not negated), every clause contains at least one negative variable, and the embedding is planar and matches the definition of the associated graph. Additionally the validity of a certificate, i.e. a satisfying assignment, can also by verified in polynomial time. Thus, Simple Planar 1–negative [2,3]–bounded 3–SAT is in NP.

To show NP-hardness a reduction from Simple Planar exactly 3–bounded 3–SAT (Definition 8.4.1) is described. This two versions differ in that Simple Planar 1–negative [2,3]–bounded 3–SAT is restricted to Boolean formulas where each clause contains at least one negative variable as well. That this variant do not bound the number of appearances for each variable to exactly three is not of importance for the reduction because instances for exactly 3–bounded restricted problems are also valid instances for [2,3]–bounded restricted problems. So for the reduction only the clauses with three positive variables, i.e. positive monotone clauses, have to be eliminated. Let $C = (x \vee y \vee z)$ such a clause. Then by replacing this clause with the two clauses $D = (x \vee x')$ and $C' = (\neg x' \vee y \vee z)$, where x' is a new variable, one positive monotone clause is eliminated. To get a satisfying assignment of the new formula from one of the original formula the new variable x' have to be set to the value of $\neg x$ only. Given a satisfying assignment of the new formula a satisfying assignment of the original formula can be easily derived by just dropping the new variable x'. It follows that the formulas are equisatisfiable. The planarity of the associated graph can be preserved by replacing C with the clause–vertex of C', a component of the variable–vertex of

x' in line with the clause–vertex of $(x \vee x')$, and corresponding edges (Figure 8.3).



(a) Before the transformation the clause–vertex C has an edge to the variable–vertex of x.



(b) For the transformation v_C is replaced by the clause–vertex of the new clause $C' = (x' \vee y \vee z)$. The new vertices for the variable x', the clause $D = (x \vee x')$, and the new edges can be embedded without causing any intersections. No further adjustments have to be done.

Figure 8.3.: Example for a planar embedding before and after the elimination of a positive monotone clause $C = (x \vee y \vee z)$.

Performing the described transformation on every such clause leads to a equisatisfiable, planar Boolean formula. Each instance with a Boolean formula that has some variable that do not appears exactly three times, i.e. is not a valid instance for Simple Planar exactly 3–bounded 3–SAT, is alternated such that it is not a valid instance for this variant either. This is done with an extension of the Boolean formula with the clause $(a \wedge b)$ (a, b new auxiliary variables) which ensures that the formula is not in 3–CNF. Therefore such an instance is not a member of Simple Planar 1–negative [2,3]–bounded 3–SAT either. Thus Simple Planar 1–negative [2,3]–bounded 3–SAT is NP–hard, and hence NP–complete. □

8.7. Simple Planar [3,4]–bounded exactly 3–SAT

Let $G = (V, E)$ be a graph and let S be a subset of V. The subset S is called *stable* or *independent* if its vertices are pairwise non-adjacent. S is a *cutset* or *separator* of G if the deletion of the vertices of S from G results in a disconnected graph. A *stable cutset* of a graph is a subset of vertices which is stable and also a cutset. Le, Mosca, and Müller have shown that the Stable Cutset problem, i.e. does there exist a stable cutset in a given graph, remains NP–complete when restricted to subgraphs of triangulations with vertices of degree at most five [LMM05]. For this proof they use the following variant of Simple Planar 3–SAT [4.3].

Definition 8.7.1. Simple Planar [3,4]–bounded exactly 3–SAT[2] [LMM05]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. Each variable occurs in at least in three and at most in four clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.

Question: Is ϕ satisfiable?

Theorem 8.7.1. *Simple Planar [3,4]–bounded exactly 3–SAT is NP–complete.*

Proof. A given instance of this variant can be verified in polynomial time. For this, it is checked that each variable appears in at least three and at most four clauses, each clause is of size three, and the embedding of the associated graph is planar. It is also possible to verify in polynomial time that a given assignment is satisfying, i.e. the validity of a certificate. Hence, this variant of Planar 3–SAT is in NP.

With a reduction from Simple Planar 4–bounded exactly 3–SAT (Definition 8.8.1) it is shown that this variant is also NP–hard. The only difference between these problems is that by the restrictions of Le, Mosca, and Müller each variable

[2] In the paper of Le, Mosca, and Müller the problem is called Restricted Planar 3–SAT.

has to appear in at least three and at most four clauses. Simple Planar 4–bounded exactly 3–SAT has no lower bound on the number of appearances for each variable.

Let a planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$ be an instance for Simple Planar 4–bounded exactly 3–SAT. If an instance do not meet the restriction of Simple Planar 4–bounded exactly 3–SAT, e.g. a variable appears less then four times, then is the Boolean formula of that instance expanded with the clause $(a \vee b)$, where a, b are new auxiliary variables. That ensures that the instance is not valid for Simple Planar [3,4]–bounded exactly 3–SAT either. Otherwise, for the reduction it is only necessary to transform ϕ in such a way that each variable is contained in at least three clauses. Now, for each variable x with only one occurrence in ϕ the Boolean formula is expanded with

$$(x \vee a_x \vee b_x) \wedge (x \vee a_x \vee c_x) \wedge (x \vee b_x \vee c_x) \wedge (a_x \vee b_x \vee c_x), \tag{8.1}$$

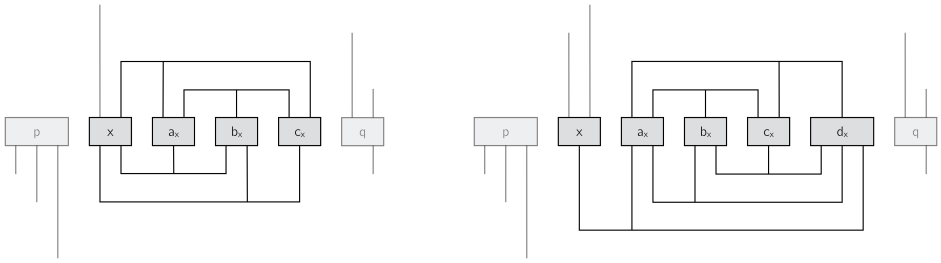
where a_x, b_x, c_x are newly introduced variables. For each variable x with only two occurrences ϕ is expanded with

$$(x \vee a_x \vee d_x) \wedge (a_x \vee b_x \vee c_x) \wedge (a_x \vee b_x \vee d_x) \wedge (a_x \vee c_x \vee d_x) \wedge (b_x \vee c_x \vee d_x), \tag{8.2}$$

where a_x, b_x, c_x, d_x are newly introduced variables. The resulting formula is equisatisfiable to ϕ because the new clauses are quasi-independent from the other clauses. They are always satisfied by assigning *true* to the new variables which leads to a remaining formula that is ϕ.

The resulting formula is still planar because the rectilinear embedding of (8.1) and (8.2) is planar and can be drawn directly next to the according variable–vertices as seen in Figure 8.4. For this a rectilinear embedding of $G(\phi)$ according to Theorem 4.1.2 is used.

The reduction can be done in polynomial time, because the formula is expanded with a constant number of clauses for each variable that appears not exactly three time. Thus Simple Planar [3,4]-bounded exactly 3–SAT is NP–hard, and hence NP–complete. □



(a) The variable x appears once in ϕ. The formula is then expanded with (8.1) to increase the number of clauses containing x to four.

(b) Here, variable x is contained in two clauses of ϕ. By adding (8.2) to the Boolean formula x appears now four times.

Figure 8.4.: Rectilinear embedding of the Boolean formulas (8.1) and (8.2).

8.8. Simple Planar 4–bounded exactly 3–SAT

A *communication network* is a set of *members* of the network that can communicate over *communication lines* connecting pairs of members. Such a network can be represented as a graph $G = (V, E)$ where the vertices and the edges of G represent the members and the communication lines. *Broadcasting* is the process to communicate a message originated at a set of members, also called *originators*, to every member of the communication network. The broadcasting should be achieved with as few as possible communication between the members. The following constraints are set for the communication of members in a communication network.

1. Each communication requires one time unit.
2. Each member can communicate with exactly one other member per time unit.
3. Each member can communicate with adjacent members only.

The *Minimum Broadcast Time* problem is: Given a communication network represented as a graph $G = (V, E)$ and a set $V_0 \subseteq V$ of originators, what is the minimum number of time units required for broadcasting.

Jansen and Müller have proved that variants of the Minimum Broadcast Time problem remain NP–complete [JM95]. The variants have restrictions on the network layout with a maximum broadcast time, also called *deadline*, of $k = 2$ time

units or with only one originator. The complexity results of these variants based on Simple Planar 4–bounded exactly 3–SAT are for network layouts of bipartite planar graphs, split graphs, chordal graphs, and grid graphs with maximum degree of at most 3, each with deadline $k = 2$.

Definition 8.8.1. Simple Planar 4–bounded exactly 3–SAT[3] [JM95]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. Each variable occurs in at most four clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between variable–vertices.

Question: Is ϕ satisfiable?

Theorem 8.8.1. *Simple Planar 4-bounded exactly 3–SAT is NP–complete.*

Proof according to [JM95]. Let a planar Boolean formula ϕ in exactly 3–CNF with an planar embedding of the associated graph $G(\phi)$ be an instance for Simple Planar 4–bounded exactly 3–SAT. It can be verified in polynomial time that each clause of ϕ has exactly three literals, each variable occurs in at most four clauses, and the associated graph has no edges between the variable–vertices and matches the definition of $G(\phi)$. Given an assignment for ϕ a verification algorithm can easily check in polynomial time whether the given assignment is satisfiable. Thus Simple Planar 4–bounded exactly 3–SAT is in NP.

It can be shown that the problem is also NP-hard with a polynomial time reduction from Simple Planar 3–SAT (Definition 4.3.1). In the first step, for every variable x that appears in $n_x > 3$ clauses a cycle of the form

$$(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \wedge \cdots \wedge (x_{n_x} \vee \neg x_1)$$

is added to the formula. Then for $i = 1, \ldots, n_x$ the i-th occurrence of x is replaced by x_i. The cycle is a constraint so that $x_1 = x_2 = \cdots = x_{n_x}$ and the x_i take over the role of the original variable x.

[3] In the paper of Jansen and Müller the problem is called Planar 3,4–SAT.

In the second, step every clause c with only two literals is expanded with a new literal $\neg x$ which is forced to be 0 by a set of clauses.

1. $(\neg a_i \vee b_i \vee d_i) \wedge (\neg a_i \vee \neg b_i \vee c_i) \wedge (\neg b_i \vee \neg c_i \vee d_i)$ for $i = 1, 2, 3$.
2. $(x \vee a_i \vee d_i)$ for $i = 1, 2, 3$.
3. $(\neg d_1 \vee \neg d_2 \vee \neg d_3)$.

It can easily be shown with a proof by contradiction that $\neg x = 0$, with the result that the new and old formula are equisatisfiable. The new clauses can be added to ϕ so that the associated graph is still planar [JM95]. This technique can be used twice to transform a single literal into a clause with exactly three literals.

It takes only polynomial time to transform ϕ into an instance ϕ' for Simple Planar 4–bounded exactly 3–SAT because each clause and each variable is only handled once with a polynomial amount of effort. Furthermore ϕ and ϕ' are equisatisfiable because the new clauses can be satisfied independently and they are forcing the additional literals of some clauses to be 0. Hence Simple Planar 4–bounded exactly 3–SAT is NP-hard, and therefore NP–complete. □

8.9. Simple Planar 3–connected exactly 3–SAT

Kratochvíl shows NP-hardness for String Graph Recognition [Kra91] with a reduction from Abstract Topological Graph Realizability (AT–graph Realizability). This problem is NP–hard because this variant of Simple Planar 3–SAT is polynomial time reducible to AT–graph Realizability. Though, at that time it was not proved that Simple Planar 3–SAT is NP–hard [Kra94]. But the NP–hardness of this problem is a corollary of Kratochvíl's later proof that a restricted version of this is NP–hard [Kra94].

A graph G is a *string graph* if and only if there exists a set R of curves in the plane, the so called *strings*, such that the intersection graph of R is isomorphic to G. The *intersection graph* $\mathcal{I}(R)$ of these strings has a vertex for each string, and an edge between two vertices if the representing strings have a non-empty intersection. The *String Graph Recognition* problem is whether a given graph G is a string graph.

A graph G with a given embedding in the plane is called a *topological graph* [KLN91b] and is denoted by G^T. An *abstract topological graph* is a tuple of a

graph G and a set of intersections $\mathcal{I}$ of the edges of G. Such a graph $(G, \mathcal{I})$ is *realizable* if there exists a topological layout G^T of G where the intersection graph $\mathcal{I}(G^T)$ is equal to the set of intersections $\mathcal{I}$.

Hence, an instance of the *Abstract Topological Graph Realizability* problem is an abstract topological graph $(G, \mathcal{I})$ and the question is whether $(G, \mathcal{I})$ is realizable.

Only in 2003 Schaefer, Sedgwick, and Štefankovič showed that Abstract Topological Graph Realizability problem is in NP [SSŠ03] and therefore is NP–complete.

Definition 8.9.1. Simple Planar 3–connected exactly 3–SAT[4] [Kra91]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.
  2. $G(\phi)$ is 3–connected.

Question: Is ϕ satisfiable?

Corollary 7 (of Theorem 8.9.1)**.** *Simple Planar 3–connected exactly 3–SAT is NP–complete.*

Proof. An instance of Simple Planar 3–connected exactly 3–SAT is in polynomial time verifiable, i.e. the Boolean formula is in exactly 3–CNF, and the embedding of the associated graph is planar, matches the definition and is 3–connected[5]. An assignment for the Boolean formula is checked in polynomial time whether it is satisfying. Hence this variant is in NP.

Each instance of 4–bounded Simple Planar 3–connected exactly 3–SAT (Definition 8.9.2) is an instance of this variant as well. Further, the Boolean formula of every instance with some variable that occurs more than 4 times is appended by the clauses $(a \vee b)$, where a and b are new variables. In this way it is ensured that the transformed instance for Simple Planar 3–connected exactly 3–SAT is

[4] In the paper of Kratochvíl the problem is called Planar 3–connected 3–SAT.
[5] 3–connectivity can be tested in linear time[Sch11].

also not accepted because the Boolean formula is either in exactly 3–CNF nor the associated graph is 3–connected. Thus the restricted version is polynomial time reducible to this variant. Hence Simple Planar 3–connected exactly 3–SAT is NP–hard, and therefore NP–complete.

□

8.9.1. Simple Planar 3–connected 4–bounded exactly 3–SAT

Definition 8.9.2. Simple Planar 3–connected 4–bounded exactly 3–SAT[6] [Kra94]

Instance: A planar Boolean formula ϕ and a planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. ϕ is in exactly 3–CNF.
  2. Each variable occurs in at most 4 clauses.
- of the associated graph:
  1. $G(\phi)$ has no edges between the variable–vertices.
  2. $G(\phi)$ is 3–connected.

Question: Is ϕ satisfiable?

Theorem 8.9.1. *Simple Planar 3–connected 4–bounded exactly 3–SAT is NP–complete* [Kra94].

8.10. Restricted Clause–Linked Planar exactly 3–bounded 3–SAT

Definition 8.10.1. Restricted Clause–Linked Planar exactly 3–bounded 3–SAT[7] [Fel+95]

Instance: A planar Boolean formula ϕ and planar embedding of the associated graph $G(\phi)$, with the following properties

- of the planar Boolean formula:
  1. Each clause of size three is positive.

[6] In the paper of Kratochvíl the problem is called 4–Bounded Planar 3–connected 3–SAT.
[7] In the paper of Fellows et al. the problem is called Restricted Clause–Linked Planar 3–SAT.

2. Each variable occurs in exactly three clauses, once negated and twice not negated.

- of the associated graph:
  1. $G(\phi)$ has no edges between variable–vertices.
  2. The set of clauses allows a linear ordering such that $G(\phi)$ is still planar when consecutive clause–vertices are connected with an edge.

Question: Is ϕ satisfiable?

Theorem 8.10.1. *Restricted Clause–Linked Planar exactly 3–bounded 3–SAT is NP–complete* [Fel+95].

9. Conclusion

9.1. Remarks

Some problems, e.g. Clause–Linked Planar 3–SAT (Definition 4.6.1), are mentioned only with their definition and complexity. In this way it is possible to obtain a comprehensive picture of the wide variety of Planar 3–SAT, even when they are not described as detailed as the other problems.

9.2. Open Problems

There are some interesting problems that remain open. One of these is a detailed comparison between NP-hardness proofs for properties of planar graphs using Planar 3–SAT (or some variant) and another known NP–hard problem. A good candidate for this comparison is the Triangulation Existence Problem. An instance of this problem is a geometric graph $G = (V, E)$ and the question is whether a triangulation $T = (V, E_T \subseteq E)$ exists. In 1977 Lloyd showed the NP–hardness of the Triangulation Existence Problem with a reduction from SAT for Boolean formula in conjunctive normal form [Llo77]. According to Schulz the construction for the NP–hardness proof is easier with a reduction from Simple Planar 3-SAT [Sch06]. The new proof is presented alongside with a similar proof showing the NP–completeness of the Pseudo–Triangulations Existence Problem [Sch06]. A polygon with exactly three convex corners is called a *pseudo–triangle*, and a *pseudo-triangulation* is a planar partition of a point set into pseudo-triangles. Other good candidates are Planar Node Cover, Planar Directed Hamiltonian Circuits, and Geometric Connected Dominating Set with alternative proofs given by Lichtenstein [Lic82].

It is not clear whether Planar 3–SAT and Simple Planar 3–SAT are the same problem, i.e. if the definition and properties of the associated graph are inter-

changeable. The main difference between the two problems is, that for Planar 3–SAT a circular order of the variable–vertices is necessary such that consecutive vertices are connected by an edge. Whereas for Simple Planar 3–SAT such an ordering is not necessary which may lead to Boolean formula that have a planar embedding according to Simple Planar 3–SAT but not to Planar 3–SAT. In contrast to that it is shown in Chapter 4 that the definition and properties of the associated graph of Planar 3–SAT and Rectilinear Planar 3–SAT are interchangeable (Theorem 4.1.2).

In 1978, Schaefer showed that the problem of deciding whether a propositional formula in CNF is satisfiable is either in P or NP–complete [Sch78]. Later, when a classification of different problems in P were developed *Schaefer's dichotomy theorem* was refined by Allender et al. [All+09]. The refinement takes into account that if AC^0 isomorphisms are considered then the Boolean Constraint Satisfaction Problem (Boolean CSP) can be classified according to the complexity classes NP, P, $\oplus$L, NL, and L. This dichotomy theorem is also valid for planar Boolean formulas but an open problem is whether it is possible to extend the result for this special group. One general approach makes Dvořák and Kupec with a partial classification of the complexity of Planar Boolean CSP [DK15].

A similar approach could be to *close some more gaps* (more than in this thesis are already closed) between the variants, e.g. if a variant is NP–complete for monotone Boolean formula, is a variant with the same restriction except that the Boolean formula is positive also NP–complete? Many problems are variants of Simple Planar 3–SAT but not necessarily for Planar 3–SAT too. Studying the same restrictions for Planar 3–SAT as well could give some insights into the previous mentioned open question whether Planar 3–SAT and Simple Planar 3–SAT are the same problem or not.

Appendices

A. List of Variants of Planar 3–SAT

All variants of PLANAR 3–SAT in this thesis are here listed in order of their appearance. New variants are marked with ν.

1. 4.1.1 - Planar 3–SAT
2. 4.1.1 - Rectilinear Planar 3–SAT
3. 4.2.1 - Planar exactly 3–SAT$^{\nu}$
4. 4.3.1 - Simple Planar 3–SAT$^{\nu}$
5. 4.4.1 - Separable Planar 3–SAT
6. 4.5.1 - Separable Simple Planar 3–SAT
7. 4.6.1 - Clause–Linked Planar 3-SAT
8. 5.1.1 - Simple Planar 1–in–3–SAT
9. 5.2.1 - Planar Positive exactly 1–in–3–SAT
10. 5.3.1 - Simple Planar Monotone 1–in–3–SAT
11. 5.3.2 - Simple Planar Monotone exactly 3–bounded 1–in–3–SAT
12. 5.4.1 - Separable Simple Planar 1–in–3–SAT
13. 6.1.1 - Planar not–all–equal 3–SAT
14. 6.2.1 - Restricted Planar Positive not–all–equal 3–SAT
15. 7.1.1 - Planar Monotone 3–SAT$^{\nu}$
16. 7.2.1 - Restricted Planar Monotone 3–SAT
17. 7.3.1 - Simple Planar 3–bounded Monotone 3–SAT
18. 7.3.2 - Simple Planar Monotone exactly 3–bounded 3–SAT

19. 7.3.3 - Restricted Simple Planar Monotone [3,4]–bounded 3–SAT
20. 7.3.4 - Restricted Simple Planar Monotone exactly 4–bounded 3–SAT
21. 7.3.5 - Simple Planar Monotone exactly 4–bounded exactly 3*–SAT
22. 7.3.6 - Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT
23. 8.1.1 - Planar 3–bounded 3–SAT$^{\nu}$
24. 8.2.1 - Planar exactly 3–bounded 3–SAT$^{\nu}$
25. 8.3.1 - Restricted Planar exactly 3–bounded 3–SAT
26. 8.4.1 - Simple Planar exactly 3–bounded 3–SAT
27. 8.5.1 - Simple Planar [2,3]–bounded 3–SAT
28. 8.6.1 - Simple Planar [2,3]–bounded 1–negative 3–SAT
29. 8.7.1 - Simple Planar [3,4]–bounded exactly 3–SAT
30. 8.8.1 - Simple Planar 4–bounded exactly 3–SAT
31. 8.9.1 - Simple Planar 3–connected exactly 3–SAT
32. 8.9.2 - Simple Planar 3–connected 4–bounded exactly 3–SAT
33. 8.10.1 - Restricted Clause–Linked exactly 3–bounded Planar 3–SAT

B. Restrictions on Planar 3–SAT

B.1. List of Restrictions

Let ϕ be a planar Boolean formula with a planar embedding of the associated graph $G(\phi)$. Then the following restriction can be put on an instance for Planar 3–SAT.

1. Restrictions on the planar Boolean formula:
   a) ϕ is in exactly 3–CNF.
   b) ϕ is monotone.
   c) ϕ is positive (respectivly negative).
   d) Each variable appears negated exactly once.
   e) Bounded variable appearances:
      i. Each variable occurs in at least two and at most three clauses, once negated, and once or twice not negated.
      ii. Each variable occurs in at most three clauses.
      iii. Each variable occurs in exactly three clauses.
      iv. Each variable occurs in exactly three clauses, once negated and twice not negated.
      v. Each variable occurs in at least three and at most four clauses.
      vi. Each variable occurs in at most four clauses.
      vii. Each variable occurs in exactly four clauses.
      viii. Each variable occurs in exactly five clauses.
   f) Each clause with three literals is positive.
   g) Each clause with three literals contains at least one negated variable.
   h) A clause may contain the same literal more than once.
2. Restrictions on the associated graph:
   a) $G(\phi)$ has no edges between the variable–vertices.

   b) $G(\phi)$ is biconnected.
   c) $G(\phi)$ is 3–connected.
   d) At each variable–vertex all edges representing a positive literal are incident to one side of the vertex, and all edges representing a negative literal are incident to the other side. The separation is according to the edges of the cycle through all variable–vertices.
   e) The set of clauses allows a linear ordering such that $G(\phi)$ is still planar when consecutive clause–vertices are connected with an edge.
3. Restrictions on the planar embedding of $G(\phi)$:
   a) It is a rectilinear embedding[1].
   b) All positive clauses are drawn on one side of the variables, and all negative negative clauses are drawn on the other side
4. Restrictions on the satisfying assignment:
   a) A clause is satisfied if exactly one literal is true.
   b) A clause is satisfied if at least one literal is true and at least one literal is false.
   c) Let P be a subset of variables in ϕ. Each variable in P is assigned with true.

B.2. Categorization of Planar 3–SAT Variants

Each of the presented variants of Planar 3–SAT can be put in some categories for possible restrictions. The category for restrictions on the planar Boolean formula is dominated by variants with bounded variable appearances. The restriction on the associated graph for Simple Planar versions is the restriction for most variants in this category. Restrictions on the planar embedding of the associated graph have only two problems, though a rectilinear embedding is no restriction at all (See Theorem 4.1.2). So, Restricted Planar Monotone 3–SAT (Definition 7.2.1) is the only proper member of this category. The reason for such a small number of variants in this category is not clear. One possibility could be that in practice variants of Simple Planar versions are capable enough. This impression is supported by the great presence of nineteen Simple Planar

[1] A rectilinear embedding is only an assumption and not a restriction (See Theorem 4.1.2)

versions against all thirty-three presented variants in this thesis. The category for restrictions on the satisfying assignment can be partitioned into two sets, that is in variants for Planar 1–in–3–SAT and Planar not–all–equal 3–SAT. Only one variant, namely Restricted Planar Positive not–all–equal 3–SAT (Definition 6.2.1), has additionally a special restriction on the satisfying assignment such that each element of a subset of variables in the Boolean formula have to be assigned with true.

1. Restrictions on the planar Boolean formula:
    - Planar exactly 3–SAT [4.2]
    - Planar Positive exactly 1–in–3–SAT [5.2.1]
    - Simple Planar Monotone 1–in–3–SAT [5.3.1]
    - Simple Planar Monotone exactly 3–bounded 1–in–3–SAT [5.3.2]
    - Restricted Planar Positive not–all–equal 3–SAT [6.2.1]
    - Planar Monotone 3–SAT [7.1.1]
    - Restricted Planar Monotone 3–SAT [7.2.1]
    - Simple Planar Monotone 3–bounded 3–SAT [7.3.1]
    - Simple Planar Monotone exactly 3–bounded 3–SAT [7.3.2]
    - Restricted Simple Planar Monotone $[3, 4]$–bounded 3–SAT [7.3.3]
    - Restricted Simple Planar Monotone exactly 4–bounded 3–SAT [7.3.4]
    - Simple Planar Monotone exactly 4–bounded exactly 3*–SAT [7.3.5]
    - Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT [7.3.6]
    - Planar 3–bounded 3–SAT [8.1.1]
    - Planar exactly 3–bounded 3–SAT [8.2.1]
    - Restricted Planar exactly 3–bounded 3–SAT [8.3.1]
    - Simple Planar exactly 3-bounded 3–SAT [8.4.1]
    - Simple Planar $[2, 3]$–bounded 3–SAT [8.5.1]
    - Simple Planar 1–negative $[2, 3]$–bounded 3–SAT [8.6.1]
    - Simple Planar exactly $[3, 4]$–bounded 3–SAT [8.7.1]
    - Simple Planar 4–bounded exactly 3–SAT [8.8.1]
    - Simple Planar 3–connected exactly 3–SAT [8.9.1]
    - Simple Planar 3–connected 4–bounded exactly 3–SAT [8.9.2]
    - Restricted Clause–Linked exactly 3–bounded Planar 3–SAT [8.10.1]

2. Restrictions on the associated graph:
    - Simple Planar 3–SAT [4.3.1]
    - Separable Planar 3–SAT [4.4.1]
    - Separable Simple Planar 3–SAT [4.5.1]
    - Clause–Linked Planar 3–SAT [4.6.1]
    - Simple Planar 1–in–3–SAT [5.1.1]
    - Planar Positive exactly 1–in–3–SAT [5.2.1]
    - Simple Planar Monotone 1–in–3–SAT [5.3.1]
    - Simple Planar Monotone exactly 3–bounded 1–in–3–SAT [5.3.2]
    - Separable Simple Planar 1–in–3–SAT [5.4.1]
    - Restricted Planar Monotone 3–SAT [7.2.1]
    - Simple Planar Monotone 3–bounded 3–SAT [7.3.1]
    - Simple Planar Monotone exactly 3–bounded 3–SAT [7.3.2]
    - Restricted Simple Planar Monotone $[3, 4]$–bounded 3–SAT [7.3.3]
    - Restricted Simple Planar Monotone exactly 4–bounded 3–SAT [7.3.4]
    - Simple Planar Monotone exactly 4–bounded exactly 3^*–SAT [7.3.5]
    - Restricted Simple Planar Monotone exactly 5–bounded exactly 3–SAT [7.3.6]
    - Simple Planar exactly 3-bounded 3–SAT [8.4.1]
    - Simple Planar $[2, 3]$–bounded 3–SAT [8.5.1]
    - Simple Planar 1–negative $[2, 3]$–bounded 3–SAT [8.6.1]
    - Simple Planar $[3, 4]$–bounded exactly 3–SAT [8.7.1]
    - 4–bounded Simple Planar exactly 3–SAT [8.8.1]
    - Simple Planar 3–connected exactly 3–SAT [8.9.1]
    - Simple Planar 3–connected 4–bounded exactly 3–SAT [8.9.2]
    - Restricted Clause–Linked exactly 3–bounded Planar 3–SAT [8.10.1]
3. Restrictions on the planar embedding of $G(\phi)$:
    - Planar Positive exactly 1–in–3–SAT [5.2.1]
    - Restricted Planar Monotone 3–SAT [7.2.1]
4. Restrictions on the satisfying assignment:
    - Simple Planar 1–in–3–SAT [5.1.1]
    - Planar Positive exactly 1–in–3–SAT [5.2.1]
    - Simple Planar Monotone 1–in–3–SAT [5.3.1]

- Simple Planar Monotone exactly 3–bounded 1–in–3–SAT [5.3.2]
- Separable Simple Planar 1–in–3–SAT [5.4.1]
- Planar not–all–equal 3–SAT [6.1.1]
- Restricted Planar Positive not–all–equal 3–SAT [6.2.1]

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[Akl+15] S. Akl, R. Benkoczi, D. R. Gaur, H. Hassanein, S. Hossain, and M. Thom. "On a class of covering problems with variable capacities in wireless networks". In: *Theoretical Computer Science* 575 (2015). Special Issue on Algorithms and Computation, pp. 42–55. DOI: 10.1016/j.tcs.2014.10.044.

[All+09] E. Allender, M. Bauland, N. Immerman, H. Schnoor, and H. Vollmer. "The Complexity of Satisfiability Problems: Refining Schaefer's Theorem". In: *J. Comput. Syst. Sci.* 75.4 (June 2009), pp. 245–254.

[BK10] M. Berg and A. Khosravi. "Computing and Combinatorics: 16th Annual International Conference, COCOON 2010, Nha Trang, Vietnam, July 19-21, 2010. Proceedings". In: ed. by M. T. Thai and S. Sahni. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Chap. Optimal Binary Space Partitions in the Plane, pp. 216–225. DOI: 10.1007/978-3-642-14031-0_25.

[Blö91] J. Blömer. "Computing sums of radicals in polynomial time". In: *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*. IEEE. 1991, pp. 670–677.

[BM04] J. M. Boyer and W. J. Myrvold. "On the Cutting Edge: Simplified O(n) Planarity by Edge Addition". In: *Journal of Graph Algorithms and Applications* 8.3 (2004), pp. 241–273. DOI: 10.7155/jgaa.00091.

[Cob64] A. Cobham. "The Intrinsic Computational Difficulty of Functions, Y. Bar-Hillel". In: *Proceedings of the 1964 Congress on Logic, Methodology and Philosophy of Science*. 1964.

[Coo71] S. A. Cook. "The Complexity of Theorem-proving Procedures". In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. STOC '71. Shaker Heights, Ohio, USA: ACM, 1971, pp. 151–158. DOI: 10.1145/800157.805047.

[Coo72] S. A. Cook. "A Hierarchy for Nondeterministic Time Complexity". In: *Proceedings of the Fourth Annual ACM Symposium on Theory of Computing*. STOC '72. Denver, Colorado, USA: ACM, 1972, pp. 187–192. DOI: 10.1145/800152.804913.

[Cor+09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. Cambridge, Massachusetts: The MIT Press, 2009.

[Dah+94] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. "The Complexity of Multiterminal Cuts". In: *SIAM Journal on Computing* 23 (1994), pp. 864–894.

[DB97] B. Das and V. Bharghavan. "Routing in ad-hoc networks using minimum connected dominating sets". In: *Communications, 1997. ICC'97 Montreal, Towards the Knowledge Millennium. 1997 IEEE International Conference on*. Vol. 1. IEEE. 1997, pp. 376–380.

[DDD16] A. Darmann, J. Döcker, and B. Dorn. "On planar variants of the monotone satisfiability problem with bounded variable appearances". In: *CoRR* abs/1604.05588 (2016).

[Deh15] A. Dehghan. "On strongly planar not-all-equal 3SAT". In: *Journal of Combinatorial Optimization* (2015), pp. 1–4. DOI: 10.1007/s10878-015-9894-6.

[DF86] M. Dyer and A. Frieze. "Planar 3DM is NP-complete". In: *Journal of Algorithms* 7.2 (1986), pp. 174–184. DOI: 10.1016/0196-6774(86)90002-7.

[Dia+12] J. Diaz, O. Pottonen, M. Serna, and E. J. van Leeuwen. "On the Complexity of Metric Dimension". In: *Proceedings of the 20th Annual European Conference on Algorithms*. ESA'12. Ljubljana, Slovenia: Springer-Verlag, 2012, pp. 419–430. DOI: 10.1007/978-3-642-33090-2_37.

[Dji95] H. N. Djidjev. "On drawing a graph convexly in the plane (extended abstract)". In: *Graph Drawing: DIMACS International Workshop, GD '94 Princeton, New Jersey, USA, October 10–12, 1994 Proceedings*. Ed. by R. Tamassia and I. G. Tollis. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 76–83. DOI: 10.1007/3-540-58950-3_358.

[DK15] Z. Dvořák and M. Kupec. "On Planar Boolean CSP". In: *Automata, Languages, and Programming: 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*. Ed. by M. M. Halldórsson, K. Iwama, N. Kobayashi, and B. Speckmann. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 432–443. DOI: 10.1007/978-3-662-47672-7_35.

[DKW02] D.-Z. Du, K. Ko, and J. Wang. *Introduction to Computational Complexity Theory*. Chinese. High Education Publisher, Beijing, 2002.

[Edm65] J. Edmonds. "Paths, Trees, and Flows". In: *Canadian Journal of Mathematics* 17 (1965), pp. 449–467. DOI: 10.4153/CJM-1965-045-4.

[Fel+95] M. R. Fellows, J. Kratochvil, M. Middendorf, and F. Pfeiffer. "The complexity of induced minors and related problems". In: *Algorithmica* 13.3 (1995), pp. 266–282. DOI: 10.1007/BF01190507.

[Gib+08] M. Gibson, G. Kanade, E. Krohn, I. A. Pirwani, and K. Varadarajan. "Algorithm Theory – SWAT 2008: 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2-4, 2008. Proceedings". In: ed. by J. Gudmundsson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. Chap. On Metric Clustering to Minimize the Sum of Radii, pp. 282–293. DOI: 10.1007/978-3-540-69903-3_26.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. 1979.

[GMM13] L. Guibas, N. Milosavljević, and A. Motskin. "Connected dominating sets on dynamic geometric graphs". In: *Computational Geometry* 46.2 (2013), pp. 160–172.

[Had75] F. Hadlock. "Finding a Maximum Cut of a Planar Graph in Polynomial Time". In: *SIAM Journal on Computing* 4.3 (1975), pp. 221–225. DOI: 10.1137/0204019.

[HM67] F. Harary and R. Melter. "The metric dimension of a graph". In: *Ars Combinatoria* 2 (1967), pp. 191–195.

[HS65] J. Hartmanis and R. E. Stearns. "On the computational complexity of algorithms". In: *Transactions of the American Mathematical Society* 117 (1965), pp. 285–306.

[HS66] F. C. Hennie and R. E. Stearns. "Two-Tape Simulation of Multitape Turing Machines". In: *J. ACM* 13.4 (Oct. 1966), pp. 533–546. DOI: 10.1145/321356.321362.

[HS78] E. Horowitz and S. Sahni. *Fundamentals of computer algorithms*. Computer Science Press, 1978.

[HSV10] M. Hauptmann, R. Schmied, and C. Viehmann. "On approximation complexity of metric dimension problem". In: *International Workshop on Combinatorial Algorithms*. Springer. 2010, pp. 136–139.

[HT08] B. Haeupler and R. E. Tarjan. "Planarity Algorithms via PQ-Trees (Extended Abstract)". In: *Electronic Notes in Discrete Mathematics* 31 (2008), pp. 143–149. DOI: 10.1016/j.endm.2008.06.029.

[HT74] J. Hopcroft and R. Tarjan. "Efficient Planarity Testing". In: *J. ACM* 21.4 (Oct. 1974), pp. 549–568. DOI: 10.1145/321850.321852.

[Hu74] T. C. Hu. "Optimum communication spanning trees". In: *SIAM Journal on Computing* 3.3 (1974), pp. 188–195.

[JLK78] D. S. Johnson, J. K. Lenstra, and A. Kan. "The complexity of the network design problem". In: *Networks* 8.4 (1978), pp. 279–285.

[JM95] K. Jansen and H. Müller. "The minimum broadcast time problem for several processor networks". In: *Theoretical Computer Science* 147.1–2 (1995), pp. 69–85. DOI: 10.1016/0304-3975(94)00230-G.

[Kam12] M. Kamiński. "MAX-CUT and containment relations in graphs". In: *Theoretical Computer Science* 438 (2012), pp. 89–95. DOI: 10.1016/j.tcs.2012.02.036.

[Kar72] R. M. Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[KLN91a] J. Kratochvíl, A. Lubiw, and J. Nešetřil. "Noncrossing Subgraphs in Topological Layouts". In: *SIAM Journal on Discrete Mathematics* 4.2 (1991), pp. 223–244. DOI: 10.1137/0404022.

[KLN91b] J. Kratochvíl, A. Lubiw, and J. Nešetřil. "Noncrossing Subgraphs in Topological Layouts". In: *SIAM Journal on Discrete Mathematics* 4.2 (1991), pp. 223–244. DOI: 10.1137/0404022.

[KM05] C. Knauer and W. Mulzer. "Minimum dilation triangulations". In: (2005).

[KMT10] Y. Kobayashi, Y. Miyamoto, and H. Tamaki. "Algorithms and Computation: 21st International Symposium, ISAAC 2010, Jeju, Korea, December 15-17, 2010, Proceedings, Part II". In: ed. by O. Cheong, K.-Y. Chwa, and K. Park. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. Chap. k-cyclic Orientations of Graphs, pp. 73–84. DOI: 10.1007/978-3-642-17514-5_7.

[Koz11] L. Kozma. "Minimum Average Distance Triangulations". In: *CoRR* abs/1112.1828 (2011).

[KR93] D. E. Knuth and A. Raghunathan. "The problem of compatible representatives". In: *eprint arXiv:cs/9301116* (June 1993).

[Kra91] J. Kratochvíl. "String graphs. II. recognizing string graphs is NP-hard". In: *Journal of Combinatorial Theory, Series B* 52.1 (1991), pp. 67–78. DOI: 10.1016/0095-8956(91)90091-W.

[Kra94] J. Kratochvíl. "A special planar satisfiability problem and a consequence of its NP-completeness". In: *Discrete Applied Mathematics* 52.3 (1994), pp. 233–252. DOI: 10.1016/0166-218X(94)90143-0.

[KRR96] S. Khuller, B. Raghavachari, and A. Rosenfeld. "Landmarks in graphs". In: *Discrete Applied Mathematics* 70.3 (1996), pp. 217–229. DOI: 10.1016/0166-218X(95)00106-2.

[Lar92] P. Laroche. "Planar 1-in-3 satisfiablility is NP-complete". In: *Comptes Rendus de I Académie des Sciences* (Jan. 1992).

[Lev73] L. A. Levin. "Universal Sequential Search Problems". Russian. In: *Probl. Peredachi Inf.* 9 (3 1973), pp. 115–116.

[Lic77] D. Lichtenstein. *A technique for proving NP- completeness results on planar graphs.* 1977., 1977.

[Lic82] D. Lichtenstein. "Planar Formulae and Their Uses". In: *SIAM J. Comput.* 11.2 (1982), pp. 329–343.

[Llo77] E. L. Lloyd. "On triangulations of a set of points in the plane". In: *Foundations of Computer Science, 1977., 18th Annual Symposium on*. Oct. 1977, pp. 228–240. DOI: 10.1109/SFCS.1977.21.

[LMM05] V. B. Le, R. Mosca, and H. Müller. "Graph-Theoretic Concepts in Computer Science: 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers". In: ed. by D. Kratsch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. Chap. On Stable Cutsets in Claw-Free Graphs and Planar Graphs, pp. 163–174. DOI: 10.1007/11604686_15.

[MG08] J. Maňuch and D. R. Gaur. "Fitting Protein Chains to Cubic Lattice is NP-complete". In: *Journal of Bioinformatics and Computational Biology* 06.01 (2008), pp. 93–106. DOI: 10.1142/S0219720008003308.

[Mor88] B. M. E. Moret. "Planar NAE3SAT is in P". In: *j-SIGACT* 19.2 (June 1988), pp. 51–54.

[MP09] J. S. B. Mitchell and E. Packer. *On Non-crossing (Projected) Spanning Trees of 3D point sets*. 2009.

[MP93] M. Middendorf and F. Pfeiffer. "On the complexity of the disjoint paths problem". In: *Combinatorica* 13.1 (1993), pp. 97–107. DOI: 10.1007/BF01202792.

[MR01] C. Moore and J. M. Robson. "Hard Tiling Problems with Simple Tiles". In: *Discrete Comput. Geom.* 26.4 (Jan. 2001), pp. 573–590. DOI: 10.1007/s00454-001-0047-6.

[MR08] W. Mulzer and G. Rote. "Minimum-weight Triangulation is NP-hard". In: *J. ACM* 55.2 (May 2008), 11:1–11:29. DOI: 10.1145/1346330.1346336.

[MW00] P. Mutzel and R. Weiskircher. "Computing Optimal Embeddings for Planar Graphs". In: *Computing and Combinatorics: 6th Annual International Conference, COCOON 2000 Sydney, Australia, July 26–28, 2000 Proceedings*. Ed. by D.-Z. Du, P. Eades, V. Estivill-Castro, X. Lin, and A. Sharma. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 95–104. DOI: 10.1007/3-540-44968-X_10.

[OD72] G. Orlova and Y. Dorfman. "Finding the Maximum Cut in a Graph". In: *engineering cybernetics*. Vol. 10. 1972, pp. 502–506.

[RS96] A. A. Rabow and H. A. Scheraga. "Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator". In: *Protein Science* 5.9 (1996), pp. 1800–1815.

[Sch06] A. Schulz. "The Existence of a Pseudo-triangulation in a given Geometric Graph". In: *Proceedings of the 22nd European Workshop on Computational Geometry*. Mar. 2006, pp. 17–20.

[Sch11] J. M. Schmidt. "Structure and Constructions of 3–ConnectedGraphs". dissertation. Freie Universität Berlin, 2011.

[Sch78] T. J. Schaefer. "The Complexity of Satisfiability Problems". In: *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*. STOC '78. San Diego, California, USA: ACM, 1978, pp. 216–226. DOI: 10.1145/800133.804350.

[SFM78] J. I. Seiferas, M. J. Fischer, and A. R. Meyer. "Separating Nondeterministic Time Complexity Classes". In: *J. ACM* 25.1 (Jan. 1978), pp. 146–167. DOI: 10.1145/322047.322061.

[Sip06] M. Sipser. *Introduction to the Theory of Computation*. Vol. 2. Thomson Course Technology Boston, 2006.

[Sla75] P. J. Slater. "Leaves of trees". In: *Congr. Numer* 14.549-559 (1975), p. 37.

[SSŠ03] M. Schaefer, E. Sedgwick, and D. Štefankovič. "Recognizing string graphs in NP". In: *Journal of Computer and System Sciences* 67.2 (2003). Special Issue on STOC 2002, pp. 365–380. DOI: 10.1016/S0022-0000(03)00045-X.

[TDK12] M. A. Tahraoui, E. Duchêne, and H. Kheddouci. "Gap vertex-distinguishing edge colorings of graphs". In: *Discrete Mathematics* 312.20 (2012), pp. 3011–3025.

[Tov84] C. A. Tovey. "A simplified NP-complete satisfiability problem". In: *Discrete Applied Mathematics* 8.1 (1984), pp. 85–89. DOI: 10.1016/0166-218X(84)90081-7.

[Wie47] H. Wiener. "Structural determination of paraffin boiling points". In: *Journal of the American Chemical Society* 69.1 (1947), pp. 17–20.

[Wu15] L. Wu. "On strongly planar 3SAT". In: *Journal of Combinatorial Optimization* (2015), pp. 1–6. DOI: 10.1007/s10878-015-9878-6.

[WW99] S. Wei-Kuan and H. Wen-Lian. "A new planarity test". In: *Theoretical Computer Science* 223.1 (1999), pp. 179–191. DOI: 10.1016/S0304-3975(98)00120-0.

[Žák83] S. Žák. "A Turing machine time hierarchy". In: *Theoretical Computer Science* 26.3 (1983), pp. 327–333. DOI: 10.1016/0304-3975(83)90015-4.