On the Complexity of Modified Instances

Dissertation
zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat)

vorgelegt dem Rat der
Fakultät für Mathematik und Informatik
der Friedrich Schiller Universität Jena

von Diplom-Informatiker Tobias Berg
geboren am 28. Dezember 1978 in Jena

Gutachter:

1. PD Dr. Harald Hempel, Friedrich-Schiller-Universität Jena
2. Prof. Dr. Ulrich Hertrampf, Universität Stuttgart
3. Prof. Dr. Rolf Niedermeier, Friedrich-Schiller-Universität Jena

Tag der letzten Prüfung des Rigorosums: 22.12.2008
Tag der öffentlichen Verteidigung: 07.01.2009

Zusammenfassung

Ein wesentlicher Beitrag, den die Komplexitätstheorie zum besseren Verständnis von Algorithmen leisten kann, ist die genaue Klassifikation algorithmischer Problemstellungen hinsichtlich ihrer Kompliziertheit. Eine herausragende Rolle nimmt dabei die Klasse der NP-vollständigen Probleme ein, welche genau jene Probleme enthält, die genauso schwer zu lösen sind wie das Problem, die Erfüllbarkeit einer aussagenlogischen Formel zu testen. Viele Probleme die in der Praxis auftauchen, zum Beispiel das Erstellen von Plänen oder das Suchen optimaler Routen, stellen sich bei genauerer Betrachtung als NP-vollständig heraus. Da es bisher noch niemandem gelungen ist, einen schnellen Algorithmus für irgendeines der zahllosen NP-vollständigen Probleme anzugeben, gehen viele Komplexitätstheoretiker inzwischen davon aus, dass NP-vollständige Probleme nicht effizient lösbar sind.

Unsere Untersuchungen haben das Ziel, festzustellen, inwiefern sich NP-vollständige Probleme unter minimaler Veränderung stabil verhalten. Genauer gesagt: es wird für verschiedene NP-vollständige Probleme untersucht, ob die Kenntnis einer Lösung (oder eines anderen Hinweises) einer Instanz eine Hilfe beim Entscheiden einer geringfügig modifizierten Instanz liefern kann. Diese Fragestellung spielt nicht nur überall dort ein große Rolle, wo NP-vollständige Probleme in dynamischen Situationen schnell gelöst werden müssen, sondern liefert auch tiefere Einsichten in die generelle Natur der Klasse der NP-vollständigen Probleme.

Wir unterscheiden vier verschiedene Möglichkeiten welcher Art der gegebene Hinweis sein kann:

1. In unserem ersten Ansatz, den wir in Kapitel 3 diskutieren, lassen wir nur solche Instanzen als Originalinstanz zu, die auch wirklich Lösungen haben. Als Hinweis sei dabei eine beliebige dieser Lösungen gegeben. Wir fragen also, ob *jede* Lösung der Originalinstanz einen nützlichen Hinweis liefert, um modifizierte Instanzen zu entscheiden. Diese Fragestellung spielt immer dann eine Rolle, wenn Probleminstanzen unvorhergesehen verändert werden.

   Wir zeigen für verschiedene NP-vollständige Sprachen, dass das Problem eine Lösung für eine modifizierte Instanz zu berechnen, abhängig von der Änderung, entweder trivial oder aber selbst wieder NP-vollständig ist. Letzteres bedeutet aber, dass der gegebene Hinweis nutzlos ist. Diese Resultate können mit Hilfe von Standardtechniken der Komplexitätstheorie gewonnen werden.

2. In Kapitel 4 untersuchen wir, ob, wenn schon nicht alle Lösungen der Originalinstanz hilfreich sind, wenigstens einige möglichst geschickt ausgewählte Lösungen helfen können. Wir fragen also, ob es Lösungen der Originalinstanz *gibt*, die einen nützlichen Hinweis liefern. Diese Fragestellung spielt dann eine Rolle, wenn man bereits vorher weiß, dass sich Instanzen eventuell ändern, und deshalb solche Lösungen gesucht werden können, die sich später als hilfreich her-

ausstellen.

Um eine Theorie der Nützlichkeit *ausgewählter* Lösungen zu entwickeln erweisen sich die Standardklassen und bekannten Reduzierungen der Komplexitätstheorie als unzureichend. Wir führen deshalb neue Komplexitätsklassen $\mathcal{C}^{\in}_{\mathrm{MOD}}/\mathcal{F}$ ein, die charakterisieren sollen wie schwer es ist, modifizierte Instanzen zu entscheiden, wenn eine ausgewählte Lösung gegeben ist. Desweiteren führen wir geeignete Reduktionsbegriffe ein, namentlich $\leq^p_{hi}$-reduction und $\leq^p_{hi}$-interreduction, mit deren Hilfe man für viele NP-vollständige Probleme auf einfache Weise die Nutzlosigkeit ausgewählter Hinweise zeigen kann.

3. In Kapitel 5 klären wir die Frage, ob das Wissen über die Nichtexistenz von Lösungen der Originalinstanz einen nützlichen Hinweis für modifizierte Instanzen liefert. Diese Ansatz ist vor allem dann interessant, wenn man eine initiale Instanz, für die man keine Lösung gefunden hat, durch geringe Modifikation in eine Instanz mit Lösung überführen will.

   Wiederum erweisen sich bekannte Klassen als unzureichend um die Komplexität von Problemen bezüglich dieser Fragestellung zu charakterisieren. Analog zu Kapitel 4 führen wir deshalb wieder passende Komplexitätsklassen $\mathcal{C}_{\mathrm{MOD}}/\notin$ und spezielle Reduktionen ein, nämlich $\leq^p_{pi}$-Reduktionen, um die Nutzlosigkeit dieses Hinweises für viele Probleme elegant beweisen zu können.

4. In Kapitel 8 untersuchen wir die Frage, was geschieht, wenn statt einer Lösung ein beliebiger polynomieller Hinweis gegeben werden darf. Dieser Hinweis kann zum Beispiel das Zwischenergebnisse eines Algorithmus', eine Hilfsrechnung oder eine Liste mehrere Lösungen der Originalinstanz beinhalten. Dies ist der allgemeinste der vier Ansätze.

Für jedes dieser vier Szenarios kommen wir zu dem Ergebnis, dass in den meisten Fällen ein gegebener Hinweis nicht besonders hilfreich ist. Wir zeigen dies für die NP-vollständigen Erfüllbarkeitsprobleme SAT, 3SAT, EX3SAT und 1-3SAT sowie für die Probleme CLIQUE, VERTEXCOVER, HAMILTONIANCYCLE, THREEDIMENSIONALMATCHING und PARTITION.

Darüber hinaus übertragen wir in Kapitel 6 unsere Theorie für NP-vollständige Probleme auf das Problem der Graph-Isomorphie, welches wahrscheinlich nicht NP-vollständig ist. Wir betrachten wiederum verschiedene Möglichkeiten von Hinweisen und adaptieren die entsprechenden Techniken welche wir für NP-vollständige Probleme entwickelt haben, um die Nutzlosigkeit solcher Hinweise auch für das Graph-Isomorphie zu beweisen. Es stellt sich heraus, dass dieses von gegebenen Hinweisen ebenfalls nicht profitieren kann.

Ganz anders verhält es sich für die in Kapitel 7 betrachtete Klasse der NP-Optimierungsprobleme. Die Kenntnis optimaler Lösungen für eine Instanz eines solchen Problem kann durchaus hilfreich sein, um eine gute Lösung für eine leicht veränderte Instanz zu finden. Wir geben folgende Beispiele für Optimierungsprobleme die von der Kenntnis (irgend)einer optimalen Lösung profitieren: MINVC, MINMAXMATCH, MinST, MinTSP$_\triangle$, MaxTSP und MaxTSP$_\triangle$. Doch nicht jedes Optimierungsproblem kann gegebene Lösungen gewinnbringend nutzen. Wir können zeigen, dass für die Probleme MinTSP und MinMaxIS jede gegebene optimale Lösung nutzlos ist, um eine gute Lösung für eine modifizierte Instanz zu finden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Complexity theory studies the inherent computational difficulty of problems. By problem we usually mean *decision problem*, i.e., it is to decide whether a given input belongs to a certain set. For instance, to decide if a Boolean formulas F is satisfiable, i.e., if there exist truth values for the variables of F that make F true, is such a decision problem. In contrast, the problem of sorting n numbers is no decision problem, since we are not interested in an answer ”yes” or ”no”, but in a sorted list of the given numbers.

These two examples, testing for satisfiability and sorting, not only differ in the output they produce, but also in the time necessary to produce this output. While it is well known that a list of n numbers can be sorted in a time that is specified by a polynomial in n, it is a famous open problem whether the same polynomial bound holds when deciding satisfiability of a formula with n variables. The problem of satisfiability, or short SAT, is not a singular example of such a problem; many real-life problems are exactly as hard as SAT. Complexity theorists introduced a special complexity class to characterize problems that share the same time complexity with SAT; these problems are called NP-complete.

In the last decades, theorists put much effort into finding an algorithm that solves efficiently, i.e., in polynomial time, an NP-complete problem — without success. But along this research, many concepts have been developed to deal with NP-complete problems in other ways, e.g., fast exponential-time exact algorithms, fixed-parameter algorithms, approximation algorithms, and heuristics. In this thesis, we discuss another possible approach to attack NP-complete problems, namely to decide slightly modified instances with help of prior knowledge.

To motivate the study of modified instances we give a simple example: Let F be a satisfiable formula and β be a satisfying assignment for F. Let F' be a formula that is constructed from F by addition of a new clause. We ask: How hard is it to decide if F' is satisfiable when the satisfying assignment β for F is given as an additional information?

In the above scenario we refer to F as the *original instance*, to F' as the *modified instance*, and to β as a *certificate* for the original instance or a *hint* in finding a solution of F'. As we will see in Section 3, an arbitrary assignment β for F is in general not helpful to decide if F' is satisfiable, not to speak of constructing such satisfying assignments. More precisely, for formulas with added clauses the satisfiability problem is computationally as hard as SAT, namely NP-complete, even if a certificate of the original formula is known.

Extending a classification from [Lib04], we distinguish the following four possible settings. In the first setting, we demand that hints are certificates of the original instance, as we did in the introductory example. We ask whether *all* such hints, i.e., all certificates of the original instance, yield a helpful information. This is relevant whenever an input instance is unexpectedly modified and we want to avoid a complete reevaluation for the modified instance, using the additional information we have. We study the question whether all certificates are useful in Chapter 3.

The second case, in contrast, applies to a scenario in which we know in advance that the original instance is going to be modified, for example when we deal with at first incorrect, but later to be corrected, data. Thus, the computation for the original instance could select among all certificates of the original instance some especially convenient certificate. In other words, we are interested if there *exists* a certificate of the original instance that helps to decide modified instances. This question is covered in Chapter 4.

In addition to the classification in [Lib04], we consider as a third possibility that the original instance has no certificate at all. The information that the original instance has no certificate could be indeed useful in deciding modified instances, although it might conflict with ones first intuition. If, for instance, a clause is added to an unsatisfiable formula, the modified formula is still not satisfiable. Since it is impossible to give a hint in the form of a certificate when no certificate exists, we best think of the hint as something like a promise that the original instance has no certificate. Results about this type kind of promise-aided computation are relevant whenever an initial instance that has no certificate is to be transformed into an instance that has a certificate. We study this scenario in Chapter 5.

Last, in the fourth setting the hint may be any polynomially bounded string. This string may represent the result of some subcomputation, a composition of selected certificates, or the result of some preprocessing step. Obviously, this last setting is the most general of all these four settings and matches a practitioners approach to the problem of modified instances. Our results for this case can be found in Chapter 8.

We develop for each of these four settings a formal framework that can be used to either prove or disprove that a problem has easy to decide modified instances. Therefore, we introduce new complexity classes and reductions. We show that, for all of the NP-complete problems that we examine, a certificate for the original instance is useless if the original instance is appropriately modified. We also show that proving hardness of deciding modified instances for some other modifications amounts to solve some long-standing open problems in complexity theory, such as the non-collapse of the polynomial hierarchy. Most of the results we establish are such negative results. We find that modified instances profit from given certificates only in some trivial cases.[1]

The list of discussed NP-complete problems consists of a variation of satisfiabil-

[1]For a quick overview on our results we point the reader to the 'Conclusions'-section at the end of each chapter.

ity problems such as SAT, 3SAT, Ex3SAT and 1-Ex3SAT as well as the remaining five basic problems from [GJ79], namely Clique, VertexCover (VC), HamiltonianCycle (HC), ThreeDimensionalMatching (3DM), and Partition. In establishing the above mentioned negative results we only consider the corresponding *decision problem* — hardness of computing certificates for modified instances follows immediately. Since it is an interesting question whether this uselessness of hints under modification is an intrinsic property of NP-complete problems, we study in Chapter 6 modified instances of the graph isomorphism problem, which is probably not NP-complete.

In an effort to find a scenario in which certificates are useful, we examine in Chapter 7 the influence of modification to optimization problems. In detail, we are interested in the question if an *optimum* solution of an original instance leads to good approximations of modified instances. This problem is known as reoptimization [Sch97, ABS03, AEMP06, EMP07, BHMW08]. Again, we consider the scenario where an arbitrary optimum solution is given as a hint and also the scenario where the optimum solution is carefully selected among all optimum solutions. We find that for many optimization problems reoptimization leads to better approximation guarantees than in the usual, non-reoptimization case, even for arbitrary certificates. We mention, as an example, that the metric travelling salesperson problem, for which no better approximation guarantee is known than a factor of $3/2$, is reoptimizable with factor $4/3$ when a single edge weight is altered. We also find that sometimes reoptimization leads to no improved approximation guarantees, the nonmetric version of the travelling salesperson problem together with the modification of a single edge being an example.

1.1 Related Work

There has been extensive research on how computationally easy problems, i.e., problems in P, benefit from given hints [Hen00, HK01, TK00, FK99, EGIN97]. In this context, the problem of modification is often being referred to as dynamic problem, or problem in a dynamic environment. The most sophisticated algorithms and data structures have been developed for dynamic graph problems. For example, fast hint-using algorithms are known for maintaining the minimum spanning forest of an undirected graph [HK01] and the biconnected components of an undirected graph [Hen00]. These dynamic graph algorithms can also be used to develop fast algorithms for static graphs, i.e., in a non-dynamic environment [TK00]. We mention in passing that there also exist graph problems in P for which no efficient fully dynamic (that is, insertions *and* deletions of edges are allowed) deterministic (that is, *not* randomized) algorithms are known, for example the maximum matching problem or the problem of finding the transitive closure of a directed acyclic graph [KMW98]. For more information about dynamic graph algorithms for polynomial problems we direct the reader to [FK99] and [EGIN97]. For a complexity theoretic view on dynamic P-problems we refer to [MSVT94] and [WS07].

In contrast, in this thesis we mainly concentrate on dynamic NP-complete-problems.

Dynamic versions of the following NP-complete problems have already been studied in the literature:

- SAT and VC [Lib04],
- Scheduling with forbidden sets [Sch97],
- planning problems from the field of Artificial Intelligence (AI) [NK92, Lib04, Lib98b], and
- decision versions of MinTSP[ABS03] and MaxTSP[AEMP06].

There are many other approaches to modification of instances, especially in the field of AI. We only mention the following ones and point out differences to our approach. In [NK92] the authors are concerned with planning problems, a prototypical problem in AI research. They ask, given an original instance and a certificate for this instance, whether there exists a certificate for a modified instance that does not differ too much from the original certificate. They point out that sometimes it is even more complicated to find such a 'similar' certificate for the modified instance than reevaluating a new one from scratch. In our approach, finding a certificate for the modified instance cannot be any harder than computing a certificate from scratch.

Closely related is the concept of supermodels and robustness [GPR98]. There, the authors address the issue, of finding a solution of the original instance x that can be turned into a certificate of a slightly modified instance x' by a small change to the solution. If such a robust certificate exists, a certificate for x' can easily be found by testing all slightly altered certificates as a candidate. In a general approach, there could be more sophisticated ideas to test if x' has a certificate than testing all certificates in the neighborhood of an original certificate. Consequently, hardness results for robust certificates cannot be transferred to our general setting.

As a further specialization of the latter concept, it can be asked how much a given instance of an *optimization problem* may be varied such that the given solution remains an optimum [Gre98, LvdPSvdV98, VHW99]. Although this line of research is useful in finding modifications for which a given certificate is useful, we allow in our approach also modifications that lead to the loss of optimality of the old optimum solution.

The concept of not starting from scratch, but to use prior knowledge, when confronted with a new problem-instance is also inherent in *persistent computations* [HK08, Kos98]. There, one is concerned with solving not only a single instance, but with processing a (possibly infinite) series of inputs. Thereby, we are allowed to reuse solutions of past input instances to compute a solution for the current instance. In contrast to our approach, in which we are interested in small and local modifications of instances, the sequential inputs for persistent computations need not to be related in any way.

As a concluding remark to this section we like to emphasize the importance of the concept of modification in a broader context. In [Gol05] four solutions to important problems are found to start from a trivial construct, and to apply an ingeniously designed sequence of iterations that yields the desired nontrivial result. Thereby, in each iteration the construct is only modified in a moderate manner. In detail, the four works that are referred to in [Gol05] are:

- the polynomial-time approximation of the permanent of non negative matrices,
- the iterative (Zig-Zag) construction of expander graphs,
- Reingold's log-space algorithm for undirected connectivity, and
- Dinur's alternative proof of the PCP Theorem.

For a nice overview on how the iterative approach is employed to these four problems we refer to [Gol05].

Chapter 2

Preliminaries

In this chapter we fix the basic notions and concepts that are used in this thesis. Most of the definitions are in the standard text books on complexity theory [BDG95, BDG90, WW86, Pap94]. We assume the reader to be familiar with the most basic set theoretical notations and the fundamental concepts of propositional logic. The advanced reader may skip the first few sections. But it is highly recommended to read at least Section 2.6.

2.1 Graph Theory

An *undirected graph* G is an ordered pair (V, E), where V are the vertices of G and the set of edges E is a subset of $\{\{u, v\} : u \neq v \wedge u, v \in V\}$. We say that u and v are *endpoints* of the edge $\{u, v\}$. For a graph G let $V(G)$ and $E(G)$ denote the set of vertices and the set of edges of G, respectively. If $V(G)$ is a finite set we say that G is a *finite graph*. We say that $|V(G)|$ is the *size* of G. Within this thesis we only consider undirected, finite graphs.

Given an undirected, finite graph $G = (V, E)$ we denote by $\overline{G} := (V, E')$, where

$$E' := \{\{u, v\} : u \neq v \wedge u, v \in V \wedge \{u, v\} \notin E\},$$

the *complement graph* of G. We say that two edges e and f are *incident* if and only if e and f share a common endpoint. Likewise, an edge e and a vertex u are called *incident* if and only if u is an endpoint of e. The two vertices of an edge e are called *adjacent* vertices, or *neighbors*. For a given vertex u in a graph G the *neighborhood* of u in G, short $N_G(u)$, is the set of vertices that are adjacent to u. The *degree* of a vertex u in a graph G, short $deg_G(u)$, is the number of edges that are incident to u, i.e., $deg(u) = |N_G(u)|$. In both cases, $N_G(u)$ and $deg_G(u)$, we usually omit the subscript if the graph G is clear from the context.

A *subgraph* of a graph G is a graph whose vertex and edge sets are subsets of the respective sets in G. A subgraph H of a graph G is said to be induced by G, if and only if $E(H) \subseteq E(G)$ and every edge in $E(G)$ that consists of vertices from $V(H)$ is contained in $E(H)$. A graph that contains all possible edges is called a *complete* graph. Let K_n denote the complete graph over n vertices.

A *path* in a graph G is a sequence $(v_1, ..., v_n)$ of vertices such that every two consecutive vertices are joined by an edge of G. A path in that all the vertices of the sequence are distinct is called a *simple path*. A *component* of a graph $G = (V, E)$ is a maximal set $V' \subseteq V$ such that for all pairs of different vertices from V' there exists a path between these two vertices. A graph is called *connected* if an only if it consists of one component. A *cycle* is a path $(v_1, ..., v_n)$ with $v_1 = v_n$. We say that a cycle $(v_1, ..., v_n)$ is a *simple cycle* if and only if v_1 and v_n are the only vertices of the cycle that appear multiple times. A simple cycle that includes all the vertices of the graph is called a *Hamiltonian cycle*. Sometimes we identify a (Hamiltonian) cycle C with the subgraph that is spanned by the edges of C. A complete subgraph of a graph is called a *clique*, whereas a subgraph that has a complete complement is called an *independent set*. A *tree* is a connected acyclic graph. A *forest* is a graph in that every component is a tree.

Two graphs (V_1, E_1) and (V_2, E_2) are *isomorphic* if and only if there exists an *isomorphism* between them, that is, there exists a bijective function $\phi : V_1 \rightarrow V_2$ such that for all $u, v \in V_1$ it holds that $\{u, v\} \in E_1 \Leftrightarrow \{\phi(u), \phi(v)\} \in E_2$. A *weighted graph* is a pair that consists of a graph (V, E) and a weight function $\omega : E \rightarrow \mathbb{R}$.

For further particulars on graph theory we point the reader to [Wes96].

2.2 Words and Languages

In theoretical computer science, *languages* are set of words over a finite alphabet. Sometimes we refer to languages as problems; the two notions are interchangeable. We use the alphabet $\Sigma = \{0, 1\}$ as our standard alphabet. We denote by ϵ the *empty word*, which contains no letters. The symbol Σ^n denotes the set of all words that consists of exactly n letters from Σ. The set of all words over the alphabet Σ is given by the set $\Sigma^* := \bigcup_{i \in \mathbb{N}} \Sigma^n$. We denote by $|x|$ the length of a word x, i.e., the number of letters that x consists of. The complement of a language A is given by $\overline{A} := \Sigma^* \setminus A$. A language A is *trivial* if and only if $A = \emptyset$ or $\overline{A} = \emptyset$.

In order to treat graphs, formulas, sets, tuples, and other instances as a member of some language, we use encodings that transform structures from a given domain to a string from Σ^*. These encodings shall be easily computable and easily invertible, that is, the encoding of an object is computable in polynomial time (see Section 2.4) and conversely also information about the object can be obtained from its encoding in polynomial time. When we encode an object x , the size of the encoded object shall properly reflect the size of x. For example, when encoding a graph $G = (V, E)$, the size of the encoded graph shall be bounded from both sides by polynomials in V or E. Such encodings exist for all the mathematical objects regarded in this thesis. To avoid unnecessary formal overhead we usually speak of graphs, formulas, or sets (just to name a few) when referring to members of some language where we formally had to use encodings by words over Σ. For instance, when considering the problem HC over the domain of graphs we use a nice encoding function enc that transforms graphs to words over Σ and just write $G \in$ HC instead of the more formal $enc(G) \in$ HC.

2.3 Propositional Logic

The *literals* of a variable x are x and $\neg x$. The literal x is called a positive literal, the literal $\neg x$ is called a negative literal. A *clause* C over a variable set V is a formula $L_1 \vee L_2 \vee ... \vee L_n$, $n \in \mathbb{N}$, where L_j is literal of a variable from V, for all $1 \leq j \leq n$. A clause with exactly one literal is a *unit clause*. Let $Var(F)$ denote the set of variables of a formula F. A Boolean formula F is in *conjunctive normal form* (*CNF*) if and only if $F = C_1 \wedge C_2 \wedge ... \wedge C_m$, $m \in \mathbb{N}$, and C_j is a clause over $Var(F)$, for all $1 \leq j \leq m$. A formula F is in *3CNF*-form if and only if F is a CNF-formula and every clause of F contains at most three literals. A 3CNF-formula F in in *EX3CNF*-form if and only if every clause of F consists of exactly three distinct literals.

Sometimes it notational benefits to represent clauses as sets of literals. For example, the clause $L_1 \vee ... \vee L_n$ is represented by the set $\{L_1, ..., L_n\}$. Likewise, we represent a CNF-formula $C_1 \wedge ... \wedge C_m$ by the set $\{C_1, ..., C_m\}$. We switch seamlessly between these two representations.

An *assignment* for a Boolean formula F over the variable set X is a function $\beta : X \rightarrow \{true, false\}$. We use the number '1' to represent the truth value 'true', and the number '0' to represent 'false'. An assignment β satisfies a CNF-formula F if and only if in each clause of F there exists at least one positive literal x with $\beta(x) = 1$ or at least one negative literal $\neg x$ with $\beta(x) = 0$. We denote by SAT the set of all satisfiable CNF-formulas. Two formulas F_1 and F_2 are said to be equivalent, abbreviated $F_1 \equiv F_2$, if and only if for all assignments β it holds that β satisfies F_1 if and only if β satisfies F_2.

2.4 Complexity Theory

2.4.1 Turing Machines

In structural complexity theory sets of languages, so called complexity classes, are studied. Often, a complexity class is a set of problems that share the same complexity in terms of resources of some computational model, for example Turing machines. In this thesis, our computational model is the multi-tape Turing machine. Due to the commonly accepted thesis of church, which says that no computational model exceeds the computational power of Turing machines, we describe our algorithms in a rather informal way, instead of giving a detailed Turing machine program. For a formal definition of Turing machines we refer to [WW86] or [Pap94]. We distinguish between deterministic Turing machines (*DTMs*) and nondeterministic Turing machines (*NTMs*). Let $M(x)$ denote the work of the Turing machine M on input x. A language A is accepted by a (nondeterministic) Turing machine M if and only if for all $x \in \Sigma^*$ it holds that

$$x \in A \Leftrightarrow M(x) \text{ accepts on some path .}$$

Given a machine M let $L(M)$ denote the set of words accepted by M. We say that a Turing machine M accepts a language A in time f, $f : \mathbb{N} \rightarrow \mathbb{N}$, if and only if for all inputs $x \in \Sigma^*$ the machine M halts after at most $f(|x|)$ computational steps.

Turing machines can be equipped with an oracle $A \subseteq \Sigma^*$. A Turing machine M with an oracle A has an additional query tape. Whenever a special query state is reached the machine M immediately receives the answer 'yes' if the word on the query tape belongs to A and the answer 'no' otherwise. A machine M equipped with an oracle A is called an *oracle Turing machine*. We use the abbreviation M^A when referring to such a machine.

2.4.2 Basic Complexity Classes

A language A belongs to the complexity class P if and only if there exist a DTM M and polynomial p such that M accepts A in time p. Since the machine M works in a time that is polynomial in the size of the input we say that M is a *deterministic polynomial-time Turing machine* (*DPTM*). Similarly, a language A belongs to the class NP if and only if there exist an NTM N and a polynomial p such that N accepts A in time p. We say that N is a *nondeterministic polynomial-time Turing machine* (*NPTM*). If $\mathcal{C}$ is a complexity class, the complement $co\mathcal{C}$ of $\mathcal{C}$ is given by the set $\{\overline{A} : A \in \mathcal{C}\}$. We say that a complexity class $\mathcal{C}$ is *nontrivial* if and only if $\mathcal{C}$ contains nontrivial languages, i.e., if and only if $\mathcal{C} \setminus \{\emptyset, \Sigma^*\} \neq \emptyset$.

We can use Turing machines to compute functions. Therefore, we equip Turing machines with a separated output tape. The content of the output tape at an accepting halting state shall represent the computed function value. A non-accepting halting state shall represent the fact, that the function value is not defined at the given input. The class FP is defined as the class of functions that can be computed by a DPTM with an output tape. The class $\mathrm{F}\Delta_2^p$ is defined as the class of functions that can be computed by a DPTM with an output tape and an oracle from NP.

For a complexity class $\mathcal{C}$, the classes $\mathrm{P}^{\mathcal{C}}$ and $\mathrm{NP}^{\mathcal{C}}$ are defined as the classes of languages that can be accepted by polynomial-time deterministic or nondeterministic oracle Turing machines that make queries to a language from $\mathcal{C}$, respectively.

2.4.3 Reductions

Reductions are a standard mean to compare the complexity of problems. In this thesis we use *many-one-reductions* [Kar72].

Definition 2.1 ([Kar72]). *Let A and B be two languages. We say that A is polynomial-time many-one reducible to B, short $A \leq_m^p B$, if and only if there exists a total function $f \in \mathrm{FP}$ such that for all $x \in \Sigma^*$ it holds that*

$$x \in A \Leftrightarrow f(x) \in B.$$

Many-one-reductions are reflexive and transitive. We say that a set B is *hard* for a complexity class $\mathcal{C}$ with respect to some reduction $\leq$ if and only if for every problem A in $\mathcal{C}$ it holds that $A \leq B$. We say that a set B is *complete* for a complexity class $\mathcal{C}$ w.r.t.

some reduction $\leq$ if and only if B is $\mathcal{C}$-hard w.r.t. $\leq_m^p$ and $B \in \mathcal{C}$. A complexity class $\mathcal{C}$ is *closed under* a reduction $\leq$ if and only if for all sets A and B it holds that

$$(A \leq B \wedge B \in \mathcal{C}) \Rightarrow A \in \mathcal{C}.$$

Two problems A and B are *equivalent* w.r.t. a certain reduction $\leq$ if and only if it holds that both $A \leq B$ and $B \leq A$. If A and B are equivalent w.r.t. $\leq_m^p$-reduction we write $A \equiv_m^p B$. The language SAT is an example of an NP-complete problem [Coo71]. The set of NP-complete problems w.r.t. $\leq_m^p$-reduction is exactly the set $\{A : A \equiv_m^p \mathrm{SAT}\}$. Let NPC denote the set of NP-complete problems.

When applying the $\leq_m^p$-reduction to scenarios in which we reduce a language A to a language B of tuples $(x_1, ..., x_n)$ we sometimes only give functions $f_1, ..., f_n \in \mathrm{FP}$ such that for all $x \in \Sigma^*$ it holds that

$$x \in A \Leftrightarrow (f_1(x), ..., f_n(x_n)) \in B.$$

It should be clear how to compose from $f_1, ..., f_n$ a function $f \in \mathrm{FP}$ such that the formally correct equivalence $x \in A \Leftrightarrow f(x) \in B$ holds.

The class P and $\leq_m^p$-reduction are related in the following way.

Observation 2.2. *Any nontrivial complexity class $\mathcal{C}$ that is closed under $\leq_m^p$-reduction contains the class* P.

Proof. Let $\mathcal{C}$ be a nontrivial complexity class and let A be a nontrivial problem in $\mathcal{C}$. Every nontrivial problem is $\leq_m^p$-complete for the class P. Thus, for all $B \in \mathrm{P}$ it holds that $B \leq A$. Since $\mathcal{C}$ is closed under $\leq_m^p$-reduction we have that for all $B \in \mathrm{P}$ it holds that $B \in \mathcal{C}$. $\square$

2.4.4 Verifiers and Certificates

In this thesis, a fundamental notion is the notion of *solutions*, or synonymous, the notion of *certificates*. For NP-problems the notion of a solution is defined with the help of the following well known theorem:

Theorem 2.3. *A language A is in* NP *if and only if there exist a predicate $B \in \mathrm{P}$ and a polynomial p such that for all $x \in \Sigma^*$ it holds that*

$$x \in A \Leftrightarrow (\exists y \in \Sigma^*)[|y| \leq p(|x|) \wedge (x, y) \in B].$$

Rephrasing this theorem, we can say that the class NP is exactly the set of languages with short and easy to verify membership proofs. In the context of the above theorem, we say that those strings y that proof membership of x in A are *certificates* or *solutions* and we call the predicate B a *polynomial-time verifier* for A. Of course, there are different verifiers for one and the same language A, but all of them share the following properties.

Definition 2.4. *A relation $V \subseteq \Sigma^* \times \Sigma^*$ is called a polynomial-time verifier if and only if*

1. *$V \in \mathrm{P}$ and*
2. *there is a polynomial p such that for all $x, \pi \in \Sigma^*$ it holds that*

$$(x, \pi) \in V \Rightarrow |\pi| \leq p(|x|).$$

For a verifier V let $L(V)$ denote the language that is accepted by V, that is,

$$L(V) = \{x : (\exists y \in \Sigma^*)[|y| \leq p(|x|) \wedge (x, y) \in V]\}.$$

In the coming all the used verifiers will be polynomial-time verifiers. Thus, no ambiguity arises when we use the shorter term 'verifier' instead of the more formal 'polynomial-time verifier' henceforth.

Whenever dealing with solutions of NP-problem instances we first need to formally specify the used verifier, thereby fixing the precise form of the solutions. For details on how different verifiers for the same problem might influence the complexity of related problems we refer for example to [Che03].

2.4.5 The Polynomial Hierarchy

As a generalization of the classes P and NP, and in close analogy to the arithmetic hierarchy in recursion theory (see [Rog67]), Meyer and Stockmeyer introduced the polynomial hierarchy [SM73, Sto76]. The classes of the polynomial hierarchy are defined as follows.

Definition 2.5 ([SM73, Sto76]). *The complexity classes Σ_i^p, Π_i^p, and Δ_i^p are inductively defined via*

1. *$\Delta_0^p = \Sigma_0^p = \Pi_0^p = \mathrm{P}$,*
2. *$\Delta_{i+1}^p = \mathrm{P}^{\Sigma_i^p}$, $\Sigma_{i+1}^p = \mathrm{NP}^{\Sigma_i^p}$, and $\Pi_{i+1}^p = co\Sigma_{i+1}^p$, for all $i \geq 1$.*

The class PH is defined by $\mathrm{PH} = \bigcup_{i \in \mathbb{N}} \Sigma_i^p$.

The inclusion structure of the polynomial hierarchy is shown in Figure 2.1. It is not clear if any of the inclusions is strict, or if the hierarchy is finite, but there exist several conditions under which the hierarchy collapses. In particular, the polynomial hierarchy satisfied the following upward collapse property.

Theorem 2.6 ([Sto76]). *Let $i \geq 1$. Then*

1. *$\Sigma_i^p = \Pi_i^p \Rightarrow \mathrm{PH} = \Sigma_i^p$,*
2. *$\Sigma_i^p = \Sigma_{i+1}^p \Rightarrow \mathrm{PH} = \Sigma_i^p$, and*
3. *$\Sigma_i^p = \Delta_i^p \Rightarrow \mathrm{PH} = \Delta_i^p$.*

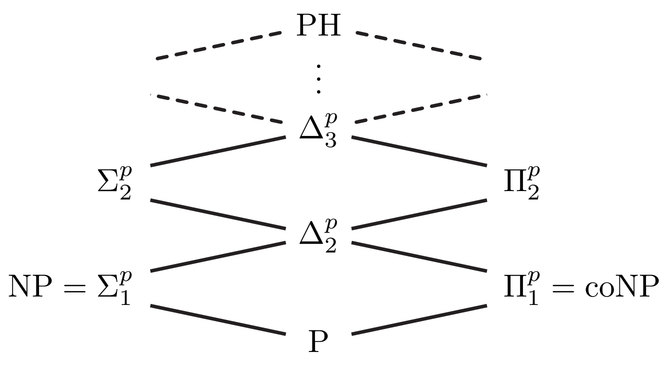

Figure 2.1: The polynomial hierarchy.

2.5 Advanced Complexity Theory

2.5.1 Counting Classes with Finite Acceptance Type

All the acceptance types considered so far had a qualitative character. For instance, when we decide an NP-language with help of an NPTM M it is only important whether a certain accepting state is reached or not. The particular number of accepting paths of M's computation is of no interest. But such consideration play an important role in the context of counting classes with finite acceptance type. These classes were first introduced in [BG82]. Here, we follow a definition from [GNW90].

Definition 2.7. *Let $acc_M(x)$ denote the number of accepting paths of an NPTM M on input x and let $C \subseteq \mathbb{N}$. A language A belongs to the counting class $C\mathrm{P}$ if and only if there exists an NPTM M such that for all $x \in \Sigma^*$ it holds that*

$$x \in A \Leftrightarrow acc_M(x) \in C.$$

It follows from this definition that $\mathrm{coNP} = \{0\}\mathrm{P}$ and $\mathrm{NP} = (\mathbb{N}\backslash\{0\})\mathrm{P}$. In this thesis we also use the complexity classes $\{0,1\}\mathrm{P}$ (which equals coNP) and $\{1\}\mathrm{P}$. It is easy to show that for every $C \subseteq \mathbb{N}$ the class $C\mathrm{P}$ is closed under $\leq_m^p$-reduction and that $C\mathrm{P}$ has complete problems. For instance, the following derivative of SAT is a prototypical $\leq_m^p$-complete problem for $C\mathrm{P}$:

$$C\text{SAT} := \{\ F\ :\ F \text{ is a Boolean formula and the number of satisfying assignments of } F \text{ is in } C\ \}.$$

It is easy to show that $\mathrm{coNP} \subseteq \{1\}\mathrm{P}$. Using this fact, we get the following corollary.

Corollary 2.8. $\{1\}\mathrm{P} \subseteq \mathrm{NP} \Rightarrow \mathrm{NP} = \mathrm{coNP}$.

Proof. Let $\{1\}\mathrm{P} \subseteq \mathrm{NP}$. Since $\mathrm{coNP} \subseteq \{1\}\mathrm{P}$ it follows that $\mathrm{coNP} \subseteq \mathrm{NP}$. $\square$

2.5.2 Function Classes

We already defined the most basic function class FP in Section 2.4.2. There, FP is defined with help of Turing machines that have an output tape. In contrast, we now base the definition of function (and relation) classes on well studied complexity classes instead of the computation of Turing machines. We follow a definition from [Gro04].

Definition 2.9 ([Gro04]). *A relation r belongs to the function class rel $\cdot\, \mathcal{C}$ if and only if there exist a predicate $B \in \mathcal{C}$ and a polynomial p such that for all $x \in \Sigma^*$ it holds that*

$$r(x) = \{y : |y| \leq p(|x|) \wedge (x, y) \in B\}.$$

A function f is contained in the class fun $\cdot\, \mathcal{C}$ if and only if

$$f \in \textit{rel} \cdot \mathcal{C} \wedge (\forall x \in \Sigma^*)[|f(x)| \leq 1].$$

The classes fun$\cdot\mathrm{NP}$ and rel$\cdot\mathrm{NP}$ are known in the literature as NPSV and NPMV (nondeterministic polynomial-time computable single/multi-valued functions) [BLS84]. An advantage of Definition 2.9 is that it yields a uniform and systematic approach to otherwise seemingly isolated notions.

2.5.3 Nonuniform Complexity Classes

The concept of Turing machines is a uniform model of computation; a Turing machine M that decides a language A uses the same algorithmic idea for all given inputs. In order to overcome this limitation, we allow that different algorithms are applied to inputs of different length. For a formalization of this idea we use the following definition from [KL80].

Definition 2.10 ([KL80]). *Let $\mathcal{F}$ be a set of functions from $\mathbb{N}$ to Σ^* and let $\mathcal{C}$ be a complexity class. A language A is contained in the nonuniform complexity class $\mathcal{C}/\mathcal{F}$ if and only if there exist a set $C \in \mathcal{C}$ and a function $f \in \mathcal{F}$ such that for all $x \in \Sigma^*$ it holds that*

$$x \in A \Leftrightarrow (x, f(|x|)) \in C.$$

In particular, we use the nonuniform classes P/poly, $\mathrm{NP}/\mathit{poly}$, and $\mathrm{coNP}/\mathit{poly}$, where the function-class poly is defined by

$$\mathit{poly} := \{f : (\exists \text{ polynomial } p)(\forall n \in \mathbb{N})[|f(n)| \leq p(n)]\}.$$

We say that a class $\mathcal{C}/\mathit{poly}$ belongs to the nonuniform polynomial hierarchy if and only if $\mathcal{C}$ belongs to the polynomial hierarchy.

At first glance, it seems that Definition 2.10 is highly dependent on the coding of the instances. Since different coding functions c_1 and c_2 could map one and the same problem instance to strings of different lengths, the following scenario could arise: While the first coding c_1 maps two different instances x and x' to words of the same length, the coding c_2 maps those instances to words of different lengths. Consequently, a nonuniform algorithm that decides these instances has the same hint for both instances in the first case. But in the second case the hints may be different.

We show that for two reasonable codings c_1 and c_2 the containment of a language A in $\mathcal{C}/poly$ when the coding c_1 is used also implicates containment of A in $\mathcal{C}/poly$ when the coding c_2 is used: Let h be the hint function that shows $A \in \mathcal{C}/poly$ when c_1 is used as coding. Note that nice codings are polynomially related, $i.e.$, for reasonable codings c_1 and c_2 it holds that $|c_1(x)| \leq p_1(|c_2(x)|)$ and $|c_2(x)| \leq p_2(|c_1(x)|)$ for all instances x and some polynomials p_1 and p_2. We use as a polynomial hint for $c_2(x)$ the string $h(0)\#h(1)\#...\#h(p_1(|c_2(x)|))$. When c_2 is easily invertible we can compute in polynomial-time from $c_2(x)$ the instance $c_1(x)$, and also the length of $c_1(x)$. The right hint $h(|c_1(x)|)$ for $c_1(x)$ can be extracted from $h(0)\#h(1)\#...\#h(p_1(|c_2(x)|))$. This shows that also $A \in \mathcal{C}/poly$ when the coding c_2 is used. In consequence when showing that $A \in \mathcal{C}/poly$ it suffices to show this for some convenient reasonable coding of A.

There are several results that relate uniform with nonuniform complexity classes. Much attention has been paid to the question, whether NP is contained in certain nonuniform complexity classes. The following theorem states the best results in this direction that are currently known.

Theorem 2.11 ([Cai07, CCHO05]).

- $\mathrm{NP} \subseteq \mathrm{P}/poly \Rightarrow \mathrm{PH} = S_2$,
- $\mathrm{NP} \subseteq \mathrm{coNP}/poly \Rightarrow \mathrm{PH} = S_2^{\mathrm{NP}}$,
- $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/poly \Rightarrow \mathrm{PH} = S_2^{\mathrm{NP} \cap \mathrm{coNP}}$.

Here, the class S_2 is a class from the so called symmetric hierarchy and is defined as follows:

Definition 2.12 ([Can96, RS98]). *A language L belongs to S_2 if and only if there exist a 3-ary predicate $V \in \mathrm{P}$ and a polynomial p such that for all x it holds that*

1. $x \in L \Rightarrow (\exists y : |y| \leq p(|x|))(\forall z : |z| \leq p(|x|))[P(x, y, z) = 1]$ *and*
2. $x \notin L \Rightarrow (\exists z : |z| \leq p(|x|))(\forall y : |y| \leq p(|x|))[P(x, y, z) = 0]$.

Containment of a language L in the class S_2, which is a superset of Δ_2^p and a subset of Σ_2^p (see [RS98]), can be interpreted as follows. Suppose that deciding the language is game of two persons that try to convince the verifier that $x \in L$ or $x \notin L$, respectively. If $x \in L$ then the first person has an irrefutable proof y that, independent of any proof z given by the second person, verifies that $x \in L$. On the other hand, if $x \notin L$ then the second person has such an irrefutable proof z for this fact, that withstands any challenge y from the first person. For more details on S_2 see [RS98].

From Theorem 2.11 we conclude that if certain nonuniform complexity classes coincide then a collapse of some uniform complexity classes occurs.

Corollary 2.13. $\mathrm{NP}/poly = \mathrm{P}/poly \Rightarrow \mathrm{PH} = S_2$.

Proof. Since $\mathrm{NP} \subseteq \mathrm{NP}/poly$ we get $\mathrm{NP} \subseteq \mathrm{P}/poly$. The assertion follows from Theorem 2.11. □

Yap has shown in [Yap83] that also the equality of other complexity classes from the nonuniform polynomial hierarchy implicates a collapse of the polynomial hierarchy.

Theorem 2.14 ([Yap83]). $\Sigma_i^p/poly = \Pi_i^p/poly \Rightarrow \Sigma_{i+2}^p = \mathrm{PH}$.

2.6 Modification Functions

In the course of this thesis we want to distinguish between different kinds of modification. On the one hand, we want to find modifications such that a certificate for an original instance is a useful hint for deciding modified instances; on the other hand, we are interested in those modification that render such a hint useless. Therefore we need to formally specify the notion of modification. For this purpose we introduce so called modification functions.

Definition 2.15. *A function* $f(x, m) : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ *is called a modification function if and only if* $f \in \mathrm{FP}$.

This definition assures that the modified instance is easy to compute from both the original instance x and the string m, which specifies the parts of the original instance that are modified. To illustrate Definition 2.15 we give a simple example of a modification function.

Example: Consider the problem SAT of all satisfiable propositional CNF-formulas. An instance for this problem is any propositional formula F. As modification we choose the deletion of a certain unit clause $\{L\}$ of F. The appropriate modification function is then given by the following function $rm(F, L)) := F \setminus \{L\}$.

In the remainder we mostly concentrate on modification functions from the 'semantic' domain of a problem, e.g., changes to the clauses of a formulas or to the edges of a graph. We do not consider modifications that work on a binary level, like flipping a single bit of an instance. We point the reader that is interested in such binary modifications to [MSVT94]. We give three reasons for our decision to not explicitly examine binary-level-modifications. First, we do not want our results to be dependent on a special encoding of the instances of a problem. Second, we are more interested in scenarios in which instances alter owing to some change within the problem domain; we are only to a lesser extent interested in consequences of faulty transmission or broken hardware that causes bit errors. Finally, results about binary modifications can often be derived from results about other modification functions when choosing an appropriate encoding of the instances.

Chapter 3

Arbitrary Solution as Hint

Throughout this thesis we are concerned with the question whether a solution for an instance x can be any helpful when x is modified in some way. This is relevant whenever instances might change from time to time. We are especially interested in modified instances of computationally hard problems, e.g., NP-complete problems.

In this chapter in particular we deal with the question whether an arbitrary certificate for the original instance can be a helpful hint in deciding modified instances. In other words: Are *all* solutions of the original instance helpful when deciding modified instances?

3.1 Problem Formalization

We want to formalize the problem whether a slightly modified instance is an element of a language A when a solution for the original instance is already known. We use the notion of a verifier and a modification function to define a decision problem $\mathrm{MOD}_c V_A$ that shall characterize the complexity of deciding modified instances.

Definition 3.1. *Let V_A be a verifier for a language $A \in \mathrm{NP}$ and c be a modification function. Then*

$$\mathrm{MOD}_c V_A := \{(x, \pi, m) : (x, \pi) \in V_A \ \ and \ \ c(x, m) \in A\}.$$

Regarding this definition, we refer to x as the original instance, π is a certificate (or solution, hint) for the original instance and m is the modification of the instance x. The modified instance, for which containment in A is to be be decided, can be obtained by applying the modification function c to x and m.

Note that the complexity of a problem $\mathrm{MOD}_c V_A$ does not stem from the part of the definition where it is verified that π is indeed a certificate for x. This verification process can be done in polynomial time. Thus, for P-hard problems $\mathrm{MOD}_c V_A$ the decision whether the modified instance $c(x, m)$ belongs to A is as hard as deciding whether $(x, \pi, m) \in \mathrm{MOD}_c V_A$. In other words: The complexity of $\mathrm{MOD}_c V_A$ characterizes the complexity of deciding modified instances $c(x, m)$.

As a last remark, observe that containment of an instance (x, π, m) in $\mathrm{MOD}_c V_A$ can be decided by verifying that π is a certificate for x, which can be done in P, and disregarding the hint π henceforth. Consequently if $\mathcal{C}$ is a complexity class that contains P — or even stronger, if $\mathcal{C}$ is closed with respect to $\leq_m^p$-reduction (see Observation 2.2) — and A is a problem in $\mathcal{C}$ then deciding modified instances cannot be any harder than the original problem A.

Observation 3.2. *Let V_A be a verifier for $A \in \mathcal{C}$ and let $\mathcal{C}$ be closed under $\leq_m^p$-reduction. Then* $\mathrm{MOD}_c V_A \in \mathcal{C}$.

Therefore, in a proof of NP-completeness of $\mathrm{MOD}_c V_A$, where $L(V_A) \in \mathrm{NP}$, it suffices to prove NP-hardness of $\mathrm{MOD}_c V_A$.

3.2 The Problem SAT

First, we modify instances of the prototypical NP-complete problem SAT. As already mentioned in the introduction, when dealing with the notion of certificate we need to specify the precise form of the certificates. This is done by fixing a specific verifier for the respective problem. We use as canonical verifier for the problem SAT the following verifier V_{SAT}:

$$(F, \beta) \in V_{\mathrm{SAT}} \Leftrightarrow \beta \text{ is a satisfying assignment for the CNF-formula } F.$$

Note that another verifier for SAT, such as the verifier V'_{SAT} with padded certificates,

$$(F, \beta\omega) \in V'_{\mathrm{SAT}} \Leftrightarrow (F, \beta) \in V_{\mathrm{SAT}} \wedge \omega \in \Sigma^{|\beta|},$$

could lead to different results (see also [Che03]). For a discussion what makes a verifier a 'natural' choice we also refer to [Krü08], where the notion of a universal verifier is introduced.

The first elementary modification of SAT-formulas that we examine is the addition of a single unit clause. We define as the corresponding modification function

$$ad(F, L) := F \cup \{\{L\}\},$$

where F is a CNF-formula and L is a literal.

Can a satisfying assignment for a Boolean formula F help to find a solution for the altered formula in that a single unit clause is added? An answer to this question has already been given by Liberatore in [Lib04]. Liberatore shows that the problem of deciding ad-modified SAT-instances is exactly as hard as the problem SAT.

Theorem 3.3 ([Lib04]). $\mathrm{MOD}_{ad} V_{\mathrm{SAT}}$ *is* NP-*complete.*

Proof. The proof is already in [Lib04]. We restate it here for convenience. To show NP-hardness of $\mathrm{MOD}_{ad} V_{\mathrm{SAT}}$ we reduce from the NP-hard problem SAT. Therefore we give three polynomial-time computable functions f_1, f_2, and f_3 such that for all $F \in \Sigma^*$ it holds that

$$F \in \text{SAT} \Leftrightarrow (f_1(F), f_2(F), f_3(F)) \in \text{MOD}_{ad}V_{\text{SAT}}. \tag{1}$$

The functions f_1, f_2, and f_3 are defined as follows. If F is no syntactically correct formula, then f_1, f_2, f_3 map to a fixed nonmember of $\text{MOD}_{ad}V_{\text{SAT}}$, say $(\{\{x\}\}, \beta', y)$ with $\beta'(x) = 0$. Equation (1) trivially holds in this case. Thus, we henceforth suppose that F is a syntactically correct CNF-formula.[1]

Let $F = \{C_1, C_2, ..., C_m\}$ and let $Var(F) = \{x_1, ..., x_n\}$. The formula $f_1(F)$ consists of exactly the clauses $(C_1 \vee y), (C_2 \vee y), ..., (C_m \vee y)$, where $y \notin Var(F)$. Obviously, any assignment β with $\beta(y) = 1$ satisfies the formula $f_1(F)$. We define $f_2(F)$ to be any of these assignments, say β with $\beta(x) = 1$, for all $x \in \{x_1, ..., x_n, y\}$. Finally, we set $f_3(F) := \neg y$. Apparently, the functions f_1, f_2, and f_3 are computable in polynomial time in the size of F.

Summarizing the construction above we have:

- $f_1(F) := \bigcup_{i=1}^{m} \{C_i \cup \{y\}\}$,
- $f_2(F) := \beta$, where $\beta(x) = 1$, for all $x \in \{x_1, ..., x_n, y\}$, and
- $f_3(F) := \neg y$.

By construction of f_1 and f_2 it holds that $(f_1(F), f_2(F)) \in V_{\text{SAT}}$. Furthermore, note that the formulas F and $f_1(F) \cup \{\{\neg y\}\}$ are equivalent with respect to satisfiability, that is, the former is satisfiable if and only if the latter is satisfiable. The equivalence

$$F \in \text{SAT} \Leftrightarrow \big(\, (f_1(F), f_2(F)) \in V_{\text{SAT}} \;\; \wedge \;\; ad\big(f_1(F), f_3(F)\big) \in \text{SAT} \, \big)$$

follows. The assertion (1) is immediate. □

Next, we examine the modification adc, which adds a *clause*, not necessarily a unit-clause, to a formula. Formally, we define

$$adc(F, C) := F \cup \{C\}.$$

Adding a unit clause $\{L\}$ to F can be seen as a special case of adding an arbitrary clause C to F. Thus, the corresponding problem $\text{MOD}_{adc}V_{\text{SAT}}$ is NP-complete as well. Consequently, certificates of the original formula are useless as a hint.

Before turning to further results, we illustrate another possible approach to show hardness of $\text{MOD}_{adc}V_{\text{SAT}}$. The modification function adc has an interesting property, that is also shared by many other of the coming modifications: each SAT-instance F can be obtained by polynomially many (in the size of F) applications of adc to an initially trivial formula. That is, the formula $F = \{C_1, ..., C_m\}$ can be constructed in a

[1] We can always treat incorrect input data in such a trivial way. Therefore, in the coming proofs we will always assume that the inputs are given in syntactically correct form.

sequence of formulas $F_1, ..., F_m$, where for example $F_i = \bigcup_{j \leq i}\{C_j\}$. Thus, if one could decide effectively, i.e., in polynomial time, if adc-modified formulas are satisfiable then this yields a polynomial time algorithm for deciding satisfiability of any formula F. Consequently, SAT $\in \mathrm{P}$, and we could conclude that the problem of deciding SAT-instances in which an additional clause is added and a certificate of the original formula is given is exactly as hard, or easy in this case, as deciding satisfiability of the formula from scratch. Namely both problems would belong to P.

There are two technical pitfalls associated with this last approach. The first difficulty is, that not all of the formulas in the sequence $F_1, ..., F_m$ must be satisfiable. Thus, for some of the formulas $F_1, ..., F_m$ no certificate for the original instance can be given. Nevertheless, we can overcome this difficulty if we are able to show that the knowledge $F_i \notin$ SAT is useful for deciding if $F_{i+1} \in$ SAT. This holds for adc and SAT, since adding another clause only further restricts the set of possible satisfying assignments. But this latter argument cannot be used for all the coming modifications[2].

Second, the above iterative argument only yields the result that if $\mathrm{MOD}_{adc}V_{\mathrm{SAT}}$ is in $\mathcal{C}$ then SAT $\in \mathrm{P}^{\mathcal{C}}$. If $\mathcal{C} = \mathrm{P}$ then this result is satisfactory, since $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$. But for other complexity classes, as for example $\mathcal{C} = \mathrm{coNP}$, the valid consequence SAT $\in \mathrm{P}^{\mathrm{coNP}}$ cannot be used to deduce further statements about uselessness of certificates. In contrast, our initial result that $\mathrm{MOD}_{adc}V_{\mathrm{SAT}} \equiv_m^p$ SAT yields such a statement.

After this slight digression, we now show that also for some other SAT-modifications the knowledge of a certificate for the original instance is useless. Similar proofs as the one of Theorem 3.3 can be given for the NP-completeness of $\mathrm{MOD}_c V_{\mathrm{SAT}}$ when the modification function c is

- the deletion of a single literal L from a clause C of a CNF-formula F, formally

$$rmlc(F,(C,L)) := \begin{cases} (F \setminus \{C\}) \cup \{C \setminus \{L\}\}, & \text{if } C \in F \text{ and } L \in C, \\ F, & \text{otherwise,} \end{cases}$$

- the negation of the single literal of a unit clause, formally

$$neg(F,L) := \begin{cases} (F \setminus \{\{L\}\}) \cup \{\{\neg L\}\}, & \text{if } \{L\} \in F, \\ F, & \text{otherwise,} \end{cases} \quad and$$

- generally, the negation of a single literal L in some clause C, formally

$$negl(F,(C,L)) := \begin{cases} (F \setminus \{C\}) \cup \{(C \cup \{\neg L\}) \setminus \{L\}\}, & \text{if } C \in F, L \in C, \\ F, & \text{otherwise.} \end{cases}$$

The respective reduction functions for a proof of NP-hardness are given in Appendix A.

[2]For example, if the modification is the removal of a triple from a 3DM-instance then the set of solutions for the modified instance is not necessarily a subset of the solutions of the original instance (also see Theorem 9.36).

At this point, the reader might be curious if all modifications c of SAT-formulas lead to an NP-complete problem $\mathrm{MOD}_c V_{\mathrm{SAT}}$. The following example demonstrates that this is not the case. Consider the modification that deletes a clause from a formula, formally

$$rmc(F, C) := F \setminus \{C\}.$$

The respective problem $\mathrm{MOD}_{rmc} V_{\mathrm{SAT}}$ belongs to P since it can be verified in polynomial time if the given assignment β satisfies the original formula F. If $(F, \beta) \in V_{\mathrm{SAT}}$ then also $rmc(F, C) \in \mathrm{SAT}$, since β is also a satisfying assignment for $rmc(F, C)$.

The same argument holds when we add a single literal L to a clause C of F, formally

$$adlc(F, (C, L)) := \begin{cases} (F \setminus \{C\}) \cup \{C \cup \{L\}\}, & \text{if } C \in F, \\ F, & \text{otherwise.} \end{cases}$$

Observation 3.4. $\mathrm{MOD}_{rmc} V_{\mathrm{SAT}} \in \mathrm{P}$ *and* $\mathrm{MOD}_{adlc} V_{\mathrm{SAT}} \in \mathrm{P}$.

3.3 The Problem EX3SAT and Interreduction

The next problem for which we examine the complexity of modified instances is the language EX3SAT — a subset of SAT that contains all satisfiable EX3CNF-formulas. We use as verifier for EX3SAT the following verifier V_{EX3SAT}:

$$(F, \beta) \in V_{\mathrm{EX3SAT}} \Leftrightarrow F \text{ is an EX3CNF-formula } \wedge (F, \beta) \in V_{\mathrm{SAT}}.$$

We examine as a first modification the addition or deletion of atomic components of EX3CNF-formulas. Since adding unit clauses to an EX3CNF-formula destroys its EX3CNF-form, we modify the formula by adding or deleting entire 3-clauses. The corresponding modification functions[3] are given by

$$adc(F, C) := \begin{cases} F \cup \{C\}, & \text{if } F \text{ is an EX3CNF-formula and } C \text{ is a 3-clause,} \\ F, & \text{otherwise,} \end{cases}$$

and

$$rmc(F, C) := \begin{cases} F \setminus \{C\}, & \text{if } F \text{ is an EX3CNF-formula,} \\ F, & \text{otherwise.} \end{cases}$$

Similar to SAT, the case where some 3-clause is deleted is easy.

Observation 3.5. $\mathrm{MOD}_{rmc} V_{\mathrm{EX3SAT}} \in \mathrm{P}$.

In contrast, when we add a 3-clause to an EX3CNF-formula, a solution for the original instance does, in general, not help to find a solution for the modified instance.

Theorem 3.6. $\mathrm{MOD}_{adc} V_{\mathrm{EX3SAT}}$ *is* NP-*complete.*

[3]Note that modification functions for different domains may go by the same name, e.g., we define a modification function adc for both, CNF-formulas and EX3CNF-formulas. It will always be clear from the context which modification function has to be applied.

Proof. To prove NP-completeness of $\text{MOD}_{adc}V_{\text{Ex3SAT}}$ it is sufficient to reduce the NP-complete problem $\text{MOD}_{ad}V_{\text{SAT}}$ (see Theorem 3.3) to $\text{MOD}_{adc}V_{\text{Ex3SAT}}$. We call such a reduction between two problems of the form $\text{MOD}_{c}V_{A}$ an *interreduction*. We show that there exist three polynomial-time computable reduction functions f_1, f_2, f_3 such that for all triples $x := (F, \beta, L)$ it holds that

$$x \in \text{MOD}_{ad}V_{\text{SAT}} \Leftrightarrow \big(f_1(x), f_2(x), f_3(x)\big) \in \text{MOD}_{adc}V_{\text{Ex3SAT}}. \qquad (2)$$

In constructing f_1 and f_2 we make use of a well known reduction function f that shows SAT $\leq^p_m$ Ex3SAT, as for example given in [DK00]. We give a short summary on how f is defined:

Let F be a CNF-formula. We give an EX3CNF-formula $F' := f(F)$ that is satisfiable if and only if F is satisfiable. Initially, the formula F' consists of the seven clauses

$$\{w_1, w_2, w_3\}, \{\neg w_1, w_2, w_3\}, \{w_1, \neg w_2, w_3\}, \{w_1, w_2, \neg w_3\},$$
$$\{\neg w_1, \neg w_2, w_3\}, \{\neg w_1, w_2, \neg w_3\}, \text{ and } \{w_1, \neg w_2, \neg w_3\},$$

where $w_1, w_2, w_3 \notin Var(F)$. Note that this formula is satisfied by an assignment β if and only if $\beta(w_1) = \beta(w_2) = \beta(w_3) = 1$.

Now, we describe how to transform a clause $C \in F$ into a set of 3-clauses of F'. Let $\{L_1, ..., L_k\}$ be a clause of F. Dependent on the number of literals k, we add the following 3-clauses to F':

- $k = 1 : \{L_1, \neg w_1, \neg w_2\}$,
- $k = 2 : \{L_1, L_2, \neg w_1\}$,
- $k = 3 : \{L_1, L_2, L_3\}$, and
- $k = 4 : \{L_1, L_2, u\}, \{\neg u, L_3, L_4\}, \{\neg L_3, u, \neg w_1\}, \{\neg L_4, u, \neg w_1\}$,

where u is a new variable not used in the construction so far. For the moment, we postpone the case $k \geq 5$. Correctness of the reduction for the cases $k \in \{1, 2, 3\}$ should be obvious. In case $k = 4$ note that with $\beta(w_1) = 1$ (which needs to hold in order for F' to be satisfiable) the conjunction of the four given clauses is equivalent to

$$(L_1 \vee L_2 \vee u) \wedge (u \Leftrightarrow (L_3 \vee L_4)).$$

Equivalence of the latter formula and the clause $\{L_1, L_2, L_3, L_4\}$ is immediate.

For the case $k \geq 5$ we inductively apply the procedure of the case $k = 4$. It should be an easy task for the reader to verify the equivalence

$$\begin{aligned}\{\{L_1, ..., L_k\}\} &\equiv (L_1 \vee L_2 \vee v) \wedge (v \Leftrightarrow (L_3 \vee \cdots \vee L_k)) \\ &\equiv \{\{L_1, L_2, v\}, \{\neg v, L_3, ..., L_k\}\} \cup \{\{v, \neg L_i\} : 3 \leq i \leq k\},\end{aligned}$$

where v is a new variable never used in the construction so far. We add the clauses $\{L_1, L_2, v\}, \{v, \neg L_3, \neg w_1\}, ..., \{v, \neg L_k, \neg w_1\}$ to F' and inductively apply this procedure to the clause $\{\neg v, L_3, ..., L_k\}\}$. As a result we obtain a formula F' which has at most quadratic size in the size of F. Clearly, the function f that maps a formula F to F' is polynomial-time computable and yields the desired result SAT $\leq_m^p$ Ex3SAT.

Note that the reduction function f has the following beneficial property. Given an assignment β for the SAT-instance F, we can easily construct a satisfying assignment β' for the EX3CNF-formula $f(F)$. This can be done in the following way. For all the variables from $Var(F)$ the assignments β and β' do not differ, that is, $\beta'(x) := \beta(x)$ for all variables $x \in Var(F)$. For each additional variable u introduced by splitting a clause $\{L_1, ..., L_k\}$, $k \geq 4$, we obtain the truth value of u by the equivalence $u \Leftrightarrow L_3 \vee ... \vee L_k$. For the remaining variables w_1, w_2, w_3 of F' we set $\beta'(w_1) = \beta'(w_2) = \beta'(w_3) = 1$. This concludes our summary on the reduction function f.

Returning to our proof of (2), we define the functions f_1, f_2, and f_3 as follows:

- $f_1(F, \beta, L) := f(F) \cup \big\{\{y_1, y_2, L\}, \{y_1, \neg y_2, L\}, \{\neg y_1, y_2, L\}\big\}$,

where y_1 and y_2 are variables that are not contained in $Var(f(F))$,

- $f_2(F, \beta, L) := \beta'$,

where β' is constructed from β as described above, and $\beta'(y_1) = \beta'(y_2) = 1$, and

- $f_3(F, \beta, L) := \{\neg y_1, \neg y_2, L\}$.

Note that f_1,f_2, and f_3 are polynomial-time computable. Furthermore, observe that by construction of f_1 and f_3 we get the equivalence

$$f_1(x) \cup \{f_3(x)\} \equiv f(F) \cup \{\{L\}\}. \tag{3}$$

To prove (2), first suppose that $x \in$ MOD$_{ad}V_{\text{SAT}}$, i.e., $(F, \beta) \in V_{\text{SAT}}$ and $F \wedge L \in$ SAT. If β is a satisfying assignment for F then β' is a satisfying assignment for $f(F)$. Thus $(f_1(x), f_2(x)) \in V_{\text{Ex3SAT}}$. On the other hand, it follows from $F \wedge L \in$ SAT that F has a satisfying assignment β with $\beta(L) = 1$. Thus, $f(F)$ has a satisfying assignment with $\beta(L) = 1$ (truth values of variables from F are preserved) and $f(F) \wedge L \in$ SAT. Using (3), we have that the EX3CNF-formula $f_1(x) \cup \{f_3(x)\}$ is contained in Ex3SAT. Thus $(f_1(x), f_2(x), f_3(x)) \in$ MOD$_{adc}V_{\text{Ex3SAT}}$.

Conversely, assume that $x \notin$ MOD$_{ad}V_{\text{SAT}}$, i.e., (i) $(F, \beta) \notin V_{\text{SAT}}$ or (ii) $F \wedge L \notin$ SAT. In the first case, there exists a clause $C_m = \{L_1, ..., L_{k_m}\}$ in F such that no literal is made true by the assignment β. The reader may verify, that the reduction function f transforms this clause C_m to a set of clauses in which exists a 3-clause that is not satisfied by β', yielding that $(f_1(x), f_2(x)) \notin V_{\text{Ex3SAT}}$. In the second case, we conclude from $F \wedge L \notin$ SAT that either F is not satisfiable or each satisfying assignment of F assigns the truth value 0 to L. By construction of f, either $f(F)$ is not satisfiable or each satisfying assignment of $f(F)$ assigns the truth value 0 to L, which implies that $f(F) \wedge L \notin$ Ex3SAT. Equivalence (3) yields that $f_1(x) \cup \big\{f_3(x)\big\} \notin$ Ex3SAT. In both cases, (i) and (ii), we get that $(f_1(x), f_2(x), f_3(x)) \notin$ MOD$_{ad}V_{\text{Ex3SAT}}$. □

We summarize our findings for satisfiability problems in Table 3.1. The NP-completeness result for MOD$_{negl}V_{\text{Ex3SAT}}$ is proven in Appendix A, as well as the results for the problem 3SAT, a special case of SAT in which each clause has *at most* three literals.

modification function	SAT / 3SAT	Ex3SAT
ad (addition of a unit clause)	NP-complete	-
adc (addition of a clause)	NP-complete	NP-complete
$adlc$ (addition of a literal to a clause)	$\in$ P	-
rm (removal of a unit clause)	$\in$ P	-
rmc (removal of a clause)	$\in$ P	$\in$ P
$rmlc$(removal of a literal from a clause)	NP-complete	-
neg (negation of a unit clause)	NP-complete	-
$negl$ (negation of a literal of a clause)	NP-complete	NP-complete

Table 3.1: Hard and easy cases when deciding modified instances of various satisfiability problems with help of arbitrary certificates.

3.4 Results for Other Problems

Up to now we introduced two techniques to prove that a language MOD$_c V_A$ is NP-complete. In the first setting, we reduce an NP-complete problem A to MOD$_c V_A$, as done in the proof of Theorem 3.3. The other technique involves a reduction from an NP-complete problem MOD$_{c'} V_B$, a so called *interreduction*. This latter technique was used in the proof of Theorem 3.6. We want to apply these both techniques to other problems MOD$_c V_A$. Since there exist hundreds of NP-complete problems, we concentrate on the six basic NP-complete problems listed in the influential book of Garey and Johnson [GJ79]. Namely, these six basic problems are

- EXACTTHREESATISFIABILITY[4] (EX3SAT),
- CLIQUE,
- VERTEXCOVER (VC),
- HAMILTONIANCYCLE (HC),
- THREEDIMENSIONALMATCHING (3DM), and
- PARTITION.

We already discussed the problem EX3SAT. For a formal definition of the other five problems we refer to the respective chapters in Appendix A. For these five problems we consider several modifications, among them the deletion or addition of atomic components of their instances, for instance addition or removal of a single edge of a graph. Our findings for the remaining five basic problems are summarized in Table 3.2. The

[4]Note that the problem we call EX3SAT is called 3SAT in [GJ79]. Since we want to distinguish between the satisfiability problem in which each clause has exactly three literals and the satisfiability problem in which each clause has at most three literals, we use the labelling EX3SAT and 3SAT, respectively.

	modification: add./rmvl.	easy (in P)	NP-complete
VC	of an edge	$\mathrm{MOD}_{rm}V_{\mathrm{VC}}$ [Lib04]	$\mathrm{MOD}_{ad}V_{\mathrm{VC}}$ [Lib04]
CLIQUE	of an edge	$\mathrm{MOD}_{ad}V_{\mathrm{CLIQUE}}$	$\mathrm{MOD}_{rm}V_{\mathrm{CLIQUE}}$
HC	of an edge	$\mathrm{MOD}_{ad}V_{\mathrm{HC}}$	$\mathrm{MOD}_{rm}V_{\mathrm{HC}}$
3DM	of a triple	$\mathrm{MOD}_{ad}V_{\mathrm{3DM}}$	$\mathrm{MOD}_{rm}V_{\mathrm{3DM}}$
PARTITION	of a natural number		$\mathrm{MOD}_{ad}V_{\mathrm{PARTITION}}$ $\mathrm{MOD}_{rm}V_{\mathrm{PARTITION}}$

Table 3.2: Hard and easy cases when deciding modified instances of the problems VC, CLIQUE, HC, 3DM, and PARTITION with help of arbitrary certificates.

respective proofs can be found in Appendix A. Also, the formal definition of the examined modification functions as well as the considered verifiers are given in Appendix A. In addition to the five basic problems, the NP-complete satisfiability problem 1-3SAT, in which each clause needs to be satisfied by exactly one literal, is considered in Appendix A.

Conclusions

In this chapter we showed that in many cases the knowledge of a certificate is completely useless to decide locally modified instances. We introduced problems of the form $\mathrm{MOD}_c V_A$, which characterize the complexity of deciding such modified instances. For many modifications c, which were defined in this chapter, we could show that $\mathrm{MOD}_c V_{\mathrm{SAT}}$ and $\mathrm{MOD}_c V_{\mathrm{Ex3SAT}}$ are NP-complete. This is equivalent to say that a certificate for an SAT- or EX3SAT-instance is a useless hint when the instance is modified by c. For some other modifications we could show the converse, namely that certificates allow to compute efficiently a solution of slightly modified instances (see also Table 3.1).

We showed our uselessness results, i.e., NP-completeness of a problem $\mathrm{MOD}_c V_A$, in two different ways. First, we reduced a standard NP-complete problem, namely SAT, to a problem of the type $\mathrm{MOD}_c V_A$. Second, we reduced an NP-complete problem of the type $\mathrm{MOD}_c V_A$ to another problem of this type. We called the latter form of reduction an interreduction. We also indicated that these two techniques suffice to show that certificates are a useless hint for instances of the six basic problems from [GJ79], when these instances are appropriately modified (see also Table 3.2).

Chapter 4

Selected Solution as Hint

The completeness proofs given in the last chapter have a drawback. What we have actually proven is only that there *exist* certificates that are useless as a hint, but not that *any* certificate is useless. We illustrate the difference at an example. Consider the problem SAT and a CNF-formula $\{C_1, ..., C_m\}$. We have proven that an assignment β with $\beta(a) = 1$ is a useless hint when deciding if the formula $ad((C_1 \vee a) \wedge ... \wedge (C_m \vee a), \neg a)$ is satisfiable (see proof of Theorem 3.3). Nevertheless, it is obvious that a satisfying assignment for $\{C_1, ..., C_m\}$ would have been a much better hint. This example shows that, although there is some assignment for the original formula that does not help to solve the modified instance, there might exist other assignments that are more helpful. The question of interest in this chapter is: Does there always exist such a useful hint?

4.1 Problem Formalization

To formalize the idea of different possible hints we introduce the notion of certificate functions. A certificate function h for a language A is a function that, given an instance x of A, outputs a certificate for this instance, if there exists one. If there is no certificate for the instance x then h outputs the empty word ϵ. Since the form of the certificate depends on the verifier we use, we define the notion of certificate functions with respect to a given verifier.

Definition 4.1. *Let V_A be a verifier for a language A. The total function h is a certificate function for V_A if and only if for all $x \in \Sigma^*$ it holds that*

$$h(x) \begin{cases} \in V_A(x), & \text{if } x \in A, \\ = \epsilon, & \text{otherwise.} \end{cases}$$

Let $cert(V_A)$ denote the set of all certificate functions for V_A. Let $cert$ be the set of all certificate functions, i.e., $cert := \bigcup_{V_A \text{ verifier}} cert(V_A)$.

Note that we do not restrict the complexity of certificate functions. They may even be nonrecursive. This model is insufficient if we actually want to implement an iterative algorithm that computes helpful certificates for later use. On the other hand, if we can show that no certificate function yields helpful hints then we have a stronger result compared to complexity-restricted hint functions. The important results in this chapter are of the latter kind.

Until now we avoided to use the term 'modification problem'; we now formally introduce this notion.

Definition 4.2. *Let c be a modification function and V_A be a verifier for a language $A \in \mathrm{NP}$. The modification problem for V_A and c is the pair (c, V_A).*

A modification problem only describes the modification c to be applied, the problem A we are dealing with (via $L(V_A)$), and the precise form of the certificates.

In this chapter we are interested in the question how easy it can be to decide if $c(x,m) \in A$ when an appropriately chosen certificate function is used to give hints. We want to express such results of easiness in terms of (c, V_A) being an element of certain complexity classes, e.g., oracle classes or nonuniform complexity classes. Unfortunately, such well studies complexity classes seem to be insufficient for this purpose. For that reason, we introduce new complexity classes $\mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$. Containment of a modification problem (c, V_A) in $\mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ expresses the fact that the decision whether a modified instance $c(x,m)$ is a member of A is a problem that belongs to $\mathcal{C}$ when adequate certificates are given. Since in later chapters we want to examine modification problems for more general functions than certificate functions, we give the following general definition, from which the definition of the classes $\mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ can easily be derived.

Definition 4.3. *Let $\mathcal{C}$ be a complexity class and $\mathcal{F}$ be a class of functions. The modification problem (c, V_A) belongs to $\mathcal{C}^{\in}_{\text{MOD}}/\mathcal{F}$ if and only if*

$$(\exists h \in \mathcal{F})(\exists C \in \mathcal{C})(\forall x, m \in \Sigma^*)\Big[\, x \in A \Rightarrow \big[c(x,m) \in A \Leftrightarrow \big(x, h(x), m\big) \in C\big]\Big].$$

The informal idea behind Definition 4.3 is the following. When a modification problem (c, V_A) is contained in the class $\mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ it follows that there exists a Turing machine (or an algorithm) that (i) obeys the constraints of $\mathcal{C}$ and (ii) that decides containment of modified instances of the form $c(x,m)$ in A only by the knowledge of the original instance x, the modification m that is applied to x, and some clever chosen certificate $h(x)$ of the original instance x. We want this algorithm to work correctly whenever the original instance x belongs to A, so that $h(x)$ actually can return a certificate for x. The superscript $\in$ in $\mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ accounts for this last property that the original instance is an element of A. In Chapter 5 we discuss the problem of deciding modified instances when the promise is given, that no solution for the original instance exists.

It is obvious by Definition 4.3 that the classes $\mathcal{C}^{\in}_{\text{MOD}}/\mathcal{F}$ satisfy the following monotonicity property.

Observation 4.4. *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes and $\mathcal{F}$ be a class of functions. If $\mathcal{C} \subseteq \mathcal{D}$ then $\mathcal{C}^{\in}_{\text{MOD}}/\mathcal{F} \subseteq \mathcal{D}^{\in}_{\text{MOD}}/\mathcal{F}$.*

Also note the following. If V_A is a verifier for a language $A \in \mathrm{NP}$ and c is a modification function then the modification problem (c, V_A) is an element of $\mathrm{NP}^{\in}_{\mathrm{MOD}}/cert(V_A)$. This formally expresses the fact that using a hint should only make easier the problem of deciding if a modified instance belongs to a language $A \in \mathrm{NP}$.

Observation 4.5. *Let V_A be a verifier for a language $A \in \mathcal{C}$, $\mathcal{C}$ be closed under $\leq^p_m$-reduction, and c be a modification function. Then $(c, V_A) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$.*

Proof. Choose the predicate C from Definition 4.3 as

$$(x, \pi, m) \in C \Leftrightarrow c(x, m) \in A$$

for all $\pi \in \Sigma^*$. Obviously $C \in \mathcal{C}$. The assertion follows. □

The class $\mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$ can best be compared to non-uniform complexity classes, in particular to $\mathcal{C}/poly$. However, there are some major differences. For $\mathcal{C}/poly$ the given advice only depends on the size of the input. In contrast, for the class $\mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$ the hint depends on the whole original instance, not just its size. On the other hand, the class $\mathcal{C}/poly$ has more freedom in choosing the advice; every polynomial string may be chosen. In contrast, for the class $\mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$ the hint must be a certificate for the original instance x.

In this chapter we are especially interested in modification problems (c, V_A) of NP-complete problems A that belong to $\mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$ for some $\mathcal{C} \subset \mathrm{NP}$, i.e., modifications problems that actually benefit from selected hints. Unfortunately, we find such positive examples only for trivial cases, i.e., cases in which every certificate of the original instance is also a certificate for the modified instance. This will be explained in detail in the next section.

4.2 Hint-independent Reducibility and SAT

Recall the verifier V_{SAT} for SAT from Section 3.2. We consider the modification functions rm, and neg defined there.

First, we consider the modification function rm, which removes a unit clause. Since any solution π of the original instance is a solution for the modified instance, every such solution π is a helpful hint. Therefore, all certificate functions $h \in cert(V_{\mathrm{SAT}})$ give helpful hints. Generally, we can state the following observation.

Observation 4.6. *If* $\mathrm{MOD}_c V_A \in \mathcal{C}$ *then* $(c, V_A) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)$.

Proof. Let $\mathrm{MOD}_c V_A \in \mathcal{C}$, $A = L(V_A)$, and h be a certificate function for V_A. If $x \in A$ we have $(x, h(x)) \in V_A$ (since h is a certificate function). Thus, if $x \in A$ it holds for all $m \in \Sigma^*$ that

$$\begin{aligned}(x, h(x), m) \in \mathrm{MOD}_c V_A &\Leftrightarrow (x, h(x)) \in V_A \wedge c(x, m) \in A \\ &\Leftrightarrow c(x, m) \in A.\end{aligned}$$

Thus, for all $x, m \in \Sigma^*$ it holds that

$$(\forall h \in cert(V_A))\big[x \in A \Rightarrow \big(c(x,m) \in A \Leftrightarrow (x, h(x), m) \in \text{MOD}_c V_A\big)\big].$$

The assertion follows immediately by Definition 4.3. □

As a consequence of this observation, we disregard in the coming any modification problem (c, V_A) for which already $\text{MOD}_c V_A$ belongs to P.

In contrast to the easy case rm, the modification function neg, which negates the sole literal of a unit clause, allows for no helpful certificates. We show this in the following way. We prove that under the assumption that $(neg, V_{\text{SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}})$ for some complexity class $\mathcal{C}$ (e.g., choose $\mathcal{C} = \text{P}$), the original problem SAT is in $\mathcal{C}$. Consequently, the modification problem is as easy as the original problem SAT and therefore hints are useless.

Theorem 4.7. *Let $\mathcal{C}$ be a complexity class that is closed under $\leq^p_m$-reduction. If $(neg, V_{\text{SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}})$ then* SAT $\in \mathcal{C}$.

Proof. The proof is already in [Lib04]. We restate it here for convenience. Let $\mathcal{C}$ be closed under $\leq^p_m$-reduction and let $(neg, V_{\text{SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}})$ via the certificate function h for V_{SAT} and the predicate $C \in \mathcal{C}$, that is, for all $F, L \in \Sigma^*$ it holds that

$$F \in \text{SAT} \Rightarrow \big[neg(F, L) \in \text{SAT} \Leftrightarrow (F, h(F), L) \in C\big]. \tag{1}$$

Let F be an arbitrary CNF-formula, $Var(F) = \{x_1, ..., x_n\}$, and $a \notin Var(F)$ and F' be the CNF-formula that can be derived from the formula

$$a \wedge [(x_1 \wedge ... \wedge x_n \wedge a) \vee (F \wedge \neg a)]$$

by straightforward expansion. Note that the formula F' still contains the unit clause $\{a\}$. Also, F' has at most quadratic size in F and can be computed in polynomial time. Apparently, F' has exactly one certificate, namely the assignment β that assigns 'true' to all the variables of F'. Thus $F' \in$ SAT and by (1) we get

$$neg(F', a) \in \text{SAT} \Leftrightarrow (F', h(F'), a) \in C.$$

Since F' has exactly one assignment, namely β, the certificate function h has no choice but $h(F') := \beta$. Thus,

$$neg(F', a) \in \text{SAT} \Leftrightarrow (F', \beta, a) \in C.$$

By propositional logic it follows that

$$\begin{aligned} F \in \text{SAT} &\Leftrightarrow \neg a \wedge [(x_1 \wedge ... \wedge x_n \wedge a) \vee (F \wedge \neg a)] \in \text{SAT} \\ &\Leftrightarrow neg(F', a) \in \text{SAT} \\ &\Leftrightarrow (F', \beta, a) \in C. \end{aligned}$$

Since F', β and a are polynomial time computable from F we have shown that SAT is $\leq^p_m$-reducible to C. Since $C \in \mathcal{C}$ and $\mathcal{C}$ is closed under $\leq^p_m$-reduction we get SAT $\in \mathcal{C}$. □

To restate Theorem 4.7 in other words: If there exists a certificate function such that a satisfying assignment for the modified instances can be found with the complexity bound given by $\mathcal{C}$ then the original satisfiability problem also can be solved with this complexity bound, not using any hint at all. This shows that when we negate a literal of a propositional formula then no solution exists that helps to decide satisfiability of the modified instance.

We summarize the key features that were used in the proof of Theorem 4.7. The essential part of the proof was to find a formula $f_1(F)$ (i.e., the formula F' in the last proof) such that

- there exists a modification $f_3(F)$ (i.e., the unit clause a in the last proof) such that

$$F \in \mathrm{SAT} \Leftrightarrow neg(f_1(F), f_3(F)) \in \mathrm{SAT},$$

- $f_1(F)$ has exactly one solution, namely $f_2(F) = \beta$, and
- $f_1, f_2, f_3 \in \mathrm{FP}$.

These properties are sufficient to show similar results for other problems. To show similar results in a succinct way, we translate these three properties to a notion of reduction. In comparison to the conditions given above we generalize in the following way: we allow different problems to be mapped to each other, e.g., we allow the graph problem B being reduced to a modification problem of formulas.

Definition 4.8. *Let V_A be a verifier for some language $A \in \mathrm{NP}$ and c be a modification function. We say that a language B is hint-independently polynomial-time reducible to (c, V_A), short $B \leq^p_{hi} (c, V_A)$, if and only if there exist three reduction functions $f_1, f_2, f_3 \in \mathrm{FP}$ such that for all $x \in \Sigma^*$ it holds that*

- $x \in B \Leftrightarrow c(f_1(x), f_3(x)) \in A$ *and*
- $V_A(f_1(x)) = \{f_2(x)\}$.

As already motivated, the functions $f_1(F) := a \wedge [(x_1 \wedge ... \wedge x_n \wedge a) \vee (F \wedge \neg a)]$, $f_2(F) := \beta$, where $\beta(x) = 1$ for all $x \in \{x_1, ..., x_n, a\}$, and $f_3(F) := a$ yield a hint-independent reduction from SAT to (neg, V_{SAT}), i.e., a reduction to SAT-instances that only allow one choice for a (selected) solution.

Observation 4.9. $\mathrm{SAT} \leq^p_{hi} (neg, V_{\mathrm{SAT}})$.

Using the notion of hint-independent reduction we are able to generalize Theorem 4.7.

Theorem 4.10. *Let V_A be a verifier for some language $A \in \mathrm{NP}$ and let $\mathcal{C}$ be some complexity class that is closed under $\leq^p_m$-reduction. Then for all languages B it holds that*

$$\big(B \leq^p_{hi} (c, V_A) \wedge (c, V_A) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V_A)\big) \Rightarrow B \in \mathcal{C}.$$

Proof. Let $(c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ and let $B \leq^p_{hi} (c, V_A)$ via $f_1, f_2, f_3 \in \text{FP}$. From $B \leq^p_{hi} (c, V_A)$ we derive that

- $x \in B \Leftrightarrow c(f_1(x), f_3(x)) \in A$ and
- $V_A(f_1(x)) = \{f_2(x)\}$.

From $(c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ we conclude that there exist a certificate function h for V_A and a predicate $C \in \mathcal{C}$ such that for all $x, m \in \Sigma^*$ it holds that

$$x \in A \Rightarrow \big[c(x, m) \in A \Leftrightarrow (x, h(x), m) \in C\big].$$

Since $V_A(f_1(x)) = \{f_2(x)\}$ it follows that $f_1(x) \in A$ (hence $V_A(f_1(x))$ is nonempty) and $h(f_1(x)) = f_2(x)$ (hence $f_1(x)$ has only one certificate). Therefore,

$$\begin{aligned} (f_1(x), f_2(x), f_3(x)) \in C &\Leftrightarrow (f_1(x), h(f_1(x)), f_3(x)) \in C \\ &\Leftrightarrow c(f_1(x), f_3(x)) \in A \\ &\Leftrightarrow x \in B. \end{aligned}$$

Thus $B \leq^p_m C$, and since $\mathcal{C}$ is closed under $\leq^p_m$-reduction we get $B \in \mathcal{C}$. □

We apply Theorem 4.10 in the following way. We show that $B \leq^p_{hi} (c, V_A)$ for two NP-complete problems $A = L(V_A)$ and B. We then conclude that even a well chosen certificate is useless if we want to decide whether the modified instance $c(x, m)$ belongs to A. This suffices because assuming that $(c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ the NP-complete language B would be an element of $\mathcal{C}$ (Theorem 4.10). Consequently, $\text{NP} \subseteq \mathcal{C}$ and also $A \in \mathcal{C}$. But then, deciding modified instances of A is not harder than deciding the original problem A, not using any hint at all.

As already shown in [Lib04], selected hints are also useless when the modification function is the function $ad\&rm$, that deletes a clause from the formula F and simultaneously adds another clause to the formula. Using the notion of hint-independent reducibility, we are able to succinctly rephrase this result as

Corollary 4.11 ([Lib04]). $\text{SAT} \leq^p_{hi} (ad\&rm, V_{\text{SAT}})$.

Proof. The proof is a carbon copy of the proof for Theorem 4.7. Only the function f_3 has to be slightly altered. □

Summarizing the results for modification problems of SAT-instances we can say the following. When an arbitrary solution is given as a hint we can show that this hint is useless if a unit-clause is added(Theorem 3.3). In contrast, when given a selected solution as hint we can show that this hint is useless if

- a unit clause is negated or
- a unit clause is added and simultaneously another unit-clause is deleted.

It is not clear if a smaller modification, e.g., the modification ad, suffices to render selected hints useless. But, we will show in a later section that the technique of hint independent reducibility is probably inappropriate to answer this question (Theorem 4.17).

4.3 Hint-independent Interreducibility and EX3SAT

Recall the verifier V_{Ex3SAT} for EX3SAT from Section 3.3. We consider the modification function $negl$, which negates a certain literal of some 3-clause of F. The formal definition of $negl$ can be adapted from the function $negl$ for SAT-instances.

For the modification function $negl$ we aim to show that no certificate function helps to decide if a modified EX3CNF-formula is satisfiable. But the techniques that were introduced in the last section cannot be applied in a straightforward manner. Thus, we present another possibility for proving uselessness of selected hints. This new method involves the following reduction of a modification problem (c, V_A) to another modification problem (c', V_B).

Definition 4.12. *Let* (c, V_A) *and* (c', V_B) *be two modification problems. The problem* (c, V_A) *is hint-independently interreducible to* (c', V_B) *if and only if there exist three functions* $g_1, g_2, g_3 \in \text{FP}$ *such that for all* $x, \pi, m \in \Sigma^*$ *it holds that*

- $V_A(x) = \{\pi\} \Rightarrow V_B(g_1(x, \pi, m)) = \{g_2(x, \pi, m)\}$,

that is, the single certificate for x *can easily be translated to the corresponding single certificate for* $g_1(x, \pi, m)$*, and*

- $c(x, m) \in A \Leftrightarrow c'\big(g_1(x, \pi, m), g_3(x, \pi, m)\big) \in B$ *(compatibility of* c *and* c'*).*

We write $(c, V_A) \leq^p_{hi} (c', V_B)$*. The meaning of* $\leq^p_{hi}$ *(in comparison to Definition 4.8) should be clear from the context.*

We give a few remarks on this last definition. The second property in Definition 4.12 assures that the modification functions c and c' are *compatible* with respect to this reduction, that is, any modification m for the instance x can be translated to a corresponding modification $g_3(x, \pi, m)$ for the instance $g_1(x, \pi, m)$.

Closely related to the notion of hint-independent interreducibility are the notions of parsimonious reduction and structure preserving reduction. A parsimonious reduction is a reduction that preserves the number of certificates. A structure preserving reduction is a parsimonious reduction that is accompanied by a polynomial-time computable function g that translates certificates of an instance x to certificates of $f(x)$. Formally:

Definition 4.13 ([Sim75]). *Let* V_A *and* V_B *be two verifiers for problems* A *and* B*, respectively. We say that* A *is parsimoniously polynomial-time reducible to* B *via* f *w.r.t.* V_A *and* V_B *if and only if*

- $(\forall x \in \Sigma^*)[|V_A(x)| = |V_B(f(x))|]$ *and*
- $f \in \text{FP}$.

Definition 4.14 ([LL78]). *Let* V_A *and* V_B *be two verifiers for problems* A *and* B*, respectively. A reduction* $A \leq B$ *via a reduction function* f *is called structure preserving w.r.t.* V_A*,*V_B*, and a function* $g \in \text{FP}$ *if and only if*

- *A is parsimoniously polynomial-time reducible to B via f w.r.t. V_A and V_B and*
- $(x, \pi) \in V_A \Rightarrow (f(x), g(x, \pi)) \in V_B$. [1]

To establish a connection between hint-independent interreduction and structure preserving reduction we note the following. Let f be a structure preserving reduction w.r.t. V_A,V_B, and a function $g \in \mathrm{FP}$. Then f and g induce a reduction that satisfies the first property of Definition 4.12. Simply choose

- $g_1(x, \pi, m) := f(x)$ and
- $g_2(x, \pi, m) := g(x, \pi)$.

So results about structure preserving reductions are of some interest in the remainder of this work. In detail, structure preserving reductions that additionally satisfy the second property of Definition 4.12 (compatibility), i.e., reductions for which a modification of the type c in an instance x translates easily to a modification of type c' in the instance $f(x)$, are frequently used henceforth.

We are now prepare to attack our initial problem, namely the problem to show that the modification function $negl$ allows for no useful hints. We aim to show that $(neg, V_{\mathrm{SAT}}) \leq^p_{hi} (negl, V_{\mathrm{Ex3SAT}})$. But before proving this fact, we make clear in which way this result justifies our initial assertion that hints are useless for $(negl, V_{\mathrm{Ex3SAT}})$. For our argumentation we need the following Theorem 4.15, which states a transitivity result for $\leq^p_{hi}$-reduction. Note that there are different kinds of $\leq^p_{hi}$-reduction involved in this theorem, hint-independent reduction and hint-independent interreduction.

Theorem 4.15. *Let $A, B, C \in \mathrm{NP}$ and let V_B and V_C be verifiers for B and C, respectively. Let c and c' be modification functions. Then*

$$\big(A \leq^p_{hi} (c, V_B) \wedge (c, V_B) \leq^p_{hi} (c', V_C)\big) \Rightarrow A \leq^p_{hi} (c', V_C).$$

Proof. Let A, B, C, V_B, V_C, c, and c' be as stated above. Let $A \leq^p_{hi} (c, V_B)$ via three functions $f_1, f_2, f_3 \in \mathrm{FP}$ and $(c, V_B) \leq^p_{hi} (c', V_C)$ via three functions $g_1, g_2, g_3 \in \mathrm{FP}$. By assumption $A \leq^p_{hi} (c, V_B)$ we have that for all $x \in \Sigma^*$ it holds that

$$x \in A \Leftrightarrow c(f_1(x), f_3(x)) \in B$$

and $V_B(f_1(x)) = \{f_2(x)\}$. Using the assumption that $(c, V_B) \leq^p_{hi} (c', V_C)$ we conclude that $V_C\Big(g_1\big(f_1(x), f_2(x), m\big)\Big) = \Big\{g_2\big(f_1(x), f_2(x), m\big)\Big\}$ for all $m \in \Sigma^*$, and in particular for $m = f_3(x)$. We also conclude that

$$x \in A \Leftrightarrow c'\Big(g_1\big(f_1(x), f_2(x), f_3(x)\big), g_3\big(f_1(x), f_2(x), f_3(x)\big)\Big) \in C.$$

[1]Note that the converse direction $(x, \pi) \in V_A \Leftarrow (f(x), g(x, \pi)) \in V_B$ follows from these two conditions.

We set $\ell_i(x) := g_i(f_1(x), f_2(x), f_3(x))$, for $i = 1, 2, 3$. Consequently, $V_C(\ell_1(x)) = \{\ell_2(x)\}$ and

$$x \in A \Leftrightarrow c'(\ell_1(x), \ell_3(x)) \in C.$$

The assertion follows from the fact that $\ell_1, \ell_2, \ell_3 \in \mathrm{FP}$. □

We apply Theorem 4.15 to the modification problem $(negl, V_{\text{Ex3SAT}})$ in the following way. As shown in Section 4.2, we have that SAT $\leq^p_{hi}$ (neg, V_{SAT}). If we are able to prove that $(neg, V_{\text{SAT}}) \leq^p_{hi} (negl, V_{\text{Ex3SAT}})$ then Theorem 4.15 yields that SAT $\leq^p_{hi}$ $(negl, V_{\text{Ex3SAT}})$. As we discussed in Section 4.2, we may interpret this last result as that no certificate function for the problem $(negl, V_{\text{Ex3SAT}})$ yields helpful hints. It remains to prove

Theorem 4.16. $(neg, V_{\text{SAT}}) \leq^p_{hi} (negl, V_{\text{Ex3SAT}})$.

Proof. We use the reduction function f from the proof of Theorem 3.6. The reader may verify that this reduction via f is structure preserving with respect to some certificate mapping function g. We may choose g such that corresponding assignments in F and $f(F)$ do not differ at the variables from F. More detailed knowledge on the reduction function f is not necessary in the remainder of this proof.

We only consider the case that the unit clause L that is going to be negated is actually contained in the original formula F. The simple idea of the proof would otherwise be unnecessarily dissembled. It is easy to modify the following proof such that it also holds for the case in which L is no part of F.

First, we define the function g_1. Let F be a CNF-formula and let $y_1, y_2 \notin Var(F)$. We define g_1 by

$$\begin{aligned} g_1(F, \beta, L) \quad := \quad & f(F \setminus \{\{L\}\}) \cup \\ & \{\{L, y_1, y_2\}, \{L, \neg y_1, y_2\}, \{L, y_1, \neg y_2\}, \{L, \neg y_1, \neg y_2\}\} \cup \\ & \{\{\neg L, y_1, y_2\}, \{\neg L, \neg y_1, y_2\}, \{\neg L, y_1, \neg y_2\}\}. \end{aligned}$$

Since f is structure preserving w.r.t. g it is obvious that g_1 is structure preserving w.r.t. the certificate mapping function g_2 that is defined by $g_2(F, \beta, L) := \beta'$, where

$$\beta'(x) = \begin{cases} g(\beta)(x), & \text{if } x \in Var(F), \\ 1, & \text{if } x \in \{y_1, y_2\}. \end{cases}$$

This shows that the first property of Definition 4.12 holds.

We now show that also the second property of Definition 4.12 holds, namely that the negation of unit clause in a CNF-formula F can be translated to a negation of a certain literal in the Ex3SAT-instance $g_1(F, \beta, L)$. Formally, we need to show that there exists $g_3 \in \mathrm{FP}$ such that

$$neg(F, L) \in \text{SAT} \Leftrightarrow negl(g_1(F, \beta, L), g_3(F, \beta, L)) \in \text{Ex3SAT}. \qquad (2)$$

We define the function g_3 as

$$g_3(F, \beta, L) := (\{L, \neg y_1, \neg y_2\}, L).$$

modification function	SAT / 3SAT	Ex3SAT	1-3SAT
$\boldsymbol{ad}$ (addition of a unit cl.)	?	-	?
$\boldsymbol{adc}$ (addition of a clause)	?	?	?
$\boldsymbol{adlc}$ (add. of a lit. to a cl.)	easy	-	$\leq_{hi}^{p}$
$\boldsymbol{rm}$ (removal of a unit cl.)	easy	-	easy
$\boldsymbol{rmc}$ (removal of a clause)	easy	easy	easy
$\boldsymbol{rmlc}$(rem. of a lit. from a cl.)	?	-	$\leq_{hi}^{p}$
$\boldsymbol{neg}$ (negation of a unit cl.)	$\leq_{hi}^{p}$	-	$\leq_{hi}^{p}$
$\boldsymbol{negl}$ (neg. of a lit. of a cl.)	$\leq_{hi}^{p}$	$\leq_{hi}^{p}$	$\leq_{hi}^{p}$

Table 4.1: Hard, easy, and uncertain cases when deciding modified instances of various satisfiability problems with help of selected certificates.

In other words, we modify $g_1(F, \beta, L)$ by negating in the clause $\{L, \neg y_1, \neg y_2\}$ the literal L. Note that

$$negl(g_1(F, \beta, L), g_3(F, \beta, L)) \in \text{Ex3SAT} \Leftrightarrow f(F \setminus \{\{L\}\}) \cup \{\{\neg L\}\} \in \text{SAT}. \quad (3)$$

It remains to prove (2). To show the $\Rightarrow$ direction of (2) let $neg(F, L) \in$ SAT and let β be a satisfying assignment of $F \setminus \{\{L\}\} \cup \{\{\neg L\}\} \equiv neg(F, L)$. It follows that $F \setminus \{\{L\}\} \in$ SAT and that $\beta(L) = 0$. Thus $f(F \setminus \{\{L\}\}) \in$ Ex3SAT with $g(\beta)$ as a satisfying assignment. Since g preserves the truth value of variables from F we conclude that $\beta'(L) = 0$. Thus $f(F \setminus \{\{L\}\}) \cup \{\{\neg L\}\} \in$ SAT and by (3) we conclude that $negl(g_1(F, \beta, L), g_3(F, \beta, L)) \in$ Ex3SAT.

For the other direction, suppose that $negl(g_1(F, \beta, L), g_3(F, \beta, L)) \in$ Ex3SAT. Let β' be a satisfying assignment for $neg(g_1(F, \beta, L), g_3(F, \beta, L))$. By (3) we have that $f(F \setminus \{\{L\}\})$ is satisfiable with an assignment β, where $\beta(L) = 0$. Thus $F \setminus \{\{L\}\} \cup \{\{\neg L\}\}$ is satisfiable and therefore $neg(F, L) \in$ SAT. □

Table 4.1 summarizes the results for satisfiability problems that were found in the last sections. It also contains additional results, which are proven in Appendix A. In Table 4.1 the shortcut $\leq_{hi}^{p}$ stands for the fact that the respective modification problem can be hint-independently reduced from some NP complete problem, in our case the problem SAT. For these problems a solution for the original instance yields no helpful hint. The shortcut 'easy' symbolizes that the respective modification problem (c, V_A) belongs to $\mathrm{P}^{\in}_{\mathrm{MOD}}/cert(V_A)$.

As the question marks in Table 4.1 indicate, the usefulness of selected hints is not clear for all kinds of modification functions. But, to show uselessness of selected hints for these cases it is likely that we have to develop new techniques. We are able to show that, unless $\mathrm{P} = \mathrm{NP}$, no $\leq_{hi}^{p}$-reduction from an NP-complete problem to (c, V_A) exists, when c and A are the respective modification function and language from a '?'-entry in Table 4.1. For the moment, we just state this result for the case $c = ad$ and $A =$ SAT.

Theorem 4.17. *Let B be an* NP-*complete problem. Then*

$$B \leq_{hi}^{p} (ad, V_{\text{SAT}}) \Rightarrow \mathrm{P} = \mathrm{NP}.$$

Proof. Let B be an NP-complete problem. Suppose that $B \leq_{hi}^{p} (ad, V_{\mathrm{SAT}})$. Thus, there exist $f_1, f_2, f_3 \in \mathrm{FP}$ such that $V_{\mathrm{SAT}}(f_1(F)) = \{f_2(F)\}$ and

$$\begin{aligned} x \in B &\Leftrightarrow ad(f_1(x), f_3(x)) \in \mathrm{SAT} \\ &\Leftrightarrow (\exists \pi \in \Sigma^*) \left[\left(ad((f_1(x), f_3(x)), \pi\right) \in V_{\mathrm{SAT}}\right] \\ &\Leftrightarrow \left(ad(f_1(x), f_3(x)), f_2(x)\right) \in V_{\mathrm{SAT}}. \end{aligned}$$

The last equivalence is owing to the fact that adding a unit clause only restricts the possible set of satisfying assignments, and since $f_1(x)$ has exactly one satisfying assignment $f_2(x)$, the formula $ad(f_1(x), f_3(x))$ can only be satisfied by this assignment $f_2(x)$. The last expression in the above sequence of equivalences clearly is decidable in polynomial time — only a verifier, a modification function, and the functions f_1, f_2, f_3 are involved. Thus $B \in \mathrm{P}$, and since B is NP-complete we get $\mathrm{P} = \mathrm{NP}$. $\square$

We are able to show a results similar to Theorem 4.17 also for the other '?'-cases of Table 4.1. The essential part of the proof of Theorem 4.17 relied on the property that the set of certificates of a formula $f_1(x)$ is restricted by the modification function ad. In other words, we used that $V_{\mathrm{SAT}}(ad, (F, L)) \subseteq V_{\mathrm{SAT}}(F)$ for all $F, L \in \Sigma^*$. We generalize Theorem 4.17 in the following way.

Corollary 4.18. *Let B be an NP-complete problem and (c, V_A) be a modification problem such that $V_A(x) \supseteq V_A(c(x, m))$, for all $x, m \in \Sigma^*$. Then*

$$B \leq_{hi}^{p} (c, V_A) \Rightarrow \mathrm{P} = \mathrm{NP}.$$

The proof for the corollary can be translated mutatis mutandis from the proof of Theorem 4.17. Its not difficult to see, that the property $V_A(x) \supseteq V_A(c(x, m))$ holds for all the '?'-cases in Table 4.1. Thus, unless $\mathrm{P} = \mathrm{NP}$, we wont be able to prove for any of these modification problems the uselessness of selected hints by using hint-independent reductions.

For the problems CLIQUE, VC, HC, 3DM and PARTITION we use the techniques of hint-independent reduction and hint-independent interreduction to obtain the results that are summarized in Table 4.2. The respective proofs can be found in Appendix A. As usual, an 'easy'-entry means that the corresponding modification problem (c, V_A) belongs to $\mathrm{P}^{\in}_{\mathrm{MOD}}/cert(V_A)$, the shortcut $\leq_{hi}^{p}$ stands for the fact that $B \leq_{hi}^{p} (c, V_A)$ for some NP-complete set B, and '?' symbolizes the fact that uselessness of selected hints is not likely to be provable via $\leq_{hi}^{p}$-reductions.

We give a final remark regarding structure preserving reductions. Note that a structure preserving reduction from SAT to another NP-complete problem $L(V_B)$ via a reduction function f often yields an upper bound for the amount of modification that is necessary to make selected hints useless. For most of these structure preserving reduction it holds that the modification neg of a formula F can be translated to a modification of a certain type c' in the B-instance $f(F)$. For this choice of modification c', the second property of Definition 4.12 is satisfied. It follows that $(neg, V_{\mathrm{SAT}}) \leq_{pi}^{p} (c', V_B)$.

	easy	?	$\leq^p_{hi}$
VC	(rm, V_{VC}) [Lib04]	(ad, V_{VC})	$(ad\&rm, V_{\text{VC}})$
CLIQUE	(ad, V_{CLIQUE})	(rm, V_{CLIQUE})	$(ad\&rm, V_{\text{CLIQUE}})$
HC	(ad, V_{HC})	(rm, V_{HC})	$(ad\&rm, V_{\text{HC}})$
3DM	(ad, V_{3DM})	(rm, V_{3DM})	$(ad\&rm, V_{\text{3DM}})$
PARTITION			$(ad, V_{\text{PARTITION}})$ $(rm, V_{\text{PARTITION}})$

Table 4.2: Hard, easy, and uncertain cases when deciding modified instances of the problems VC, CLIQUE, HC, 3DM, and PARTITION with help of selected certificates.

The above argument also holds when we choose an other modification problem than (neg, V_{SAT}) as our starting point — the only condition on this other modification problem (c, V_B) is that selected certificates are not useful, i.e., $A \leq^p_{hi} (c, V_B)$ for some NP-complete problem A.

4.4 A Connection Between (c, V_A) and $\text{MOD}_c V_A$

In Chapter 3 we showed that there *exists* a certificate that is useless for deciding formulas in which a unit clause is negated. In this chapter we showed that an even stronger statement holds, namely that *any* certificate is useless. Since the first statement is a special case of the latter, we expect that the formal results for uselessness of selected hints, e.g., SAT $\leq^p_{hi} (neg, V_{\text{SAT}})$, translate to uselessness results for arbitrary hints, e.g., NP-completeness of $\text{MOD}_{neg} V_{\text{SAT}}$.

Although it might seem obvious that the above mentioned result holds, there are some technical difficulties. We first need to establish two simple lemmata.

Lemma 4.19. $\text{NP} \setminus \text{NPC}$ *is closed under* $\leq^p_m$*-reduction.*

Proof. Assume to the contrary that $\text{NP} \setminus \text{NPC}$ in not closed under $\leq^p_m$-reduction. Consequently, there exist languages $A, B \subseteq \Sigma^*$ such that

$$\underbrace{A \leq^p_m B}_{(i)} \wedge \underbrace{B \in \text{NP} \setminus \text{NPC}}_{(ii)} \wedge \underbrace{A \notin \text{NP} \setminus \text{NPC}}_{(iii)}.$$

By (iii) it suffices to regard the following two cases:

Case 1: $A \in \text{NPC}$ **:** From (i) we conclude that B is NP-hard. From (ii) we conclude that $B \in \text{NP}$. Thus $B \in \text{NPC}$, which contradicts (ii).

Case 2: $A \notin \mathrm{NP}$**:** From (ii) it follows that $B \in \mathrm{NP}$. Since NP is closed under $\leq_m^p$-reduction we have together with (i) that $A \in \mathrm{NP}$. This contradicts the assumption of this case.

□

Lemma 4.20. *Let A be a nontrivial language from* NP*, i.e., $A \in \mathrm{NP} \setminus \{\emptyset, \Sigma^*\}$. If it holds for all classes $\mathcal{C}$ with $\mathcal{C} \subseteq \mathrm{NP}$ and $\mathcal{C}$ closed under $\leq_m^p$-reduction that*

$$A \in \mathcal{C} \Leftrightarrow \mathcal{C} = \mathrm{NP}$$

then $A \in \mathrm{NPC}$.

Proof. Let $A \in \mathrm{NP} \setminus \{\emptyset, \Sigma^*\}$.

Case 1: $\mathrm{P} = \mathrm{NP}$**:** All nontrivial problems are $\leq_m^p$-complete for P. Thus, A is complete for $\mathrm{P} = \mathrm{NP}$. Consequently, $A \in \mathrm{NPC}$, independent of the other preconditions.

Case 2: $\mathrm{P} \neq \mathrm{NP}$**:** We prove the assertion by contraposition. Assume that $A \notin \mathrm{NPC}$. Thus $A \in \mathrm{NP} \setminus \mathrm{NPC}$. We choose $\mathcal{C} = \mathrm{NP} \setminus \mathrm{NPC}$. Now, the left hand side of the equation

$$A \in \mathcal{C} \Leftrightarrow \mathcal{C} = \mathrm{NP}$$

is valid; the right hand side fails to hold, since $\mathrm{NPC} \neq \emptyset$. The assertion follows from the fact that $\mathrm{NP} \setminus \mathrm{NPC}$ is closed under $\leq_m^p$-reduction (see Lemma 4.19).

□

Now, we are prepared to formally prove that results for uselessness of selected hints are also results for uselessness of arbitrary hints.

Theorem 4.21. *Let (c, V_A) be a modification problem, $A = L(V_A) \in \mathrm{NP}$, and B be an* NP*-complete language. If $B \leq_{hi}^p (c, V_A)$ then $\mathrm{MOD}_c V_A$ is* NP*-complete.*

Proof. Let $B \in \mathrm{NPC}$ and let $B \leq_{hi}^p (c, V_A)$. By Lemma 4.20 it suffices to show that for all classes $\mathcal{C}$ with $\mathcal{C} \subseteq \mathrm{NP}$ and $\mathcal{C}$ closed under $\leq_m^p$-reduction it holds that

$$\mathrm{MOD}_c V_A \in \mathcal{C} \Leftrightarrow \mathcal{C} = \mathrm{NP}.$$

The direction form right to left is trivial (see Observation 3.2). For the other direction assume that $\mathrm{MOD}_c V_A \in \mathcal{C}$ for some class $\mathcal{C} \subseteq \mathrm{NP}$ that is closed under $\leq_m^p$-reduction. By definition of $\mathrm{MOD}_c V_A$ it holds for all $x, \pi, m \in \Sigma^*$ that

$$(x, \pi) \in V_A \wedge c(x, m) \in A \Leftrightarrow (x, \pi, m) \in \mathrm{MOD}_c V_A.$$

If $x \in A$ then for any certificate function h it holds that $(x, h(x)) \in V_A$. Therefore, the statement

$$x \in A \Rightarrow \left[c(x, m) \in A \Leftrightarrow (x, h(x), m) \in \mathrm{MOD}_c V_A\right]$$

is valid for any certificate function h. Since $\mathrm{MOD}_c V_A \in \mathcal{C}$ we get $(c, V_A) \in \mathcal{C}_{\mathrm{MOD}}^{\in} / cert(V_A)$. The assertion $\mathcal{C} = \mathrm{NP}$ follows from Theorem 4.10. □

4.5 Beyond the $\leq_{hi}^{p}$-reducibility Method

In the preceding sections we introduced the notions of hint-independent reducibility and hint-independent interreducibility. We applied these techniques to several modification problems. Unfortunately, in some cases the $\leq_{hi}^{p}$-reducibility method does not apply, e.g., it does not apply to the modification problem (ad, V_{SAT}) (see remark to Theorem 4.17). We do not settle the question whether (ad, V_{SAT}) has useful hints in this section. But, we prove that when the modification is the simultaneous addition of several unit clauses then selected hints are not likely to make the modification problem very easy.

To show this result we need ideas and notations from the proof of Cook's Theorem [Coo71], which states that the problem SAT is NP-complete w.r.t. $\leq_{m}^{p}$-reduction. We recall the necessary details from Cook's proof in the following subsection.

4.5.1 Cook's Reduction

Let $A \in \mathrm{NP}$ and M be an NPTM such that $L(M) = A$. In the course of this subsection, we demonstrate how to construct from M and an input x a CNF-formula $f_{cook}(M, x)$ such that $M(x)$ accepts if and only if $f_{cook}(M, x)$ is satisfiable.

W.l.o.g. we consider an NPTM M for A that is normalized, that is, there exists a polynomial p such that for each input x of M every computational path of M has exactly length $p(|x|)$. We may assume that M uses a single work tape that is infinite to the right but has a leftmost cell. This yields a numbering $0, 1, 2, \ldots$ of the cells from left to right. Suppose that $Q = \{q_0, \ldots, q_k\}$ are the states of M, q_0 being the starting state of M. Suppose that $\Sigma = \{\sigma_0, \ldots, \sigma_l\}$ is the alphabet of M, σ_0 being the blank symbol. We may assume that the head of the machine is in the leftmost cell at the beginning of the computation.

Since M is a nondeterministic machine, there exist multiple rules that may be applied at a time. We suppose that there always exists an applicable rule ,i.e., the transition function is total. Also, we assume that at any time the machine M may choose among at most two such rules. If there are two applicable rules ,i.e., two rules with the same left hand side, we fix a left and a right rule. Otherwise we refer to the single applicable rule as left rule. Finally, we may assume that the input of M is written in the leftmost cells and that M accepts the input x if and only if M is in state q_k after exactly $p(|x|)$ steps of computation.

The formula $f_{cook}(M, x)$ consists of several CNF-subformulas, namely

- the formula $S(x)$ that represents the initial configuration of M on input x,
- the formulas $F_t(M, |x|)$, where $1 \leq t \leq p(|x|)$, that characterize the possible transitions between the configurations of $M(x)$, and
- the formula $E(|x|)$ that characterizes the accepting condition of M.

In defining these formulas we use the following variables:

- $c_{t,i,\sigma}$, $0 \leq t, i \leq p(|x|)$, $\sigma \in \Sigma$. The variable $c_{t,i,\sigma}$ shall be true if and only if after M's tth computational step the cell with number i is occupied by the symbol σ,
- $h_{t,i}$, $0 \leq t, i \leq p(|x|)$. The variable $h_{t,i}$ shall be true if and only if after M's tth computational step M's head is on the cell with number i,
- $s_{t,q}$, $0 \leq t \leq p(|x|)$, $q \in Q$. The variable $s_{t,q}$ shall be true if and only if after M's tth computational step M is in state q, and
- v_t, $1 \leq t \leq p(|x|)$. The variable v_t shall be true if and only if M has applied the left rule in its tth computational step.

The set Y_t is defined to contain all variables that characterize the configuration after the tth computational step, that is,

$$Y_t := \{\, c_{t,i,\sigma},\ h_{t,i},\ s_{t,q}\ :\ 0 \leq i \leq p(n),\ \sigma \in \Sigma,\ q \in Q \,\}.$$

Now, we give the formula $S(x)$, which states that the machine M starts correctly. Let $x = x_0 ... x_{n-1}$. Then

$$S(x) := \bigwedge_{0 \leq c < n} (c_{0,c,x_c}) \wedge \bigwedge_{n \leq c \leq p(|x|)} (c_{0,c,\sigma_0}) \wedge h_{0,0} \wedge s_{0,q_0} \wedge \bigwedge_{y \in Y_0'} (\neg y),$$

where Y_0' contains the remaining variables from Y_0. The formula $F_t(M, |x|)$ consists of the conjunction of the following expressions:

- $\bigwedge_{0 \leq c \leq p(|x|)} \left[\left((c_{t-1,c,\sigma} \wedge \neg h_{t-1,c}) \Rightarrow c_{t,c,\sigma} \right) \wedge \left((\neg c_{t-1,c,\sigma} \wedge \neg h_{t-1,c}) \Rightarrow \neg c_{t,c,\sigma} \right) \right]$
- for each left rule $\sigma q \longrightarrow \sigma' q' r$ the two expressions:

$$\bigwedge_{0 \leq c < p(|x|)} \left((c_{t-1,c,\sigma} \wedge h_{t-1,c} \wedge s_{t-1,q}) \Rightarrow v_t \right) \qquad \text{and}$$

$$\bigwedge_{0 \leq c < p(|x|)} \left[(c_{t-1,c,\sigma} \wedge h_{t-1,c} \wedge s_{t-1,q} \wedge v_t) \Rightarrow \left(s_{t,q'} \wedge \bigwedge_{q'' \neq q'} \neg s_{t,q''} \right.\right.$$
$$h_{t,c+1} \wedge \bigwedge_{c' \neq c+1} \neg h_{t,c'}$$
$$\left.\left. c_{t,c,\sigma'} \wedge \bigwedge_{\sigma'' \neq \sigma'} \neg c_{t,c,\sigma''} \right) \right]$$

- a similar expression for left rules of the form $\sigma q \longrightarrow \sigma' q' l$ or $\sigma q \longrightarrow \sigma' q' 0$
- the above expression with v_t substituted by $\neg v_t$ for every right rule.

The formula $E(|x|)$, which shall represent the accepting condition of M, is given by $E(|x|) := z_{p(|x|),q_k}$.

The formula f_{cook} is given by $f_{cook}(M, x) := S(x) \wedge \bigwedge_{1 \leq t \leq p(|x|)} F_t(M, |x|) \wedge E(|x|)$. For a proof that f_{cook} yields a reduction from A to SAT we refer to the standard literature [DK00, BDG95, WW86, Coo71, Wec00].

We just mention the following property. Each two different paths of M's computation lead to assignments that differ when restricted to the variables v_t, $1 \leq t \leq p(|x|)$. Therefore each accepting path of M yields a different satisfying assignment for $f_{cook}(M, x)$. The converse, namely that two different satisfying assignment of $f_{cook}(M, x)$ lead to two different accepting paths of $M(x)$, also holds. Consequently, the function f_{cook} yields a parsimonious reduction from A to SAT.

It is plausible that Cook's reduction is also structure preserving. In more detail, we choose M as a machine that nondeterministically guesses a proof in a first phase and verifies it afterwards. Such a machine M always exists (see Theorem 2.3). For this choice of machine M a satisfying assignment for $f_{cook}(M, x)$ can be computed easily from the nondeterministically guessed bits, i.e., the certificate, that is guessed in the first phase of M's computation.

Note in the passing, that we can use Cook's reduction to prove that the class $\mathrm{NP}^{\in}_{\mathrm{MOD}}/cert$ has $\leq^p_{hi}$-complete problems.

Theorem 4.22. *(neg, V_{SAT}) is $\leq^p_{hi}$-complete for $\mathrm{NP}^{\in}_{\mathrm{MOD}}/cert$.*

Proof. It is obvious that $(neg, V_{\mathrm{SAT}}) \in \mathrm{NP}^{\in}_{\mathrm{MOD}}/cert$ (Observation 4.5). To show hardness of (neg, V_{SAT}) let (c, V_A) be a modification problem from $\mathrm{NP}^{\in}_{\mathrm{MOD}}/cert(V_A)$. It suffices to show that $(c, V_A) \leq^p_{hi} (neg, V_{\mathrm{SAT}})$. We give a structure preserving reduction g_1 (with respect to V_A, V_{SAT}, and a function $g_2 \in \mathrm{FP}$) and a function $g_3 \in \mathrm{FP}$ such that for all $x, \pi, m \in \Sigma^*$ it holds that

$$c(x, m) \in A \Leftrightarrow neg\big(g_1(x, \pi, m), g_3(x, \pi, m)\big) \in \mathrm{SAT}.$$

Let M be an NPTM for $L(V_A)$ that works in two phases. First, M nondeterministically guesses a certificate for the input x and afterwards this certificate is verified deterministically by V_A. Without loss of generality we may assume that the machine M obeys the restrictions of Cook's Theorem.

The formula $g_1(x, \pi, m)$ contains the two subformulas F_x and $F_{c(x,m)}$ defined by $F_x := f_{cook}(M, x)$ and $F_{c(x,m)} := f_{cook}(M, c(x, m))$. Let Y_x and $Y_{c(x,m)}$ be the set of variables of F_x and $F_{c(x,m)}$, respectively. Let $Y_x \cap Y_{c(x,m)} = \emptyset$ and $z \notin Y_x \cup Y_{c(x,m)}$. We define

$$g_1(x, \pi, m) := \Big[\big(z \wedge F_x \wedge \bigwedge_{y \in Y_{c(x,m)}} y\big) \vee \big(\neg z \wedge F_{c(x,m)} \wedge \bigwedge_{y \in Y_x} y\big)\Big] \wedge z,$$

or more exactly, the expanded CNF-form of this formula, which has at most quadratic size and still contains the unit clause z. Since f_{cook} yields a structure preserving reduction from A to SAT w.r.t. the verifiers V_A and V_{SAT}, the function g_1 also induces a structure preserving reduction. Furthermore, we set $g_3(x, \pi, m) := z$, that is, we negate the unit clause z. Then

$$\begin{aligned} c(x, m) \in L(V_A) \quad &\Leftrightarrow \quad f_{cook}(M, c(x, m)) \in \mathrm{SAT} \\ &\Leftrightarrow \quad \Big[\big(z \wedge F_x \wedge \bigwedge_{y \in Y_{c(x,m)}} y\big) \vee \big(\neg z \wedge F_{c(x,m)} \wedge \bigwedge_{y \in Y_x} y\Big] \wedge \neg z \in \mathrm{SAT} \\ &\Leftrightarrow \quad neg(g_1(x, \pi, m), g_3(x, \pi, m)) \in \mathrm{SAT}. \end{aligned}$$

□

We show in Appendix A that $(neg, V_{\text{SAT}}) \leq_{hi}^{p} (c, V_A)$ for many other modification problems (c, V_A). Since $\leq_{hi}^{p}$ is transitive (a fact for which we omit the proof) all the following modification problems are $\leq_{hi}^{p}$-complete for $\text{NP}_{\text{MOD}}^{\in}/cert$ (also see Figure 9.1): $(negl, V_{\text{SAT}})$, (neg, V_{3SAT}), $(negl, V_{\text{3SAT}})$, $(negl, V_{\text{Ex3SAT}})$, $(adlc, V_{\text{1-3SAT}})$, $(rmlc, V_{\text{1-3SAT}})$, $(neg, V_{\text{1-3SAT}})$, $(negl, V_{\text{1-3SAT}})$, $(ad\&rm, V_{\text{CLIQUE}})$, and $(ad\&rm, V_{\text{VC}})$.

4.5.2 Results

We now return to our proof that the addition of several unit clauses is not likely to be a modification that allows for useful hints. More precisely, we will show that if the modification problem $(ad^{id}, V_{\text{SAT}})$ was easy then some unlikely collapses would occur. Here, id is the identity function $id(n) := n$, for all $n \in \mathbb{N}$, whereas the notion of the modification function ad^{id} is fixed by the following definition.

Definition 4.23. *Let c be a modification function. The modification function c^k, which performs k modifications of the form c at the same time, is inductively defined by*

$$\begin{aligned} c^1(x,(m_1)) &:= c(x,m_1),\\ c^k(x,(m_1,\dots,m_k)) &:= c\big(\,c^{k-1}(x,(m_1,\dots,m_{k-1})),m_k\big). \end{aligned}$$

For a function $f : \mathbb{N} \to \mathbb{N}$ let

$$c^f(x,(m_1,\dots,m_k)) := \begin{cases} c^k(x,(m_1,\dots,m_k)), & \textit{if } k \leq f(|x|),\\ x, & \textit{otherwise.}\end{cases}$$

Note that the $\leq_{hi}^{p}$-reducibility method cannot be applied to the modification problem $(ad^{id}, V_{\text{SAT}})$ (see Corollary 4.18). But, we are able to show that the modification problem $(ad^{id}, V_{\text{SAT}})$ is not likely to be very easy via the following Theorem 4.24. Note that Theorem 4.24 will be strengthened afterwards with respect to the amount of modification. Nevertheless we prove this weaker theorem first, as the proof would become slightly intricate otherwise.

Theorem 4.24. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction. Then*

$$(ad^{id}, V_{\text{SAT}}) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_{\text{SAT}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly.$$

Proof. Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction. Let $(ad^{id}, V_{\text{SAT}}) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_{\text{SAT}})$ via the certificate function h for V_{SAT} and the predicate $C \in \mathcal{C}$, that is, for all $k \in \mathbb{N}$ and all $F, L_1, \dots, L_k \in \Sigma^*$ it holds that

$$F \in \text{SAT} \Rightarrow \big[ad^{id}\big(F,(L_1,\dots,L_k)\big) \in \text{SAT} \Leftrightarrow \big(F, h(F), (L_1,\dots,L_k)\big) \in C\big]. \quad (4)$$

We show that under this assumptions HC $\in \mathcal{C}/poly$. [2]

[2]The problem SAT could have also been chosen but we consider the problem HC in order to make the proof clearer.

Let G be a graph with $n \geq 3$ vertices; graphs with less than three vertices cannot contain a Hamiltonian cycle. Consider a nondeterministic polynomial-time Turing machine M for HC that obeys the restrictions of Cook's theorem. Additionally, we assume that $|G|$ (the size of the coding of G as input for M) only depends on n, the number of vertices of G, and is strictly increasing with n. This is achieved by coding G via an adjacency matrix.

Referring to the notations introduced in Section 4.5.1, we define $F_{M,|G|} := \bigwedge_{0 \leq t \leq p(|G|)} F_t(M,|G|) \wedge E(|G|)$. Intuitively, $F_{M,|G|}$ contains all those parts of $f_{cook}(M,G)$ that are only dependent on the size of the input but not the precise form of the input. The formula $F_{M,|G|}$ resembles the work of the machine M on an unknown input of size $|G|$. The key idea of the proof is that the input G can be coded into $F_{M,|G|}$ by addition of $S(G)$, which solely consists of unit clauses. Thus, the number of unit clauses that need to be added is bounded by id. In the remainder of this proof we elaborate this idea more formally.

Note that any complete graph K_n with $n \geq 3$ vertices contains a Hamiltonian cycle. Thus the formula $f_{cook}(M,K_n) = F_{M,|K_n|} \wedge S(|K_n|)$ is satisfiable. Since $|G|$ only depends on the number of vertices of G, the formula $F_{M,|G|}$ is satisfiable for any graph G with more than two vertices. By (4) it holds for all $k \in \mathbb{N}$, all $L_1, ..., L_k \in \Sigma^*$, and all G with more than two vertices that

$$\left[ad^{id}\left(F_{M,|G|},(L_1,...,L_k)\right) \in \text{SAT} \Leftrightarrow \left(F_{M,|G|}, h(F_{M,|G|}), (L_1,...,L_k)\right) \in C\right].$$

Since $F_{M,|G_1|} = F_{M,|G_2|}$ if and only if $|G_1| = |G_2|$ we can unambiguously define

$$h'(i) := \begin{cases} h(F_{M,|G|}), & \text{if } i \text{ is the size of some graph } G, \\ \epsilon, & \text{otherwise.} \end{cases}$$

Let $S'(G)$ denote the tuple of unit clauses of $S(G)$. Formally, if $S(G) = L_1 \wedge ... \wedge L_k$ then $S'(G) = (L_1, ..., L_k)$. The predicate C' defined by

$$(G,\omega) \in C' \Leftrightarrow (F_{M,|G|}, \omega, S'(G)) \in C$$

belongs to $\mathcal{C}$ since $\mathcal{C}$ is closed under $\leq_m^p$-reduction. Note that the number of elements in $S'(G)$ is smaller than $id(\left|F_{M,|G|}\right|)$. Now

$$\begin{aligned} G \in \text{HC} &\Leftrightarrow f_{cook}(M,G) \in \text{SAT} \\ &\Leftrightarrow F_{M,|G|} \wedge S(G) \in \text{SAT} \\ &\Leftrightarrow ad^{id}(F_{M,|G|}, S'(G)) \in \text{SAT} \\ &\Leftrightarrow (F_{M,|G|}, h(F_{M,|G|}), S'(G)) \in C \\ &\Leftrightarrow (G, h(F_{M,|G|})) \in C' \\ &\Leftrightarrow (G, h'(|G|)) \in C'. \end{aligned}$$

Since $h' \in poly$ we get HC $\in \mathcal{C}/poly$. □

As we already mentioned, Theorem 4.24 can be strengthened to also hold for a smaller amount of modification. In detail, we use a padding idea, that is, an instance is padded in a way that the amount of modification becomes very small in comparison to the length of the padded instance.

Theorem 4.25. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

$$(ad^q, V_{\text{SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly.$$

Proof. We refer to the notions from the proof of Theorem 4.24. Let $q(n) = \sqrt[K]{n}$ for some $K \geq 1$ and let $r(n) := n^K$. Let pad_r be the function that pads a formula F by adding unit clauses over new distinct variables such that the padded formula has size $r(|F|)$. Formally, for a formula F over the variables $x_1, ..., x_n$ we define

$$pad_r(F) := F \cup \bigcup_{i=n+1}^{N} \{\{x_i\}\},$$

where N is chosen to be the smallest number such that $|pad_r(F)| \geq r(|F|)$. For a reasonable coding of formulas the padding function pad_r has some useful properties:

- $pad_r \in \text{FP}$,
- $F \in \text{SAT} \Leftrightarrow pad_r(F) \in \text{SAT}$, for all $F \in \Sigma^*$, and
- $F \wedge S \in \text{SAT} \Leftrightarrow pad_r(F) \wedge S \in \text{SAT}$, for all formulas S that do not contain any of the padding variables $x_{n+1}, ..., x_N$.

In the proof of Theorem 4.24, the number of unit clauses that need to be added is bounded by $|S'(G)|$ and since

$$|S'(G)| \;\leq\; \left|F_{M,|G|}\right| \;=\; q(r(\left|F_{M,|G|}\right|)) \;\leq\; q\Big(\left|pad_r(F_{M,|G|})\right|\Big)$$

the number of unit clauses to be added is bounded by the function q in the size of $pad_r(F_{M,|G|})$. When substituting $F_{M,|G|}$ by $pad_r(F_{M,|G|})$ in the proof of Theorem 4.24 we obtain the desired result. For a more detailed elaboration of this idea also see the coming Theorem 4.27. □

The following corollaries show that if the modification problem (ad^q, V_{SAT}), where $q(n) = \sqrt[K]{n}$ and $K \in \mathbb{N} \setminus \{0\}$, would be easy, e.g., an element of $\text{P}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}})$ or $(\text{NP} \cap \text{coNP})^{\in}_{\text{MOD}}/cert(V_{\text{SAT}})$, then some collapses would occur that most computer scientist do not believe to happen. The corollaries use the best known collapse results when $\text{NP} \subseteq \text{P}/poly$ or $\text{NP} \subseteq (\text{NP} \cap \text{coNP})/poly$, respectively (also see Theorem 2.11 in the Preliminaries).

Corollary 4.26. *Let $K \in \mathbb{N} \setminus \{0\}$ and $q(n) = \sqrt[K]{n}$. Then*

- $(ad^q, V_{\text{SAT}}) \in \text{P}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}}) \;\Rightarrow\; S_2 = \text{PH}$,

- $(ad^q, V_{\text{SAT}}) \in \text{coNP}^{\in}_{\text{MOD}}/cert(V_{\text{SAT}}) \;\Rightarrow\; S_2^{\text{NP}} = \text{PH}$, *and*
- $(ad^q, V_{\text{SAT}}) \in (\text{NP} \cap \text{coNP})^{\in}_{\text{MOD}}/cert(V_{\text{SAT}}) \;\Rightarrow\; S_2^{\text{NP}\cap\text{coNP}} = \text{PH}$.

Up to now, we only considered the modification problem (ad^q, V_{SAT}). Now, we want to transfer our result to other modification problems (c, V_A). The following theorem is helpful for this task.

Theorem 4.27. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, (c, V_A) be a modification problem, $A = L(V_A)$, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. If there exist functions $f_1, f_3 \in \text{FP}$ such that for all $k > 1$ and all $F, L_1, ..., L_k \in \Sigma^*$ it holds that*

- $F \in \text{SAT} \Leftrightarrow f_1(F) \in A$ *and*
- $ad^q(F, (L_1, ..., L_k)) \in \text{SAT} \Leftrightarrow c(f_1(F), f_3((L_1, ..., L_k))) \in A$

then the following statement holds:

$$(c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly.$$

Proof. The proof is similar to the proof of Theorem 4.24. Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction. Let $(c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)$ via the certificate function h for V_A and the predicate $C \in \mathcal{C}$, that is, for all $x, m \in \Sigma^*$ it holds that

$$x \in A \Rightarrow \big[c\big(x, m\big) \in A \Leftrightarrow \big(x, h(x), m\big) \in C\big]. \tag{5}$$

Let $f_1, f_3 \in \text{FP}$ such that for all $k > 1$ and all $F, L_1, ..., L_k \in \Sigma^*$ it holds that

$$F \in \text{SAT} \Leftrightarrow f_1(F) \in A, \tag{6}$$

$$ad^q(F, (L_1, ..., L_k)) \in \text{SAT} \Leftrightarrow c(f_1(F), f_3((L_1, ..., L_k))) \in A. \tag{7}$$

We show that under this assumptions $\text{HC} \in \mathcal{C}/poly$.

W.l.o.g. we may assume that G is a graph with $n \geq 3$ vertices. Consider a nondeterministic polynomial-time Turing machine M for HC that obeys the restrictions of Cooks Theorem. We assume that G is coded via an adjacency matrix. The formulas $F_{M,|G|}$ and $S(G)$ are defined as in the proof of Theorem 4.24 . Let $S'(G)$ denote the tuple of unit clauses of $S(G)$. Let $r(n) := n^K$ and pad_r be the padding function from the proof of Theorem 4.25. For a formula F we abbreviate $pad_r(F)$ by F^{pad}. Following the arguments from the proof of Theorem 4.25 we get

$$|S'(G)| \leq q\left(\left|F^{pad}_{M,|G|}\right|\right) \tag{8}$$

for all graphs G. Note that for each graph G the formula $F_{M,|G|}$ is satisfiable. Consequently, $F^{pad}_{M,|G|}$ is satisfiable and therefore, by (6), $f_1(F^{pad}_{M,|G|}) \in A$. It follows from (5) that for all graphs G and all $m \in \Sigma^*$ it holds that

$$c\big(f_1(F^{pad}_{M,|G|}), m\big) \in A \Leftrightarrow \big(f_1(F^{pad}_{M,|G|}), h(f_1(F^{pad}_{M,|G|})), m\big) \in C. \tag{9}$$

We define a function $h' \in poly$ and a predicate C by

$$h'(i) := h(f_1(F^{pad}_{M,i})),$$

$$(G, \omega) \in C' \Leftrightarrow (f_1(F^{pad}_{M,|G|}), \omega, f_3(S'(G))),$$

for all $i \in \mathbb{N}$ and all $G, \omega \in \Sigma^*$. Obviously, $C' \leq^p_m C$ and since $C \in \mathcal{C}$ and $\mathcal{C}$ is closed under $\leq^p_m$-reduction we get $C' \in \mathcal{C}$. Now

$$\begin{aligned}
G \in \mathrm{HC} &\Leftrightarrow f_{cook}(M, G) \in \mathrm{SAT} \\
&\Leftrightarrow F_{M,|G|} \wedge S(G) \in \mathrm{SAT} \\
&\Leftrightarrow F^{pad}_{M,|G|} \wedge S(G) \in \mathrm{SAT} \\
&\overset{(8)}{\Leftrightarrow} ad^q(F^{pad}_{M,|G|}, S'(G)) \in \mathrm{SAT} \\
&\overset{(7)}{\Leftrightarrow} c(f_1(F^{pad}_{M,|G|}), f_3(S'(G))) \in A \\
&\overset{(9)}{\Leftrightarrow} (f_1(F^{pad}_{M,|G|}), h(f_1(F^{pad}_{M,|G|})), f_3(S'(G))) \in C \\
&\Leftrightarrow (f_1(F^{pad}_{M,|G|}), h'(|G|), f_3(S'(G))) \in C \\
&\Leftrightarrow (G, h'(|G|)) \in C'.
\end{aligned}$$

This implies $\mathrm{HC} \in \mathcal{C}/poly$. □

Theorem 4.27 implicates an interesting corollary, namely that the choice of the verifier V_{SAT} is not crucial in Theorem 4.25.

Corollary 4.28. *Let $\mathcal{C}$ be closed under $\leq^p_m$-reduction, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Let V'_{SAT} be any verifier for* SAT. *Then*

$$(ad^q, V'_{\mathrm{SAT}}) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V'_{\mathrm{SAT}}) \Rightarrow \mathrm{NP} \subseteq \mathcal{C}/poly.$$

Proof. We choose $(c, V_A) = (ad^q, V'_{\mathrm{SAT}})$ in Theorem 4.27. The functions $f_1(F) := F$ and $f_3(m) := m$ satisfy the conditions posed on f_1 and f_3. □

We use Theorem 4.27 to show that selected solutions are also not likely to be a good hint for some other modification problems, such as $(rmlc^q, V_{\mathrm{SAT}})$, $(ad^q, V_{3\mathrm{SAT}})$, $(adc^q, V_{\mathrm{Ex3SAT}})$, $(ad^q, V_{1\text{-}3\mathrm{SAT}})$, (ad^q, V_{VC}), $(rm^q, V_{\mathrm{CLIQUE}})$, (rm^q, V_{HC}), and the modification problem $(rm^q, V_{3\mathrm{DM}})$. The respective proofs are given in Appendix A.

4.6 Computationally Restricted Certificate Functions

Certificate function may be nonrecursive, in general. We already mentioned that this approach leads to strong results of uselessness of selected hints. But, for some modification problems, like (ad, V_{SAT}), we were unable to prove uselessness of selected hints w.r.t. such a powerful certificate function. In this section we show that if the certificate function is appropriately restricted then (ad, V_{SAT}) is not likely to have helpful selected certificates.

Only some special cases of certificate function are computable by computationally restricted machines, e.g., the functions that compute the maximum or the minimum solution with respect to some ordering are computable in $\mathrm{F}\Delta_2^p$. Even worse, in [HNOS96] the authors show that no certificate functions for V_{SAT} can exist in the class $fun \cdot \mathrm{NP}$, unless $\mathrm{NP} \subseteq (\mathrm{NP} \cap \mathrm{coNP})/poly$, and consequently $S_2^{\mathrm{NP}\cap\mathrm{coNP}} = \mathrm{PH}$ (see Theorem 2.11). In the spirit of this result, we show that the existence of a certificate function $h \in fun \cdot \mathrm{NP}$ that renders the modification problem (ad, V_{SAT}) a member of $\mathrm{coNP}^{\in}_{\text{MOD}}/(cert(V_{\text{SAT}}) \cap fun \cdot \mathrm{NP})$ is even more unlikely.

Theorem 4.29. $(ad, V_{\text{SAT}}) \in \mathrm{coNP}^{\in}_{\text{MOD}}/(cert(V_{\text{SAT}}) \cap fun \cdot \mathrm{NP}) \Rightarrow \{1\}\mathrm{P} \subseteq \mathrm{NP}$.

Proof. Let $(ad, V_{\text{SAT}}) \in \mathrm{coNP}^{\in}_{\text{MOD}}/(cert(V_{\text{SAT}}) \cap fun \cdot \mathrm{NP})$ via the certificate function $h \in fun \cdot \mathrm{NP}$ for V_{SAT} and the predicate $B \in \mathrm{coNP}$, that is, for all $F, L \in \Sigma^*$ it holds that

$$F \in \text{SAT} \Rightarrow \left[ad(F, L) \in \text{SAT} \Leftrightarrow (F, h(F), L) \in B\right]. \tag{10}$$

We show that $\{1\}\text{SAT} \in \mathrm{NP}$ under this assumption. Therefore, we give an NPTM M that decides $\{1\}\text{SAT}$.

Let F be a CNF-formula and let $Var(F) = \{x_1, ..., x_n\}$. We describe a machine M that decides $\{1\}\text{SAT}$ rather informally: In a first stage, the machine M nondeterministically guesses the hint $h(F)$. Afterwards, in a second nondeterministic phase, the machine M verifies that the guess is correct. This is possible since the certificate function h belongs to $fun \cdot \mathrm{NP}$, or equivalently $(\exists C \in \mathrm{NP})\,[h(x) = y \Leftrightarrow (x, y) \in C]$. For the correct hint $h(F)$ this might result in several accepting paths, whereas for all other guesses no path accepts. In a third phase, the machine M deterministically tests on all the second-phase-accepting-paths whether $(F, h(F)) \in V_{\text{SAT}}$. If $(F, h(F)) \notin V_{\text{SAT}}$ then $F \notin \text{SAT}$, since h is a certificate function. In this case, all the paths become rejecting paths[3]. If $(F, h(F)) \in V_{\text{SAT}}$ then a fourth phase is appended.

Before we describe the work of the machine M in this fourth phase we state a few facts. The fourth phase is only reached if $F \in \text{SAT}$. Furthermore, the certificate $h(F)$ for F is given by the guess in the first phase of M's computation. Consequently, we have a simplified form of (10), namely

$$ad(F, L) \in \text{SAT} \Leftrightarrow (F, h(F), L) \in B.$$

The key idea of the proof is to exploit the following characterization of {0,1}SAT:

$$F \in \{0,1\}\text{SAT} \quad \Leftrightarrow \quad (\forall 1 \le i \le n)\,[\neg\,((F \wedge x_i) \in \text{SAT} \wedge (F \wedge \neg x_i) \in \text{SAT})]\,.$$

Since we only deal with satisfiable formulas in the fourth phase of M we use this equivalence to accept only those formulas that belong to $\{1\}\text{SAT}$: For each ω the test

[3]Note that all second-phase-accepting-paths behave similar after the third phase since all these paths belong to the same guess of $h(F)$.

whether $(\exists 1 \le i \le n)\big[(F,\omega,x_i) \in B \wedge (F,\omega,\neg x_i) \in B\big]$ can be achieved within coNP, since coNP is closed under union and intersection. Consequently, the set

$$A := \{(F,\omega) : (\forall 1 \le i \le n)\,[\neg\,((F,\omega,x_i) \in B \wedge (F,\omega,\neg x_i) \in B)]\}$$

belongs to NP. Now, the fourth phase of M simply consist of checking whether $(F, h(F)) \in A$ using an NPTM for A. Correctness follows from the following equivalences

$$\begin{aligned} F \in \{0,1\}\mathrm{SAT} &\Leftrightarrow (\forall 1 \le i \le n)\,[\neg\,((ad(F,x_i) \in \mathrm{SAT} \wedge ad(F,\neg x_i) \in \mathrm{SAT}))] \\ &\Leftrightarrow (\forall 1 \le i \le n)\,[\neg\,((F,h(F),x_i) \in B \wedge (F,h(F),\neg x_i) \in B)] \\ &\Leftrightarrow (F,h(F)) \in A. \end{aligned}$$

□

Since $\{1\}\mathrm{P} \subseteq \mathrm{NP} \Rightarrow \mathrm{NP} = \mathrm{coNP}$ (Corollary 2.8) we get the following corollary.

Corollary 4.30. $(ad, V_{\mathrm{SAT}}) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V_{\mathrm{SAT}}) \cap fun \cdot \mathrm{NP}) \Rightarrow \mathrm{NP} = \mathrm{coNP}$.

In order to show a similar result for other modification problems (c, V_A) we establish the following theorem.

Theorem 4.31. *Let* (c, V_A) *be a modification problem,* $A = L(V_A)$, $f_1, f_3 \in \mathrm{FP}$, *and*

- $ad(F,L) \in \mathrm{SAT} \Leftrightarrow c(f_1(F), f_3(L)) \in A$.[4]

Then $(c, V_A) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V_A) \cap fun \cdot \mathrm{NP}) \Rightarrow \{1\}\mathrm{P} \subseteq \mathrm{NP}$.

Proof. The proof is similar to the proof of 4.29. We assume the reader to be familiar with this proof. Let $(c, V_A) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V_A) \cap fun \cdot \mathrm{NP})$ via the certificate function $h \in fun \cdot \mathrm{NP}$ for V_{SAT} and the predicate $B \in \mathrm{coNP}$, that is, for all $x, m \in \Sigma^*$ it holds that

$$x \in A \Rightarrow \big[c(x,m) \in A \Leftrightarrow (x, h(x), m) \in B\big].$$

We show that $\{1\}\mathrm{SAT} \in \mathrm{NP}$ under this assumption.

Let F be a CNF-formula and let $Var(F) = \{x_1, ..., x_n\}$. We give a five-phased NPTM M that decides if $F \in \{1\}\mathrm{SAT}$. In a first stage, the machine M deterministically computes $f_1(F)$. In a second stage, it nondeterministically guesses a hint $h(f_1(F))$. Afterwards, in a third nondeterministic phase, it verifies that the guess is correct. For the correct hint $h(f_1(F))$ this might result in several accepting paths; for all other guesses no path accepts. In a fourth phase, the machine M deterministically test on all the third-phase-accepting-paths whether $(f_1(F), h(f_1(F))) \in V_A$. If $(f_1(F), h(f_1(F))) \notin V_A$ then $f_1(F) \notin A$ and M rejects on these paths. Otherwise ,if $(f_1(F), h(f_1(F))) \in V_A$, then a fifth phase is appended.

The fifth phase is only reached if $f_1(F) \in A$. Consequently,

$$c(f_1(F), f_3(F,L)) \in A \Leftrightarrow (f_1(F), h(f_1(F)), f_3(L)) \in B$$

[4]Note the similarity of this condition to the second condition of Theorem 4.27.

for all $F, m \in \Sigma^*$. We define a predicate C by

$$(F, \omega) \in C \Leftrightarrow (\forall 1 \leq i \leq n) \left[(f_1(F), \omega, f_3(x_i)) \in \overline{B} \vee (f_1(F), \omega, f_3(\neg x_i)) \in \overline{B}\right].$$

Obviously, $C \in \mathrm{NP}$. Now, the fifth phase of M consist of checking if $(F, h(f_1(F))) \in C$, using an NPTM for C. Correctness follows from the following equivalences:

$$\begin{aligned}
F \in \{0,1\}\text{SAT} &\Leftrightarrow (\forall 1 \leq i \leq n)\,[\neg\,(\,ad(F, x_i) \in \text{SAT} \wedge ad(F, \neg x_i) \in \text{SAT})] \\
&\Leftrightarrow (\forall 1 \leq i \leq n)\Big[\neg\Big(\,c(f_1(F), f_3(x_i)) \in A \wedge \\
&\qquad\qquad\qquad c(f_1(F), f_3(\neg x_i)) \in A\Big)\Big] \\
&\Leftrightarrow (\forall 1 \leq i \leq n)\Big[\neg\,(\,(f_1(F), h(f_1(F)), f_3(x_i)) \in B \wedge \\
&\qquad\qquad\qquad (f_1(F), h(f_1(F)), f_3(\neg x_i)) \in B)\Big] \\
&\Leftrightarrow (F, h(f_1(F))) \in C.
\end{aligned}$$

Thus, after the fifth phase we have an accepting path if and only $F \in \{1\}\text{SAT}$. □

We just mention here, that Theorem 4.31 can be used to show that also the modification problems $(rmlc, V_{\text{SAT}})$, $(ad, V_{3\text{SAT}})$, (adc, V_{Ex3SAT}), $(ad, V_{1\text{-}3\text{SAT}})$, (ad, V_{VC}), (rm, V_{CLIQUE}), (rm, V_{HC}), and $(rm, V_{3\text{DM}})$ are probably not a member of the class $\text{coNP}^{\in}_{\text{MOD}}/(cert(V_A) \ \cap \ fun \cdot \text{NP})$. The respective proof are given in Appendix A. Similar to Corollary 4.28 we can also use Theorem 4.31 to show that (ad, V'_{SAT}) is probably no member of $\text{coNP}^{\in}_{\text{MOD}}/(cert(V_A) \ \cap \ fun \cdot \text{NP})$ for any choice of verifier V'_{SAT} for SAT.

Conclusions

In this chapter we examined the scenario in which the given certificate is not an arbitrary certificate, but may be chosen carefully among all certificates. Thereby, we assumed, that the original instance has at least one certificate. To formalize this problem it was necessary to introduce the notions of a certificate function and a modification problem. We also introduced new complexity classes $\mathcal{C}^{\in}_{\text{MOD}}/\mathcal{F}$ to categorize the complexity of modification problems. We showed how easiness results for arbitrary strings, e.g., $\text{MOD}_c V_A \in \text{P}$, can be translated to easiness results for selected hints, e.g., $(c, V_A) \in \text{P}^{\in}_{\text{MOD}}/certV_A$. Conversely, we also showed how to translate uselessness results for selected certificates to uselessness results for arbitrary certificates (Theorem 4.21).

We proved uselessness of selected hint in two ways: (i) by a hint-independent reduction from an NP-complete problem A to a modification problem (c, V_B) or (ii) by a hint-independent interreduction between two modification problems (c, V_A) and (c', V_B). We applied these two methods to the problems SAT and EX3SAT. These two techniques are also used in Appendix A to show that a selected certificate is a useless hint for instances of the six basic problems from [GJ79], when these instances are appropriately modified. We also indicated how structure preserving reductions might help to find hint-independent interreductions.

For some modification problems, e.g., (ad, V_{SAT}), we were unable to show easiness or hardness. New techniques are necessary to answer this question, since hint-independent (inter)reductions are insufficient to show uselessness of selected hints for (ad, V_{SAT}).

Open problem 1. *Are selected certificates of any use when deciding ad-modified* SAT-*instances?*

As a first step in answering this open question, we found the following two results:

1. Unless the polynomial hierarchy collapses, the problem (ad, V_{SAT}) has no useful selected certificates that are easily computable.

2. Unless the polynomial hierarchy collapses, the related problem $(ad^{id}, V_{\text{SAT}})$ has no useful selected certificates.

We also gave sufficient conditions such that the latter results can be translated to other modification problems.

Chapter 5

No Solution as a Promise

In this chapter we are concerned with original instances that have no certificate, and therefore no certificate can be given as a hint. This describes an scenario in which an earlier computation yielded that the original instance has no solutions, and now we want to decide if a modified instance has one.

5.1 Problem Formalization

Can the knowledge that the original instance has no certificates be any useful in deciding modified instances? It can, as the following trivial example illustrates: Consider the problem SAT and the modification function ad. Apparently, if the original instance F is not satisfiable then the modified instance $ad(F, L)$ also has no certificate. Thus $ad(F, L) \notin A$.

Since no certificates are involved in this decision problem the precise form of the certificates needs not to be specified via a verifier. All the results obtained in this section are independent of specific verifiers. Thus, we can simplify what we understand as modification problem within this chapter.

Definition 5.1. *Let c be a modification function and $A \in \mathrm{NP}$. Then the modification problem for A and c is the pair (c, A).*

Note that the notion of a modification problem is overloaded (Definition 4.2 and Definition 5.1) but it should be clear from notation, (c, V_A) vs. (c, A), which one is meant.

Given a modification problem (c, A), we now formalize the problem of deciding containment of $c(x, m)$ in A when the promise is given that $x \notin A$. In analogy to Chapter 4 we define new classes $\mathcal{C}_{\mathrm{MOD}}/\notin$ that characterize the complexity of the aforementioned modification problem (c, A).

Definition 5.2. *Let $\mathcal{C}$ be a complexity class. The modification problem (c, A) belongs to $\mathcal{C}_{\mathrm{MOD}}/\notin$ if and only if*

$$(\exists C \in \mathcal{C})(\forall x, m \in \Sigma^*)\left[\, x \notin A \Rightarrow \left[c(x, m) \in A \Leftrightarrow \big(x, m\big) \in C\right]\right].$$

In a nutshell, Definition 5.2 says that if $(c, A) \in \mathcal{C}_{\text{MOD}}/\notin$ then a modified instance $c(x, m)$ can also be decided with help of a $\mathcal{C}$-predicate C whenever the original instance has no solutions. As usual, the modification problem cannot be harder than the original problem, as the following observation shows.

Observation 5.3. *Let c be a modification function, $\mathcal{C}$ be a complexity class closed under $\leq_m^p$-reduction, and $A \in \mathcal{C}$. Then $(c, A) \in \mathcal{C}_{\text{MOD}}/\notin$.*

Proof. Choose $C := \{(x, m) : c(x, m) \in A\}$ in Definition 5.2. □

We use Definition 5.2 to rephrase our initial example in a more formal way.

Observation 5.4. $(ad, \text{SAT}) \in \mathrm{P}_{\text{MOD}}/\notin$.

Proof. Choose $C = \emptyset$ in Definition 5.2. The empty set belongs to P. □

Also, by Definition 5.2 the following monotonicity property is obvious.

Observation 5.5. *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes and $\mathcal{F}$ be a class of functions. If $\mathcal{C} \subseteq \mathcal{D}$ then $\mathcal{C}_{\text{MOD}}/\notin \subseteq \mathcal{D}_{\text{MOD}}/\notin$.*

5.2 Promise-independent Reducibility and SAT

Do there exist problems that do *not* benefit from the hint, that the original instance has no certificates? We show existence of such a problem via the following theorem.

Theorem 5.6. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction. Then*

$$(rm, \text{SAT}) \in \mathcal{C}_{\text{MOD}}/\notin \Rightarrow \text{SAT} \in \mathcal{C}.$$

Proof. Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction and let $(rm, \text{SAT}) \in \mathcal{C}_{\text{MOD}}/\notin$ via the predicate $C \in \mathcal{C}$, that is, for all $F, L \in \Sigma^*$ it holds that

$$F \notin \text{SAT} \Rightarrow \big[rm(F, L) \in \text{SAT} \Leftrightarrow (F, L) \in C\big].$$

Let F be a CNF-formula and $a \notin Var(F)$. Apparently, $F \wedge a \wedge \neg a \notin \text{SAT}$. Thus,

$$\begin{aligned} F \in \text{SAT} &\Leftrightarrow F \wedge a \in \text{SAT} \\ &\Leftrightarrow rm(F \wedge a \wedge \neg a, \neg a) \in \text{SAT} \\ &\Leftrightarrow (F \wedge a \wedge \neg a, \neg a) \in C. \end{aligned}$$

Since $f_1(F) := F \wedge a \wedge \neg a$ and $f_2(F) := \neg a$ are polynomial time computable from F and since rm belongs to FP we have shown that SAT is $\leq_m^p$-reducible to C. Since $\mathcal{C}$ is closed under $\leq_m^p$ we get $\text{SAT} \in \mathcal{C}$. □

We conclude from the last theorem that the knowledge that no solution for the original instance exists is generally not helpful for the problem SAT and the modification function rm. If the problem (rm, SAT) was in some class $\mathcal{C}_{\mathrm{MOD}}/\notin$ then also $\mathrm{SAT} \in \mathcal{C}$.

We want to condense the key idea of the above proof into an appropriate notion of a reduction. Basically, if we want to show that the knowledge '$x \notin A$' is useless for a problem A it is sufficient to give for all $x \in \Sigma^*$

- an instance $f_1(x) \notin A$
- for which the decision problem for the modified instances $c(f_1(x), f_2(x))$ resembles the complexity of some problem B that is as hard as A.

Definition 5.7. *Let $A \in \mathrm{NP}$ and c be a modification function. We say that a problem B is promise-independently polynomial-time reducible to (c, A), short $B \leq_{pi}^{p} (c, A)$, if and only if there exist two functions $f_1, f_2 \in \mathrm{FP}$ such that for all $x \in \Sigma^*$ the following two conditions hold:*

- $f_1(x) \notin A$,
- $x \in B \Leftrightarrow c\big(f_1(x), f_2(x)\big) \in A$.

It should be obvious how to translate the proof of Theorem 5.6 to the result that $\mathrm{SAT} \leq_{pi}^{p} (rm, \mathrm{SAT})$.

Corollary 5.8. $\mathrm{SAT} \leq_{pi}^{p} (rm, \mathrm{SAT})$.

We use the notion of hint-independent reducibility to generalize the assertion of Theorem 5.6 to other modification problems (c, A).

Theorem 5.9. *Let $A \in \mathrm{NP}$, c be a modification function, and $\mathcal{C}$ be a complexity class that is closed under $\leq_m^p$-reduction. For all languages B it holds that*

$$B \leq_{pi}^{p} (c, A) \wedge (c, A) \in \mathcal{C}_{\mathrm{MOD}}/\notin \Rightarrow B \in \mathcal{C}.$$

Proof. Let $(c, A) \in \mathcal{C}_{\mathrm{MOD}}/\notin$ via the predicate $C \in \mathcal{C}$, that is, for all $x, m \in \Sigma^*$ it holds that

$$x \notin A \Rightarrow \big(c(x, m) \in A \Leftrightarrow (x, m) \in C\big).$$

Since $B \leq_{pi}^{p} (c, A)$ we conclude that $f_1(x) \notin A$, for all $x \in \Sigma^*$. Therefore we get the following equivalences:

$$\begin{aligned} x \in B &\Leftrightarrow c(f_1(x), f_2(x)) \in A \\ &\Leftrightarrow (f_1(x), f_2(x)) \in C. \end{aligned}$$

Thus, we have shown that $B \leq_m^p C$, and since $\mathcal{C}$ is closed under $\leq_m^p$-reduction we get $B \in \mathcal{C}$. □

Consequently, if we are able to show for a problem A that $A \leq_{pi}^{p} (c, A)$ then we have that the no-solution promise is useless when modifying instances of A by c. Also, if we can show for two problems A and B with $A \equiv_m^p B$ (and in particular for NP-complete problems A and B) that $A \leq_{pi}^{p} (c, B)$ we can conclude likewise that the no-solution promise is useless.

5.3 Promise-independent Interreducibility and EX3SAT

In analogy to Section 4.3 we aim to show the uselessness of the no-solution promise in an alternative way, namely by reduction from other modification problems for which such uselessness results are already known. The following definition proposes an appropriate notion of interreduction for this purpose.

Definition 5.10. *Let (c, A) and (c', B) be two modification problems as defined in Definition 5.1. The problem (c, A) is promise-independently interreducible to (c', B), short $(c, A) \leq^p_{pi} (c', B)$, if and only if there exist two functions $g_1, g_2 \in \mathrm{FP}$ such that for all $x, m \in \Sigma^*$ it holds that*

- *$x \notin A \Rightarrow g_1(x, m) \notin B$ and*
- *$c(x, m) \in A \Leftrightarrow c'(g_1(x, m), g_2(x, m)) \in B$.*

In close analogy to Section 4.3, we show uselessness of the no-solution promise for a modification problem (c', C) by $\leq^p_{pi}$-reduction from some appropriate modification problem (c, B). The following quasi-transitivity result, that connects both kinds of promise-independent reductions, will be used.

Theorem 5.11. *Let $A \in \mathrm{NP}$ and let (c, B) and (c', C) be modification problems. Then*

$$\big(A \leq^p_{pi} (c, B) \ \wedge \ (c, B) \leq^p_{pi} (c', C)\,\big) \Rightarrow \ A \leq^p_{pi} (c', C).$$

Proof. Let $A \leq^p_{pi} (c, B)$ via reduction functions f_1 and f_2 and $(c, B) \leq^p_{pi} (c', C)$ via reduction functions g_1 and g_2. It is an easy task to show that $A \leq^p_{hi} (c', C)$ via the two reduction functions $h_1, h_2 \in \mathrm{FP}$ defined by

- $h_1(x) := g_1\big(f_1(x), f_2(x)\big)$ and
- $h_2(x) := g_2\big(f_1(x), f_2(x)\big)$. □

By the combination of Corollary 5.8, Theorem 5.9 and Theorem 5.11 it suffices to show that $(rm, \mathrm{SAT}) \leq^p_{pi} (c, A)$ to prove that the modification problem (c, A) cannot benefit from the hint that the original instance has no solution. Thereby, the problem (c, A) also becomes a problem that can be used as a starting point for a promise-independent interreduction.

Note that any reduction $A \leq^p_m B$ that is compatible with respect to the modifications c and c' yields a promise-independent interreduction $(c, A) \leq^p_{pi} (c', B)$. To show promise-independent interreducibility of two modification problems it often suffices to slightly alter previous hint-independent interreductions from Section 4.3 or Appendix A. We demonstrate this at the example of the modification problem $(negl, \mathrm{Ex3SAT})$. Note that is is shown in Appendix A, Observation 9.3, that $\mathrm{SAT} \leq^p_{pi} (neg, \mathrm{SAT})$. Thus, by application of Theorem 5.11, the following Theorem 5.12 yields that $\mathrm{SAT} \leq^p_{pi} (negl, \mathrm{Ex3SAT})$.

Theorem 5.12. $(neg, \text{SAT}) \leq^p_{pi} (negl, \text{Ex3SAT})$.

Proof. We use the $\leq^p_{hi}$-reduction given in the proof of Theorem 4.16, and in particular the function g_1 defined there as [1]

$$\begin{aligned} g_1(F, \beta, L) \quad := \quad & f(F \setminus \{\{L\}\}) \cup \\ & \{\{L, y_1, y_2\}, \{L, \neg y_1, y_2\}, \{L, y_1, \neg y_2\}, \{L, \neg y_1, \neg y_2\}\} \cup \\ & \{\{\neg L, y_1, y_2\}, \{\neg L, \neg y_1, y_2\}, \{\neg L, y_1, \neg y_2\}\}. \end{aligned}$$

We give two reduction functions g'_1 and g'_2 that realize the reduction $(neg, \text{SAT}) \leq^p_{pi} (negl, \text{Ex3SAT})$. We define g'_1 by

$$g'_1(F, L) := g_1(F, \epsilon, L),$$

where ϵ denotes the empty word[2]. The reader may verify that for any fixed literal L the function g'_1 is a reduction function for SAT $\leq^p_m$ Ex3SAT, thus satisfying the first property in Definition 5.10, namely

$$F \notin \text{SAT} \Rightarrow g'_1(F, L) \notin \text{Ex3SAT}.$$

Furthermore, the reduction function g'_1 yields a reduction that is compatible with respect to the modification functions neg and $negl$. We use the reduction function g_3 from the proof of Theorem 4.16 to define a function g'_2 that translates a negation of a unit clause $\{L\}$ in F to a negation of a literal in $g'_1(F, L)$:

$$g'_2(F, L) := g_3(F, \beta, L) = (\{L, \neg y_1, \neg y_2\}, L).$$

Now, it is not difficult to show that the property

$$neg(F, L) \in \text{SAT} \Leftrightarrow negl(g'_1(F, L), g'_2(F, L)) \in \text{Ex3SAT}$$

holds. Hence g'_1 and g'_2 yield the desired $\leq^p_{pi}$ reduction. □

Table 5.1 summarizes our results for modified satisfiability problems when the no-solution promise is given. The missing proofs for $\leq^p_{pi}$-reducibility of the respective modification problems are given in Appendix A.

Regarding the problems VC, Clique, HC ,3DM, and Partition our results are summarized in Table 5.2. For the proofs we refer to the respective sections in Appendix A.

[1]Note that for the sake of simplicity we only consider the case $L \in F$, just as we did in the proof of Theorem 4.16. For the definition of the reduction function f we refer to the proof of Theorem 3.6.

[2]Since g_1 does not depend on β, any other polynomial string besides ϵ could have been chosen as well.

modification function	SAT / 3SAT	Ex3SAT	1-3SAT
$\boldsymbol{ad}$ (addition of a unit cl.)	easy	-	easy
$\boldsymbol{adc}$ (addition of a clause)	easy	easy	easy
$\boldsymbol{adlc}$ (add. of a lit. to a cl.)	$\leq^p_{pi}$	-	$\leq^p_{pi}$
$\boldsymbol{rm}$ (removal of a unit cl.)	$\leq^p_{pi}$	-	$\leq^p_{pi}$
$\boldsymbol{rmc}$ (removal of a clause)	$\leq^p_{pi}$	$\leq^p_{pi}$	$\leq^p_{pi}$
$\boldsymbol{rmlc}$(rem. of a lit. from a cl.)	easy	-	$\leq^p_{pi}$
$\boldsymbol{neg}$ (negation of a unit cl.)	$\leq^p_{pi}$	-	$\leq^p_{pi}$
$\boldsymbol{negl}$ (neg. of a lit. of a cl.)	$\leq^p_{pi}$	$\leq^p_{pi}$	$\leq^p_{pi}$

Table 5.1: Hard and easy cases when deciding modified instances of various satisfiability problems with help of the no-solution promise.

	easy ($\in \mathrm{P}^{\notin}_{\mathrm{MOD}}$)	hard ($\leq^p_{hi}$-(inter)reducible)
VC	(ad, VC)	(rm, VC)
CLIQUE	(rm, CLIQUE)	(ad, CLIQUE)
HC	(rm, HC)	(ad, HC)
3DM		$(ad, \mathrm{3DM})$ $(rm, \mathrm{3DM})$
PARTITION		$(ad, \mathrm{PARTITION})$ $(rm, \mathrm{PARTITION})$

Table 5.2: Hard and easy cases when deciding modified instances of the problems VC, CLIQUE, HC, 3DM, and PARTITION with help of the no-solution promise.

5.4 Composing the Cases

In Chapter 4 we showed that many problems do not benefit from selected certificates. But, we assumed that the original instance has at least one certificate that can be given as a hint. In this chapter we examined the scenario in which no certificate for the original instance exists. Now, we combine the results for these both cases such that the benefit of a selected certificate can be judged independent of whether the original instance has a solution or not. Therefore we introduce the notion $\mathcal{C}_{\mathrm{MOD}}/\mathcal{F}$, which is similar to the definition of $\mathcal{C}^{\in}_{\mathrm{MOD}}/\mathcal{F}$, except that no demand on the membership of the original instance to the problem A is put.

Definition 5.13. *Let $\mathcal{C}$ be a complexity class and $\mathcal{F}$ be a class of functions. The modification problem (c, V_A) belongs to $\mathcal{C}_{\mathrm{MOD}}/\mathcal{F}$ if and only if*

$$(\exists h \in \mathcal{F})(\exists C \in \mathcal{C})(\forall x, m \in \Sigma^*) \left[c(x, m) \in A \Leftrightarrow \big(x, h(x), m\big) \in C\right] .$$

With regard to the function class $cert$, containment of a modification problem (c, V_A) in $\mathcal{C}_{\text{MOD}}/cert(V_A)$ expresses the fact that modified instances of the form $c(x, m)$ can be decided with help of a selected certificates of the original instance and a $\mathcal{C}$-predicate, independent of whether the original instance has a solution or not. It is intuitively clear that the complexity of a modification problem (c, V_A) w.r.t. $\mathcal{C}_{\text{MOD}}/cert(V_A)$ is dominated by the harder of the both aforementioned cases, namely the case where the original instance x has a solution and the case $x \notin A$. This fact is formally expressed by the following theorem.

Theorem 5.14. *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes that are closed under $\leq_m^p$-reduction, let (c, V_A) be a modification problem, and let $A = L(V_A)$. Then*

$$(c, V_A) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_A) \ \wedge \ (c, A) \in \mathcal{D}_{\text{MOD}}/\notin \ \Rightarrow \ (c, V_A) \in (\mathcal{C} \cup \mathcal{D})_{\text{MOD}}/cert(V_A).$$

Proof. Let $(c, V_A) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_A)$ via the certificate function h and the predicate $C \in \mathcal{C}$. Let $(c, V_A) \in \mathcal{D}_{\text{MOD}}/\notin$ via a predicate $D \in \mathcal{D}$. Consequently,

- $(\forall x, m \in \Sigma^*) \left[x \in A \Rightarrow \left[c(x, m) \in A \Leftrightarrow \big(x, h(x), m\big) \in C\right]\right]$ and
- $(\forall x, m \in \Sigma^*) \left[x \notin A \Rightarrow \left[c(x, m) \in A \Leftrightarrow \big(x, m\big) \in D\right]\right]$.

We define a predicate $E \in \mathcal{C} \cup \mathcal{D}$ by

$$(x, \pi, m) \in E \Leftrightarrow \begin{cases} (x, \pi, m) \in C, & \text{if } (x, \pi) \in V_A, \\ (x, m) \in D, & \text{otherwise.} \end{cases}$$

Now it obviously holds that $c(x, m) \in A \Leftrightarrow (x, h(x), m) \in E$, which concludes the proof of the theorem. □

Although the last theorem gives an upper bound on the complexity of a modification problem (c, V_A) w.r.t. $\mathcal{C}_{\text{MOD}}/cert(V_A)$, it fails to provide any results about uselessness of hints. Such results can be given with help of the following observation.

Observation 5.15. *Let (c, V_A) be a modification problem, $A = L(V_A)$, and $\mathcal{C}$ be a complexity class that is closed under $\leq_m^p$-reduction. Let $B \leq_{hi}^p (c, V_A)$ or $B \leq_{pi}^p (c, A)$ for some language B. It holds that*

$$(c, V_A) \in \mathcal{C}_{\text{MOD}}/cert(V_A) \Rightarrow B \in \mathcal{C}.$$

Proof. Let $(c, V_A) \in \mathcal{C}_{\text{MOD}}/cert(V_A)$. It follows by the definition of $\mathcal{C}_{\text{MOD}}^{\in}/cert(V_A)$ that $(c, V_A) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_A)$. It also follows by (i) the definition of $\mathcal{C}_{\text{MOD}}/\notin$ and (ii) the fact that a certificate function for V_A outputs the empty word ϵ for all inputs $x \notin A$, that $(c, A) \in \mathcal{C}_{\text{MOD}}/\notin$. Now, the assertion follows by Theorem 4.10, in case that $B \leq_{hi}^p (c, V_A)$, and from Theorem 5.9, in case that $B \leq_{pi}^p (c, A)$. □

Thus, if we can show for two NP-complete problems $A = L(V_A)$ and B that $B \leq_{hi}^p (c, V_A)$ or $B \leq_{pi}^p (c, A)$, we conclude that (c, V_A) has no useful selected hints. Assume to the contrary that $(c, V_A) \in \mathcal{C}_{\text{MOD}}/cert(V_A)$ for some complexity class $\mathcal{C} \subset \text{NP}$ that

modification function	SAT / 3SAT	Ex3SAT	1-3SAT
$\boldsymbol{ad}$ (addition of a unit cl.)	?	-	?
$\boldsymbol{adc}$ (addition of a clause)	?	?	?
$\boldsymbol{adlc}$ (add. of a lit. to a cl.)	$\leq^p_{pi}$	-	$\leq^p_{hi}, \leq^p_{pi}$
$\boldsymbol{rm}$ (removal of a unit cl.)	$\leq^p_{pi}$	-	$\leq^p_{pi}$
$\boldsymbol{rmc}$ (removal of a clause)	$\leq^p_{pi}$	$\leq^p_{pi}$	$\leq^p_{pi}$
$\boldsymbol{rmlc}$(rem. of a lit. from a cl.)	?	-	$\leq^p_{hi}, \leq^p_{pi}$
$\boldsymbol{neg}$ (negation of a unit cl.)	$\leq^p_{hi}, \leq^p_{pi}$	-	$\leq^p_{hi}, \leq^p_{pi}$
$\boldsymbol{negl}$ (neg. of a lit. of a cl.)	$\leq^p_{hi}, \leq^p_{pi}$	$\leq^p_{hi}, \leq^p_{pi}$	$\leq^p_{hi}, \leq^p_{pi}$

Table 5.3: The complexity of modified satisfiability problems.

	modification: add./rmvl.	ad	rm
VC	of an edge	?	$\leq^p_{pi}$
CLIQUE	of an edge	$\leq^p_{pi}$	?
HC	of an edge	$\leq^p_{pi}$	?
3DM	of a triple	$\leq^p_{pi}$	$\leq^p_{pi}$
PARTITION	of a number	$\leq^p_{hi}, \leq^p_{pi}$	$\leq^p_{hi}, \leq^p_{pi}$

Table 5.4: The complexity of VC, CLIQUE, HC, 3DM, and PARTITION when modified.

is closed under $\leq^p_m$-reduction. Then, also $B \in \mathcal{C}$ by the above observation. Since B is NP-complete we get NP $\subseteq \mathcal{C}$, a contradiction.

The following Tables 5.3 and 5.4 summarize the results for the complexity of modified instances when we use selected certificates, but the original instance does not necessarily has a solution. Abbreviations in the table should be clear from previous comments.

Conclusions

In this chapter we examined the case that the original instance has no solution and therefore no certificate can be given as a hint to decide modified instances. Instead, we get the promise that the original instance has no certificates. Since no verifiers are necessary to specify the form of certificates, we adapted the notion of a modification problem to be independent of a specific verifier. We introduced new complexity classes $\mathcal{C}_{\text{MOD}}/\notin$ to categorize the complexity of these modification problems. We used these classes to state easiness and hardness results, i.e., we gave modification problems where the no-solution promise is useless and problems where it is not.

We proved uselessness of the no-solution promise in two ways. First, we showed uselessness by a promise-independent reduction from an NP-complete problem A to a modification problem (c, B). Second, we showed uselessness by an promise-independent interreduction between two modification problems (c, A) and (c', B). Exemplarily, we applied the two methods of promise-independent reduction and promise-independent interreduction to the problems SAT and Ex3SAT. In Appendix A these two techniques are also used to show that the no-solution promise is useless for instances of the six basic problems from [GJ79], when these instances are appropriately modified. In contrast to the scenario in which selected certificates are given, we could show for all of the examined modification problems that either the no-solution promise is useless or that it yields a polynomial-time algorithm to decide modified instances.

In the last part of this chapter, we combined the results for selected hints from Chapter 4 and the results about the no-solution promise from this chapter. In consequence, we are able to appraise the use of selected hints independent of whether the original instance has a solution. Not to our surprise, we find that all considered modification problem are not easy, i.e., probably not an element of $\mathrm{P}_{\mathrm{MOD}}/cert$, when a selected certificate or the no-solution promise is given (see Tables 5.3 and 5.4). There are still a few open cases for which no unconditional uselessness results could be proven. But, for these open cases we showed in Chapter 4 and in Appendix A that selected certificates are probably hard to compute. Thus, in a scenario where no restriction is put on whether the original instance must have a certificate we find that selected hints are useless in all cases that we considered.

The reader may conjecture that selected hints are useless for *all* modification problems (c, V_A). This is not the case, as modification functions and verifiers may be chosen in very artificial ways. In Chapter 8 (Theorem 8.1) for example, we give a verifier V'_{SAT} for SAT such that $(ad, V'_{\mathrm{SAT}}) \in \mathrm{P}_{\mathrm{MOD}}/cert(V'_A)$. Also, the function that substitutes each given formula by a fixed 'yes'-instance of SAT is an extreme example of a modification function that makes deciding modified formulas easy.

Chapter 6

Non-NP-complete Problems

In this chapter we are concerned with the applicability of our techniques to problems that are probably not NP-complete. A candidate for such a problem is the graph isomorphism problem.

The graph-isomorphism problem, abbreviated GI, is to decide for a given pair of graphs whether they are isomorphic. In other words, the problem is to find a bijective function between the vertices of the graphs such that the edge incidences are respected. Formally,

$$\text{GI} := \{(G, H) : \text{there exists an isomorhism between } G \text{ and } H\}.$$

The problem GI is one of the few problems that is neither known to be NP-complete nor known to be in P. The problem GI belongs to SPP [AK06], and is useless as an oracle for Σ_2^p [Sch87]. As a consequence of these facts, NP-completeness of GI would imply the collapse of the polynomial-time hierarchy to its second level [Sch87].

When dealing with problems that possibly belong to P[1] we have to use a stronger reduction than $\leq_m^p$-reduction, e.g., $\leq_m^{log}$-reduction. Despite this fact we stick to $\leq_m^p$-reduction and to modification functions from FP for the following reason. Our theory of uselessness of hints was developed for problems $A \in \mathrm{NP}$ with a verifier V_A. Since the notion of a verifier becomes pathological for problems from P our theory applies best to problems from $\mathrm{NP} \setminus \mathrm{P}$. Also, if GI $\in \mathrm{P}$ then hints become less important since the problem is tractable even *without* the use of any hint.

We use the following verifier for GI:

$$((G, H), \phi) \in V_{\text{GI}} \Leftrightarrow \phi \text{ is an isomorphism from } G \text{ to } H.$$

For this verifier, we address in the next section 6.1 the issue of deciding modified GI-instances when an arbitrary solution is given. Afterwards, in Section 6.2 we address this issue for selected certificates. In Section 6.3 we study the scenario in which the promise is given that the original instance has no solution.

[1]Actually, even P-hardness of GI w.r.t. some adequate reduction is uncertain. The strongest known result towards this direction is that GI is hard under $\leq_{log}^p$-reduction for the complexity class DET, the class of problems that are solvable by NC^1 circuits with additional oracle gates that compute the determinant of integer matrices [Tor04].

6.1 Uselessness of Arbitrary Certificate

Our theory of arbitrary hints, which was developed in Chapter 3, and in particular the notion

$$\text{MOD}_c V_A := \{(x, \pi, m) : (x, \pi) \in V_A \ \ \textit{and} \ \ c(x, m) \in A\},$$

is independent of V_A being a verifier for an NP-*complete* problem, and is also applicable to the verifier V_{GI}. Also, Observation 3.2, which says that the problem $\text{MOD}_c V_A$ cannot be any harder than A, holds for the special case V_{GI}. Thus, if we want to show, that a problem $\text{MOD}_c V_{\text{GI}}$ is as hard as the problem GI it suffices to show hardness of $\text{MOD}_c V_{\text{GI}}$, i.e., that GI $\leq_m^p$ $\text{MOD}_c V_{\text{GI}}$.

When looking at GI-instances (G, H), it suffices to consider graphs G and H with the same number of vertices and edges. Otherwise $(G, H) \notin$ GI and consequently $((G, H), \phi, m) \notin \text{MOD}_c V_{\text{GI}}$ for all $\phi, m \in \Sigma^*$ and any modification function c.

The most elementary nontrivial possibility of modification is the addition or removal of a single edge in each of G and H. Let ad_2 and rm_2 denote the corresponding modification functions. The subscript '2' accounts for the property that two edges are added to or removed from the GI-instance. Formally, ad_2 and rm_2 are defined by

$$ad_2\big((G, H), (e, f)\big) := \Big(\big(V(G), E(G) \cup \{e\}\big) \ , \ \big(V(H), E(H) \cup \{f\}\big) \Big),$$

$$rm_2\big((G, H), (e, f)\big) := \Big(\big(V(G), E(G) \setminus \{e\}\big) \ , \ \big(V(H), E(H) \setminus \{f\}\big) \Big).$$

We show existence of useless hints for GI w.r.t. these both modification function via the following results.

Theorem 6.1. *GI* $\leq_m^p$ $\text{MOD}_{ad_2} V_{GI}$.

Proof. Let $x = (G, H)$ be a GI-instance. Let $V(G) = \{v_1, ..., v_n\}$ and let $V(H) = \{w_1, ..., w_n\}$. Let G' denote the graph defined by

- $V(G') := V(G) \cup V(H) \cup \{u_i : 1 \leq i \leq 4\}$ and
- $E(G') := E(G) \cup E(H) \cup \{\{v_i, u_1\}, \{w_i, u_3\} : 1 \leq i \leq n\}$.

Now the reduction functions $f_1, f_2, f_3 \in \text{FP}$ are given by

- $f_1(x) := (G', G')$,
- $f_2(x) := id$, and
- $f_3(x) := (\{u_1, u_2\}, \{u_3, u_4\})$.

See Figure 6.1 for an illustration of the instance $ad_2(f_1(x), f_3(x))$. It is obvious that $f_2(x) \in V_{\text{GI}}(f_1(x))$. Furthermore, the instance $ad(f_1(x), f_3(x))$ is in GI if and only if the two $n+1$-vertex components of $ad(f_1(x), f_3(x))$ are isomorphic. The latter is equivalent to the proposition that G and H are isomorphic. Thus

$$x \in \text{GI} \Leftrightarrow (f_1(x), f_2(x)) \in V_{\text{GI}} \ \wedge \ ad(f_1(x), f_3(x)) \in \text{GI}.$$

□

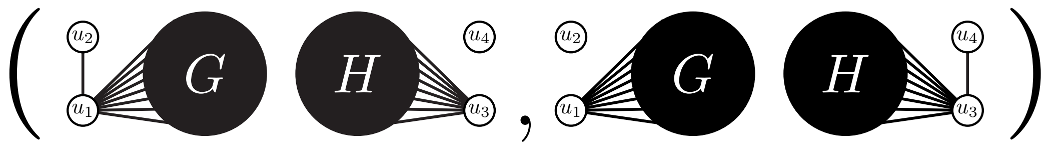

Figure 6.1: The GI-instance $ad_2(f_1((G,H)), f_3((G,H)))$.

The problem $\text{MOD}_{ad_2}V_{\text{GI}}$ is as hard as the problem GI. We conclude that there exist certificates that are useless as a hint. An analogous result holds for the modification function rm_2.

Theorem 6.2. *GI* $\leq^p_m \text{MOD}_{rm_2}V_{GI}$.

Proof. The proof is similar to the proof of Theorem 6.1. The only difference to the proof of Theorem 6.1 is that the edges $\{u_1, u_2\}$ and $\{u_3, u_4\}$ are already contained in both copies of G'. The modification consists of deleting $\{u_3, u_4\}$ from one copy of G' and $\{u_1, u_2\}$ from the other one, such that the resulting GI-instance is identical to the instance $ad_2(f_1((G,H)), f_3((G,H)))$ from the proof of Theorem 6.1. $\square$

6.2 Uselessness of Selected Certificate

The notions and definitions that we established in Chapter 4, and which we applied to NP-complete problems, are also applicable to the problem GI, which is probably not NP-complete. In particular we use Theorem 4.10, which we restate here for convenience.

> **Theorem 4.10**: *Let V_A be a verifier for some language $A \in \text{NP}$ and let $\mathcal{C}$ be some complexity class that is closed under $\leq^p_m$-reduction. Then for all languages B it holds that*
>
> $$\left(B \leq^p_{hi} (c, V_A) \wedge (c, V_A) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_A)\right) \Rightarrow B \in \mathcal{C}.$$

We apply Theorem 4.10 in the following way. We show that GI $\leq^p_{hi} (c, V_{\text{GI}})$ for some modification function c. Under the assumption that modified instances are easy to decide with selected certificates, i.e., if $(c, V_{\text{GI}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V_{\text{GI}})$ for some complexity class $\mathcal{C} \subset \{A : A \leq^p_m \text{GI}\}$, we get by Theorem 4.10 that the language GI would become an element of $\mathcal{C}$. But, if GI $\in \mathcal{C}$ there is no use of deciding the modified instance with help of selected certificates and a $\mathcal{C}$-predicate, since the modified instance can be decided directly, without the hint.

Unfortunately, we are unable to show uselessness of selected certificates for a small, e.g., constant, amount of modification. Any attempt to prove GI $\leq^p_{hi} (c, V_{\text{GI}})$ for some small modification c was frustrated by the apparently much more constrained nature of GI in comparison to the NP-complete problems that are studied in Appendix A. The smallest modification for which we are able to show uselessness of selected hints

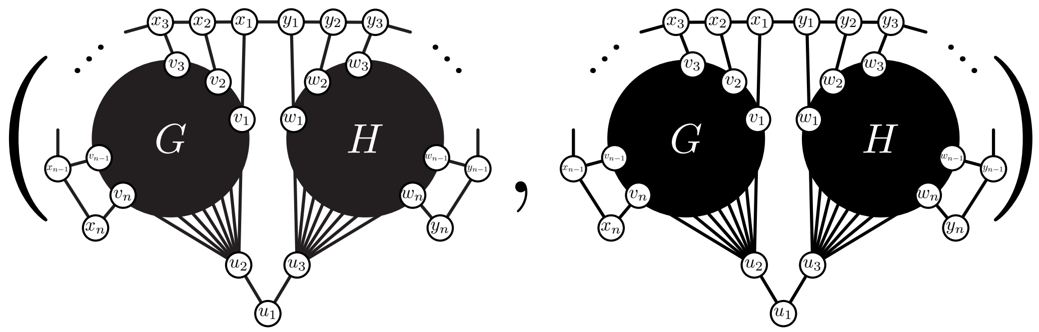

Figure 6.2: The GI-instance (G', G') before modification.

is the simultaneous addition, or removal, of $q(|(G, H)|)$ edges in both graphs, where $q(n) = \sqrt[K]{n}$ and $K \in \mathbb{N}$ a constant.

Theorem 6.3. *Let $K \in \mathbb{N} \setminus \{0\}$ and $q(n) = \sqrt[K]{n}$. Then GI $\leq_{hi}^{p} (rm_2^q, V_{GI})$.*

Proof. Let (G, H) be a GI-instance. In a nutshell, the proof is as follows. First, we construct from (G, H) a GI instance $f_1((G, H)) := (G', G')$ that has exactly one isomorphism, namely, the obvious isomorphism id. Afterwards, we modify (G', G') to $(G', G')^m$ by deleting $2|V(G)|$ edges in both parts of (G', G'). We then show that the modified instance $(G', G')^m$ belongs to GI if and only if $(G, H) \in$ GI. Last, we use a padding argument in a way that the $2|V(G)|$ many modification become relatively small in comparison to the size of (G', G').

Let $V(G) = \{v_1, ..., v_n\}$ and $V(H) = \{w_1, ..., w_n\}$. Let ϕ_{id} denote the mapping that is given by $\phi_{id}(v_i) := w_i$, for all $1 \leq i \leq n$. We may assume in the following that ϕ_{id} is no isomorphism between G and H and $n \geq 3$; the other cases can be dealt with separately. We construct an auxiliary graph G' as follows:

- $V(G') := V(G) \cup V(H) \cup \{x_i, y_i : 1 \leq i \leq n\} \cup \{u_1, u_2, u_3\}$,
- $$\begin{aligned} E(G') := E(G) \cup E(H) \cup \{&\{v_i, x_i\}, \{v_i, u_2\}, \{w_i, y_i\}, \{w_i, u_3\} : 1 \leq i \leq n\} \cup \\ &\{\{x_i, x_{i+1}\}, \{y_i, y_{i+1}\} : 1 \leq i \leq n-1\} \cup \\ &\{\{x_1, y_1\}, \{u_1, u_2\}, \{u_1, u_3\}\}. \end{aligned}$$

The instance $f_1((G, H))$ is given by (G', G') and is depicted in Figure 6.2 .

We now show that id is the only isomorphism for (G', G'). Assume to the contrary that there exists another isomorphism $\phi \neq id$ for (G', G'). Note that since $n \geq 3$ the vertices u_2 and u_3 are the only vertices of maximum degree in G'. The vertex u_1 is the only vertex that has distance 1 to both of the vertices of maximum degree, thus $\phi(u_1) = u_1$.

Case 1: $\phi(u_2) = u_3$**:** Note that there are only two vertices in G' that have degree 2 and distance greater than 1 from u_2 or u_3, namely x_n and y_n. Among these two vertices x_n and y_n, the vertex x_n is the only vertex with distance 2 to u_2 and the vertex y_n is the only vertex with distance 2 to $u_3 = \phi(u_2)$. Thus $\phi(x_n) = y_n$.

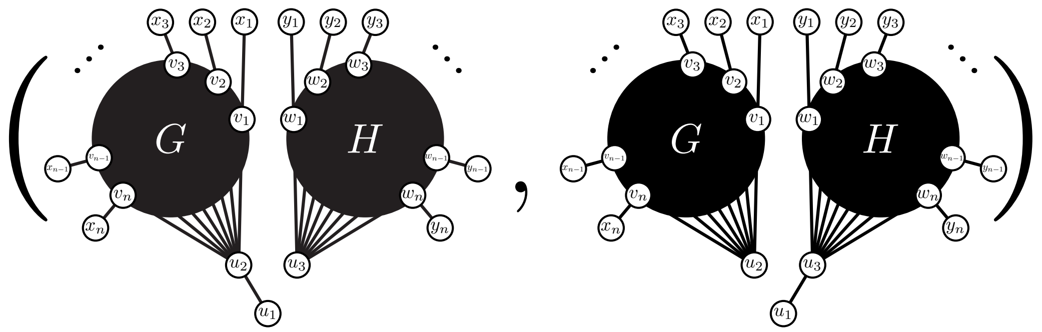

Figure 6.3: The modified GI-instance $(G', G')^m$.

The vertex x_{n-1} is the sole vertex with distance 1 to x_n and distance 2 to u_2. The same holds for the vertices y_{n-1}, y_n and u_3. Consequently, $\phi(x_{n-1}) = y_{n-1}$. Inductive application of this argument, namely that x_i (y_i) is the sole vertex with distance 1 to x_{i-1} (y_{i-1}) and distance 2 to u_2 (u_3), we obtain $\phi(x_i) = y_i$ for all $i \leq n$.

For every vertex x_i, $1 \leq i \leq n$, there is only one adjacent vertex that has distance 1 to u_2, namely v_i. The same holds for y_i and u_3, which have w_i as a common neighbor. Consequently, $\phi(v_i) = w_i$ for all $1 \leq i \leq n$. Thus, ϕ_{id} yields an isomorphism on (G, H), a contradiction to our assumptions.

Case 2: $\phi(u_2) = u_2$**:** Using similar arguments as in Case 1, we can show that $\phi(v) = v$ for all $v \in V(G')$. But then $\phi = id$, a contradiction to the assumption $\phi \neq id$.

Thus, (G', G') has exactly one isomorphism.

Now we modify (G', G') to $(G', G')^m$ such that $(G', G')^m \in$ GI $\Leftrightarrow (G, H) \in$ GI. The modification consists of the deletion of all the edges $\{x_i, x_{i+1}\}$ and $\{y_i, y_{i+1}\}$, for $1 \leq i \leq n-1$, as well as the deletion of the edge $\{x_1, y_1\}$. All these edges are deleted in both copies of G'. Furthermore, we delete the edge $\{u_1, u_3\}$ in one copy of G' and the edge $\{u_2, u_3\}$ in the other copy. For an illustration of the modified instance $(G', G')^m$ see Figure 6.3. Note that $(G', G')^m$ consists of graphs that have two components each, one component with $2n + 1$ vertices and another component with $2n + 2$ vertices. It is easy to see that the respective components are isomorphic if and only if G and H are isomorphic. Consequently, $(G', G')^m \in$ GI $\Leftrightarrow (G, H) \in$ GI.

Up to now, we know that GI $\leq^p_{hi}$ $(rm_2^{2|V(G)|}, V_{\text{GI}})$. Now, we show that GI $\leq^p_{hi}$ (rm_2^q, V_{GI}), where $q(n) = \sqrt[K]{n}$ and $K \in \mathbb{N} \setminus \{0\}$. Therefore, we pad the graph G' by adding to G' a component G_{pad} with at least $2^k |V(G)|^k$ vertices and such that $G_{pad} \neq G'$ and (G_{pad}, G_{pad}) has only the trivial isomorphism id. An example of such a graph G_{pad} is a long enough path in which an additional vertex is appended to one of the two vertices with distance 2 to an endpoint of the path. Correctness of this construction is obvious. □

Theorem 6.4. *Let $K \in \mathbb{N} \setminus \{0\}$ and $q(n) = \sqrt[K]{n}$. Then GI $\leq^p_{hi}$ (ad_2^q, V_{GI}).*

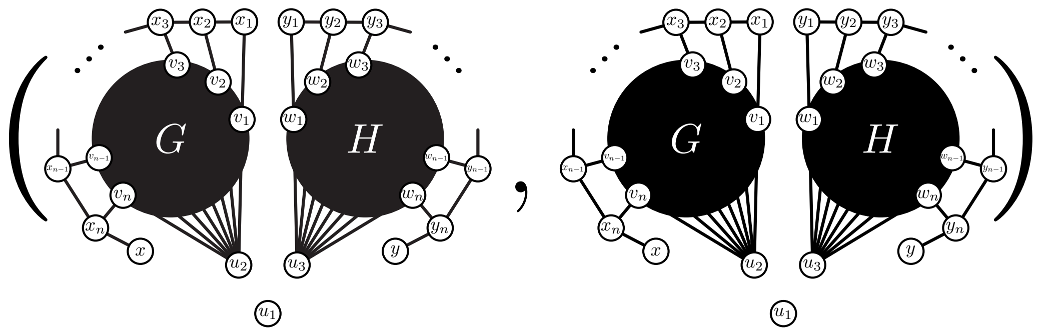

Figure 6.4: The GI-instance (G', G') when the modification is ad^q.

Proof. The proof is similar to the proof of Theorem 6.3. The auxiliary graph G' is almost as in the proof of Theorem 6.3, except that the edges $\{u_1, u_2\}$,$\{u_1, u_3\}$, and $\{x_1, y_1\}$ are missing and that new vertices x and y are introduced, together with the edges $\{x_n, x\}$ and $\{y_n, y\}$. The instance $f_1((G, H))$ is again given by (G', G'). See Figure 6.4 for an illustration of (G', G'). Using similar arguments as above, it is obvious that also for this graph G' the GI-instance (G', G') has only the trivial isomorphism id. We modify (G', G') by adding all the missing edges between the vertices $x, x_1, ..., x_n$ and all the edges between the vertices $y, y_1, ..., y_n$, which results in two $n + 1$-cliques. Also, we add in one copy of G' the edge $\{u_1, u_2\}$ and in the other copy of G' the edge $\{u_1, u_3\}$. It is easy to see, that this modified instance $(G', G')^m \in$ GI if and only if $(G, H) \in$ GI.

The assertion follows by a padding argument: We simply add large enough graphs G_{pad}, e.g., graphs of size $2^k \cdot |V(G)|^{2k}$, to G'. To make this construction work, we also assume that the GI-instance (G_{pad}, G_{pad}) only has one isomorphism, namely id. □

6.3 No Solution as a Promise

In this section we answer the question if the knowledge that two graphs are not isomorphic is of any use when modifying these graphs. We use our framework from Section 5 to answer this question. In detail, we use Theorem 5.9, which we restate here for convenience.

Theorem 5.9: *Let $A \in$ NP, c be a modification function, and $\mathcal{C}$ be a complexity class that is closed under $\leq_m^p$-reduction. For all languages B it holds that*

$$B \leq_{pi}^p (c, A) \wedge (c, A) \in \mathcal{C}_{\text{MOD}}/\notin \Rightarrow B \in \mathcal{C}.$$

By this theorem, it suffices to prove that GI $\leq_{pi}^p (c,\text{GI})$ in order to show that the no-solution promise is a useless hint for the modification problem (c,GI). We are able to show this for the two modification functions ad_2 and rm_2.

Theorem 6.5. *GI $\leq_{pi}^p (c, GI)$ for $c \in \{ad_2, rm_2\}$.*

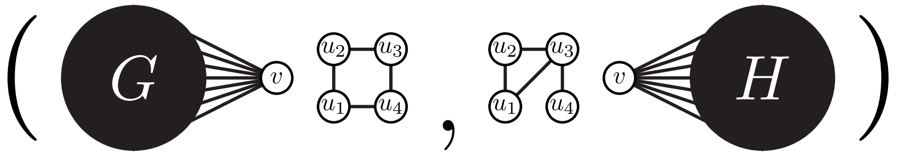

Figure 6.5: The GI-instance (G', H') from the proof of GI $\leq^p_{pi}$ (rm_2, GI).

Proof. Let (G, H) be a GI-instance with $V(G) = V(H) = \{v_1, ..., v_n\}$. We assume that $n \geq 4$; the finite many special cases with $n < 4$ can be handled separately. We construct from (G, H) the GI instance $f_1((G, H))$ as illustrated in Figure 6.5. That is, $f_1((G, H)) = (G', H')$, where $V(G') = V(H') = V(G) \cup \{v, u_1, ..., u_4\}$,

$$E(G') = E(G) \cup \{\{v_i, v\} : 1 \leq i \leq n\} \cup \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, u_4\}, \{u_4, u_1\}\},$$

$$E(H') = E(H) \cup \{\{v_i, v\} : 1 \leq i \leq n\} \cup \{\{u_1, u_2\}, \{u_2, u_3\}, \{u_3, u_1\}, \{u_3, u_4\}\}.$$

Obviously, G' and H' are not isomorphic, since G' has no degree-one vertex in its single 4-vertex component, but H' has. Thus, $f_1(G, H) \notin$ GI. The function f_1 is the same for both modifications ad_2 and rm_2.

Now, let $c = rm_2$. We modify (G', H') by deleting the edge $\{u_2, u_3\}$ in both of G' and H'. In consequence, the resulting graphs have as their single four-vertex components paths of length three. The $n + 1$-vertex components of the modified graphs are isomorphic if and only if G and H are isomorphic. This shows the assertion.

In case $c = ad_2$, the modification is the addition of the edge $\{u_2, u_4\}$ in both of G' and H', which results in isomorphic four-vertex components. □

Conclusions

In this chapter we showed that the techniques that we have developed in the last few chapters are also applicable to problems that are probably not NP-complete. In particular, we showed for the problem GI, which contains pairs of isomorphic graphs, that

1. there exist useless certificates when the modification is the removal or the addition of two edges,

2. all certificates are useless when the modification is the removal or the addition of polynomially many edges, and

3. the no-solution promise is a useless hint when the modification is the removal or the addition of two edges.

We were unable to show uselessness of selected hint for a small amount of modification.

Open problem 2. *Does GI $\leq^p_{hi}$ (ad_2^k, V_{GI}) or GI $\leq^p_{hi}$ (rm_2^k, V_{GI}) hold for some constant $k \geq 1$?*

Chapter 7

Approximated Solutions of Modified Instances

We have seen in the last chapters that often a solution for the original instance is a completely useless hint when we ask for a solution of a slightly modified instance. For practical purposes these are rather unsatisfying results. In this section we show that when we weaken our demands on the quality of solutions then modified instances can indeed benefit from given hints.

In detail, we ask if an optimum solution for the original instance can be helpful to find a not necessarily optimum, but sufficiently good, solution for a modified instance. In other words, can we find good approximated solutions for modified instances of optimization problems?

7.1 Approximation Basics

We start by defining the notion of an optimization problem. For each input instance of an optimization problem there is a set of feasible solutions associated to it. For each feasible solution a certain cost is defined. The task is to find a feasible solution with optimum cost. Slightly rephrasing definitions from [ACG$^+$99], we define an NP-optimization problem as a 3-tuple $(V_B, cost, goal)$, where

- V_B is a verifier for some language $B \in \mathrm{NP}$. We say that $V_B(x)$ is the set of feasible solutions of x.
- Given an instance x and a feasible solution $y \in V_B(x)$, $cost(x, y)$ is a polynomial-time computable positive rational and
- $goal$ is either min or max.

We denote by NPO the class of all NP-optimization problems. As it is usual in the literature, we define an optimization problem B in the following way:

PROBLEM: **PROBLEM NAME** (which begins with 'Min' or 'Max')

INSTANCE: Specifies the input instances for B.

SOLUTION: Specifies the feasible solutions for an input x.

MEASURE: Defines how the cost of a feasible solution is measured.

From this description, the formal triple $(V_B, cost, goal)$ can be derived. For an example of an NP-optimization problem see the definition of MINTSP at the beginning of Section 7.4.

For an NP-optimization problem the task is to find a feasible solution y that achieves the best objective value, i.e., a solution y with

$$cost(x, y) = goal\{cost(x, y') : y' \in V_B(x)\}.$$

We denote by $opt(x)$ the cost of the optimum solution for an instance x. If y is a solution for x then the *performance ratio* of y is defined as

$$R(x, y) = \begin{cases} cost(x, y)/opt(x), & \text{if } goal = min, \\ opt(x)/cost(x, y), & \text{if } goal = max. \end{cases}$$

Thus $R(x, y)$ is always at least 1; the closer it is to 1, the closer the solution is to the optimum. For a function $f : \mathbb{N} \to [1, \infty)$, we say that an algorithm A is an $f(n)$-*approximation algorithm* if and only if for every input x it holds that $R(x, A(x)) \leq f(|x|)$.

We categorize the hardness of optimization problems by sorting them into the following descending chain of complexity classes. We say that an optimization problem $(V_B, cost, goal) \in \mathrm{NPO}$ belongs to the class

- APX: if and only if there exist a constant function $f(n) = \delta$ and an algorithm A such that A is an $f(n)$-approximation algorithm. In this case we say that $(V_B, cost, goal)$ is δ-approximable.

- PTAS (polynomial-time approximation scheme), or has a PTAS: if and only if there exists an algorithm A such that for every input pair (x, ε), $\varepsilon \in \mathbb{R}$ and $\varepsilon > 0$, the algorithm A computes a solution y with $R(x, y) \leq 1 + \varepsilon$ in time $p_\varepsilon(|x|)$, where p_ε is a polynomial that depends on ε, but not on x.

- FPTAS (fully polynomial-time approximation scheme), or has an FPTAS: if and only if there exists an algorithm A and a polynomial p such that for every input pair (x, ε), $\varepsilon \in \mathbb{R}$ and $\varepsilon > 0$, the algorithm A computes a solution y with $R(x, y) \leq 1 + \varepsilon$ in time $p(|x|, 1/\varepsilon)$.

It is obvious that $\mathrm{FPTAS} \subseteq \mathrm{PTAS} \subseteq \mathrm{APX} \subseteq \mathrm{NPO}$. Unless $\mathrm{P} = \mathrm{NP}$ these inclusions are strict. For more details on approximation we refer to [ACG+99].

Finally, we say that an optimization problem is *solvable with an absolute-error guarantee of* k if there exists an algorithm A such that for every input x the solution $A(x)$ differs from an optimum solution by at most k, i.e., $|cost(A(x)) - opt(x)| \leq k$. From a practitioners point of view an approximation algorithm with an absolute error guarantee of 1 is the best one can get. But, such a constant error approximation even exists for problems that do not belong to APX, unless P=NP.[1]

7.2 Approximation of modification problems

In the context of this thesis, we are interested in approximating solutions of slightly modified instances of optimization problems, assuming that an *optimum* solution for the original instance is known. This problem is referred to as *reoptimization* in the literature [Sch97, ABS03, AEMP06, EMP07, BHMW08]. Again, we distinct between the following two scenarios: 1.) knowledge of an arbitrary optimum solution and 2.) knowledge of a carefully selected optimum solution of the original instance. In the literature, so far only the first approach is considered. We will show our positive results, i.e., good approximation algorithms, to be valid for any given solution. In contrast we will prove that all our negative results, i.e., nonexistence of an FPTAS or nonexistence of an $f(n)$-approximation, are valid independent of the given optimum solution.

To formally distinguish between these both cases, we introduce, in analogy to previous chapters, the two notions $\mathrm{MOD}_c(V_B, cost, goal)$ and $(c, (V_B, cost, goal))$. Here c is a modification function and $(V_B, cost, goal) \in \mathrm{NPO}$. We say that an algorithm A is an $f(n)$-*approximation algorithm* for $\mathrm{MOD}_c(V_B, cost, goal)$ if and only for all original instances $x \in \Sigma^*$, for all modifications $m \in \Sigma^*$, and for *all* optimum solutions y_{opt} of x, the algorithm A computes from (x, y_{opt}, m) a solution y^A for $c(x,m)$ such that $R(c(x,m), y^A) \le f(|c(x,m)|)$.

To define the notion of $f(n)$-approximability for the problem $(c, (V_B, cost, goal))$ we substitute the phrase 'for *all* optimum solutions y_{opt} of x' by 'there *exists* an optimum solution y_{opt} of x'. Hence, we say that an algorithm A is an $f(n)$-*approximation algorithm* for $(c, (V_B, cost, goal))$, if for all original instances $x \in \Sigma^*$, for all modifications $m \in \Sigma^*$, there *exists* an optimum solution y_{opt} of x such that the algorithm A computes from (x, y_{opt}, m) a solution y^A for $c(x,m)$ with $R(c(x,m), y^A) \le f(|c(x,m)|)$.

We can use the above notion of $f(n)$-approximability to state, in analogy to the last section, what containment of $\mathrm{MOD}_c(V_B, cost, goal)$ and $(c, (V_B, cost, goal))$ in APX means. Similar definitions can also be given to state that a reoptimization problem $\mathrm{MOD}_c(V_B, cost, goal)$ or $(c, (V_B, cost, goal))$ has an PTAS, has an FPTAS, or has an approximation with an absolute-error guarantee.

7.3 Some Trivial Results

Given an optimization problem B and a modification function c, an algorithm that computes a solution for $c(x,m)$ may ignore the hint, i.e., the optimum solution of the orig-

[1]The problem MINTSP* with $cost^*(G,T) = \big(\sum_{e\in T} c(e)\big) \cdot \big(\max_{e \in E(G)} c(e) \cdot |V(G)|\big)^{-1}$ does not belong to APX, unless $\mathrm{P} = \mathrm{NP}$. On the other hand, for every feasible solution T of a MINTSP*-instance G it holds that $cost(G,T) \le 1$. Consequently, it it not hard to find a feasible solution with an absolute error of 1.

inal instance x. Thus $\mathrm{MOD}_c B$ and (c, B) are not harder than B.

Observation 7.1. *Let $B \in \mathrm{NPO}$. If B is $f(n)$-approximable (has a PTAS, has an FPTAS, or is solvable with an absolute-error guarantee of k) then $\mathrm{MOD}_c B$ and (c, B) are $f(n)$-approximable (have a PTAS, have an FPTAS, or are solvable with an absolute-error guarantee of k).*

Furthermore, any result for approximability of (c, B) is also valid for the problem $\mathrm{MOD}_c B$.

Observation 7.2. *If (c, B) is $f(n)$-approximable (has a PTAS, has an FPTAS, or is solvable with an absolute-error guarantee of k), then $\mathrm{MOD}_c B$ is $f(n)$-approximable (has a PTAS, has an FPTAS, or is solvable with an absolute-error guarantee of k, respectively).*

This last result is valid because the respective definitions of $f(n)$-approximability only differ by the quantifier for the hint y.

A nice criterion for NP-hardness of a reoptimization problem $\mathrm{MOD}_c A$ is given in [BHMW08].

Lemma 7.3 ([BHMW08]). *Let A be an NP-hard optimization problem, c be a modification function, and B be a subset of the input instances of A that is solvable in polynomial time. If c is a modification function, such that every input instance x of A can be obtained from a B-instance by polynomial many applications of c, then $\mathrm{MOD}_c A$ is NP-hard.*

For a proof we refer to [BHMW08] — in a nutshell, the proof uses the fact that Turing reductions are sufficient to show NP-hardness of an optimization problem. By application of this Lemma, it is an easy task for the reader to verify, that all coming reoptimization problems $\mathrm{MOD}_c A$ are NP-hard.

Next we show two simple examples of reoptimization problems for which *any* optimum solution y_{opt} of the original instance x can be used to get a very good solution for a modified instance $c(x, m)$. In detail, we give two examples where the old optimum solution y_{opt} is slightly modified and gives a solution for the modified instances that differs by at most 1 or 2, respectively, from $opt(c(x, m))$. The first example is the problem MinVertexCover (MinVC, for definition see [ACG+99]) and the modification function ad, which adds an edge to G. Note that MinVC is not approximable with factor 1.37 in the usual non-reoptimization case [DS05].

Theorem 7.4. $\mathrm{MOD}_{ad}\textsc{MinVC}$ *is approximable with an absolute error of 1.*

Proof. Let G^o be the original instance, C^o_{opt} be a minimum vertex cover for G^o, and $\{u, v\}$ be the edge that is added to G^o. Each solution for $G^m := ad(G^o, \{u, v\})$ has at least size $|C^o_{opt}|$, otherwise C^o_{opt} was no optimum solution for G^o. Thus, $opt(G^m) \geq |C^o_{opt}|$. Furthermore, by adding one of u or v to C^o_{opt} we get a vertex cover for G^m of size at most $|C^o_{opt}| + 1 \leq opt(G^m) + 1$. □

Our second exemplary problem, which is a bit more interesting, is the problem MINMAXMATCH. It is defined as follows:

PROBLEM: MINMAXMATCH

INSTANCE: Graph $G = (V, E)$.

SOLUTION: A maximal matching E', i.e., a subset $E' \subseteq E$ such that no two edges in E' share a common endpoint and every edge in $E \setminus E'$ shares a common endpoint with some edge in E'.

MEASURE: Cardinality of the matching, i.e., $|E'|$.

The problem MINMAXMATCH is APX-complete [YG80] and has a factor 2 approximation [ACG+99], but is approximable with a constant error of 2 when reoptimization is used.

Theorem 7.5. MOD$_{ad}$MINMAXMATCH *is approximable with an absolute error of 2.*

Proof. Let $G^o = (V, E)$ be the original instance, M^o_{opt} be a maximal matching of minimum size for G, and $e = \{u, v\}$ be an edge to be added. Let $G^m := ad(G^o, e)$ be the modified graph and M^m_{opt} be an optimum solution for G^m. We claim that $\left|M^m_{opt}\right| \geq \left|M^o_{opt}\right| - 1$. Assume to the contrary that $\left|M^m_{opt}\right| \leq \left|M^o_{opt}\right| - 2$.

Case 1: $e \notin M^m_{opt}$, e and M^m_{opt} share no common vertex: Then M^m_{opt} is not maximal, a contradiction.

Case 2: $e \notin M^m_{opt}$, e and M^m_{opt} share at least one common vertex with M^m_{opt}: Thus, the matching M^m_{opt} is also a maximal matching for G^o, but is smaller than M^o_{opt}, a contradiction.

Case 3: $e \in M^m_{opt}$: When removing the edge e from M^m_{opt} we obtain a matching of size $\left|M^m_{opt}\right| - 1$. But, this matching does not need to be maximal in G^o. If this is the case then $M^m_{opt} \setminus \{e\}$ can be expanded to a maximum matching by adding at most two other edges f and g, where $f, g \neq e$, that contain the vertices u and v respectively. This matching $M^m_{opt} \setminus \{e\} \cup \{f, g\}$ is a maximal matching for G^o of size at most $\left|M^m_{opt}\right| + 1$, and hence smaller than M^o_{opt}. This contradicts the fact, that M^o_{opt} is optimum for G^o.

This shows that $\left|M^m_{opt}\right| \geq \left|M^o_{opt}\right| - 1$.

An algorithm that approximates a minimum maximal matching for G^m with an absolute error of 2 works as follows. First, we test if the matching M^o_{opt} is still maximal for G^m. If this is the case we output M^o_{opt}, which results in a solution with an absolute error of at most 1. If M^o_{opt} is not maximal in G^m, it can be made maximal by adding the edge e to M^o_{opt}. This yields an approximation with an absolute error of at most 2. □

Using similar arguments, it should be an easy task for the reader to verify that the problems MOD$_{rm}$MINVC and MOD$_{rm}$MINMAXMATCH also are approximable with an absolute error guarantee of 1 and 2, respectively. For the majority of unweighted approximation problems listed in [ACG+99] we were able to give algorithms that approximate a solution for slightly modified instances with an absolute error of 1 or 2. The respective modifications are chosen in a canonical way. We desist from giving a detailed list of all of these easy problems. But, we give in Appendix B an example of an unweighted optimization problem that has *no* absolute-error approximation, unless $\mathrm{P} = \mathrm{NP}$. Namely, we show this for the problem MINMAXIS.

7.4 The Travelling Salesperson Problem (MINTSP)

In the last section we indicated that many unweighted optimization problems have an approximation with an absolute-error of at most 1 or 2. If we deal with *weighted* modification problems, for example weighted graphs or formulas, a solution for the modified instance with some absolute error seems hard to find. In this section we exemplarily study approximability of weighted modification problems for the famous travelling salesperson problem (TSP).

PROBLEM: MINTSP

INSTANCE: A complete graph $G = K_n$ and a function $w : E(G) \to \mathbb{N}$ assigning a weight to each edge of G.

SOLUTION: A Hamiltonian cycle T in G. Such a cycle is called a tour.

MEASURE: The length of the tour T, i.e., $\sum\limits_{e\in T} w(e)$.

A problem MAXTSP can be defined in the same way and is studied in Appendix B. The problem MINTSP is NPO-complete and admits no $p(n)$-approximation, p polynomial,unless $\mathrm{P} = \mathrm{NP}$.

For MINTSP and MAXTSP, the task of reoptimization has already been addressed in [AEMP06]. The modification that is considered in [AEMP06] is the addition of a vertex, i.e., a new city, to the graph, thereby also assigning a cost to all newly introduced edges. Let adv denote the corresponding modification function. It is shown in [AEMP06] that MOD_{adv}MINTSP is not approximable with ratio $2^{p(n)}$, p polynomial, unless $\mathrm{P} = \mathrm{NP}$.

In this thesis, we examine reoptimization w.r.t. edge-cost modifications, that is, small modifications to the weight-function w. In detail, we consider the modification functions

- $inc((G, w), (e, i)) := (G, w')$, where $w'(e) := w(e) + i$, and
- $dec((G, w)), (e, i) := (G, w')$, where $w'(e) := w(e) - i$,

and $w'(e') = w(e')$ for all other edges $e' \neq e$. Reoptimization of MINTSP w.r.t. inc and dec has already been examined in [BFH+07] and [BHMW08]. For example, the following result is already in [BFH+07].

Theorem 7.6 ([BFH+07]). *Let p be a polynomial. Unless $\mathrm{P} = \mathrm{NP}$, the problems MOD_{dec}MINTSP and MOD_{inc}MINTSP are not $p(n)$-approximable.*

Thus, when *some unselected* optimum solution of the original instance is given, the problem of reoptimizing MinTSP is as hard as the problem MinTSP itself. We are able to improve on this result, by showing that *no* solution is sufficient for this task. First, we prove hardness of (inc, MinTSP).

Theorem 7.7. *Unless* $\mathrm{P} = \mathrm{NP}$, (inc, MinTSP) *is not* $2^{|V|}$*-approximable.*

Proof. Let (inc, MinTSP) be $2^{|V|}$-approximable. Let A be an algorithm that, given an original instance G^o, an optimum solution T^o_{opt} of G^o, and a modification m, computes a solution T^A for $G^m := inc(G^o, m)$ with $cost(G^m, T^A) \leq 2^{|V(G^m)|} opt(G^m)$. We show that HC $\in$ P under this assumption.

Let $G = (\{v_1, ..., v_n\}, E)$ be a graph. We assume the reader to be familiar with the proof of Theorem 9.27. Let G' be defined as in the proof of Theorem 9.27. Using G' we construct a TSP-instance $G^o = (K_{|V(G')|}, w)$ where

$$w(e) := \begin{cases} 1, & \text{if } e \in E(G') \setminus \{\{v_1^1, v_4^1\}\}, \\ 2, & \text{if } e = \{v_1^1, v_4^1\}, \\ 2^{8n} \cdot (8n+1), & \text{otherwise.} \end{cases}$$

Recall from the proof of Theorem 9.27 that C' is the sole Hamiltonian cycle of the graph $\big(V(G'), E(G') \setminus \{v_1^1, v_4^1\}\big)$, therefore C' is the sole optimum tour in G^o having $cost(G^o, C') = |V(G')| = 8n$. Thus, the optimum solution that A gets as part of its input has to be C'. We modify G^o by incrementing the weight of the edge $\{v_3^1, v_5^1\}$ from 1 to $2^{8n} \cdot (8n+1)$. Let G^m denote the modified graph and let m be such that $G^m := inc(G^o, m)$.

When applying A to the input (G^o, C', m) we obtain a solution T^A. We claim that

$$G \in \text{HC} \Leftrightarrow cost(G^m, T^A) \leq 2^{8n} \cdot (8n+1),$$

which yields a polynomial time algorithm for HC.

To show sufficiency, suppose that G has a Hamiltonian cycle. This cycle induces a tour with cost $8n+1$ in G^m, i.e., $opt(G^m) \leq 8n+1$. Since A gives a $2^{|V|}$-approximation we conclude that

$$cost(G^m, T^A) \leq 2^{8n} \cdot opt(G^m) \leq 2^{8n} \cdot (8n+1).$$

To show necessity, suppose that $cost(G^m, T^A) \leq 2^{8n} \cdot (8n+1)$. Since all edge-weights are nonzero, T^A is a tour that does not use an edge of weight $2^{8n} \cdot (8n+1)$. Therefore, the tour T^A only uses edges that correspond to some edge in the graph $(V(G'), E(G') \setminus \{\{v_3^1, v_5^1\}\})$. Hence, the graph $(V(G'), E(G') \setminus \{\{v_3^1, v_5^1\}\})$ has a Hamiltonian cycle and, as we have seen in the proof of Theorem 9.27, it follows that G has a Hamiltonian cycle. $\square$

We can show the same non-approximability results as in Theorem 7.7 for the modification function dec.

Theorem 7.8. *Unless* $\mathrm{P} = \mathrm{NP}$, (dec, MinTSP) *is not* $2^{|V|}$*-approximable.*

Proof. *(Sketch)* We use the same construction as in the proof of Theorem 7.7, but assign the following weights to the edges of G^o:

$$w(e) := \begin{cases} 1, & \text{if } e \in E(G') \setminus \{\{v_1^1, v_4^1\}, \{v_3^1, v_5^1\}\}, \\ 2^{8n} \cdot (8n+1), & \text{if } e = \{v_3^1, v_5^1\}, \\ 2^{8n} \cdot (8n+2), & \text{otherwise.} \end{cases}$$

Note that each Hamiltonian tour through G' has to use one of the edges $\{v_1^1, v_4^1\}$ or $\{v_3^1, v_5^1\}$. Consequently, C' is the sole optimum tour in G^o.

We modify G^o to G^m by decreasing the weight of the edge $\{v_1^1, v_4^1\}$ from $2^{8n} \cdot (8n+2)$ to 1. Thus, if $G \in$ HC then G^m has a tour of size $8n$, otherwise a tour in G^m has cost at least $2^{8n} \cdot (8n+1)$. Consequently,

$$G \in \text{HC} \Leftrightarrow cost(G^m, T^A) \leq 2^{8n} \cdot 8n.$$

□

7.5 Metric TSP (MINTSP$_\triangle$)

In this section we study the travelling salesperson problem when restricted to special instances. We pose the restriction that for every instance (G, w) the triangle inequality shall be satisfied, that is, for every three different vertices $u, v, z \in V(G)$ it holds that $w(\{u, v\}) + w(\{v, z\}) \geq w(\{u, z\})$. The problem of finding an optimum tour in such restricted TSP-instances is called the metric TSP (TSP$_\triangle$).

PROBLEM: MINTSP$_\triangle$

INSTANCE: A complete graph $G = K_n$ and a function $w : E(G) \rightarrow \mathbb{N}$ assigning a weight to each edge of G. Each triple of vertices from $V(G)$ shall satisfy the triangle inequality.

SOLUTION: A Hamiltonian cycle C in G.

MEASURE: The length of the cycle C, i.e., $\sum\limits_{e \in C} c(e)$.

A problem MAXTSP$_\triangle$ is defined in the same way.

The best known approximation results for MINTSP$_\triangle$ is a $\frac{3}{2}$-approximation due to Christofides [Chr76]. It is unlikely that MINTSP$_\triangle$ has a PTAS since it is APX-complete [PY93]. Regarding the modification adv it has been shown in [AEMP06] that

- MOD$_{adv}$MINTSP$_\triangle$ is approximable with ratio $\frac{4}{3}$,
- MOD$_{adv^k}$MINTSP$_\triangle$, for $k \in \mathbb{N}$, is approximable with ratio $\frac{3}{2} - \frac{1}{(4k+2)}$,

where the modification function adv^k is the addition of k vertices to the original graph (also see Definition 4.23). In this thesis, we consider as modification the change of single edge weights. Thereby, we are only interested in weight changes that do not lead to a violation of the triangle inequality. Otherwise, we had to change several other weights in the graph to reestablish the triangle inequality, which contradicts our approach of *minimal* modification. The modification functions that increases the weight of a single edge and respects the triangle inequality is given by

$$inc_{\Delta}(G,(e,i)) := \begin{cases} inc(G,(e,i)), & \text{if } G \text{ and } inc(G,(e,i)) \text{ satisfy} \\ & \text{ the triangle inequalilty,} \\ ((\emptyset,\emptyset),\emptyset), & \text{otherwise.} \end{cases}$$

A modification function dec_{Δ} is defined in the same way. In [BHMW08] the authors show that dec_{Δ}-MINTSP$_{\Delta}$ is $7/5$-approximable. In the same paper the authors established the following lemma.

Lemma 7.9 ([BHMW08]). *Let G^o be a complete, weighted graph, $e \in E(G)$, $i \in \mathbb{N}$, $c \in \{dec_{\Delta}, inc_{\Delta}\}$, and $G^m := c(G^o,(e,i))$. If G^m is not the empty graph $((\emptyset,\emptyset),\emptyset)$, i.e., G^o and G^m satisfy the triangle inequality, then every edge incident to e has cost at least $i/2$.*

Proof. Let $\{u,v\}$ be the modified edge. We just show the assertion for the modification dec_{Δ} and edges incident to u. Let c_e denote the weight of the edge e in G^o. Let $z \in V(G) \setminus \{u,v\}$. Applying the triangle inequality to the triangle induced by u,v, and z we get

$$\begin{array}{lllll} w(\{u,z\}) & + & w(\{z,v\}) & \geq & c_e & \text{(triangle inequality in } G^o\text{)}, \\ w(\{u,z\}) & + & (c_e - i) & \geq & w(\{z,v\}) & \text{(triangle inequality in } G^m\text{)}. \end{array}$$

Summarizing these inequalities we get $2 \cdot w(\{u,z\}) - i \geq 0$. □

Using Lemma 7.9, we improve on the factor $7/5$ given in [BHMW08] by showing

Theorem 7.10. MOD$_{dec_{\Delta}}$MINTSP$_{\Delta}$ *is $4/3$-approximable.*

Proof. Given an original graph G^o, a modified graph $G^m := dec_{\Delta}(G^o,(e,i))$, and an optimum tour T^o_{opt} for G^o, an algorithm A that approximates a tour for G^m with a factor $4/3$ works as follows. The case $G^m = ((\emptyset,\emptyset),\emptyset)$ is trivial. Otherwise, A computes a solution T^m_{Chr} for G^m using Christofides's algorithm [Chr76]. We now argue that the better of the both tours T^o_{opt} and T^m_{Chr} yields a $4/3$-approximation.

Without going into detail, we mention that in general the cost of T^m_{Chr} is bounded by the size of a minimum spanning tree of G^m (short, $MST(G^m)$) added to the size of a minimum perfect matching M between the vertices of odd degree in $MST(G_m)$. Also, the size of M is bounded by $\frac{1}{2}opt(G^m)$. For details see [Chr76]. By Lemma 7.9, an optimum tour T^m_{opt} in G^m uses at least one edge of weight $i/2$. Since T^m_{opt} without that $i/2$-weight edge is a spanning tree, we have that $MST(G^m) \leq opt(G^m) - \frac{i}{2}$. Thus, $cost(G^m, T^m_{Chr}) \leq \frac{3}{2}opt(G^m) - \frac{i}{2}$.

On the other hand, note that $opt(G^o) \leq opt(G^m) + i$, since optimum solutions in G^o and G^m differ by at most i. Also,

$$cost(G^m, T^o_{opt}) = \begin{cases} opt(G^o) - i, & \text{if } e \text{ is part of } T^o_{opt}, \\ opt(G^o), & \text{otherwise,} \end{cases}$$

and therefore $cost(G^m, T^o_{opt}) \leq opt(G^o)$. By combining the two inequalities we get that $cost(G^m, T^o_{opt}) \leq opt(G^m) + i$

In case $i \leq \frac{1}{3} opt(G^m)$ the tour T^o_{opt} yields a $4/3$-approximation in G^m. In case $i > \frac{1}{3} opt(G^m)$ the tour T^m_{Chr} is a $4/3$-approximation in G^m. □

This proof translates mutatis mutandis to inc_Δ.

Theorem 7.11. *The problem* inc_Δ-MinTSP$_\Delta$ *is* $4/3$*-approximable.*

Proof. Let $G^m := inc_\Delta(G^o, (e, i))$. Obviously, $opt(G^o) \leq opt(G^m)$. Also,

$$cost(G^m, T^o_{opt}) = \begin{cases} opt(G^o) + i, & \text{if } e \text{ is part of } T^o_{opt}, \\ opt(G^o), & \text{otherwise,} \end{cases}$$

and therefore $cost(G^m, T^o_{opt}) \leq opt(G^o) + i$. By combining the two inequalities we get that $cost(G^m, T^o_{opt}) \leq opt(G^m) + i$. The rest of the proof is a carbon copy of the proof of Theorem 7.10. □

We can generalize the above idea, to the case in which more than one edge is decreased or increased.

Theorem 7.12. MOD$_{dec^k_\Delta}$MinTSP$_\Delta$ *and* MOD$_{inc^k_\Delta}$MinTSP$_\Delta$ *are approximable with ratio* $\frac{3k+1}{2k+1}$*, for all* $k \in \mathbb{N}$.

Proof. (Sketch) As in the proof of Theorem 7.10 we output the better one of the old solution and the solution obtained by Christofides's algorithm. Let $i_1, \ldots, i_k$ be the numbers by which the edges are decreased/increased. The analysis relies on the facts that

$$cost(G^m, T^o_{opt}) \leq opt(G^m) + k \cdot \max_{1 \leq j \leq k} i_j \text{ , and}$$

$$cost(T^m_{Chr}) \leq \frac{3}{2} opt(G^m) - \max_{1 \leq j \leq k} \frac{i_j}{2}.$$

If $\max_{1 \leq j \leq k} i_j \leq opt(G^m)/(2k+1)$ then T^o is a $\frac{3k+1}{2k+1}$-approximation, otherwise T^m_{Chr} yields such a bound. □

As another generalization, it is shown in [BFH+07] that reoptimization also yields improved bounds for input instances that satisfy a relaxed form of triangle inequality, namely the β-triangle inequality $w\{u, z\} \leq \beta \cdot (w\{u, v\} + w\{v, z\})$ for $\beta \geq 1$.

Besides these positive results, we show the following lower bound for approximability.

Theorem 7.13. *There is no* FPTAS *for* $(dec_\Delta, \text{MinTSP}_\Delta)$ *and* $(inc_\Delta, \text{MinTSP}_\Delta)$*, unless* P = NP.

Proof. First, we show that $(dec_\Delta, \textsc{MinTSP}_\Delta)$ has no FPTAS. Assume to the contrary that there is an FPTAS for $(dec_\Delta, \textsc{MinTSP}_\Delta)$. Let p be a polynomial and A be an algorithm that, given a weighted graph G^o, an optimum tour in G^o, a modification m, and $\varepsilon > 0$, outputs a tour T^A for $G^m := dec_\Delta(G^o, m)$ with $cost(G^m, T^A) \leq (1+\varepsilon) \cdot opt(G^m)$ in time $p(|V(G)|, 1/\varepsilon)$. We show that under this assumption HC $\in$ P.

Let G be a graph. We assume the reader to be familiar with the proof of Theorem 9.27, and in particular with the construction of the graph G'. Using G' we construct a $\textsc{MinTSP}_\Delta$-instance $G^o = (K_{|V(G')|}, w)$ where

$$w(e) := \begin{cases} 2, & \text{if } e \in E(G') \setminus \{\{v_1^1, v_4^1\}\}, \\ 3, & \text{otherwise.} \end{cases}$$

Note that any complete graph with weights 2 and 3 satisfies the triangle inequality, and even so if the weight of a single edge is reduced to 1.

Let $n := |V(G)|$ and $n' := |V(G^o)| = 8n$. First, note that the tour

$$T^o_{opt} = (v_1^1, v_2^1, v_3^1, v_5^1, v_4^1, v_6^1, v_7^1, v_8^1, v_1^2, \ldots, v_8^{n-1}, v_1^n, v_2^n, v_3^n, v_5^n, v_4^n, v_6^n, v_7^n, v_8^n, v_1^1),$$

is the sole optimum tour in G^o and has cost $2n'$. Now we modify the graph G^o by decreasing the cost of the edge $\{v_1^1, v_4^1\}$ to 1. The resulting graph is the graph G^m. Note that T^o_{opt} still has cost $2n'$ in G^m, therefore $opt(G^m) \leq 2n'$. In addition, we claim that G^m has a tour with cost $2n' - 1$ if and only if G has a Hamiltonian cycle. For a proof of this fact, note that a tour with cost $2n' - 1$ has to use the edge $\{v_1^1, v_4^1\}$ and has to avoid all edges with cost 3. Consequently, it traverses the gadget H_1, and also all other gadgets H_i, via $(v_3^i, v_2^i, v_1^i, v_4^i, v_5^i, v_8^i, v_7^i, v_6^i)$, $1 \leq i \leq n$. This is possible if and only if G is Hamiltonian (see proof of Theorem 9.27).

Let $\varepsilon = 1/(3n')$. For every tour T with $cost(G^m, T) = opt(G^m) + 1$ we have

$$cost(G^m, T) = \left(1 + \frac{1}{opt(G^m)}\right) opt(G^m) \geq \left(1 + \frac{1}{2n'}\right) opt(G^m) > (1+\varepsilon) opt(G^m).$$

Consequently, the output T^A of the algorithm $A(G^o, T^o_{opt}, m, \varepsilon)$ is an optimum solution of G^m. Now, $cost(G^m, T^A) = 2n' - 1$ if and only if $G \in$ HC. The assertion follows from the fact that the running time of $A(G^o, T^o_{opt}, m, \varepsilon)$ is bounded by $p(1/\varepsilon, n') = p(24n, 8n)$.

The proof for $(inc_\Delta, \textsc{MinTSP})$ is essentially the same, except that the edge $\{v_1^1, v_8^n\}$ has weight 1 in G^o and has weight 3 in G^m. □

Conclusions

In this chapter, we presented the concept of *approximation* of slightly modified instances, also known as reoptimization. We showed that reoptimization can help to improve the approximability of hard optimization problems. In detail, we indicated that reoptimization leads for most[2] unweighted optimization problems to approximations with an absolute error of 1 or 2.

For (natural) weighted optimization problems we were unable to find such absolute error approximations. But, we gave an example of a reoptimization problem, namely $\mathrm{MOD}_{inc_\triangle}\textsc{MinTSP}_\triangle$, where modification leads to an improved approximation ratio of $4/3$.

Open problem 3. *Do* $\mathrm{MOD}_{inc_\triangle}\textsc{MinTSP}_\triangle$ *and* $\mathrm{MOD}_{dec_\triangle}\textsc{MinTSP}_\triangle$ *have a* δ*-approximation with* $\delta < 4/3$*?*

In Appendix B we even give a PTAS for an reoptimization problem that is APX-complete in its classical non-reoptimization variant. It is an open question if the same result holds for $\textsc{MinTSP}_\triangle$.

Open problem 4. *Is there a* PTAS *for* $inc_\triangle$-$\textsc{MinTSP}_\triangle$ *or* $dec_\triangle$-$\textsc{MinTSP}_\triangle$*?*

As a first step in this direction, we showed that there is no FPTAS for these two problems. Besides the above mentioned positive results, we also gave an example where reoptimization does not help at all, namely when altering an edge-weight of (nonmetric) TSP-instances.

Table 7.1 summarizes our main results for reoptimization. It also includes the results that are given in Appendix B.

[2] An example for an unweighted reoptimization problem that has *no* absolute error guarantee, unless $\mathrm{P} = \mathrm{NP}$, is the problem $\textsc{MinMaxIS}$, which is studied in Appendix B.

	best known approximation	reoptimization for inc/inc_{Δ}	reoptimization for dec/dec_{Δ}
MinVC	no 1.37-app. [DS05]	absolute error 1	absolute error 1
MinTSP	no $2^{\lvert V\rvert}$-approx.	no $2^{\lvert V\rvert}$-approx.	no $2^{\lvert V\rvert}$-approx.
MinTSP$_{\Delta}$	3/2-approx. [Chr76]	4/3-approx. no FPTAS	4/3-approx. no FPTAS
MaxTSP	4/3-approx. [Ser84]	5/4-approx. no FPTAS	? no FPTAS
MaxTSP$_{\Delta}$	8/7-approx. [CN07]	PTAS no FPTAS	PTAS no FPTAS
MinST	1.55-approx. [RZ05]	$\frac{4}{3}$-approx. [BBH$^+$08] no FPTAS	1.3-approx. [BBH$^+$08] no FPTAS
MinMaxIS	no $\lvert V\rvert^{1-\varepsilon}$-app. [Hal93]	no $\lvert V\rvert^{1-\varepsilon}$-approx.	no $\lvert V\rvert^{1-\varepsilon}$-approx.

Table 7.1: An overview on reoptimization results

Chapter 8

Polynomial Strings as Hint

In this section we are concerned with the question what happens when we loose our restriction on the hint being a certificate of the original instance. Instead, we allow any polynomially bounded string[1] as a hint. The applicability of this approach to practice stems from the scenario in which a computation produces intermediate results or additional information about how the solution was obtained. This information might be useful for further computations.

To formalize the idea of polynomially bounded strings as possible hint we introduce the notion of hint functions.

Definition 8.1. *A total function h is a hint function if and only if the size of the output of h is polynomially bounded in the length of its argument, that is, there exists a polynomial p such that for all $x \in \Sigma^*$ it holds that $|h(x)| \leq p(|x|)$. Let str denote the set of all hint functions.*

Using Definition 5.13 we can define classes $\mathcal{C}_{\text{MOD}}/str$, which characterize the complexity of deciding if a modified instance belongs to a problem A when the hint $h(x)$ is given, where h is a hint function. We may write $(c, A) \in \mathcal{C}_{\text{MOD}}/str$ instead of the more formal $(c, V_A) \in \mathcal{C}_{\text{MOD}}/str$, since no certificates are involved in the definition of $\mathcal{C}_{\text{MOD}}/str$.

Note that $cert(V_A) \subseteq str$, for any verifier V_A. Thus, the following observation is obvious by Definition 5.13.

Observation 8.2. $(c, V_A) \in \mathcal{C}_{\text{MOD}}/cert(V_A) \Rightarrow (c, L(V_A)) \in \mathcal{C}_{\text{MOD}}/str.$

In other words, each modification problem (c, V_A) that is easy with selected hints (which includes the no-solution promise) is also easy when using hint functions. This is not surprising, as the hint function may also output selected certificates or the empty word.

But not only easiness results for $\mathcal{C}_{\text{MOD}}/cert(V_A)$ translate to $\mathcal{C}_{\text{MOD}}/str$; we can also reuse our results about improbable easiness of certain modification problems.

[1]We bound the length of the string in order to avoid algorithms that use too much space.

Theorem 8.3. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, (c, A) be a modification problem, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$, . If there exist functions $f_1, f_3 \in \mathrm{FP}$ such that for all $k \in \mathbb{N}$ and all $F, L_1, ..., L_k \in \Sigma^*$ it holds that*

- $F \in \mathrm{SAT} \Leftrightarrow f_1(F) \in A$ *and*
- $ad^q(F, (L_1, ..., L_k)) \in \mathrm{SAT} \Leftrightarrow c(f_1(F), f_3((L_1, ..., L_k))) \in A$

then the following statement holds:

$$(c, A) \in \mathcal{C}_{\mathrm{MOD}}/str \Rightarrow \mathrm{NP} \subseteq \mathcal{C}/poly.$$

Proof. Let $(c, A) \in \mathcal{C}_{\mathrm{MOD}}/str$. Thus $(c, A) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/str$. Now, the proof translates mutatis mutandis from the proof of Theorem 4.24. The main difference is that every occurrence of $cert(V_A)$ is replaced by str. $\square$

Choosing $f_1(F) := F$ and $f_3((L_1, ..., L_k)) := (L_1, ..., L_k)$ in Theorem 8.3 we get

Corollary 8.4. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. If $(ad^q, V_{\mathrm{SAT}}) \in \mathcal{C}_{\mathrm{MOD}}/str$ then $\mathrm{NP} \subseteq \mathcal{C}/poly$.*

We conclude from the last corollary that if $(ad^q, V_{\mathrm{SAT}}) \in \mathrm{P}_{\mathrm{MOD}}/str$ then the polynomial hierarchy collapses to its second level.

8.1 The Table-Lookup Method

The power of polynomial hints allows us to show that some modification problems (c, A) for which we were not able to prove containment in $\mathrm{P}_{\mathrm{MOD}}/cert(V_A)$ are a member of $\mathrm{P}_{\mathrm{MOD}}/str$. For example, while it is not known if $(ad, V_{\mathrm{SAT}}) \in \mathrm{P}_{\mathrm{MOD}}/cert(V_{\mathrm{SAT}})$ (see Table 5.3), the modification problem (ad^k, V_{SAT}), k being a constant, can be shown to be a member of $\mathrm{P}_{\mathrm{MOD}}/str$.

Observation 8.5. $(ad^k, V_{\mathrm{SAT}}) \in \mathrm{P}_{\mathrm{MOD}}/(str \cap \mathrm{F}\Delta_2^{\mathrm{p}})$.

The following proof of this fact stands exemplarily for the proof of the upcoming Observation 8.6.

Proof. Let F be a formula and let $n = |Var(F)|$. First, we show that the number of ad^k-modified formulas of a formula F is bounded by a polynomial $q(|F|)$. Just for the proof, suppose that the unit clauses are added sequentially, instead of all at once. Thus, even before the last addition of a unit clause there exist at most $n + (k - 1)$ different variables in the hitherto existing modified formula F'. For each $x \in Var(F')$ one of the two unit clauses $\{x\}$ or $\{\neg x\}$ may be added to F'. Consequently, only one of at most $2(n + (k - 1))$ unit clauses of $Var(F')$ and one of two unit clauses over some new variable may be added at any given time. Thus, for each individual addition of a single unit clause the number of possible modified instances is bounded by the polynomial $2n + 2k$. The total number of modified instances after k applications of ad is therefore bounded by $q(n) = (2n + 2k)^k$, which yields a polynomial upper bound in the size of $|F|$, for every constant k.

We choose as hint string a table that contains all of the at most $q(|F|)$ possible modified formulas together with a satisfying assignment for each formula, if existent. Note that each modified formula is bounded in its size by a polynomial $r(|F|)$. Thus, the size of the table is polynomial in $|F|$.

Now, the decision process for $ad^q(F, m) \in \text{SAT}$ is just a matter of table lookup. This table can be computed from F by a function $f \in \text{F}\Delta_2^{\text{p}}$ that (i) computes all possible modified instances and (ii) uses its oracle to compute bitwise a satisfying assignments for each modified instance. □

Observation 8.6 ([Lib04]). *Let $A \in \text{NP}$ and c be a modification function. If there exist polynomials q and r such that for all $x \in \Sigma^*$ it holds that*

$$\left| \bigcup_{m \in \Sigma^*} \{c(x, m)\} \right| \le q(|x|)$$

and for all $m \in \Sigma^*$ it holds that $|c(x, m)| \le r(|x|)$ then $(c, A) \in \text{P}_{\text{MOD}}/str$.

Note that the hint function that shows $(c, A) \in \text{P}_{\text{MOD}}/str$ in Observation 8.6 does not need to be necessarily a member of $\text{F}\Delta_2^{\text{p}}$. The function probably cannot compute the at most $q(n)$ modified instances in polynomial time.

Observation 8.6 only gives a sufficient condition for when a problem (c, V_A) belongs to $\text{P}_{\text{MOD}}/str$. There also exist modification problems (c, V_A) for which exponentially many modified instances can be decided in polynomial time with a polynomial string as hint. We give the following (rather artificial) example of such a problem. Consider the following padded version of SAT:

$$\text{SAT}' := \{(F, \omega) : F \in \text{SAT} \land \omega = \{0, 1\}^{|F|}\}.$$

Obviously, SAT$'$ is NP-complete. For the modification function

$$c'((F, \omega), \omega') := \begin{cases} (F, \omega'), & \text{if } |\omega'| = |\omega|, \\ (F, \omega), & \text{otherwise,} \end{cases}$$

and a given formula F there exist $2^{|F|}$ different modified SAT$'$-instances. Nevertheless each of those instances can be decided with the 1-bit-hint '1' if F is satisfiable and '0' if F is not satisfiable. Thus, $(c', \text{SAT}') \in \text{P}_{\text{MOD}}/str$. Another problem of this kind, namely 'CycleInHamiltonianReduction' (CHR), which uses the padding idea in a more practical fashion, is given in [CDLS02].

From Observation 8.6 we also conclude that if the *size* of the modification is small enough then the respective modification problem is easy when hint strings are allowed. In detail, if the size of the modification is bounded by $O(log(|x|))$, where x is the instance that is going to be modified, then there exist at most $|\Sigma|^{O(log(|x|))}$ modified instances — one for each possible modification. Also, the size of each modified instance must be polynomial in $|x|$ since $c \in \text{FP}$. A table that contains all the modified instances can be computed in polynomial time. For each modified instance in the table, a solution, e.g., the minimum solution, can be found by a function from $\text{F}\Delta_2^{\text{p}}$.

Corollary 8.7. *Let $A \in \mathrm{NP}$, c be a modification function, and let the second argument m of c be bounded by $O(log(|x|))$, where x is the first argument of c. Then $(c, A) \in \mathrm{P}_{\mathrm{MOD}}/(str \cap \mathrm{F}\Delta_2^{\mathrm{p}})$.*

On an sidenote, we mention that the modification function $ad^{log(|x|)}$ is no such modification function as desired in the last corollary, since the size of the modification is $O\left(log\left(|x|\right) \cdot log\left(|Var(F)|\right)\right)$. On the other hand, the modification function ad^k, where k is a constant, is clearly a modification function of the desired form since $|Var(F)| \leq |x|$.

At first glance, it seems that hint-strings are more powerful than selected certificates, since they are more flexible when choosing a hint. But, a closer look reveals that Observation 8.6 can be rephrased as a results for selected certificates when an appropriate verifier is chosen.

Theorem 8.8. *Let $A \in \mathrm{NP}$ and c be a modification function If there exist polynomials q and r such that for all $x \in \Sigma^*$ it holds that*

$$\left| \bigcup_{m \in \Sigma^*} \{c(x, m)\} \right| \leq q(|x|)$$

and for all $m \in \Sigma^*$ it holds that $|c(x, m)| \leq r(|x|)$ then there exists a verifier V_A' for A such that $(c, V_A') \in \mathrm{P}_{\mathrm{MOD}}/cert(V_A')$.

Proof. Let $A \in \mathrm{NP}$, V_A be an arbitrary verifier for A, and c be a modification function. Let q and r be as above. Let p be some polynomial such that $(x, \pi) \in V_A \Rightarrow |\pi| \leq p(|x|)$, that is, p bounds the size of certificates w.r.t. V_A. Consider the verifier V_A' defined by

$$\begin{aligned}(x, \omega) \in V_A' \Leftrightarrow \omega = ((x, \pi), (x_1, \pi_1), ..., (x_\ell, \pi_\ell)) \ \wedge \ (x, \pi) \in V_A \wedge \\ \ell \leq q(|x|) \ \wedge \ (\forall 1 \leq i \leq \ell)[|x_i| \leq r(|x|) \wedge |\pi| \leq p(|x_i|)].\end{aligned}$$

In other words, the certificate ω for x not only contains information whether $x \in A$, or equivalently, if there exists π with $(x, \pi) \in V_A$, but it also contains pairs of other strings, which we interpret as a modified instance x_i together with a corresponding certificate π_i for x_i, if existent. Note that V_A' is indeed a verifier for A since (i) $|\omega|$ is polynomial in the size of x, (ii) there exists an ω with $(x, \omega) \in V_A'$ if and only if $x \in A$, and (iii) $(x, \omega) \in V_A'$ can be decided in polynomial time.

We choose as hint function for (c, V_A') the function h that, given an instance x, outputs a string $((x, \pi), (x_1, \pi_1), ..., (x_\ell, \pi_\ell))$, where

- π is a certificate for x w.r.t. V_A, if existent,
- $\{x_1, ..., x_\ell\}$ is the set of all possible modified instances, of which there are at most $q(|x|)$, and
- for all $i \leq \ell$, the string π_i is a certificate for x_i w.r.t. V_A, if existent.

To decide if a modified instance $c(x,m)$ belongs to A, we just need to find the corresponding modified instance x_i in the table given by the hint. All we then need to do is to evaluate whether $(x_i, \pi_i) \in V_A$, which is possible in polynomial time. $\square$

This last theorem illustrates that the results in Chapter 4 are highly dependent on the chosen verifier. For instance, it shows that there exists a verifier V'_{SAT} for SAT such that $(ad, V'_{\mathrm{SAT}}) \in \mathrm{P}^{\in}_{\mathrm{MOD}}/cert(V_{\mathrm{SAT}})$.[2]

8.2 Connection to Preprocessing

In this section we outline a connection between the problem of modification with hint strings and a related problem, namely the problem of preprocessing. First, we give an informal summary on preprocessing.

Subject to preprocessing are problems of pairs (x,y) for which the first part of the input x is known before the rest y, and the known part x remains the same for several subsequent inputs y. In this case, it could be beneficial to preprocess from the known part x an additional information $h(x)$ that is helpful for deciding future inputs. The extra cost for preprocessing is justified when the complexity of deciding coming instances is thereby significantly decreased. We refer to the preprocessing of the known part as *compilation*. For more details see [CDLS02, Lib01, Lib98a, CD97, SK96].

8.2.1 Uniform Preprocessing

The following subsection consists of three parts. First, we give a brief summary on preprocessing. Following [CDLS02], we define classes of uniform compilability and show hardness results for these classes with the help of an appropriate notion of reduction. Second, we show how preprocessing results relate to modification problems. Last, we demonstrate our results at an example, albeit a fairly artificial example.

We start by formally defining classes of uniform compilability. These classes capture our intuitive notion of compilability. In the next subsection we also consider *nonuniform* compilability classes, but for the moment we postpone a discussion on why these nonuniform classes are necessary.

Definition 8.9 ([CDLS02], Definition 2.4). *A language $A \subseteq \Sigma^* \times \Sigma^*$ of pairs belongs to the class $\rightsquigarrow\mathcal{C}$ (in words, compilable to $\mathcal{C}$) if and only if*

$$(\exists C \in \mathcal{C})(\exists h \in str)(\forall x, y \in \Sigma^*)[(x,y) \in A \Leftrightarrow (h(x), y) \in C].$$

Informally, a language of pairs (x,y), where x is the known part and y is given online, belongs to $\mathcal{C}$ if there exists a $\mathcal{C}$-predicate C such that $(x,y) \in A$ can also be decided

[2] Since a table of all ad-modified formulas is computable in polynomial time, we even have $(ad, V'_{\mathrm{SAT}}) \in \mathrm{P}^{\in}_{\mathrm{MOD}}/(cert(V_{\mathrm{SAT}}) \cap \mathrm{F}\Delta^{\mathrm{p}}_2)$. We can contrast this result with the fact that if $(ad, V'_{\mathrm{SAT}}) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V_{\mathrm{SAT}}) \cap fun \cdot \mathrm{NP})$ then $\mathrm{NP} = \mathrm{coNP}$ (see Theorem 4.31).

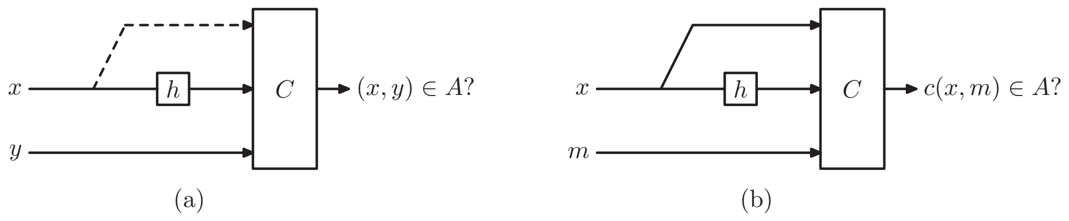

Figure 8.1: Schematic depiction of the classes (a) $\rightsquigarrow\mathcal{C}$ and (b) $\mathcal{C}_{\text{MOD}}/str$.

by asking whether the preprocessed string $h(x)$ together with the missing part y of the input belongs to C. The class $\rightsquigarrow\mathcal{C}$ and the class $\mathcal{C}_{\text{MOD}}/str$ are quite similar according to their definitions. Both aim to find a string that only depends on either the original instances or the known part of the input to decide instances that are later to be specified, either by an unknown modification or an online given part y. A comparison of these classes is depicted in Figure 8.1. The dotted line in Figure 8.1 indicates that the fixed part of the input may w.l.o.g. be assumed to be known to C, since h can give x as part of its output.

It has already been shown in [CDLS02] that classical complexity classes and compilability classes share the same inclusion structure.

Theorem 8.10 ([CDLS02], Theorem 2.11). *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes that are closed under $\leq_m^p$-reduction and that have $\leq_m^p$-complete problems. Then*

$$\rightsquigarrow\mathcal{C} \subseteq \rightsquigarrow\mathcal{D} \;\Leftrightarrow\; \mathcal{C} \subseteq \mathcal{D}.$$

We now aim to establish a notion of non-compilability, which then leads to hardness results for modification problems. In [CDLS02] the authors propose the following reduction to show such hardness results.

Definition 8.11 ([CDLS02], Definition 2.5). *A $\leq_{comp}$-reduction between two languages of pairs A and B is a triple (f_1, f_2, g), where $f_1, f_2 \in str$ and $g \in \text{FP}$, such that for all pairs of strings (x, y) it holds that*

$$(x,y) \in A \Leftrightarrow \big(f_1(x), g(f_2(x), y)\big) \in B.$$

For a discussion on the benefits of this last definition we refer to [CDLS02]. In [CDLS02] it is also demonstrated how to find complete $\leq_{comp}$-problems for a class $\rightsquigarrow\mathcal{C}$, provided that a complete problem for $\mathcal{C}$ exists. In detail, given the $\leq_m^p$-complete problem A for $\mathcal{C}$, the problem $\epsilon A := \{(\epsilon, y) : y \in A\}$ is $\leq_{comp}$-complete for $\rightsquigarrow\mathcal{C}$. Intuitively, the problem ϵA cannot benefit from preprocessing, since the fixed part x is empty, and therefore there is no possibility to take advantage of preprocessing.

Theorem 8.12 ([CDLS02], Theorem 2.3). *Let $\mathcal{C}$ be a complexity class and let A be $\leq_m^p$-complete in $\mathcal{C}$. Then ϵA is $\rightsquigarrow\mathcal{C}$-complete w.r.t. $\leq_{comp}$-reduction.*

To show $\leq_{comp}$-hardness of a problem B for the class $\rightsquigarrow\mathcal{C}$ it suffices to show that $\epsilon A \leq_{comp} B$ for some problem A that is $\leq_m^p$-complete in $\mathcal{C}$.

We now turn to our discussion on how compilability and the problem of modified instances are related. We already mentioned, that compilability classes and modification problems are similar regarding to their definitions. The similarity between the two complexity classes $\mathcal{C}_{\text{MOD}}/str$ and $\rightsquigarrow\mathcal{C}$ is formally expressed by the following theorem.

Theorem 8.13. *Let (c, A) be a modification problems and $\mathcal{C}$ be a complexity class that is closed under $\leq_m^p$-reduction. Then*

$$\{(x, m) : c(x, m) \in A\} \in \rightsquigarrow\mathcal{C} \Leftrightarrow (c, A) \in \mathcal{C}_{\text{MOD}}/str.$$

Proof. Let $S := \{(x, m) : c(x, m) \in A\}$. For the '$\Rightarrow$'-direction assume that $S \in \rightsquigarrow\mathcal{C}$. Consequently, there exist $B \in C$ and $h \in str$ such that

$$(\forall x, m \in \Sigma^*)\ [(x, m) \in S \Leftrightarrow (h(x), m) \in B].$$

We define a new set $B' := \{(x, \omega, m) : (\omega, m) \in B\ \wedge x \in \Sigma^*\}$ and get that

$$(\forall x, m \in \Sigma^*)\ [c(x, m) \in A \Leftrightarrow (x, h(x), m) \in B'].$$

Since $B' \in \mathcal{C}$ we have that $(c, A) \in \mathcal{C}_{\text{MOD}}/str$.

For the other direction, assume that $(c, A) \in \mathcal{C}_{\text{MOD}}/str$. Therefore, there exist $B \in \mathcal{C}$ and $h \in str$ such that

$$(\forall x, m \in \Sigma^*)\ [c(x, m) \in A \Leftrightarrow (x, h(x), m) \in B].$$

We define $h'(x) := (x, h(x))$ and $B' := \{((x, \omega), m) : (x, \omega, m) \in B\}$ and get that

$$(\forall x, m \in \Sigma^*)\ [(x, m) \in S \Leftrightarrow (h'(x), m) \in B'].$$

Since $B' \in \mathcal{C}$ and $h' \in str$ we have that $(c, A) \in \mathcal{C}_{\text{MOD}}/str$. $\square$

We use Theorem 8.13 to establish a connection between uselessness of hint strings for a modification problem (c, A) and $\rightsquigarrow\mathcal{C}$-completeness of the corresponding language of pairs $\{(x, m) : c(x, m) \in A\}$.

Corollary 8.14. *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes that are closed under $\leq_m^p$-reduction and (c, A) be a modification problem for which $B := \{(x, m) : c(x, m) \in A\}$ is $\rightsquigarrow\mathcal{C}$-complete w.r.t. $\leq_{comp}$-reduction. If $(c, A) \in \mathcal{D}_{\text{MOD}}/str$ then $\mathcal{C} \subseteq \mathcal{D}$.*

Proof. Let $\mathcal{C}$, $\mathcal{D}$, (c, A), and B be as above. Let $(c, A) \in \mathcal{D}_{\text{MOD}}/str$. By Theorem 8.13 we get $B \in \rightsquigarrow\mathcal{D}$. It has been shown in [CDLS02] (Theorem 2.2 in [CDLS02]) that if $\mathcal{D}$ is closed under $\leq_m^p$-reduction then $\rightsquigarrow\mathcal{D}$ is closed under $\leq_{comp}$-reduction. Since B is $\rightsquigarrow\mathcal{C}$-complete we conclude that $\rightsquigarrow\mathcal{C} \subseteq \rightsquigarrow\mathcal{D}$. By Theorem 8.10 we get that $\mathcal{C} \subseteq \mathcal{D}$. $\square$

We use this last corollary to argue for useless of hint strings in the following way. For example, assume that (c, A) is a modification problem for an NP-complete language A. Also, assume that we showed $\rightsquigarrow\mathrm{NP}$-completeness of the corresponding language of pairs $\{(x, m) : c(x, m) \in A\}$. If there are useful hints for (c, A), that is, if $(c, A) \in \mathcal{C}_{\mathrm{MOD}}/str$ for some complexity class $\mathcal{C} \subset \mathrm{NP}$, then $\mathrm{NP} \subseteq \mathcal{C}$ by Corollary 8.14. Consequently, we have that the NP-complete problem A is also in $\mathcal{C}$, without using any hints.

Finally, we illustrate our arguments with a rather artificial example. We give more practical examples in the next subsection.

Example: Consider the following problem $\overline{\mathrm{EQ}}$:

$$\begin{aligned}\overline{\mathrm{EQ}} := \{(F, G) :\ & F \text{ and } G \text{ are Boolean formulas, } Var(F) \subseteq Var(G)\text{, and}\\ & \text{there exist an assignment } \beta \text{ over } Var(G) \text{ such that}\\ & \beta(G) \neq \beta|_{Var(F)}(F)\ \}.\end{aligned}$$

Here, $\beta(G)$ denotes the truth value of G under the assignment β and $\beta|_{Var(F)}(F)$ denotes the truth value of F when the assignment β is restricted to the variables of $Var(F)$. Informally spoken, the problem $\overline{\mathrm{EQ}}$ consists of pairs of formulas that are nonequivalent, but where the notion of equivalence is adapted to hold for formulas with different variables. Note that $\overline{\mathrm{EQ}}$ is NP-complete.

We modify $\overline{\mathrm{EQ}}$-instances with the following modification sub_2:

$$sub_2((F, G), G') := (F, G'),$$

that is, the second formula G is substituted with a new formula G'. We aim to apply Corollary 8.14 in order to show that polynomial strings are useless when deciding sub_2-modified $\overline{\mathrm{EQ}}$-instance. Therefore, we need the following result.

Lemma 8.15. *$B := \{((F, G), G') : sub_2((F, G), G') \in \overline{\mathrm{EQ}}\}$ is $\rightsquigarrow\mathrm{NP}$-complete w.r.t. $\leq_{comp}$-reduction.*

Proof. Containment of B in $\rightsquigarrow\mathrm{NP}$ is clear from the facts that $\overline{\mathrm{EQ}} \in \mathrm{NP}$ and $sub_2 \in \mathrm{FP}$. To prove $\leq_{comp}$ -hardness of B we show that $\epsilon\mathrm{SAT} \leq_{comp} B$ via the following reduction functions f_1, f_2, and g:

- $f_1(x) := (x_1 \vee \neg x_1, x_1)$,[3]
- $f_2(x) := x$, and
- $g(x, F) := \begin{cases} \neg F, & \text{if } x = \epsilon, \\ x_1 \vee \neg x_1, & \text{otherwise.} \end{cases}$

[3]We assume that formulas are coded in way such that the variable x_1 is contained in any formula F.

It suffices to show that

$$(x, F) \in \epsilon\text{SAT} \Leftrightarrow (f_1(x), g(f_2(x), F)) \in B. \tag{1}$$

We consider two cases, $x = \epsilon$ and $x \neq \epsilon$. If $x \neq \epsilon$ then $(x, F) \notin \epsilon\text{SAT}$. Also,

$$(f_1(x), g(f_2(x), F)) = ((x_1 \vee \neg x_1, x_1), x_1 \vee \neg x_1)$$

is no element of B, since $sub_2((x_1 \vee \neg x_1, x_1), x_1 \vee \neg x_1)$ yields two equivalent formulas.
If $x = \epsilon$ then

$$\begin{aligned}(x, F) \in \epsilon\text{SAT} &\Leftrightarrow F \in \text{SAT}\\ &\Leftrightarrow (\exists \beta \in \Sigma^*)[(\neg F, \beta) \notin V_{\text{SAT}}]\\ &\Leftrightarrow \{\beta : \beta \text{ is an assignment over } Var(F)\} \neq V_{\text{SAT}}(\neg F)\\ &\Leftrightarrow (x_1 \vee \neg x_1, \neg F) \in \overline{\text{EQ}}\\ &\Leftrightarrow sub_2((x_1 \vee \neg x_1, x_1), \neg F) \in \overline{\text{EQ}}\\ &\Leftrightarrow (f_1(x), g(f_2(x), F)) \in B.\end{aligned}$$

□

By Corollary 8.14 we conclude that $(sub_2, \overline{\text{EQ}})$ has no useful polynomial hints.

8.2.2 Nonuniform Preprocessing

The results on uniform compilability that are given in the last section suffer from a severe technical problem. The notions of $\rightsquigarrow\mathcal{C}$ and $\leq_{comp}$-reduction are useful to state *strict and unconditional* results of compilability or non-compilability; but these notions are insufficient to express non-compilability results for problems that are *unlikely* to be compilable. For example, it is shown in [CDLS02] that if the problem of constrained satisfiability, which is defined by

$$\begin{aligned}\text{C-SAT} := \{(F, \beta) : & \ F \text{ is a CNF-formula and } \beta \text{ is a partial assignment}\\ & \text{that can be extended to a satisfying assignment of } F\},\end{aligned}$$

is solvable in deterministic polynomial time with the help of preprocessing then $\text{NP}/poly \subseteq \text{P}/poly$. However, we are unable to formally derive this fact using uniform compilability classes and $\leq_{comp}$-reduction. To overcome this shortcoming, new classes of *nonuniform* compilability are introduced in [CDLS02].

Definition 8.16 ([CDLS02], Definition 2.7). *A language $A \subseteq \Sigma^* \times \Sigma^*$ of pairs belongs to the class $\Vert\!\rightsquigarrow C$ (in words, non-uniformly compilable to $\mathcal{C}$) if and only if*

$$(\exists B \in \mathcal{C})(\exists h \in str)(\forall x, y \in \Sigma^*)[(x, y) \in A \Leftrightarrow (h(x, |y|), y) \in B].$$

Here, preprocessing not only depends on the known fixed part x of the input, but also on the *size* of the unknown part y. We assume that $|y|$ is given in unary notion. This definition corresponds to scenarios where the size of the missing part of the input, or equivalently, a polynomial upper bound on this size, is known in advance. The problem C-SAT is an example of such a problem.

Again, we establish a notion of hardness for classes of the form $\Vert\!\!\leadsto\mathcal{C}$. But, $\leq_{comp}$-reduction are not useful for this purpose (for details see [CDLS02]). Therefore, we introduce a more suitable notion of reduction.

Definition 8.17 ([CDLS02], Definition 2.8). *A $\Vert\!\!\leadsto$reduction between two languages of pairs A and B is a triple (f_1, f_2, g), where $f_1, f_2 \in str$ and $g \in \mathrm{FP}$, such that for all pairs (x, y) it holds that*

$$(x, y) \in A \;\Leftrightarrow\; \big(f_1(x, |y|), g(f_2(x, |y|, y))\big) \in B.$$

This definition extends in a natural way the definition of $\leq_{comp}$ such that the size of the unknown part y is incorporated. We just mention that for a complexity class $\mathcal{C}$ that is closed under $\leq_m^p$-reduction the complexity class $\Vert\!\!\leadsto\mathcal{C}$ is closed under $\Vert\!\!\leadsto-$reduction (see [CDLS02], Theorem 2.8). Again, we can show that complete problems for a class $\mathcal{C}$ yield complete problems for the class $\Vert\!\!\leadsto\mathcal{C}$.

Theorem 8.18 ([CDLS02], Theorem 2.9). *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction and A be a $\mathcal{C}$-complete problem w.r.t. $\leq_m^p$-reduction. Then ϵA is $\Vert\!\!\leadsto\mathcal{C}$-complete w.r.t. $\Vert\!\!\leadsto$reduction.*

It has already been shown in [CDLS02] (see Propositions 3.1 and 3.2 there) that the problems C-SAT and the problem ConstrainedVertexCover,

$$\text{C-VC} := \{(G, k), V') : G \text{ has a vertex cover } C \text{ of size at least } k \text{ with } V' \subseteq C\}$$

are $\leadsto\mathrm{NP}$-complete.

Nonuniform compilability classes and classical nonuniform complexity classes are related in the following way.

Theorem 8.19 ([CDLS02], Theorem 2.12). *Let $\mathcal{C}$ and $\mathcal{D}$ be complexity classes that are closed under $\leq_m^p$-reduction and that have $\leq_m^p$-complete problems. Then*

$$\Vert\!\!\leadsto\mathcal{C} \subseteq \Vert\!\!\leadsto\mathcal{D} \;\Leftrightarrow\; \mathcal{C}/poly \subseteq \mathcal{D}/poly.$$

Using this framework, we now exemplarily show how hardness results for nonuniform compilability classes can be used to show that certain modification problems have probably no useful polynomial hints.

Example 1: As a first example, we examine the already discussed modification problem (ad^{id}, SAT). Assume that $(ad^{id}, \mathrm{SAT}) \in \mathcal{C}_{\mathrm{MOD}}/str$, where $\mathcal{C}$ is closed under $\leq_m^p$-reduction. We have already argued at the beginning of this chapter, that this assumption implies that $\mathrm{NP} \subseteq \mathcal{C}/poly$ (Theorem 8.3). Using compilability classes, we now show, under the same assumption, the slightly weaker collapse $\mathrm{NP}/poly \subseteq \mathcal{C}/poly$.

First, we show that if $(ad^{id}, \mathrm{SAT}) \in \mathcal{C}_{\mathrm{MOD}}/str$ then the problem C-SAT would become an element of $\|\!\!\leadsto\mathcal{C}$. This is easy to see, as adding unit clauses over certain variables to a formula F corresponds to fixing the values of satisfying assignments for these variables. Thus, we reduced the question whether a fixed formula F has a satisfying assignment that extends a given partial assignment to the question whether the formula F in thats the respective unit clauses are added is satisfiable. This shows that C-SAT $\in \|\!\!\leadsto C$, a formal proof is omitted. Since C-SAT is $\|\!\!\leadsto\mathrm{NP}$-complete and $\|\!\!\leadsto\mathcal{C}$ is closed with respect to $\|\!\!\leadsto$reduction we conclude that $\|\!\!\leadsto\mathrm{NP} \subseteq \|\!\!\leadsto\mathcal{C}$. The assertion $\mathrm{NP}/poly \subseteq \mathcal{C}/poly$ follows by Theorem 8.19. Consequently, if $(ad^{id}, \mathrm{SAT}) \in \mathrm{P}_{\mathrm{MOD}}/str$ then $\mathrm{PH} = S_2$ (see Corollary 2.13) and if $(ad^{id}, \mathrm{SAT}) \in \mathrm{coNP}_{\mathrm{MOD}}/str$ then $\mathrm{PH} = \Sigma_3^p$ (see Theorem 2.14). These are consequences that are considered unlikely by most theoretical computer scientists.

Example 2: A similar results as in Example 1 holds for the modification problem (app, VC), where app is the modification function that takes as input a VC-instance (G, k) as well as a set $\{v_1, ..., v_\ell\}$ of vertices of G and outputs an instance (G', k) where at each vertex v_i a new vertex u_i is appended. Formally, for a VC-instance (G, k) and a subset $V' := \{v_1, ..., v_\ell\}$ of $V(G)$ we define

$$app((G,k),V') := \Big(\big(V(G) \cup \{u_1, ..., u_\ell\}, E(G) \cup \{\{v_i, u_i\} : 1 \leq i \leq \ell\}\big), k\Big).$$

Containment of the modification problem (app, VC) in $\mathcal{C}_{\mathrm{MOD}}/str$, where $\mathcal{C}$ is closed under $\leq_m^p$-reduction, would lead to C-VC $\in \|\!\!\leadsto\mathcal{C}$. To see this, let (app, VC) in $\mathcal{C}_{\mathrm{MOD}}/str$. Note that in a modified instance $app((G, k), \{v_1, ..., v_\ell\})$ with new vertices $u_1, ..., u_\ell$ at least one of v_i or u_i has to be contained in a minimum vertex cover, for each $1 \leq i \leq \ell$. But choosing v_i is always better than choosing u_i. Thus, we may assume that a vertex cover for $app((G, k), V')$ does not contain $u_1, ..., u_\ell$, but contains all vertices from V'. Therefore, the question whether $app((G, k), V')$ is an element of VC is reduced to the question whether $((G, k), V') \in$ C-VC. This shows that C-VC $\in \|\!\!\leadsto\mathcal{C}$.

Since C-VC is $\|\!\!\leadsto\mathrm{NP}$-complete we get $\mathrm{NP}/poly \subseteq \mathcal{C}/poly$, with the already in Example 1 mentioned implications for special choices of $\mathcal{C}$.

Example 3: As a last example, we examine a problem that is already discussed in [Lib04]. There, the problem of satisfiability is considered, together with the modification that allows for an arbitrary change to the original formula as long as the set of used variables remains unaltered. Assume that this modification problem is a member of $\mathcal{C}_{\mathrm{MOD}}/str$, $\mathcal{C}$ closed under $\leq_m^p$-reduction. The analysis relies on the $\|\!\!\leadsto\mathrm{NP}$ completeness of the set

$$n\mathrm{SAT} := \{(i, F) : F \in \mathrm{SAT} \wedge i = 1^{|Var(F)|}\},$$

which is just mentioned in [Lib04] without a proof.

Lemma 8.20. *$n\mathrm{SAT}$ is $\|\!\!\leadsto\mathrm{NP}$ complete w.r.t. $\|\!\!\leadsto$reduction.*

Proof. We show that ϵSAT is $\Vert\!\!\leadsto$-reducible to nSAT. Therefore we give three functions $f_1, f_2 \in str$ and $g \in \mathrm{FP}$ such that

$$(x, F) \in \epsilon\mathrm{SAT} \Leftrightarrow (f_1(x, |F|), g(f_2(x, |F|), F)) \in n\mathrm{SAT}.$$

Let F be a CNF-formula with $Var(F) = \{x_1, ..., x_n\}$. We assume that formulas are coded in a reasonable way such that $n \leq |F|$. We define $f_1(x, |F|) := 1^{|F|}$, $f_2(x, |F|) := x$, and

$$g(x, F) := \begin{cases} pad(F), & \text{if } x = \epsilon, \\ x_1 \wedge \neg x_1, & \text{otherwise,} \end{cases}$$

where $pad(F)$ is generated from F by addition of unit clauses $(x_{n+1}), ..., (x_{|F|})$, resulting in a CNF-formula with $|F|$ variables. We distinguish two cases. If $x \neq \epsilon$ then $(x, F) \notin \epsilon$SAT and the desired equivalence

$$(x, F) \in \epsilon\mathrm{SAT} \Leftrightarrow (1^{|F|}, x_1 \wedge \neg x_1) \in n\mathrm{SAT}$$

is valid. If $x = \epsilon$, the following equivalences yield the desired results

$$\begin{aligned}(\epsilon, F) \in \epsilon\mathrm{SAT} &\Leftrightarrow F \in \mathrm{SAT} \\ &\Leftrightarrow pad(F) \in \mathrm{SAT} \\ &\Leftrightarrow (1^{|F|}, pad(F)) \in n\mathrm{SAT}.\end{aligned}$$

□

We now return to our proof of hardness for the modification that leaves the set of variables unchanged. A helpful polynomial hint function h that renders this problem a member of $\mathcal{C}_{\mathrm{MOD}}/str$ also yields a helpful function h' for preprocessing. The function h' is given by $h'(1^n) := h(F_n)$, for an arbitrary but fixed formula F_n over n variables. Now, satisfiability of an online given formula F' over n variables can be decided by modifying the formula F_n to F' and testing whether F' is satisfiable, using the hint $h'(1^n) = h(F_n)$. By assumption we get $n\mathrm{SAT} \in \Vert\!\!\leadsto\mathcal{C}$. Thus, $\mathrm{NP}/poly \subseteq \mathcal{C}/poly$.

Conclusions

In this chapter we studied the scenario in which a polynomial hint is used to decide modified instances. We introduced the notion of a hint function and obtained complexity classes of the form $\mathcal{C}_{\mathrm{MOD}}/str$, which allowed us to categorize the complexity of modification problems with respect to polynomial strings as a hint.

First, we observed that some of our unlikeliness results for selected hints from Section 4 can be used to show also unlikeliness for polynomial hints. In detail, we showed this for the problem (ad^q, SAT), where $q(n) = \sqrt[K]{n}$ and $K \in \mathbb{N} \setminus \{0\}$, but this result also holds for many other modification problems.

The power of polynomial hints allowed us to show that some modification problems (c, A), for which we have not been able to prove containment in $\mathrm{P}_{\mathrm{MOD}}/cert(V_A)$, are a member of $\mathrm{P}_{\mathrm{MOD}}/str$. We showed these results with the help of the so called table-lookup method — a very simple method that involves a table of solutions for all possible modified instances. We applied this technique to the modification problem (ad^k, V_{SAT}), k a constant, and showed that this problem benefits from polynomial strings. But still, the complexity of some modification problems is not clear.

Open problem 5. *Does the modification problem* (ad^{log}, SAT) *benefit from polynomial hints?*

In the last part of this chapter, we showed how the concept of compilability relates to modification problems. We summarized the necessary details about compilability in order to prove hardness results for compilability classes. We translated unconditional results on non-compilability, i.e, $\leq_{comp}$-hardness for a class $\leadsto C$, to unconditional uselessness results for modification problems that have polynomial hints. Also, we showed exemplarily how results about probable non-compilability, i.e., $\|\!\leadsto$-hardness for a class $\|\!\leadsto\mathcal{C}$, can be translated to results about probable uselessness of polynomial hints.

Chapter 9

Appendix A - Decision Problems

In this chapter, we examine the NP-complete problems

- THREESATISFIABILITY (3SAT),
- ONEINTHREESATISFIABILITY (1-3SAT),
- CLIQUE,
- VERTEXCOVER (VC),
- HAMILTONIAN CYCLE (HC),
- THREEDIMENSIONALMATCHING (3DM), and
- PARTITION

with respect to several modification functions.We also give the missing proofs for the problems SAT and EX3SAT.

Easiness results of the form $\mathrm{MOD}_c V_A \in \mathrm{P}$ and $(c, V_A) \in \mathrm{P}^{\in}_{\mathrm{MOD}}/cert(V_A)$ that are due to the trivial argumentation that each certificate of the original instance is also a certificate for the modified instance are omitted. For an overview on how *hardness* results for the other problems are obtained, we refer to Figure 9.1. An arrow in Figure 9.1 indicates that hardness of one problem can be derived from hardness of the other either by

- $\leq^p_m$-(inter)reduction: We use this technique to show NP-completeness for problems of the form $\mathrm{MOD}_c V_A$ on the left side of Figure 9.1 The number in the box on the arrow points to the respective theorem where this fact is proven.
- hint-independent (inter)reduction: We use this technique for problems of the form (c, V_A) on the right side of Figure 9.1. The number in the box on the arrow points to the respective theorem where this fact is proven.

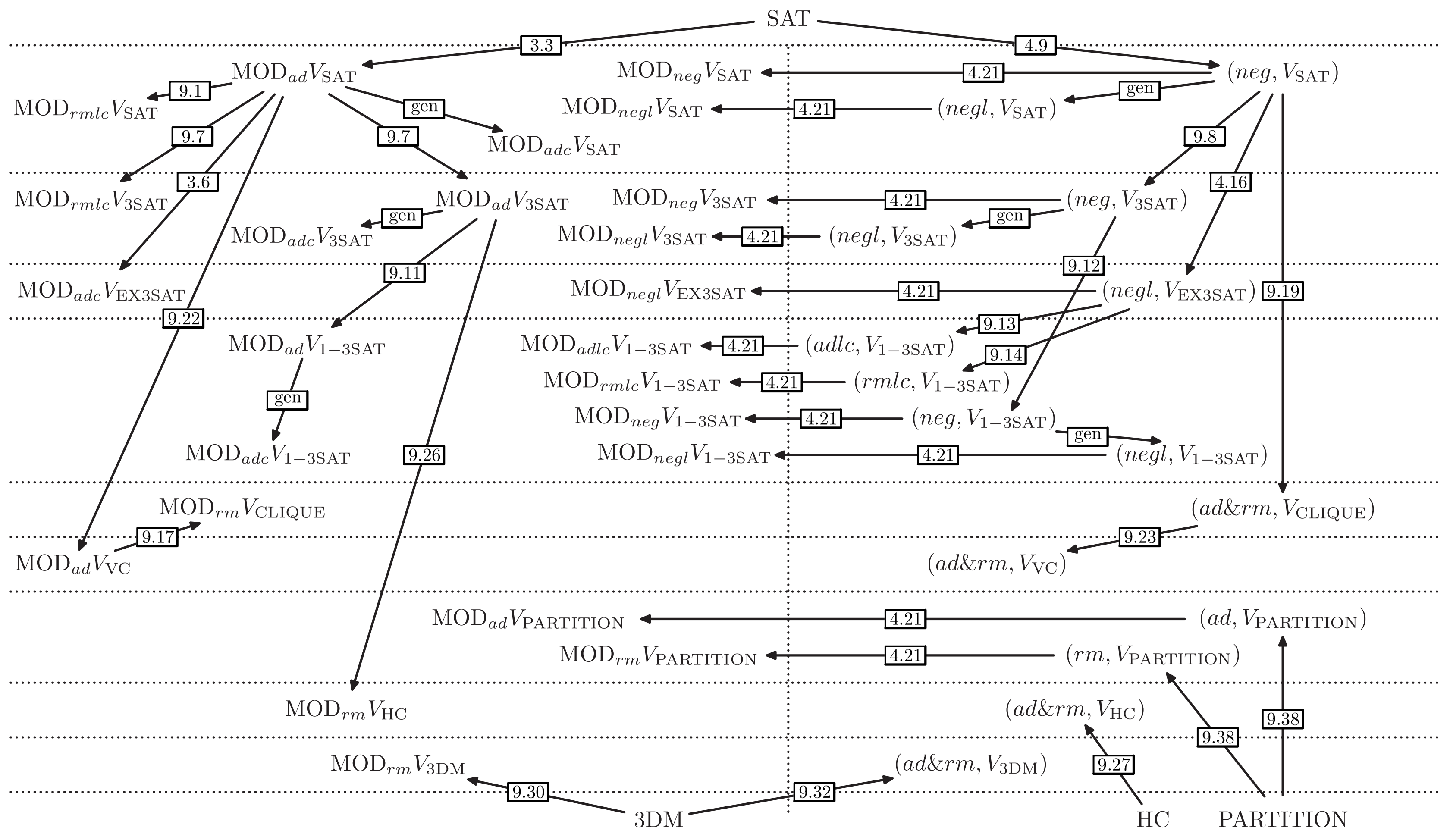
SAT
MOD$_{ad}V_{\text{SAT}}$
MOD$_{neg}V_{\text{SAT}}$
(neg, V_{SAT})
MOD$_{rmlc}V_{\text{SAT}}$
MOD$_{negl}V_{\text{SAT}}$
$(negl, V_{\text{SAT}})$
MOD$_{adc}V_{\text{SAT}}$
MOD$_{rmlc}V_{\text{3SAT}}$
MOD$_{ad}V_{\text{3SAT}}$
MOD$_{neg}V_{\text{3SAT}}$
(neg, V_{3SAT})
MOD$_{adc}V_{\text{3SAT}}$
MOD$_{negl}V_{\text{3SAT}}$
$(negl, V_{\text{3SAT}})$
MOD$_{adc}V_{\text{EX3SAT}}$
MOD$_{negl}V_{\text{EX3SAT}}$
$(negl, V_{\text{EX3SAT}})$
MOD$_{ad}V_{1-\text{3SAT}}$
MOD$_{adlc}V_{1-\text{3SAT}}$
$(adlc, V_{1-\text{3SAT}})$
MOD$_{rmlc}V_{1-\text{3SAT}}$
$(rmlc, V_{1-\text{3SAT}})$
MOD$_{neg}V_{1-\text{3SAT}}$
$(neg, V_{1-\text{3SAT}})$
MOD$_{adc}V_{1-\text{3SAT}}$
MOD$_{negl}V_{1-\text{3SAT}}$
$(negl, V_{1-\text{3SAT}})$
MOD$_{rm}V_{\text{CLIQUE}}$
$(ad\&rm, V_{\text{CLIQUE}})$
MOD$_{ad}V_{\text{VC}}$
$(ad\&rm, V_{\text{VC}})$
MOD$_{ad}V_{\text{PARTITION}}$
$(ad, V_{\text{PARTITION}})$
MOD$_{rm}V_{\text{PARTITION}}$
$(rm, V_{\text{PARTITION}})$
MOD$_{rm}V_{\text{HC}}$
$(ad\&rm, V_{\text{HC}})$
MOD$_{rm}V_{\text{3DM}}$
$(ad\&rm, V_{\text{3DM}})$
3DM
HC
PARTITION
3.3
4.9
4.21
gen
9.1
9.7
3.6
9.22
9.8
4.16
9.12
9.11
9.19
9.13
9.14
9.26
9.17
9.23
9.27
9.38
9.30
9.32

Figure 9.1: Overview on how hardness results are obtained for arbitrary and selected certificates.

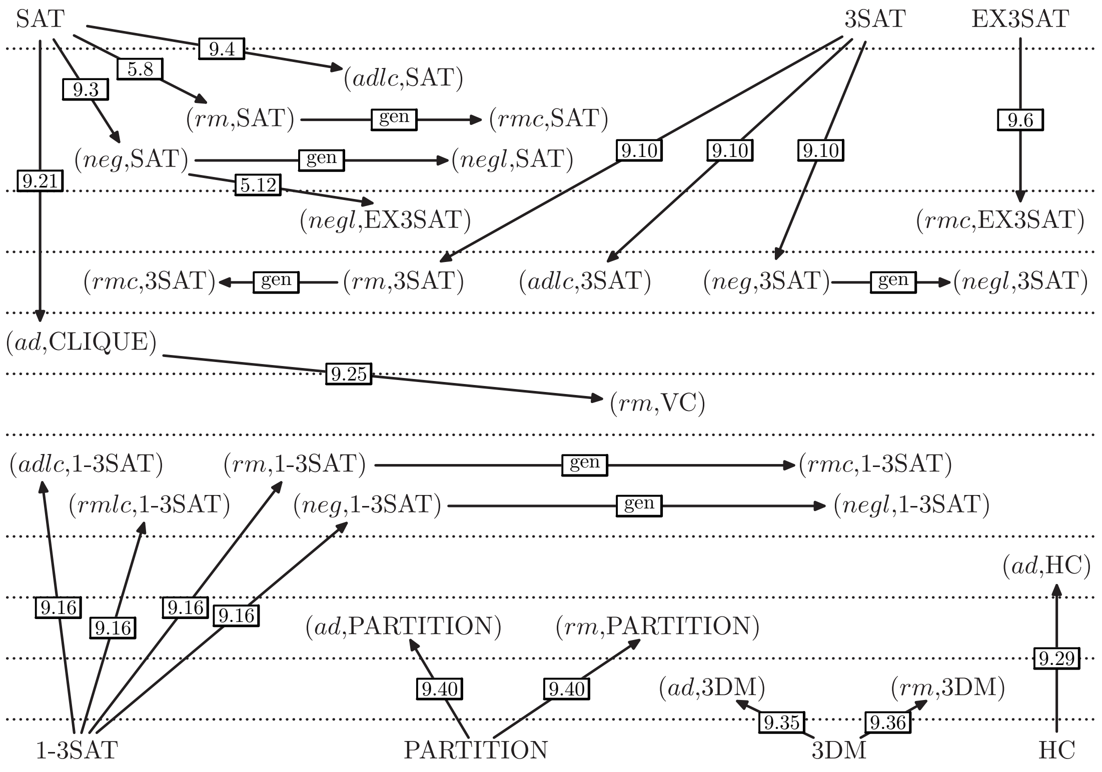

Figure 9.2: Overview on how hardness results are obtained when the no-solution promise is given.

- generalization from some other modification function: This case applies, if some modification is a special case of the other, and therefore hardness results can be transferred. A box with the letters ’gen’ symbolizes these cases.

- generalization from a (c, V_A)-problem to a MOD$_c V_A$ problem: As we have seen in Section 4.4 (Theorem 4.21), if each selected certificate is useless as a hint then an arbitrary certificate must be useless a fortiori. A box filled with ’4.21’ symbolizes the cases where hardness of MOD$_c V_A$ is obtained by this argumentation.

Similarly, Figure 9.2 illustrates how we prove our results in the case that the no-solution promise is given. Again, easy cases are omitted. The main tool to prove uselessness of the no-solution promise are promise-independent reductions. But as we see in Figure 9.2, also some cases of generalization and some promise-independent *inter*reductions are given — although these results could have been proven also with some promise-independent reduction and only little more effort.

9.1 Satisfiability (SAT)

Arbitrary solution as hint

Theorem 9.1. $\mathrm{MOD}_{ad}V_{\mathrm{SAT}} \leq_m^p \mathrm{MOD}_{rmlc}V_{\mathrm{SAT}}$.

Proof. Let F be a formula, $Var(F) = \{x_1, ..., x_n\}$, β be a satisfying assignment for F, and $y \notin Var(F)$. We define

- $f_1(F, \beta, L) := F \cup \bigcup_{i=1}^{n} \{\{x_i, y\}, \{\neg x_i, y\}\}$,
- $f_2(F, \beta, L) := \beta'$, with $\beta'(y) = 1$ and $\beta'(x_i) = \beta(x_i)$, $1 \leq i \leq n$, and
- $f_3(F, \beta, L) := (\{L, y\}, y)$.

Apparently, these functions yield the desired reduction. □

Selected solution as hint

Theorem 9.2. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, V'_{SAT} be a verifier for* SAT, $K \in \mathbb{N} \setminus \{0\}$, *and* $q(n) = \sqrt[K]{n}$. *Then*

1. $(rmlc^q, V'_{\mathrm{SAT}}) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V'_{\mathrm{SAT}}) \Rightarrow \mathrm{NP} \subseteq \mathcal{C}/poly$,
2. $(rmlc, V'_{\mathrm{SAT}}) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V'_{\mathrm{SAT}}) \cap fun \cdot \mathrm{NP}) \Rightarrow \{1\}\mathrm{P} \subseteq \mathrm{NP}$.

Proof. The reduction functions f_1 and f_3 that are given in the proof of NP-completeness of $\mathrm{MOD}_{rmlc}V_{\mathrm{SAT}}$ (Theorem 9.1) can easily be altered to hold for multiple modifications of the form $rmlc^q$. Consequently, the conditions in Theorem 4.27 are satisfied. The assertions follow by application of Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Observation 9.3. $\mathrm{SAT} \leq_{pi}^p (neg, \mathrm{SAT})$.

Proof. Let F be a formula and let $a \notin Var(F)$. We define

- $f_1(F) := F \wedge a \wedge \neg a$,
- $f_2(F) := \neg a$.

Apparently, this yields a $\leq_{pi}^p$-reduction. □

Observation 9.4. $\mathrm{SAT} \leq_{pi}^p (adlc, \mathrm{SAT})$.

Proof. Let F be a formula and let $a, b \notin Var(F)$. We define

- $f_1(F) := F \wedge a \wedge \neg a$,
- $f_2(F) := (\{a\}, b)$.

Apparently, this yields a $\leq_{pi}^p$-reduction. □

9.2 EXACTTHREESATISFIABILITY (EX3SAT)

Selected solution as hint

Theorem 9.5. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, V'_{Ex3SAT} be a verifier for* EX3SAT*, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

1. $(adc^q, V'_{\text{Ex3SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V'_{\text{Ex3SAT}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly$,
2. $(adc, V'_{\text{Ex3SAT}}) \in \text{coNP}^{\in}_{\text{MOD}}/(cert(V'_{\text{Ex3SAT}}) \cap fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. The reduction functions f_1 and f_3 that are given in the proof of NP-completeness of $\text{MOD}_{adc}V_{\text{Ex3SAT}}$ (Theorem 3.6) can easily be altered to hold for multiple modifications of the form adc^q. Consequently, the conditions in Theorem 4.27 are satisfied. The assertions follow by application of Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Observation 9.6. EX3SAT $\leq_{pi}^p (rmc, \text{Ex3SAT})$.

Proof. Let F be an EX3SAT-instance and let $a, b, c \notin Var(F)$. Let E denote the formula that consists of all eight 3-clauses that can be built with the variables $a, b,$ and c. We define

- $f_1(F) := F \wedge E$,
- $f_2(F) := \{\neg a, \neg b, \neg c\}$.

Apparently, this yields a $\leq_{pi}^p$-reduction. □

9.3 THREESATISFIABILITY (3SAT)

Formally, the problem 3SAT is defined as

$$\text{3SAT} := \{F : F \in \text{SAT and } F \text{ has at most three literals per clause}\}$$

and the verifier we use for 3SAT is given by

$$(F, \pi) \in V_{\text{3SAT}} \Leftrightarrow (F, \pi) \in V_{\text{SAT}} \text{ and each clause of } F \text{ has at most three literals.}$$

Note that in contrast to the problem SAT the modification function $adlc$ for 3SAT may only add literals to clauses of size at most two.

Arbitrary solution as hint

Theorem 9.7.

1. $\text{MOD}_{ad}V_{\text{SAT}} \leq^p_m \text{MOD}_{ad}V_{\text{3SAT}}$,
2. $\text{MOD}_{ad}V_{\text{SAT}} \leq^p_m \text{MOD}_{rmlc}V_{\text{3SAT}}$.

Proof. Let f be the reduction function for SAT $\leq^p_m$ EX3SAT as described in the proof of Theorem 3.6. Recall from this proof that a satisfying assignment β' for $f(F)$ is polynomial time computable from a satisfying assignment β of F. Let g denote the respective function that computes β'.

Proof of 1.) Let (F, β, L) be a $\text{MOD}_{ad}V_{\text{SAT}}$-instance. We reduce via

- $f_1(F, \beta, L) := f(F)$,
- $f_2(F, \beta, L) := g(F, \beta)$, and
- $f_3(F, \beta, L) := L$.

Correctness follows by arguments similar to the ones in the proof of Theorem 3.6.

Proof of 2.) Let F be a formula, $Var(F) = \{x_1, ..., x_n\}$, and $y \notin Var(f(F))$. We reduce via

- $f_1(F, \beta, L) := f(F) \cup \bigcup_{i=1}^{n}\{\{x_i, y\}, \{\neg x_i, y\}\}$,
- $f_2(F, \beta, L) := \beta'$, where $\beta'(y) = 1$ and $\beta'(x) = g(F, \beta)(x)$ for $x \in Var(f(F)) \setminus \{y\}$,
- $f_3(F, \beta, L) := (\{L, y\}, y)$.

□

Selected solution as hint

Theorem 9.8. $(neg, V_{\text{SAT}}) \leq^p_{hi} (neg, V_{3\text{SAT}})$.

Proof. The proof is similar to the proof of Theorem 4.16. We just describe how to modify the proof of Theorem 4.16 to obtain a proof for $(neg, V_{\text{SAT}}) \leq^p_{hi} (neg, V_{3\text{SAT}})$:

- $g_1(F, \beta, L) := \begin{cases} f\big(F \setminus \{\{L\}\}\big) \cup \{\{L\}\}, & \text{if } \{L\} \in F, \\ f(F), & \text{otherwise,} \end{cases}$
- $g_3(F, \beta, L) := L$.

The rest of the proof can be translated mutatis mutandis. □

Theorem 9.9. *Let $\mathcal{C}$ be closed under $\leq^p_m$-reduction, $V'_{3\text{SAT}}$ be a verifier for* 3SAT*,* $c \in \{ad, rmlc\}$*,* $K \in \mathbb{N} \setminus \{0\}$*, and* $q(n) = \sqrt[K]{n}$*. Then*

1. $(c^q, V'_{3\text{SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V'_{3\text{SAT}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly$*,*
2. $(c, V'_{3\text{SAT}}) \in \text{coNP}^{\in}_{\text{MOD}}/(cert(V'_{3\text{SAT}}) \cap fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. The reduction functions f_1 and f_3 that are given in the proofs of NP-completeness of $\text{MOD}_{ad}V_{3\text{SAT}}$ and $\text{MOD}_{rmlc}V_{3\text{SAT}}$ (Theorem 9.7) can easily be altered to hold for multiple modifications of the form c^q. Consequently, the conditions in Theorem 4.27 are satisfied. The assertions follow by application of Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Theorem 9.10.

1. $3\text{SAT} \leq^p_{pi} (adlc, 3\text{SAT})$*,*
2. $3\text{SAT} \leq^p_{pi} (rm, 3\text{SAT})$*, and*
3. $3\text{SAT} \leq^p_{pi} (neg, 3\text{SAT})$.

Proof. Let F be a 3SAT-instance and $a, b \notin Var(F)$.
Proof of 1.) Copy of the proof for Observation 9.4.
Proof of 2.) Copy of the proof for Theorem 5.6.
Proof of 3.) Copy of the proof for Observation 9.3.

□

9.4 ONEINTHREESATISFIABILITY (1-3SAT)

We study as a last satisfiability problem the language

$$\text{1-3SAT} := \{\, F : F \in \text{SAT, each clause of } F \text{ consists of at most three literals, and there exists a satisfying assignment of } F \text{ that satisfies exactly one literal in each clause of } F \,\}.$$

If β is an assignment that satisfies a formula F in the sense of 1-3SAT we say that β one-satisfies F. The problem 1-3SAT is NP-complete [Sch78]. We use the following verifier $V_{\text{1-3SAT}}$ for the problem 1-3SAT:

$$(F, \beta) \in V_{\text{1-3SAT}} \quad \Leftrightarrow \quad (F, \beta) \in V_{\text{3SAT}} \wedge \beta \text{ one-satisfies } F.$$

Arbitrary solution as hint

Theorem 9.11. $\text{MOD}_{ad}V_{\text{3SAT}} \leq_m^p \text{MOD}_{ad}V_{\text{1-3SAT}}$.

Proof. We use a reduction 3SAT $\leq_m^p$ 1-3SAT from [HMRS98] (proof of Theorem 3.8 (3SAT $\rightarrow$ 1-EX3SAT) in [HMRS98]). There, the reduction function f maps

- each 3-clause $C_j := \{z_p, z_q, z_r\}$ to the set of clauses

$$C'_j := \{\{z_p, u^j, v^j\}, \{\neg z_q, u^j, w^j\}, \{v^j, w^j, t^j\}, \{\neg z_r, v^j, x^j\}\},$$

- each 2-clause $C_j := \{z_p, z_q\}$ to the set of clauses

$$C'_j := \{\{z_p, u^j, v^j\}, \{\neg z_q, u^j, w^j\}, \{v^j, w^j, t^j\}, \{\neg a^j, v^j, x^j\}, \{a^j, d^j, e^j\}, \{a^j, e^j, f^j\}, \{d^j, e^j, f^j\}\},$$

where $u^j, v^j, w^j, t^j, x^j, a^j, d^j, e^j, f^j$ are new distinct variables local to C'_j. We expand the domain of f to also handle unit clauses. The function f shall map

- each unit clause $C_j := \{z_p\}$ to the formula $C'_j := \{\{z_p\}\}$.

This transformation is done for every clause of F. The reduction function f is formally given by

$$f(F) := \bigcup_{C_j \in F} C'_j.$$

For a proof that this function f yields the reduction 3SAT $\leq_m^p$ 1-3SAT we refer to [HMRS98]. It is also shown in [HMRS98] that a satisfying assignment for F can be transformed in polynomial time to an assignment that one-satisfies $f(F)$. Compatibility with the modification function ad follows from the fact that unit clauses are mapped to unit clauses. Consequently an added unit clause in F results in an added unit clause in $f(F)$. The corresponding modification is computable in polynomial time. This concludes the proof of the theorem. □

Selected solution as hint

Theorem 9.12. $(neg, V_{3\text{SAT}}) \leq^p_{hi} (neg, V_{1\text{-}3\text{SAT}})$.

Proof. A closer examination of the modification function f from [HMRS98], which is also used in the proof of Theorem 9.11, reveals that f is actually a structure preserving reduction that is compatible with the modification function neg. For details we refer to [HMRS98]. □

Theorem 9.13. $(negl, V_{\text{Ex3SAT}}) \leq^p_{hi} (adlc, V_{1\text{-}3\text{SAT}})$.

Proof. It is sufficient to give three functions $g_1, g_2, g_3 \in$ FP such that for each $x = (F, \beta, (C, L))$ the following two conditions hold:

$$V_{\text{Ex3SAT}}(F) = \{\beta\} \Rightarrow V_{1\text{-}3\text{SAT}}\big(g_1(x)\big) = \{g_2(x)\}, \tag{1}$$

$$negl(F, (C, L)) \in \text{Ex3SAT} \Leftrightarrow adlc(g_1(x), g_3(x)) \in \text{1-3SAT}. \tag{2}$$

Let $F = \{C_1, ..., C_m\}$ be an Ex3SAT-formula. The function g_1 is mapping each 3-clause $C_j = \{z_p, z_q, z_r\}$ to the set of clauses

$$C'_j := \big\{\{z_p, s^j, t^j\}, \{\neg z_p, \neg u^j, \neg v^j\}, \{\neg z_q, t^j, \neg v^j\}, \{z_r, \neg w^j\}, \{s^j, \neg w^j, \neg y^j\}\big\},$$

where $s^j, t^j, u^j, v^j, w^j, y^j$ are new distinct variables local to C'_j. Formally, the reduction function g_1 is given by

$$g_1(F, \beta, (C, L)) := \bigcup_{C_j \in F} C'_j,$$

a function that is computable in polynomial time. Now the assertion is a consequence of the following two claims.

Claim 1: There exists $g_2 \in$ FP such that (1) holds.

Proof of Claim 1: Let $V_{\text{Ex3SAT}}(F) = \{\beta\}$. Thus, in each clause $C_j = \{z_p, z_q, z_r\}$ of F at least one literal is satisfied. Consequently, the set of clauses C'_j is one-satisfiable when we expand β to the variables $s^j, t^j, u^j, v^j, w^j, y^j$ according to Table 9.1. The function g_2 is defined to yield the assignment β that appropriately is expanded to the additional variables $s^j, t^j, u^j, v^j, w^j, y^j$, $1 \leq j \leq m$. Apparently, the function g_2 is polynomial-time computable.

It remains to show that the assignment $g_2(\beta)$ is the sole one-satisfying assignment for $g_1(F, \beta, (C, L))$. Assume to the contrary that $g_1(F, \beta, (C, L))$ has two one-satisfying assignments β_1 and β_2. Note that for all clauses C_j the truth values of the variables of C_j uniquely determine the truth values of the variables $s^j, t^j, u^j, v^j, w^j, y^j$ in C'_j (see Table 9.1). We conclude that the assignments β_1 and β_2 differ on some variable from F. Furthermore, for each clause $C_j \in F$ both assignments β_1 and β_2 induce a satisfying assignment for C_j when restricted to variables from C_j. Consequently, when restricting β_1 and β_2 to the variables from F we obtain two distinct satisfying assignments for F.

			clause (I) $\{z_p, s^j, t^j\}$		clause (II) $\{\neg z_p, \neg u^j, \neg v^j\}$		clause (III) $\{\neg z_q, t^j, \neg v^j\}$	clause (IV) $\{z_r, \neg w^j\}$	clause (V) $\{s^j, \neg w^j, \neg y^j\}$
$\beta(z_p)$	$\beta(z_q)$	$\beta(z_r)$	$\beta(s^j)$	$\beta(t^j)$	$\beta(u^j)$	$\beta(v^j)$		$\beta(w^j)$	$\beta(y^j)$
0	0	0	0	1	·	·	$\natural$		
			1	·	·	·		0	$\natural$
0	0	1	0	1	·	·	$\natural$		
			1	0	1	1		1	1
0	1	0	0	1	1	1		0	1
			1	0	·	1	$\natural$		
0	1	1	0	1	1	1		1	1
			1	0	·	1	$\natural$		
1	0	0	0	0	0	1		0	1
					1	0	$\natural$		
1	0	1	0	0	0	1		1	0
					1	0	$\natural$		
1	1	0	0	0	0	1	$\natural$		
					1	0		0	1
1	1	1	0	0	0	1	$\natural$		
					1	0		1	0

Table 9.1: This table shows how to obtain the unique one-satisfying assignment, if existent, for the set of clauses C'_j depending on the truth values of the literals z_p, z_q, and z_r. The assignment is constructed from left to right. A case distinction is performed when ambiguity arises. A '$\cdot$'-entry symbolizes that the corresponding variable is not considered since a contradiction can be derived independently. The '$\natural$'-entry stands for a contradiction of the corresponding clause with the partial assignment constructed so far.

This contradicts our initial assumption, which concludes the proof of Claim 1.

Claim 2: There exists $g_3 \in \mathrm{FP}$ such that (2) holds.

Proof of Claim 2: Let $C_k = \{L_1, L_2, L\}$ be the clause of F in that L will be negated. Note that the function f_1 maps the clause C_k to the clause

$$C'_k := \big\{\{L_1, s^k, t^k\}, \{\neg L_1, \neg u^k, \neg v^k\}, \{\neg L_2, t^k, \neg v^k\}, \{L, \neg w^k\}, \{s^k, \neg w^k, \neg y^k\}\big\},$$

We define g_3 by

$$g_3(F, \beta, (C_k, L)) := (\{L, \neg w^k\}, s^k),$$

that is, we add the literal s^k to the clause $\{L, \neg w^k\}$ in C'_k. Obviously, $g_3 \in \mathrm{FP}$. It remains to show that (2) holds.

To show sufficiency, let β be a satisfying assignment for $negl(F, (C_k, L))$. For each clause C_j, $j \neq k$, from $negl(F, (C_k, L))$ the assignment β can be expanded to a one-satisfying assignment for C'_j according to Table 9.1. The satisfying assignment β for $negl(F, (C_k, L))$, which in particular satisfies $\{L_1, L_2, \neg L\}$, can be expanded to a one-

			clause (I) $\{z_p,s^j,t^j\}$		clause (II) $\{\neg z_p,\neg u^j,\neg v^j\}$		clause (III) $\{\neg z_q,t^j,\neg v^j\}$	modified clause (IV) $\{z_r,s^j,\neg w^j\}$	clause (V) $\{s^j,\neg w^j,\neg y^j\}$
$\beta(z_p)$	$\beta(z_q)$	$\beta(z_r)$	$\beta(s^j)$	$\beta(t^j)$	$\beta(u^j)$	$\beta(v^j)$		$\beta(w^j)$	$\beta(y^j)$
0	0	0	0	1	·	·	$\natural$		
			1	0	1	1		1	1
0	0	1	0	1	·	·	$\natural$		
			1	·	·	·		$\natural$	
0	1	0	0	1	1	1		0	1
			1	0	·	1	$\natural$		
0	1	1	0	1	1	1		1	0
			1	0	·	1	$\natural$		
1	0	0	0	0	0	1		0	1
					1	0	$\natural$		
1	0	1	0	0	0	1		1	0
					1	0	$\natural$		
1	1	0	0	0	0	1	$\natural$		
					1	0		0	1
1	1	1	0	0	0	1	$\natural$		
					1	0		1	0

Table 9.2: This table shows the unique one-satisfying assignment for the modified set of clauses $\{\{z_p, s^k, t^k\}, \{\neg z_p, \neg u^k, \neg v^k\}, \{\neg z_q, t^k, \neg v^k\}, \{z_r, s^k, \neg w^k\}, \{s^k, \neg w^k, \neg y^k\}\}$ depending on the truth values of the literals z_p, z_q, and z_r. For further explanation see Table 9.1

satisfying assignment of

$$\{\{L_1, s^k, t^k\}, \{\neg L_1, \neg u^k, \neg v^k\}, \{\neg L_2, t^k, \neg v^k\}, \{L, s^k, \neg w^k\}, \{s^k, \neg w^k, \neg y^k\}\},$$

according to Table 9.2. In consequence, we obtain a one-satisfying assignment for the formula $adlc(g_1(x), g_3(x))$.

For the other direction assume that β' one-satisfies $adlc(g_1(x), g_3(x))$. Thus, for each set of clauses C'_j, $j \neq k$, the assignment β' restricted to variables from C_j yields a satisfying assignment for C_j (see Table 9.1). For the remaining set of clauses

$$\{\{L_1, s^k, t^k\}, \{\neg L_1, \neg u^k, \neg v^k\}, \{\neg L_2, t^k, \neg v^k\}, \{L, s^k, \neg w^k\}, \{s^k, \neg w^k, \neg y^k\}\}$$

in $adlc(g_1(x), g_3(x))$ the restriction of β' to L_1, L_2, L yields a satisfying assignment for the clause $(L_1, L_2, \neg L)$ (see Table 9.2). In total we obtain a satisfying assignment for $negl(F, (C_k, L))$. This concludes the proof of Claim 2. □

Theorem 9.14. $(negl, V_{\text{Ex3SAT}}) \leq^p_{hi} (rmlc, V_{\text{1-3SAT}})$.

Proof. The proof is similar to the proof of Theorem 9.13. Let $F = \{C_1, ..., C_m\}$ be an Ex3SAT-formula. The function g_1 is mapping each 3-clause $C_j = \{z_p, z_q, z_r\}$ to the set of clauses

$$C'_j := \{\{z_p, s^j, t^j\}, \{\neg z_p, \neg u^j, \neg v^j\}, \{\neg z_q, t^j, \neg v^j\}, \{\neg z_r, s^j, \neg w^j\}, \{s^j, \neg w^j, \neg y^j\}\},$$

where $s^j, t^j, u^j, v^j, w^j, y^j$ are new distinct variables local to C'_j. The reduction functions g_1 and g_3 are given by

$$g_1(F, \beta, (C, L)) := \bigcup_{C_j \in F} C'_j,$$

$$g_3(F, \beta, (C_k, z_r)) := (\{z_r, s^k, \neg w^k\}, s^k).$$

We can prove claims that are similar to the ones in the proof of Theorem 9.13. Again, Table 9.1 and Table 9.2 are helpful to establish these claims. We leave a formal proof to the reader. □

Theorem 9.15. *Let $\mathcal{C}$ be closed under $\leq^p_m$-reduction, $V'_{\text{1-3SAT}}$ be a verifier for* 1-3SAT, *$K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

1. $(ad^q, V'_{\text{1-3SAT}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V'_{\text{1-3SAT}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly$,
2. $(ad, V'_{\text{1-3SAT}}) \in \text{coNP}^{\in}_{\text{MOD}}/(cert(V'_{\text{1-3SAT}}) \cap fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. The reduction functions f_1 and f_3 that are implied by the proof of NP-completeness of $\text{MOD}_{ad}V_{\text{1-3SAT}}$ (Theorem 9.11) can easily be altered to hold for multiple modifications of the form ad^q. Consequently, for all $F, m \in \Sigma^*$ it holds that

- $F \in \text{3SAT} \Leftrightarrow f_1(F) \in \text{1-3SAT}$,
- $ad^q(F, m) \in \text{3SAT} \Leftrightarrow ad^q(f_1(F), f_3(m)) \in \text{1-3SAT}$.

By the construction in the proof of Corollary 9.9 there exist functions $g_1, g_3 \in \text{FP}$ such that for all $F, m \in \Sigma^*$ it holds that

- $F \in \text{SAT} \Leftrightarrow g_1(F) \in \text{3SAT}$ and
- $ad^q(F, m) \in \text{SAT} \Leftrightarrow ad^q(g_1(F), g_3(m)) \in \text{3SAT}$.

By composition of these functions we get functions $h_1, h_3 \in \text{FP}$, for which

- $F \in \text{SAT} \Leftrightarrow h_1(F) \in \text{1-3SAT}$ and
- $ad^q(F, m) \in \text{SAT} \Leftrightarrow ad^q(h_1(F), h_3(m)) \in \text{1-3SAT}$.

Now, the assertions follow from Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Observation 9.16.

1. $\text{1-3SAT} \leq^p_{pi} (adlc, \text{1-3SAT})$,
2. $\text{1-3SAT} \leq^p_{pi} (rm, \text{1-3SAT})$,
3. $\text{1-3SAT} \leq^p_{pi} (rmlc, \text{1-3SAT})$, *and*

4. 1-3SAT $\leq_{pi}^{p}$ $(neg, \text{1-3SAT})$.

Proof. Let F be a 1-3SAT-instance and $a, b, c \notin Var(F)$. The following functions f_1 and f_2 yield the desired $\leq_{pi}^{p}$-reduction:
Proof of 1.)

- $f_1(F) := F \wedge a \wedge \neg a$,
- $f_2(F) := (\{\neg a\}, b)$.

Proof of 2.)

- $f_1(F) := F \wedge a \wedge \neg a$,
- $f_2(F) := \neg a$.

Proof of 3.)

- $f_1(F) := F \wedge a \wedge b \wedge c \wedge (a \vee b \vee \neg c)$,[1]
- $f_2(F) := ((a \vee b \vee \neg c), b)$.

Proof of 4.)

- $f_1(F) := F \wedge a \wedge b \wedge (a \vee b)$,
- $f_2(F) := a$.

□

[1]We choose this rather 'complicated' construction to avoid modified instances in which the same clause appears twice.

9.5 Clique

The problem Clique is defined as follows:

$$\text{Clique} := \{(G,k) : \text{the graph } G \text{ contains a clique of size at least } k\}.$$

The verifier V_{Clique} we use for the problem Clique is given by

$$((G,k),C) \in V_{\text{Clique}} \Leftrightarrow C \subseteq V(G) \ \wedge\ C \text{ induces a } k\text{-clique on } G.$$

We consider the modification functions ad, rm, and $ad\&rm$, which add an edge to a graph G, or remove an edge, or simultaneously add and remove an edge, respectively. Additionally, we examine the case where a single edge is added and k is incremented by one. Formally,

- $ad((G,k),e) := ((V(G), E(G) \cup \{e\}), k)$,
- $rm((G,k),e) := ((V(G), E(G) \setminus \{e\}), k)$,
- $ad\&rm((G,k),(e,f)) := \begin{cases} \big((V(G),(E(G)\cup\{e\})\setminus\{f\}),k\big), & \text{if } f \in E(G), \\ (G,k) & \text{otherwise,} \end{cases}$
- $adinc((G,k),e) := ((V(G), E(G) \cup \{e\}), k+1)$.

Arbitrary solution as hint

We antedate a result from the next section. There, we show that $\text{MOD}_{ad}V_{\text{VC}}$ is NP-complete. Thus, to show NP-completeness of $\text{MOD}_{rm}V_{\text{Clique}}$, it is sufficient to show the following.

Theorem 9.17. $\text{MOD}_{ad}V_{\text{VC}} \leq^p_m \text{MOD}_{rm}V_{\text{Clique}}$.

Proof. The standard reduction function for VC $\leq^p_m$ Clique, as for example given in [GJ79], which uses the strong connection

$$(G,k) \in \text{VC} \Leftrightarrow (\overline{G}, |V(G)| - k) \in \text{Clique}$$

is structure preserving. Also it is compatible w.r.t. the modifications ad and rm, since adding an edge in G corresponds to deleting an edge in the complementary graph $\overline{G}$. □

Because $\text{MOD}_{ad}V_{\text{Clique}}$ is easy (proof omitted), we examine if knowing a k-clique of G can help in finding a $k+1$-clique in an inc-modified graph. This problem is NP-complete.

Theorem 9.18. $\text{MOD}_{adinc}V_{\text{Clique}}$ *is* NP*-complete.*

Proof. We reduce Clique to $\text{MOD}_{adinc}V_{\text{Clique}}$ via the three functions f_1, f_2, and f_3 defined as

- $f_1((G,k)) := \Big(\big(V(G) \cup \{v_1, ..., v_k, y\}\ ,\ E(G) \cup \{\{v_i, v_j\} : 1 \leq i < j \leq k\} \cup \{\{v, y\} : v \in V(G) \setminus \{u\}\}\ \big), k\Big)$,
- $f_2((G,k)) := \{v_1, ..., v_k\}$,
- $f_3((G,k)) := \{u, y\}$,

where $v_1, ..., v_k, y$ are pairwise different vertices that are not contained in G and u is fixed vertex from $V(G)$. The functions f_1, f_2, and f_3 are polynomial-time computable.

Note that $(f_1(G,k), f_2(G,k)) \in V_{\text{CLIQUE}}$. To show CLIQUE $\leq_m^p$ MOD$_{adinc}V_{\text{CLIQUE}}$ it only remains to prove the equivalence

$$(G,k) \in \text{CLIQUE} \Leftrightarrow adinc\big(f_1((G,k)), f_3((G,k))\big) \in \text{CLIQUE}.$$

To show sufficiency, suppose that (G,k) contains a k-clique $C \subseteq V(G)$. The graph $f_1((G,k))$ joined with the edge $\{u,y\}$ contains a $k+1$-clique, namely $C \cup \{y\}$. Thus $adinc(f_1((G,k)), f_3((G,k))) \in$ CLIQUE.

Conversely, assume that the graph $f_1((G,k))$ joined with the edge $\{u,y\}$ contains a $k+1$-clique C. No vertex from $\{v_1, ..., v_k\}$ can be part of this $k+1$-clique C. Therefore $C \subseteq V(G) \cup \{y\}$. The set $C \setminus \{y\}$ is a clique of size at least k in G. Thus $(G,k) \in$ CLIQUE. □

Selected solution as hint

We aim to show that for the modification $ad\&rm$ no certificate function is helpful to decide if the modified instances have appropriate sized cliques. In order to prove this, we use a reduction function f from [Kar72] that shows SAT $\leq_m^p$ CLIQUE. There, the author gives the following function f that maps a formula $F = \{C_1, ..., C_m\}$ to the pair (G, m), where G is given by

- $V(G) = \{\ (L,k)\ :\ L \in C_k\}$ and
- $E(G) = \{\ \big((L,i),(L',j)\big)\ :\ i \neq j\ \wedge\ L \neq \neg L'\}$.

The reader may verify that this yields a reduction from SAT to CLIQUE. Note that the reduction function f has the following useful properties: For any formula $F = \{C_1, ..., C_m\}$

(P1) there exists no $m+1$-clique in $f(F)$,

(P2) if $C_k = \{L\}$ is a unit clause of F then each m-clique of $f(F)$ contains the vertex (L,k),

(P3) if F has exactly one satisfying assignment then $f(F)$ has exactly one m-clique,[2]

[2]It is stated frequently in the literature that this reduction is parsimonious, but it is not. For example, the formula $x_1 \wedge (x_1 \vee x_2 \vee x_3)$ has four satisfying assignments, but the graph $f(F)$ has only three cliques of size two.

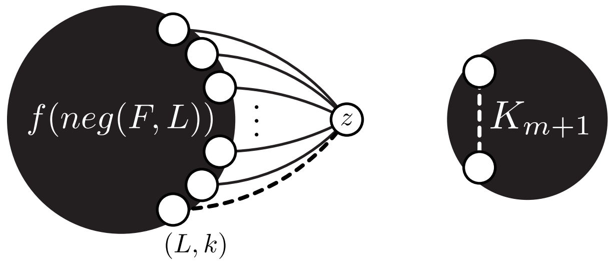

Figure 9.3: The graph G from the proof of Theorem 9.19. A dashed line indicates a modified edge.

(P4) if $V_{\text{SAT}}(F) = \{\beta\}$ then the sole clique of $f(F)$ can be computed from F and β by a function $g \in \text{FP}$.

Property $(P1)$ and property $(P2)$ follow from the fact that the vertex sets $\{(L,k) : L \in C_k\}$, $1 \leq k \leq m$, are independent sets in $f(F)$. Property $(P3)$ can easily be verified by contraposition. The last property is obvious.

Theorem 9.19. $(neg, V_{\text{SAT}}) \leq^p_{hi} (ad\&rm, V_{\text{CLIQUE}})$.

Proof. We give three reduction functions $g_1, g_2, g_3 \in \text{FP}$ such that for all $F, \beta, m \in \Sigma^*$ it holds that

$$V_{\text{SAT}}(F) = \{\beta\} \Rightarrow V_{\text{CLIQUE}}(g_1(F,\beta,L)) = \{g_2(F,\beta,L)\}, \tag{3}$$

$$neg(F,L) \in \text{SAT} \Leftrightarrow ad\&rm\big(g_1(F,\beta,L), g_3(F,\beta,L)\big) \in \text{CLIQUE}. \tag{4}$$

To define g_1 we use the above mentioned reduction function f for SAT $\leq^p_m$ CLIQUE. We distinguish two cases, $\{L\} \in F$ and $\{L\} \notin F$. The total functions g_1, g_2, g_3 can then be composed of their partial counterparts.

We start with the easier case $\{L\} \notin F$. Therefore, F is not modified at all. We define

$$g_1(F,\beta,L) := f(F) \quad \text{and} \quad g_2(F,\beta,L) := g(F,\beta),$$

where g is the certificate mapping function from (P4). The equivalence (3) follows from (P3) and (P4). If we define $g_3(F,\beta,L)$ to be some tuple of edges not contained in $f(F)$ we have that $ad\&rm(g_1(F,\beta,L), g_3(F,\beta,L)) = g_1(F,\beta,L) = f(F)$. Now, the equivalence (4) is a consequence of the fact that f is a reduction function for SAT $\leq^p_m$ CLIQUE.

Now, let $\{L\} \in F$, $F = C_1 \wedge \cdots \wedge C_m$, and $C_k = \{L\}$. Our starting point for the definition of $g_1((F,\beta,L))$ is the graph $G' := f(neg(F,L))$. To G' we add a new vertex z and connect it to all the vertices from G' except the vertex (L,k), which corresponds to the unit clause $C_k = \{L\}$. We also add to G' an extra $m+1$-clique. The resulting graph is the graph G, as depicted in Figure 9.3. Formally, G is defined as follows:

- $V(G) := V(G') \cup \{z, u_1, ..., u_{m+1}\}$ and
- $E(G) := E(G') \cup \{\{v, z\} : v \in V(G) \wedge v \neq (L, k)\} \cup \{\{u_i, u_j\} : 1 \leq i < j \leq m+1\}$.

We claim that (3) holds when we define g_1 and g_2 as

$$g_1(F, \beta, L) := (G, m+1) \text{ and}$$

$$g_2(F, \beta, L) := \{u_1, \dots, u_{m+1}\}.$$

Therefore, we show that the single $m+1$-clique of G is $\{u_1, ..., u_{m+1}\}$ (independent of $|V_{\text{SAT}}(F)|$): By property (P1), the graph G' cannot contain a clique of size greater than m. Therefore, by property (P2) each $m+1$-clique of G other than $\{u_1, ..., u_{m+1}\}$ has to contain the edge $\{z, (L, k)\}$. Since this edge is not present in G we conclude that the sole $m+1$-clique of G is given by $g_2(F, \beta, L)$.

Finally, we define g_3 as

$$g_3(F, \beta, L) := (\{z, (L, k)\}, \{u_1, u_2\}),$$

that is, we add the missing edge between z and (L, k) and remove an edge in the only $m+1$-clique of G.

To see that (4) holds assume that $neg(F, L) \in$ SAT. Thus $f(neg(F, L))$ contains an m-clique C and $ad\&rm(G, \{z, (L, k)\}, \{u_1, u_2\})$ contains the $m+1$ clique $C \cup \{z\}$.

For the other direction, assume that $ad\&rm(G, \{z, (L, k)\}, \{u_1, u_2\})$ contains an $m+1$ clique. Obviously, this clique does not contain any of the vertices $u_1, ..., u_{m+1}$. Thus, the subgraph of G induced by $V(G') \cup \{z\}$ contains an $m+1$-clique and therefore $f(neg(F, L))$ has an m-clique. This implies $neg(F, L) \in$ SAT. □

Theorem 9.20. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, V'_{CLIQUE} be a verifier for* CLIQUE, *$K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

1. $(rm^q, V'_{\text{CLIQUE}}) \in \mathcal{C}^{\in}_{\text{MOD}}/cert(V'_{\text{CLIQUE}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly$,
2. $(rm, V'_{\text{CLIQUE}}) \in \text{coNP}^{\in}_{\text{MOD}}/(cert(V'_{\text{CLIQUE}}) \cap fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. The proof of NP-completeness of $\text{MOD}_{ad}V_{\text{VC}}$ (Theorem 9.22) can easily be altered to hold for multiple modifications of the form ad^q. Using the close connection between VC and CLIQUE we can derive by composition of the respective reductions (also see Theorem 9.15) two functions $f_1, f_3 \in \text{FP}$ such that for all $F, m \in \Sigma^*$ it holds that

- $F \in \text{SAT} \Leftrightarrow f_1(F) \in$ CLIQUE,
- $ad^q(F, m) \in 3\text{SAT} \Leftrightarrow rm^q(f_1(F), f_3(m)) \in$ CLIQUE.

The assertions follow from Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Theorem 9.21. SAT $\leq_{pi}^{p}$ $(ad, \textsc{Clique})$.

Proof. Again, we use the reduction function f for SAT $\leq_m^p$ $\textsc{Clique}$ from [Kar72], as described at the beginning of the last subsection. Let $F = \{C_1, ..., C_m\}$. We construct a graph G by adding two additional vertices z_1 and z_2 to the graph $f(F)$. Then we connect z_1 and z_2 to every vertex in $f(F)$. Formally,

- $V(G) := V(f(F)) \cup \{z_1, z_2\}$,
- $E(G) := E(f(F)) \cup \{\{z_1, v\}, \{z_2, v\} : v \in V(f(F))\}$,

where $z_1, z_2 \notin V(f(F))$. By property $(P1)$ of the reduction f it is clear that $f(F)$ has at most an m-clique. Therefore, G has at most an $m+1$-clique. Now the desired $\leq_{pi}^{p}$-reduction is given by

- $f_1(F) := (G, m+2)$,
- $f_2(F) := \{z_1, z_2\}$.

Correctness of the reduction follows from the fact that $ad(f_1(F), f_2(F))$ has an $m+2$-clique if and only if $f(F)$ has an m-clique. The latter is the case if and only if F is satisfiable. □

9.6 VertexCover (VC)

A *vertex cover* of a graph G is a subset of the vertices of G that covers all the edges, that is, for every edge $e \in E(G)$ one of its endpoints belongs to the vertex cover. The decision problem VC is defined as follows,

$$\text{VC} := \{(G,k) : G \text{ has a vertex cover of size at most } k\}.$$

We fix our verifier V_{VC} for VC as

$$((G,k),C) \in V_{\text{VC}} \Leftrightarrow C \subseteq V(G) \ \wedge \ |C| \leq k \ \wedge \ C \text{ is a vertex cover of } G$$

We consider the modifications ad, rm, and $ad\&rm$ as defined in Section 9.5.

Arbitrary solution as hint

Theorem 9.22 ([Lib04]). $\text{MOD}_{ad}V_{\text{SAT}} \leq_m^p \text{MOD}_{ad}V_{\text{VC}}$.

Proof. The proof is already in [Lib04]. We restate it here for convenience. We construct three reduction functions f_1, f_2, and f_3 such that for all $x := (F, \beta, L)$ it holds that

$$x \in \text{MOD}_{ad}V_{\text{SAT}} \Leftrightarrow ((f_1(x), f_2(x), f_3(x)) \in \text{MOD}_{ad}V_{\text{VC}}. \tag{5}$$

To define f_1, we use a reduction function f from [BC94] that shows SAT $\leq_m^p$ VC. We give a short summary on the work of the function f.

Let $F = \{C_1, ..., C_m\}$ be CNF-formula and $Var(F) = \{x_1, .., x_n\}$. The function f maps F to a graph G_F that consists of truth-setting components, satisfaction testing components, and communication edges between these components. For each variable x_i in F we add as a truth-setting-component the subgraph $(\{x_i, \neg x_i\}, \{x_i, \neg x_i\})$ to G_F, that is, we add the two vertices x_i and $\neg x_i$ joined by a single edge. For each clause C_i we add to G_F as a satisfaction testing component the $|C_i|$-clique that consists of the vertices $a_i^1, ..., a_i^{|C_i|}$. Furthermore, for each clause $C_i = \{L_{i_1}, ..., L_{i_{|C_i|}}\}$ the communication edges are distributed between the truth-setting components and the satisfaction testing components according to the literals in this clause, that is, if $L_{i_1} = x_j$ then a_i^1 is connected to x_j and so on. Figure 9.4 illustrates the graph G_F that is constructed by f when $F = \big\{\{x_1, \neg x_2, x_3\}, \{\neg x_2, \neg x_3, x_4, \neg x_5\}, \{\neg x_4, \neg x_5\}\big\}$. If we set

$$f(F) := (G_F, n + \sum_{i=1}^{m}(|C_i| - 1)).$$

we have the complete definition of f. For a proof of the fact that G_F contains a vertex cover of size $n + \sum_{i=1}^{m}(|C_i| - 1)$ if and only if the CNF-formula F is satisfiable we refer to [BC94].

To show (5) we define

- $f_1((F, \beta, L)) := \Big(\big(V(G_F) \cup \{y\}, E(G_F)\big), n + \sum_{i=1}^{m}(|C_i| - 1)\Big),$

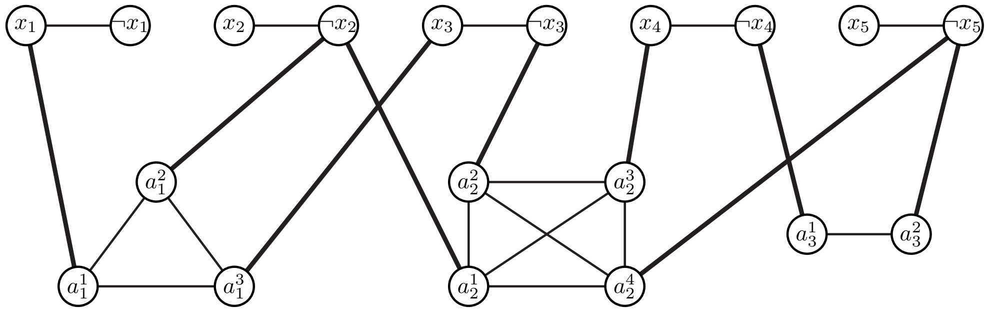

Figure 9.4: A showcase example for the reduction SAT $\leq_m^p$ VC if $F = \{\{x_1, \neg x_2, x_3\}, \{\neg x_2, \neg x_3, x_4, \neg x_5\}, \{\neg x_4, \neg x_5\}\}$.

where y is a new vertex not contained in $V(G_F)$ and

- $f_2((F, \beta, L)) := C'$,

where C' is constructed from the satisfying assignment β as explained in the following. For each variable x_i from F we put the vertex x_i into C' if $\beta(x_i) = 1$ and we add $\neg x_i$ to C' if $\beta(x_i) = 0$. For each clause C_i we search for a literal that is made true by the assignment β. If there exits such a literal L_{i_j}, with a corresponding vertex a_i^j, we add all the vertices $\{a_i^k : 1 \leq k \leq |C_i|, k \neq j\}$ to C'. If no such literal L_{i_j} exists, we put all the vertices $\{a_i^k : 1 \leq k \leq |C_i|\}$ into C'. Finally, we define

- $f_3(x) := \{y, L\}$,

that is, we modify the VC-instance $f_1(x)$ by adding an edge between the isolated vertex y and the literal L. The vertex y now acts as a satisfaction testing component for the unit clause $\{L\}$. All three functions f_1, f_2, and f_3 are polynomial-time computable.

To prove (5), first suppose that $x \in \text{MOD}_{ad}V_{\text{SAT}}$, that is, $(F, \beta) \in V_{\text{SAT}}$ and $F \wedge L \in$ SAT. Since β is a satisfying assignment for F, the vertex cover C' contains exactly $n + \sum_{i=1}^{m}(|C_i| - 1)$ vertices, which implies $(f_1(x), f_2(x)) \in V_{\text{VC}}$. Furthermore, observe that the VC-instance $ad(f_1(x), f_3(x))$ equals $f(F \wedge L)$. Since f is a reduction function for SAT $\leq_m^p$ VC it holds that $f(F \wedge L) \in$ VC. Thus, also $ad(f_1(F), f_2(F)) \in$ VC.

Conversely, assume that $x \notin \text{MOD}_{ad}V_{\text{SAT}}$, that is, (i) $(F, \beta) \notin V_{\text{SAT}}$ or (ii) $F \wedge L \notin$ SAT. In the first case, f_2 transforms (F, β, L) to a cover that has more than $n + \sum_{i=0}^{m}(|C_i| - 1)$ vertices. Such a cover can be no solution for the VC-instance $(G_F, n + \sum_{i=0}^{m}(|C_i| - 1))$. Thus $(f_1(x), f_2(x)) \notin V_{\text{VC}}$. In the second case, we conclude that $ad(f_1(x), f_3(x))$, which equals $f(F \wedge L)$, is not contained in VC, since otherwise the formula $F \wedge L$ was satisfiable. In both cases, we get that $(f_1(x), f_2(x), f_3(x)) \notin \text{MOD}_{ad}V_{\text{VC}}$. □

Selected solution as hint

Theorem 9.23. $(ad\&rm, V_{\text{CLIQUE}}) \leq_{hi}^p (ad\&rm, V_{\text{VC}})$.

Proof. The standard reduction function f for $\textsc{Clique} \leq_m^p$ VC (see [GJ79]) is structure preserving and an addition/deletion of an edge in the $\textsc{Clique}$-instance (G, k) corresponds to a deletion/addition of an edge in the VC-instance $f((G, k))$. □

Theorem 9.24. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, V'_{VC} be a verifier for* VC, $K \in \mathbb{N} \setminus \{0\}$*, and $q(n) = \sqrt[K]{n}$. Then*

1. $(ad^q, V'_{\mathrm{VC}}) \in \mathcal{C}^{\in}_{\mathrm{MOD}}/cert(V'_{\mathrm{VC}}) \Rightarrow \mathrm{NP} \subseteq \mathcal{C}/poly$,
2. $(ad, V'_{\mathrm{VC}}) \in \mathrm{coNP}^{\in}_{\mathrm{MOD}}/(cert(V'_{\mathrm{VC}}) \cap fun \cdot \mathrm{NP}) \Rightarrow \{1\}\mathrm{P} \subseteq \mathrm{NP}$.

Proof. The reduction functions f_1 and f_3 that are given in the proof of NP-completeness of $\mathrm{MOD}_{ad}V_{\mathrm{VC}}$ (Theorem 9.22) can easily be altered to hold for multiple modifications of the form ad^q. Consequently, the conditions in Theorem 4.27 are satisfied. The assertions follow by application of Theorem 4.27 and Theorem 4.31, respectively. □

No solution as promise

Theorem 9.25. $(ad, \textsc{Clique}) \leq_{pi}^p (rm, \mathrm{VC})$.

Proof. The standard reduction function f for $\textsc{Clique} \leq_m^p$ VC, as for example given in [GJ79], is parsimonious. The reduction f is also compatible with respect to the modification functions ad and rm, that is, the addition of an edge in the $\textsc{Clique}$-instance (G, k) corresponds to the deletion of an edge in the VC-instance $f((G, k))$. Therefore, both properties of Definition 5.7 are satisfied. □

9.7 HAMILTONIANCYCLE (HC)

The problem HC consists of all graphs that have a Hamiltonian cycle. We use the following verifier V_{HC} for HC:

$$(G, C) \in V_{\text{HC}} \Leftrightarrow C \subseteq E(G) \wedge C \text{ forms a Hamiltonian cycle in G}.$$

Note that a cycle of G usually is defined as a sequence of vertices. This would lead to several cycles, with different starting points and directions, all corresponding to essentially the *same* Hamiltonian cycle. With the verifier V_{HC} a graph G has exactly as many certificates as the graph G contains 'different' Hamiltonian cycles. Nevertheless, for notational benefits we often refer to Hamiltonian cycles in G as a sequence of the vertices of G.

We consider the three modification functions ad, rm, and $ad\&rm$ as defined by

- $ad(G, e) := (V(G), E(G) \cup \{e\})$,
- $rm(G, e) := (V(G), E(G) \setminus \{e\})$, and
- $ad\&rm(G, (e, f)) := \begin{cases} \big(V(G)\ ,\ (E(G) \cup \{e\}) \setminus \{f\}\ \big), & \text{if } f \in E(G), \\ G, & \text{otherwise.} \end{cases}$

Arbitrary solution as hint

Theorem 9.26. $\text{MOD}_{ad}V_{3\text{SAT}} \leq_m^p \text{MOD}_{rm}V_{\text{HC}}$.

Proof. We construct three reduction functions f_1, f_2, and f_3 such that for all $x := (F, \beta, L)$ it holds that

$$x \in \text{MOD}_{ad}V_{3\text{SAT}} \Leftrightarrow ((f_1(x), f_2(x), f_3(x)) \in \text{MOD}_{rm}V_{\text{HC}}. \tag{6}$$

We use an idea from [PS76], where it is shown that EX3SAT $\leq_m^p$ HC. First, we give a sketch of the proof in [PS76]. Then we show how to reduce 3SAT to HC, using basically the same idea. Last, we will alter this reduction from 3SAT to HC in a way that is becomes compatible w.r.t. ad and rm.

First, we summarize the proof for EX3SAT $\leq_m^p$ HC from [PS76]. The following two graph gadgets A and B, which are depicted in Figure 9.5 a) and Figure 9.5 d), are very helpful in the proof. We just mention the following property of A (for details see [PS82]). Assume that A is an induced subgraph of some Hamiltonian graph G. Furthermore, assume that the vertices $z_1, ..., z_{12}$ are only incident to edges from $E(A)$. Then, each Hamiltonian cycle of G traverses the gadget A by using one of the ways depicted in Figure 9.5 b). Thus, the graph A behaves as if it was just a pair of edges $\{u, u'\}$ and $\{v, v'\}$ of G with the additional restriction that each Hamiltonian cycle of G traverses exactly one of them. We represent this as shown in Figure 9.5 c). A similar property holds for the gadget B. In short, each Hamiltonian path through B, can traverse the gadget by using an arbitrary subset of the edges $\{\{u_i, u_{i+1}\} : i = 1, 2, 3\}$, except the case where it uses all three of these edges.

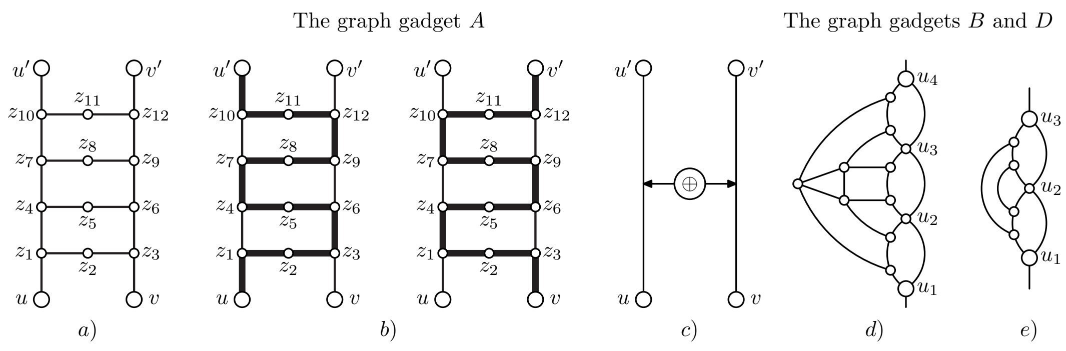

Figure 9.5: The graph gadgets A,B, and D, which are shown in a),d), and e), respectively. The only two possible ways to traverse A are given in b). A pictorial representation of the gadget A is given in c).

Let $F = \{C_1, ..., C_m\}$ be a EX3CNF-formula and let $Var(F) = \{x_1, ..., x_n\}$. We show how to construct a graph $G = f(F)$ such that $F \in$ EX3SAT $\Leftrightarrow f(F) \in$ HC. For each variable x_i, $1 \leq i \leq n$, we have two vertices v_i and w_i as well as an 'upper' and a 'lower' copy of the edge $\{v_i, w_i\}$[3]. These two edges shall represent the fact that the variable x_i is set to either 'false' or 'true', respectively. We also have the edges $\{w_i, v_{i+1}\}$, $1 \leq i \leq n-1$, that connect these so called truth-setting components.

For each 3-clause C_j, $1 \leq j \leq m$, we add a copy B^j of the graph B. Let u^j denote the vertices from B^j that correspond to the u-vertices in B. We add the edges $\{u_1^1, v_1\}$, $\{u_4^m, w_n\}$, and $\{u_4^j, u_1^{j+1}\}$, $1 \leq j \leq m-1$, to the graph constructed so far. Now, we take into account the exact form of the clauses of F by 'connecting' edges from the truth-setting component with edges from the copies of B. We use the gadget A to 'connect' the edge $\{u_i^j, u_{i+1}^j\}$ with the 'upper' copy of $\{v_k, w_k\}$ in case that the ith literal of C_j is $\neg x_k$ and with the 'lower' copy of $\{v_k, w_k\}$ if it is x_k. This concludes the construction of G. For a proof of correctness we refer to [PS82].

Now we alter this reduction to hold for 3SAT-formulas as well. The construction of the needed graph G' works exactly as above, except when unit clauses or 2-clauses appear. For each unit-clause x_k of F we add a new vertex y on the 'lower' copy of the edge $\{v_k, w_k\}$. For each negated unit-clause $\neg x_k$ of F we add a vertex on the 'upper' copy of the edge $\{v_k, w_k\}$. This new vertex y is added as direct neighbor of v_k so that it does not interfere with any (potentially to be added) A-vertices on this edge. As a consequence, for a unit clause $\{x_k\}$ ($\{\neg x_k\}$) each Hamiltonian cycle trough G' has to use the 'lower' ('upper') copy of the edge $\{v_k, w_k\}$, thereby fixing the variable x_k.

For each 2-clause C_j of F we add to G' a copy D^j of the gadget D depicted in Figure 9.5 e) . This gadget D has the same property as B, namely that not both of the edges $\{u_1, u_2\}$ and $\{u_2, u_3\}$ can be used when trying to traverse D - but any other selection allows a traversal. The edges $\{u_1^j, u_2^j\}$ and $\{u_2^j, u_3^j\}$ are connected to the truth setting components in the usual manner. This concludes the construction of G' it should be clear from the arguments given in [PS82] that $F \in$ 3SAT $\Leftrightarrow G' \in$ HC.

[3]These both edges will later be made distinct by inserting vertices along the edges.

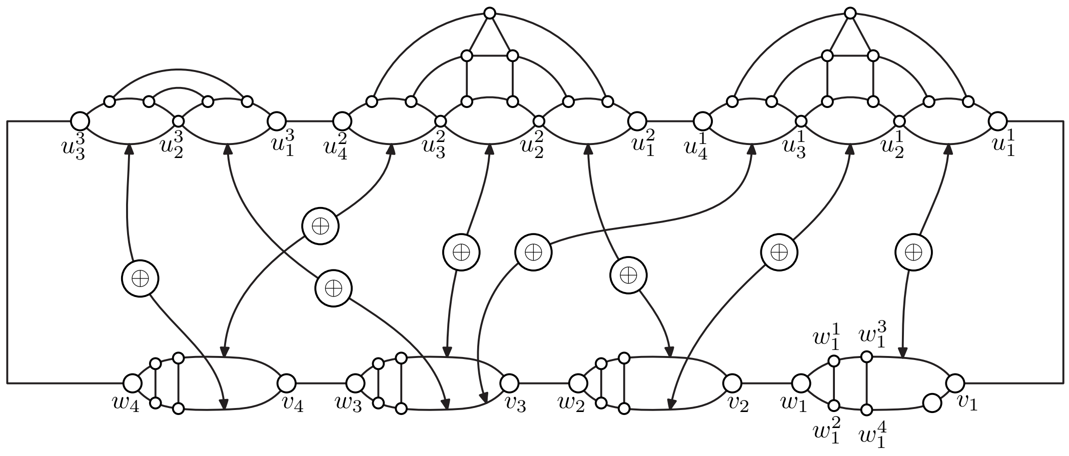

Figure 9.6: A showcase example for the reduction function f_1, where the input instance is $F = (\neg x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_3 \vee x_4) \wedge x_1$.

In a last step, we now alter the reduction such that the addition of a unit clause in F is equivalent to the removal of an edge in some graph $G'' := f_1(F, \beta, L)$. Our starting point is the graph G' from above. For each vertex w_k in G', $1 \leq k \leq n$, we add new vertices w_k^1,w_k^2,w_k^3,w_k^4 to the copies of $\{v_k, w_k\}$ as indicated by Figure 9.6. Additionally, we add the edges $\{w_k^1, w_k^2\}$ and $\{w_k^3, w_k^4\}$. This concludes the construction of $G'' = f_1(F, \beta, L)$.

To show (6) note that a cycle through G'' can easily be computed from a satisfying assignment for F, and vice versa. Let f_2 be the respective function that maps assignments to cycles. Thus,

$$(F, \beta) \in V_{\text{3SAT}} \Leftrightarrow f_1(F, \beta, L), f_2(F, \beta, L) \in V_{\text{HC}}.$$

It remains to show that there exists a function $f_3 \in \text{FP}$ such that

$$ad(F, L) \in \text{3SAT} \Leftrightarrow rm(f_1(F, \beta, L), f_2(F, \beta, L)) \in \text{HC}.$$

The function $f_3(F, \beta, L)$ is given by

$$f_3(F, \beta, L) := \begin{cases} \{w_k^2, w_k^4\}, & \text{if } L = x_k, \\ \{w_k^1, w_k^3\}, & \text{if } L = \neg x_k. \end{cases}$$

This modification forces each Hamiltonian cycle of the modified graph to use the lower (upper) copy of $\{v_k, w_k\}$ in case that $L = x_k$ ($L = \neg x_k$). This is possible if and only if F has a satisfying assignment β with $\beta(L) = 1$ ($\beta(L) = 0$), or equivalently, $F \wedge L \in$ SAT. □

Selected solution as hint

Theorem 9.27. HC $\leq_{hi}^{p}$ $(ad\&rm, V_{\text{HC}})$.

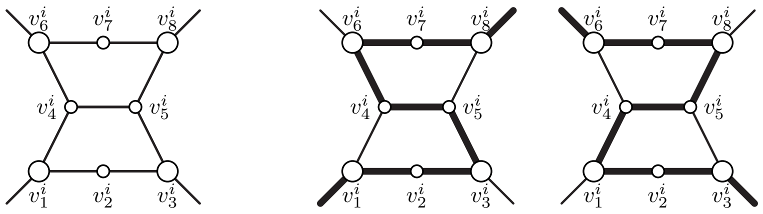

Figure 9.7: The graph gadget H_i and its two Hamiltonian traversals. The vertices $v_1^i, v_3^i, v_6^i, v_8^i$ are the only vertices that may be connected to other vertices.

Proof. We give three reduction functions $f_1, f_2, f_3 \in \text{FP}$ such that

- $G \in \text{HC} \Leftrightarrow ad\&rm(f_1(G), f_3(G)) \in \text{HC}$ and
- $V_{\text{HC}}(f_1(G)) = \{f_2(G)\}$.

In defining f_1 we follow a construction from [Krü05]. There, the author shows how to transform a given graph G to a graph G' that has certain Hamiltonian cycles that correspond to Hamiltonian cycles of G and one additional Hamiltonian cycle that does not correspond to any of the Hamiltonian cycles of G. This additional Hamiltonian cycle will be the hint that is given by f_2. The gadget H_i, which is depicted in Figure 9.7, is a useful tool when constructing G'. So, before turning to formal definitions of the reduction functions, we state a useful structural property of the gadget H_i.

Claim: *Each Hamiltonian cycle through the gadget H_i contains either the path $(v_1^i, v_2^i, v_3^i, v_5^i, v_4^i, v_6^i, v_7^i, v_8^i)$ or the path $(v_3^i, v_2^i, v_1^i, v_4^i, v_5^i, v_8^i, v_7^i, v_6^i)$.*

A proof of the claim by exhaustive case distinction is in [Krü05].

Now, we describe the transformation of a graph G into a graph $G' := f_1(G)$. Let $v_1, ..., v_n$ be the vertices of G. For each vertex v_i, $1 \leq i \leq n$, we add the gadget H_i to the graph G', which results in a graph that consists of n gadgets $H_1, ..., H_n$. Let $v_1^i, ..., v_8^i$ denote the vertices of H_i as depicted in Figure 9.7. For each gadget H_i, $1 \leq i \leq n-1$, we connect the vertices v_8^i and v_1^{i+1}. Furthermore, we add the edge $\{v_8^n, v_1^1\}$. Consequently, the resulting graph contains the Hamiltonian cycle

$$C' := (v_1^1, v_2^1, v_3^1, v_5^1, v_4^1, v_6^1, v_7^1, v_8^1, v_1^2, ..., v_8^{n-1}, v_1^n, v_2^n, v_3^n, v_5^n, v_4^n, v_6^n, v_7^n, v_8^n, v_1^1).$$

As a last step in the construction of G' we want to establish a connection between Hamiltonian cycles in G and corresponding Hamiltonian cycles in G'. Therefore, for each edge $\{v_i, v_j\} \in E(G)$ we add the edges $\{v_3^i, v_6^j\}$ and $\{v_3^j, v_6^i\}$ to the graph constructed so far. The resulting graph is the graph G'. For an illustration of the construction of G' also see Figure 9.8. We define

- $f_1(G) := G' \setminus \{v_1^1, v_4^1\}$,[4]

[4]Within this proof, we use the shorthand $G \setminus e$ to denote the graph G in that e is removed.

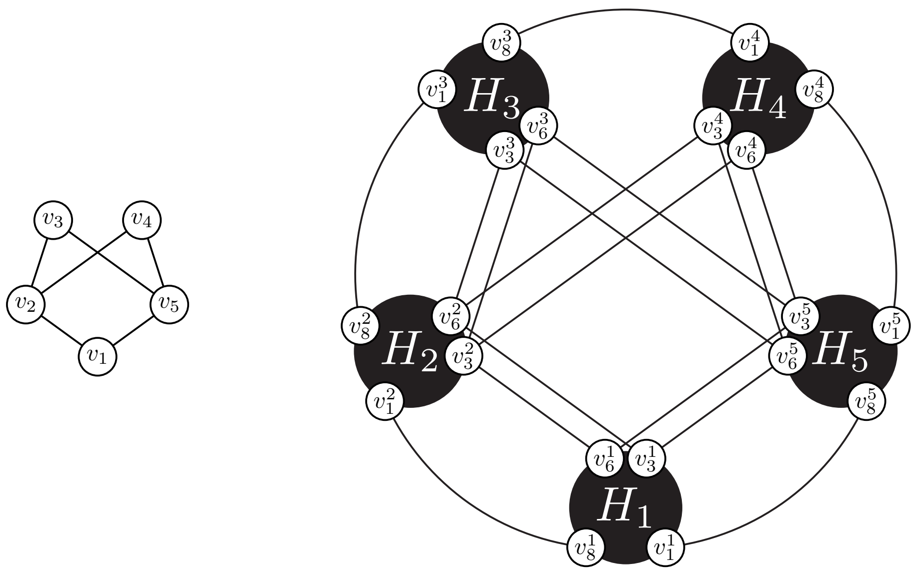

Figure 9.8: An example for the graph G' (on the right) that is constructed from the graph G (on the left).

- $f_2(G) := C'$, and
- $f_3(G) := (\{v_1^1, v_4^1\}, \{v_3^1, v_5^1\})$,

that is, we modify G' by adding the edge $\{v_1^1, v_4^1\}$ and removing the edge $\{v_3^1, v_5^1\}$.

In the remainder of the proof we restrict HC to graphs with at least three vertices. This is justified, since there are only finitely many graphs with fewer than three vertices, all of them non-members of HC. By mapping these graphs to a fixed, appropriately chosen HC-instance the reduction functions f_1,f_2, and f_3 remain polynomial-time computable.

Now we show that $V_{\text{HC}}(f_1(G)) = \{f_2(G)\}$, i.e., that the graph $G' \setminus \{v_1^1, v_4^1\}$ has C' as single Hamiltonian cycle. By the claim, a Hamiltonian cycle C through $G' \setminus \{v_1^1, v_4^1\}$ has to traverse the gadget H_1 along one of the two mentioned possible paths. One of these paths contains the edge $\{v_1^1, v_4^1\}$. Consequently, C uses the other path and enters/leaves the gadget H_i at v_1^1 and v_8^1. But then, the rest of the cycle is predetermined and is equal to C'.

What remains to be proven is the equivalence

$$G \in \text{HC} \Leftrightarrow G' \setminus \{v_3^1, v_5^1\} \in \text{HC}.$$

To show necessity, suppose that $G \in \text{HC}$. Let $(v_{i_1}, ..., v_{i_n}, v_{i_1})$ denote a Hamiltonian cycle in G. Then clearly

$$(\underbrace{v_3^{i_1}, v_2^{i_1}, v_1^{i_1}, v_4^{i_1}, v_5^{i_1}, v_8^{i_1}, v_7^{i_1}, v_6^{i_1}}_{\text{path through } H_{i_1}}, v_3^{i_2}, ..., v_6^{i_{n-1}}, \underbrace{v_3^{i_n}, v_2^{i_n}, v_1^{i_n}, v_4^{i_n}, v_5^{i_n}, v_8^{i_n}, v_7^{i_n}, v_6^{i_n}, v_3^{i_1}}_{\text{path through } H_{i_n}})$$

is a Hamiltonian cycle in G' that does not use the edge $\{v_3^1, v_5^1\}$.

Conversely, let C denote a Hamiltonian cycle in $G' \backslash \{v_3^1, v_5^1\}$. Note that C has to leave or enter the gadget H_1 through the vertex v_1^3, which is a vertex of degree two in $G' \setminus \{v_3^1, v_5^1\}$ and has only one incident edge in the gadget H_1. Therefore, the above claim ensures that the gadget H_1 is traversed by C via the path $(v_3^1, v_2^1, v_1^1, v_4^1, v_5^1, v_8^1, v_7^1, v_6^1)$. Since any Hamiltonian cycle that enters the gadget H_1 at v_3^1 has to enter every other gadget H_i at v_3^i we conclude that the cycle C has the form

$$(\underbrace{v_3^{i_1}, v_2^{i_1}, v_1^{i_1}, v_4^{i_1}, v_5^{i_1}, v_8^{i_1}, v_7^{i_1}, v_6^{i_1}}_{\text{path through } H_{i_1}}, v_3^{i_2}, ..., v_6^{i_{n-1}}, \underbrace{v_3^{i_n}, v_2^{i_n}, v_1^{i_n}, v_4^{i_n}, v_5^{i_n}, v_8^{i_n}, v_7^{i_n}, v_6^{i_n}, v_3^{i_1}}_{\text{path through } H_{i_n}}).$$

Therefore the edges $\{v_{i_j}\, v_{i_{j+1}}\}, 1 \leq j \leq n-1$, and $\{v_{i_n}, v_{i_1}\}$ are contained in $E(G)$. Now, since $n > 2$, the path $(v_{i_1}, ..., v_{i_n}, v_{i_1})$ forms a Hamiltonian cycle in G, which implies $G \in \text{HC}$. $\square$

Theorem 9.28. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, let V'_{HC} be a verifier for* HC, *$K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

1. $(rm^q, V'_{\text{HC}}) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V'_{\text{HC}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly$,
2. $(rm, V'_{\text{HC}}) \in \text{coNP}_{\text{MOD}}^{\in}/(cert(V'_{\text{HC}}) \ \cap \ fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. The reduction functions f_1 and f_3 that are given in the proof of NP-completeness of $\text{MOD}_{rm}V_{\text{HC}}$ (Theorem 9.26) can easily be altered to hold for multiple modifications of the form rm^q. The rest of the proof is similar to the proof of Theorem 9.15. $\square$

No solution as promise

Theorem 9.29. $\text{HC} \leq_{pi}^p (ad, \text{HC})$.

Proof. Let G be a graph and let v be a vertex of G. Let $u_1, ..., u_k$ be the neighbors of v. We give two reduction functions $g_1, g_2 \in \text{FP}$ such that

- $g_1(G) \notin \text{HC}$,
- $G \in \text{HC} \Leftrightarrow ad(g_1(G), g_2(G)) \in \text{HC}$.

Let $G \setminus \{v\}$ denote the subgraph induced by the vertices $V(G) \setminus \{v\}$. We construct an auxiliary graph G' that is defined by

- $V(G') := V(G \setminus \{v\}) \cup \{v_1, v_2, v_3\}$ and
- $E(G') := E(G \setminus \{v\}) \cup \{\{v_1, v_2\}, \{v_2, v_3\}\} \cup \bigcup_{i=1}^k \{\{v_1, u_i\}, \{v_3, u_i\}\}$,

where v_1, v_2, and v_3 are three new vertices that are not contained in $V(G)$. Informally, the vertex v is expanded to a 3-chain of vertices v_1, v_2, and v_3, in which the outer vertices v_1 and v_3 are connected to the old neighbors of v.

Now the functions g_1 and g_2 are defined as follows:

- $g_1(G) := (V(G'), E(G') \setminus \{\{v_1, v_2\}\})$,
- $g_2(G) := \{v_1, v_2\}$.

Obviously, the graph $g_1(G)$ does not contain a Hamiltonian Cycle, since the vertex v_2 has degree one. Furthermore, $ad(g_1(G), g_2(G)) = G'$. It is not difficult to verify that G' has a Hamiltonian cycle if and only if G has one. We leave this as an easy task for the reader. $\square$

9.8 THREEDIMENSIONALMATCHING (3DM)

The problem 3DM is defined as follows. Input instances to 3DM are sets of triples over an alphabet Σ. A set $S \subseteq \Sigma^3$ is contained in 3DM if and only if it has a three-dimensional matching, or for short, a 3D-matching. A 3D-matching of a set S is a subset M of S that covers each element of S at its respective coordinate, but in a way such that no two elements of M agree in the same coordinate. We choose as verifier for 3DM the verifier V_{3DM} that is characterized by

$$(S, M) \in V_{\text{3DM}} \Leftrightarrow S \subseteq \Sigma^3 \wedge\ M \subseteq S \text{ is a 3D-matching for } S.$$

The elementary modifications of interest are the addition of a triple t to the set of triples S, the deletion of such a triple from S, and the simultaneous addition and deletion of triples. The respective modification functions are given by

- $ad(S,t) := S \cup \{t\}$,
- $rm(S,t) := S \setminus \{t\}$, and
- $ad\&rm(S,(t_1,t_2)) := \begin{cases} (S \cup \{t_1\}) \setminus \{t_2\}, & \text{if } t_2 \in S, \\ S, & \text{otherwise.} \end{cases}$

Before turning to the results, we state a few facts about 3D-matchings and introduce some helpful notions. Let S_i denote the set of different alphabetic characters that occur as ith component in the set $S \subseteq \Sigma^3$, e.g., $S_1 := \{x : (\exists y, z \in \Sigma)\, [(x,y,z) \in S]\}$. Note that a set S can only have a 3D-matching if $|S_1| = |S_2| = |S_3|$. We call such sets S with $|S_1| = |S_2| = |S_3|$ well-formed. For well-formed sets S let $\{x_1, ..., x_\ell\} := S_1$, $\{y_1, ..., y_\ell\} := S_2$, and $\{z_1, ..., z_\ell\} := S_3$.

Arbitrary solution as hint

Observation 9.30. $\text{MOD}_{ad}V_{\text{3DM}} \in \text{P}$.

Proof. Let M be a 3D-matching for a set S. Containment of $ad(S,(x,y,z))$ in 3DM can be decided with the following polynomial-time algorithm.

First, determine if $x \in S_1$, $y \in S_2$, and $z \in S_3$. If this is the case then M is a 3D-matching for $ad(S,(x,y,z))$ and the algorithm outputs "yes". If the first case does not apply, test if $x \notin S_1$, $y \notin S_2$ and $z \notin S_3$. Then a 3D-matching for $ad(S,(x,y,z))$ is given by $M \cup \{(x,y,z)\}$ and the algorithm outputs "yes". Otherwise, if none of the above cases apply, the algorithm outputs "no" since the set $ad(S,(x,y,z))$ is not well-formed. □

Theorem 9.31. $\text{3DM} \leq_m^p \text{MOD}_{rm}V_{\text{3DM}}$.

Proof. We show existence of three polynomial-time computable functions f_1, f_2, and f_3 such that for all sets S it holds that

$$S \in \text{3DM} \Leftrightarrow \big(f_1(S), f_2(S)\big) \in V_{\text{3DM}} \wedge rm\big(f_1(S), f_3(S)\big) \in \text{3DM}. \qquad (7)$$

Assume that S is a well-formed 3DM-instance, otherwise the proof is trivial. Let $S_1 = \{x_1, ..., x_\ell\}$, $S_2 = \{y_1, ..., y_\ell\}$, and $S_3 = \{z_1, ..., z_\ell\}$. The basic idea for the definition of the reduction function f_1 originates from [UN96]. In this paper a function $f \in$ FP is given such that, for any set of triples S, the set $f(S)$ contains (i) at least two 3D-matchings if $S \in$ 3DM (ii) contains exactly one 3D-matching if $S \notin$ 3DM. In detail, the function f from [UN96] is given by

$$f(S) := S \cup M' \cup CAV,$$

where x_i^+, y_i^+, z_i^+, $1 \leq i \leq \ell$, are new variables that are not contained in $S_1 \cup S_2 \cup S_3$,

- $M' := \{(x_i, y_i, z_i^+) : 1 \leq i \leq \ell\} \cup \{(x_i^+, y_i^+, z_i) : 1 \leq i \leq \ell\}$, and
- $CAV := \{(x_i^+, y_{i+1}^+, z_i^+) : 1 \leq i \leq \ell - 1\} \cup \{(x_\ell^+, y_1^+, z_\ell^+)\}$.

Note that M' is a 3D-matching for $f(S)$. Informally, the set M' may be referred to as an extra matching and CAV may be seen as cover set for the additionally introduced variables.

Now we are prepared to define the functions f_1, f_2, and f_3:

- $f_1(S) := f(S)$,
- $f_2(S) := M'$, and
- $f_3(S) := (x_1^+, y_1^+, z_1)$.

The functions f_1, f_2, and f_3 are polynomial-time computable. Before turning to the proof that these functions f_1, f_2, f_3 satisfy (7), we establish a helpful claim.

Claim: Let M be a 3D-matching for $rm(f_1(S), f_3(S))$. Then $M \cap M' = \emptyset$.

Proof of claim: Let M be a 3D-matching for $f_1(S) \setminus \{(x_1^+, y_1^+, z_1)\}$. The triple (x_1^+, y_2^+, z_1^+) is contained in M since it is the only element of $f_1(S) \setminus \{(x_1^+, y_1^+, z_1)\}$ that contains the character x_1^+. Thus, the element y_2^+ is already covered and therefore $(x_2^+, y_2^+, z_2) \notin M$. Considering the element x_2^+ we conclude that $(x_2^+, y_3^+, z_2^+) \in M$. Repeated use of this argument yields that $CAV \subseteq M$. Since CAV covers all of the additional variables x_i^+, y_i^+, z_i^+, $1 \leq i \leq \ell$, and all of the triples from M' contain at least one of these additional variables, we conclude that $M \cap M' = \emptyset$. This proves the claim.

To show (7), it is sufficient to show the equivalence

$$S \in \text{3DM} \Leftrightarrow rm(f_1(S), f_3(S)) \in \text{3DM}$$

since $\big(f_1(S), f_2(S)\big) \in V_{\text{3DM}}$ is obviously true.

To prove sufficiency, suppose that $S \in$ 3DM and let M be a 3D-matching for S. Thus, $M \cup CAV$ is apparently a matching for $f(S) = f_1(S)$. Since $f_3(S) \notin M \cup CAV$

the set $M \cup CAV$ is a matching for $f_1(S) \setminus \{f_3(S)\}$, which implies $rm(f_1(S), f_3(S)) \in$ 3DM.

Conversely, assume that $rm\big(f_1(S), f_3(S)\big) \in$ 3DM and let M be a matching for $rm\big(f_1(S), f_3(S)\big)$. Using the claim, we obtain the result that $M' \cap M = \emptyset$. Therefore $M \subseteq S \cup CAV$. Now, the set $M \setminus CAV$ is a matching for S. Consequently, $S \in$ 3DM. □

Selected solution as hint

Theorem 9.32. 3DM $\leq^p_{hi} (ad\&rm, V_{\text{3DM}})$.

Proof. We give three reduction functions $g_1, g_2, g_3 \in$ FP such that

- $V_{\text{3DM}}(g_1(S)) = \{g_2(S)\}$ and
- $S \in$ 3DM $\Leftrightarrow ad\&rm(g_1(S), g_3(S)) \in$ 3DM.

Recall the proof of Theorem 9.31 and the function f defined there. The three functions g_1, g_2 and g_3 are given by

- $g_1(S) := f(S) \setminus \{x_1^+, y_2^+, z_1^+\}$,
- $g_2(S) := M'$, and
- $g_3(S) := \big((x_1^+, y_2^+, z_1^+), (x_1^+, y_1^+, z_1)\big)$.

First, we show that $V_{\text{3DM}}(g_1(S)) = \{g_2(S)\}$. Obviously, $g_2(S)$ is a 3D-matching for $g_1(S)$. To show uniqueness of the solution $g_2(S)$, assume to the contrary that $g_1(S)$ has another 3D-matching M. The triple (x_1^+, y_1^+, z_1) has to be contained in M since it is the only element of $f(S) \setminus \{(x_1^+, y_2^+, z_1^+)\}$ that contains the character x_1^+. Thus, the element y_1^+ is already covered and therefore $(x_\ell^+, y_1^+, z_\ell^+) \notin M$. Considering the element x_ℓ^+ we conclude that $(x_\ell^+, y_\ell^+, z_\ell) \in M$. Repeated use of this argument yields that $M' = M$, a contradiction.

Furthermore the set $ad\&rm(g_1(S), g_3(S))$ and the set $rm(f_1(S), f_3(S))$ from the proof of Theorem 9.31 are identical. Regarding the latter set we already have shown in the proof of Theorem 9.31 that

$$S \in \text{3DM} \Leftrightarrow rm(f_1(S), f_3(S)) \in \text{3DM}.$$

The assertion follows. □

Next, we show an analog to Theorem 4.25, that is, we show that for the modification problem (rm^q, V_{3DM}), where $q(n) = \sqrt[K]{n}$ and $K \in \mathbb{N} \setminus \{0\}$, the existence of useful hints is not likely. We could proceed similar to the case of HC where we gave a reduction that had the necessary properties in order to be applicable to Theorem 4.27. Such a reduction can easily be found by altering a well known reduction Ex3SAT $\leq$ 3DM from [GJ79]. Nevertheless, we demonstrate how to prove this result directly, as the proof demonstrates a more general idea.

Theorem 9.33. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction, $K \in \mathbb{N} \setminus \{0\}$, and $q(n) = \sqrt[K]{n}$. Then*

$$(rm^q, V_{\text{3DM}}) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_{\text{3DM}}) \Rightarrow \text{NP} \subseteq \mathcal{C}/poly.$$

Proof. Let $\mathcal{C}$, K, and q be as stated above and let $(rm^q, V_{\text{3DM}}) \in \mathcal{C}_{\text{MOD}}^{\in}/cert(V_{\text{3DM}})$ via the certificate function h for V_{3DM} and the predicate $C \in \mathcal{C}$, that is, for all 3DM-instances S, all $k \leq q(|S|)$, and every series $m = (m_1, ..., m_k)$ of triples from S it holds that

$$S \in \text{3DM} \Rightarrow \left[rm^q(S, m) \in \text{3DM} \Leftrightarrow (S, h(S), m) \in C\right].$$

We show that under these assumptions 3DM $\in \mathcal{C}/poly$.

Recall the definition of S_i, $i = 1, 2, 3$, given at the beginning of this chapter and let $\ell = |S_1| = |S_2| = |S_3|$. We consider a coding of 3DM-instances such that $|S| = |S'|$ for all well formed sets S and S' with $|S_1| = |S_1'|$. This can be achieved by coding S via $0/1$ entries in a three-dimensional array of dimension $\ell \times \ell \times \ell$. We also want to pad these 3DM-instances, such that the conditions posed in the proof of Theorem 4.25, as restated below, are satisfied. Namely, these conditions can be restated for 3DM-instances by posing that there exists a function $pad_r \in$ FP such that for all $S \in \Sigma^*$ it holds that

- $pad_r \in$ FP,
- $|pad_r(S)| \geq |S|^K$,
- $S \in$ 3DM $\Leftrightarrow pad_r(S) \in$ 3DM,
- $S \setminus M \in$ 3DM $\Leftrightarrow pad_r(S) \setminus M \in$ 3DM, for all $M \subseteq S$.

Such a padding function pad_r does exist, e.g., by adding to S the triples (a_i, a_i, a_i), $\ell + 1 \leq i \leq |S|^K$, where $a_i \notin S_1 \cup S_2 \cup S_3$.

Let S be some well formed 3DM-instance with $|S_1| = \ell$. Let $K_\ell := S_1 \times S_2 \times S_3$ denote the 3DM instance that contains all possible triples over the coordinates from S. Note that $|S| = |K_\ell|$ and that $|K_i| \neq |K_j|$, for $i \neq j$, for our coding of 3DM-instances. Furthermore $K_\ell \in$ 3DM and therefore for all $m \in \Sigma^*$ it holds that

$$rm^q(K_\ell, m) \in \text{3DM} \Leftrightarrow (K_\ell, h(K_\ell), m) \in C.$$

Let $\overline{S}$ denote the missing triples in S, i.e., $\overline{S} = K_\ell \setminus S$. If we define

- $h'(|K_\ell|) := h(pad_r(K_\ell))$[5] and
- $(S, \omega) \in C' \Leftrightarrow (pad_r(K_\ell), \omega, \overline{S}) \in C$

[5]Note that the elements of S_1, S_2, and S_3 may be thought of as the natural numbers $\{1, ..., \ell\}$. Therefore K_ℓ is not dependent on the structure of the sets S_1, S_2, S_3 but only on their size ℓ.

then obviously $h' \in poly$ and since $\mathcal{C}$ is closed under $\leq_m^p$-reduction also $C' \in \mathcal{C}$. Now, the following equivalences hold

$$\begin{aligned} S \in \text{3DM} &\Leftrightarrow rm^{id}(K_\ell, \overline{S}) \in \text{3DM} \\ &\Leftrightarrow rm^q(pad_r(K_\ell), \overline{S}) \in \text{3DM} \\ &\Leftrightarrow (pad_r(K_\ell), h(pad_r(K_\ell)), \overline{S}) \in C \\ &\Leftrightarrow (S, h(pad_r(K_\ell))) \in C' \\ &\Leftrightarrow (S, h'(|K_\ell|)) \in C' \\ &\Leftrightarrow (S, h'(|S|)) \in C'. \end{aligned}$$

This shows that $\text{3DM} \in \mathcal{C}/poly$. □

This last result can easily be adapted to also hold for modification problems (c^q, V_A) in which all instances of the same size can be obtained by applying the modification c^{id} to a fixed instance $K_\ell \in A$. For the problem HC for instance, K_ℓ is given by a complete graph with ℓ vertices, from which every other graph with ℓ vertices can be obtained by edge deletions.

Theorem 9.34. *Let $\mathcal{C}$ be closed under $\leq_m^p$-reduction and V'_{3DM} be a verifier for* HC. *Then* $(rm, V'_{\text{3DM}}) \in \text{coNP}^{\in}_{\text{MOD}}/(cert(V'_{\text{3DM}}) \cap fun \cdot \text{NP}) \Rightarrow \{1\}\text{P} \subseteq \text{NP}$.

Proof. We refer to the already mentioned reduction Ex3SAT $\leq_m^p$ 3DM from [GJ79]. It is easy to derive from this reduction a reduction 3SAT $\leq$ 3DM via $f_1, f_3 \in \text{FP}$ such that

- $F \in \text{3SAT} \Leftrightarrow f_1(F) \in \text{3DM}$ and
- $ad^q(F, m) \in \text{3SAT} \Leftrightarrow rm^q(f_1(F), f_3(m)) \in \text{3DM}$.

The rest of the proof is similar to the proof of Theorem 9.15. □

No solution as promise

Theorem 9.35. 3DM $\leq_{pi}^p$ $(ad, \text{3DM})$.

Proof. Let S be a collection of triples and let $x \in S_1$, $x' \notin S_1$, $y' \notin S_2$, and $z' \notin S_3$. We define

- $f_1(S) := S \cup \{(x, y', z')\}$ and
- $f_2(S) := (x', y', z')$.

The set $f_1(S)$ is not well-formed, hence $f_1(S) \notin \text{3DM}$. The assertion follows from the obvious equivalence

$$S \in \text{3DM} \Leftrightarrow S \cup \{(x, y', z'), (x', y', z')\} \in \text{3DM}.$$

□

Theorem 9.36. $3\mathrm{DM} \leq_{pi}^{p} (rm, 3\mathrm{DM})$.

Proof. Let S be a collection of triples and let $x \in S_1$, $y' \notin S_2$, and $z' \notin S_3$. Let

- $f_1(S) := S \cup \{(x, y', z')\}$,
- $f_2(S) := (x, y', z')$.

The set $f_1(S)$ is not well-formed, hence $f_1(S) \notin 3\mathrm{DM}$. Also, $rm(f_1(S), f_2(S)) = S$. The assertion follows immediately. $\square$

9.9 Partition

For the problem Partition we depart form the classical problem definition given in [GJ79], which uses finite sets and size functions. We do so for description-simplicity — the definition from [GJ79] is slightly intricate for our purposes. Instead, we use sequences of natural numbers as instances for Partition.

$$\begin{aligned}\textsc{Partition} := \big\{(a_1, ..., a_n) \in (\mathbb{N} \setminus \{0\})^n \ : \ n \in \mathbb{N} \text{ and there exists} \\ I \subset \{1, ..., n\} \text{ with } \sum_{i \in I} a_i = \sum_{i \in \{1,...,n\} \setminus I} a_i \ \big\}.\end{aligned}$$

We use the following verifier $V_{\textsc{Partition}}$ for the problem Partition:

$$\begin{aligned}(S, I) \in V_{\textsc{Partition}} \quad \Leftrightarrow \quad & S = (a_1, ..., a_n) \in (\mathbb{N} \setminus \{0\})^n \ \wedge \ I \subset \{1, ..., n\} \\ & \wedge 1 \in I \ \wedge \ \sum_{i \in I} a_i = \sum_{i \in \{1,...,n\} \setminus I} a_i.\end{aligned}$$

The condition $1 \in I$ is a necessary condition for excluding the trivial second partitions $\{1, ..., n\} \setminus I$ as a certificate; an instance S shall not have two solution when there really is only one.

We consider the modification functions ad and rm that add or delete a single natural number from a given sequence, respectively. Formally,

- $ad((x_1, ..., x_n), y) := (x_1, ..., x_n, y)$ and
- $rm((x_1, ..., x_n), y) := \begin{cases} (x_1, ..., x_{i-1}, x_{i+1}, ..., x_n), & \text{if } y = x_i, \\ (x_1, ..., x_n), & \text{otherwise.} \end{cases}$

Arbitrary solution as hint

Theorem 9.37. Partition $\leq^p_m$ $\text{MOD}_{ad}V_{\textsc{Partition}}$.

Proof. Let $S = (a_1, ..., a_n)$. We define

- $f_1(S) := (a_1, ..., a_n, \sum_{i=1}^n a_i)$,
- $f_2(S) := \{1, ..., n\}$, and
- $f_3(S) := \sum_{i=1}^n a_i$.

Obviously, $\big(f_1(S), f_2(S)\big) \in V_{\textsc{Partition}}$. Consequently, it remains to show that

$$S \in \textsc{Partition} \Leftrightarrow ad\big(f_1(S), f_3(S)\big) \in \textsc{Partition}.$$

To show sufficiency, suppose that I is a certificate for S. Thus, $I \cup \{n + 1\}$ is a certificate for $ad\big(f_1(S), f_3(S)\big) = (a_1, ..., a_n, \sum_{i=1}^n a_i, \sum_{i=1}^n a_i)$.

Conversely, assume that I is a certificate for $(a_1, ..., a_n, \sum_{i=1}^n a_i, \sum_{i=1}^n a_i)$. The sum of the sequence $(a_1, ..., a_{n+2})$ is $3 \cdot \sum_{i=1}^n a_i$. It follows that

$$\sum_{i \in I} a_i = \sum_{i \in \{1,...,n+2\} \setminus I} a_i = \frac{3}{2} \sum_{i=1}^n a_i.$$

Thus, it is impossible that both $n+1$ and $n+2$ are contained in I or $\{1, ..., n+2\} \setminus I$. W.l.o.g. generality assume that $n+1 \in I$ and $n+2 \in \{1, ..., n+2\} \setminus I$. Thus,

$$\sum_{i \in I \setminus \{n+1\}} a_i = \sum_{i \in \{1,...,n\} \setminus I} a_i,$$

which implies that $S = (a_1, ..., a_n) \in$ PARTITION. □

Theorem 9.38. PARTITION $\leq_m^p$ MOD$_{rm}V_{\text{PARTITION}}$.

Proof. We use the same reduction functions as in the proof of Theorem 9.37:

- $f_1(S) := (a_1, ..., a_n, \sum_{i=1}^n a_i)$,
- $f_2(S) := \{1, ..., n\}$,
- $f_3(S) := \sum_{i=1}^n a_i$,

Obviously, $\big(f_1(S), f_2(S)\big) \in V_{\text{PARTITION}}$. Furthermore, $rm\big(f_1(S), f_3(S)\big) = S$. The desired result is immediate. □

Selected solution as hint

The problem PARTITION seems especially susceptible to slight modifications. Not only that both adding and deleting an element from the sequence S yield NP-complete problems MOD$_c V_{\text{PARTITION}}$, but we can also prove that generally *no* solution yields a helpful hint in this case. This can be proven by hint-independent reductions from PARTITION to $(ad, V_{\text{PARTITION}})$ and $(rm, V_{\text{PARTITION}})$, respectively. The reductions given in the proofs of Theorems 9.37 and 9.38 are hint-independent reductions since $V_{\text{PARTITION}}(f_1(S)) = \{f_2(S)\}$ in both cases.

Corollary 9.39. PARTITION $\leq_{hi}^p (c, V_{\text{PARTITION}})$ *for* $c \in \{ad, rm\}$.

No solution as promise

Theorem 9.40. PARTITION $\leq_{pi}^p (rm, \text{PARTITION})$.

Proof. Let $S = (a_1, ..., a_n)$. We obtain the desired $\leq_{pi}^p$-reduction by defining

- $f_1(S) = \big(a_1, ..., a_n, (\sum_{i=1}^n a_i) + 1\big)$ and
- $f_2(S) = (\sum_{i=1}^n a_i) + 1$.

□

The reduction functions f_1 and f_2 from the last proof can also be used to show that PARTITION $\leq_{pi}^p (ad, \text{PARTITION})$.

Chapter 10

Appendix B - Optimization Problems

10.1 MAXIMUMTRAVELLINGSALESPERSON (MAXTSP)

We consider the problem MAXTSP as defined in Section 7.4. Without reoptimization, the best known approximation result for MAXTSP is a $4/3$-approximation [Ser84]. In contrast to MINTSP, the general MAXTSP problem benefits from reoptimization. It has been shown in [AEMP06] that MOD$_{adv}$MAXTSP is approximable with ratio $\frac{5}{4}$ (for the definition of adv see Section 7.4). In thesis we show that MOD$_{inc}$MAXTSP is $5/4$-approximable. Before we prove this result, we introduce some additional notation.

Definition 10.1 ([LP86]). *Let G be a graph and $f : V(G) \rightarrow \mathbb{N}$. An f-factor of G is a subgraph H of G such that $deg_H(v) = f(v)$, for all $v \in V(G)$.*

When f is a constant function, i.e., $f(v) = k$ for all $v \in V(G)$ and some fixed $k \in \mathbb{N}$,we get the notion of a k-factor. Note that a 1-factor of G is a perfect matching of G. A 2-factor of G is a partition of G into node disjoint cycles, or short, a cycle cover.

Lemma 10.2. *Let G be a weighted graph and $P = (p_1, ..., p_m)$, $m \geq 2$ a constant, be a path in G. There is a polynomial-time algorithm that finds among all 2-factors of G that contain the path P a 2-factor of maximum weight.*

Proof. A maximum weight 2-factor for G that respects a given path $(p_1, ..., p_m)$ is induced by a maximum weight f-factor for G, where

$$f(x) = \begin{cases} 0, & \text{if } x \in \{p_2, ..., p_{m-1}\}, \\ 1, & \text{if } x \in \{p_1, p_m\}, \\ 2, & \text{if } x \in V(G) \setminus \{p_1, ..., p_m\}. \end{cases}$$

Chapter 10.1. of [LP86] contains a reduction function, call it g, such that for any graph G it holds that G has an f-factor if and only if $g(G)$ has a perfect matching. We can alter this reduction to yield a similar statement for weighted graphs. Utilizing an algorithm from [Gab74] for finding a maximum weight perfect matching, we obtain an $O(n^3)$ algorithm for finding a maximum weight f-factor of G. □

We are now prepared to prove our main result of this section.

Theorem 10.3. MOD$_{inc}$MAXTSP *is* $5/4$*-approximable.*

Proof. Let G^o denote the original instance and let T^o_{opt} be a maximum tour for G^o. Let $G^m := inc(G^o, (e, i))$ denote the modified graph, $e = \{u, v\}$, and let T^m_{opt} be a maximum tour for G^m. Note that $cost(G^o, T^o_{opt}) \geq cost(G^m, T^m_{opt}) - i$, since otherwise T^o_{opt} was not an optimum tour for G^o. Also, we assume that $e \notin T^o_{opt}$ and $e \in T^m_{opt}$, otherwise T^o_{opt} is an optimum tour in G^m and is chosen as output when compared to other solutions that are obtained in the coming.

1. Case: $|V(G^o)|$ is even: From T^o_{opt} we can, in an obvious way, obtain a perfect matching M with $cost(G^o, M) \geq cost(G^o, T^o_{opt})/2 \geq (cost(G^m, T^m_{opt}) - i)/2$. By adding the edge e to M we obtain a set M' that contains a path of length 3 and with $cost(G^m, M') \geq (cost(G^m, T^m_{opt}) + i)/2$.

Now, consider a 2-factor $F = (C_1, ..., C_\ell)$ of G^m such that (a) e is contained in C_1, (b) $|C_1| \geq 5$, and (c) F is of maximum weight among all 2-factors of G^m that satisfy (a) and (b). We can find such a 2-factor in polynomial-time by constructing a maximum weighted 2-factor for G^m that contains the path (r, s, t, u, v) (see Lemma 10.2), for all possibilities of expanding e to a cycle-free path of length four, and selecting the costliest of these 2-factors. Since $e \in T^m_{opt}$ we have that $cost(G^m, F) \geq cost(G^m, T^m_{opt})$.

Applying the method of Serdyukov [Ser84], we iteratively, for $p = 1, ..., \ell$, delete an edge from C_p and add this edge to M' such that the modified set M' is still a union of paths. For $C_1 = (r, s, t, u, v, ...)$ this is possible since r, s, t are endpoints of a path in M' (only u and v are no endpoints) but only two of them can be endpoints of the same path. Hence, one of $\{s, r\}$ or $\{s, t\}$ can be added to M'. For all other cycles C_p, this is possible since only vertices from already processed cycles C_j, $1 \leq j < p$, can have degree 2 in M'. Thus, all (of at least 3) vertices of C_p are endpoints of some path in M' but only two of them can be endpoints of the same path.

By this procedure, the 2-factor F and the set of edges M' are transformed into two sets of paths P_1 and P_2 that satisfy

$$\begin{aligned} cost(G^m, P_1) + cost(G^m, P_2) &= cost(G^m, M') + cost(G^m, F) \\ &\geq (cost(G^m, T^m_{opt}) + i)/2 + cost(G^m, T^m_{opt}). \end{aligned}$$

By taking the costlier of P_1 and P_2 we get a set of paths with cost larger than $\frac{3}{4}(cost(G^m, T^m_{opt}) + \frac{i}{4}$. Let T denote the completion of this partial tour to a cycle in G^m; since G^m is a complete graph T always exists.

Now for $i \leq \frac{1}{5}cost(G^m, T^m_{opt})$ the solution T^o_{opt} yields a $5/4$-approximation, for $i > \frac{1}{5}cost(G^m, T^m_{opt})$ the tour T yields such a bound.

2. Case: $|V(G^o)|$ is odd**:** We construct from T^o_{opt} a set of paths M with at most one path of length 2 such that (a) $cost(G^o, M) \geq cost(G^o, T^o_{opt})/2$, (b) inserting e into M prolongs the longest path in M, and (c) u and v are no endpoints in $M \cup \{e\}$. Thus, $M' := M \cup \{e\}$ only consists of paths of length 1, with the exception of a single path of length at most 4 containing u and v, and u, v being no endpoints of the path.

Now, consider a maximum cost 2-factor $F = (C_1, ..., C_\ell)$ of G^m that contains e as an edge in C_1 and $|C_1| \geq 8$. Since M' contains at most 3 vertices that are not an endpoint of a path in M', the cycle C_1 contains 3 consecutive vertices that are endpoints in M'. The rest of the proof translates mutatis mutandis from the case $|V(G^o)|$ is even.

□

A similar result for the modification dec has not been found.

Open problem 6. *Does* MaxTSP$_\Delta$ *benefit from reoptimization when the modification is* dec*, i.e., is* MOD$_{dec}$MaxTSP$_\Delta$ δ*-approximable for some* $\delta < \frac{4}{3}$*?*

In analogy to the proof of Theorem 7.13 we are able to show the following non-approximability result.

Theorem 10.4. *There is no* FPTAS *for* (inc, MaxTSP) *and* (dec, MaxTSP)*, unless* $\text{P} = \text{NP}$.

For a proof we refer to the coming Theorem 10.8, in which it is shown that already some restricted versions of (dec, MaxTSP) and (inc, MaxTSP) have no FPTAS.

10.2 MaximumMetricTravellingSalesperson (MaxTSP$_\triangle$)

We consider the problem MaxTSP$_\triangle$ as defined in Section 7.5. The problem MaxTSP$_\triangle$ is approximable with ratio $\frac{8}{7}$ when no hint is given [CN07]. It has been shown in [AEMP06] that MOD$_{adv}$MaxTSP$_\triangle$ admits a PTAS.

In this thesis, we prove that there exists a PTAS for MOD$_{dec_\triangle}$MaxTSP$_\triangle$. Therefore, we use the following theorem about approximability of alternative solutions for MaxTSP$_\triangle$ — a result interesting in its own right.

Theorem 10.5. *Let G be a* MaxTSP$_\triangle$*-instance, T_{opt} be a maximum tour in G, and e be an edge of T_{opt}. We can find in polynomial-time a tour T' such that e does not belong to T' and $cost(G, T') \geq \left(1 - \frac{2}{|V(G)|-2}\right) cost(G, T_{opt})$.*

Proof. Let $T_{opt} = (v_1, ..., v_n)$ be a maximum tour of $G = (K_n, w)$. We assume without loss of generality that $e = \{v_n, v_1\}$. First, we prove that there exist three consecutive vertices v_{i-1}, v_i, and v_{i+1} in T^o_{opt} such that

$$w(\{v_{i-1}, v_i\}) + w(\{v_i, v_{i+1}\}) \leq \frac{2}{n-2} cost(G, T_{opt}).$$

Assume to the contrary that $w(\{v_{i-1}, v_i\}) + w(\{v_i, v_{i+1}\}) > \frac{2}{n-2} cost(G, T_{opt})$, for all $2 \leq i \leq n-1$. But then, by summarizing the weights of the paths $(v_{2i-1}, v_{2i}, v_{2i+1})$, $1 \leq i \leq \lfloor (n-1)/2 \rfloor$ in T_{opt} we get that $cost(G, T_{opt}) > (\frac{n-1}{2} - \frac{1}{2}) \cdot \frac{2}{n-2} \cdot cost(G, T_{opt})$, a contradiction.

By deleting the edges $\{v_n, v_1\}$, $\{v_{i-1}, v_i\}$, and $\{v_i, v_{i+1}\}$ from T_{opt} and inserting the edges $\{v_n, v_i\}$, $\{v_i, v_1\}$, and $\{v_{i-1}, v_{i+1}\}$ we obtain a tour T' that does not contain the edge e. Because of the triangle inequality the cost of the tour T_{opt} increases when taking the path (v_n, v_i, v_1) instead of the shortcut (v_n, v_1). Thus, T' is shortened by at most $w(\{v_{i-1}, v_i\}) + w(\{v_i, v_{i+1}\})$ compared to T_{opt}. □

Corollary 10.6. MOD$_{dec_\triangle}$MaxTSP$_\triangle$ *has a* PTAS.

Proof. We may assume that the modified edge e does belong to a maximum tour T^o_{opt} in the original graph G^o, but not to a maximum tour T^m_{opt} in the modified graph G^m. Let $\varepsilon > 0$. In case that $\varepsilon \geq \frac{2}{|V(G)|-2}$ Theorem 10.5 yields a solution T' with $cost(G^o, T') \geq (1-\varepsilon) cost(G^o, T^o_{opt})$. The assertion follows from the facts that $cost(G^o, T^o_{opt}) \geq cost(G^m, T^m_{opt})$ and that $cost(G^o, T') = cost(G^m, T')$. In case that $\varepsilon < \frac{2}{|V(G)|-2}$ we perform a brute force search for a maximum tour in G^m. □

Note that the proof of the above Theorem does not use the assertion that the modified graph G^m satisfies the triangle inequality. Thus, we also have a PTAS for the more general modification dec that might output modified instances that violate the triangle inequality.

Theorem 10.7. MOD$_{inc_\triangle}$MaxTSP$_\triangle$ *has a* PTAS.

Proof. Let $\varepsilon \geq 0$ and G be a MAXTSP$_\Delta$-instance. If $\varepsilon < \frac{2}{|V(G)|-2}$ we perform a brute force search for an optimum solution in G^m. In the other case, $\varepsilon \geq \frac{2}{|V(G)|-2}$, we output the old solution T^o_{opt}. This suffices since $cost(G^m, T^o_{opt}) \geq (1-\varepsilon)cost(G^m, T^m_{opt})$.

To see this, take an optimum tour T^m_{opt} of G^m. If e is contained in T^m_{opt} we use Theorem 10.5 to get a tour T' that does not contain e and for which

$$cost(G^m, T') \geq \left(1 - \frac{2}{|V(G)|-2}\right) cost(G^m, T^m_{opt}) \geq (1-\varepsilon)cost(G^m, T^m_{opt}).$$

If e is not contained in T^m_{opt} we set $T' := T^m_{opt}$. Since the modified edge e is not contained in T' and since T^o_{opt} is maximum in G^o we have

$$cost(G^m, T') = cost(G^o, T') \leq cost(G^o, T^o_{opt}) \leq cost(G^m, T^o_{opt}).$$

The assertion follows immediately. □

Theorem 10.8. *There is no* FPTAS *for* $(inc_\Delta,$ MAXTSP$_\Delta)$ *and* $(dec_\Delta,$ MAXTSP$_\Delta)$.

Proof. The proof is similar to the proof of Theorem 7.13, except that we assign the weights 2, 3, and 4 to the edges, in order to satisfy the triangle inequality. □

10.3 MINIMUMSTEINERTREE (MINST)

The problem MINST is defined as follows:

PROBLEM: MINST

INSTANCE: A complete graph $G = K_n$, a function $w : E(G) \to \mathbb{N}$ assigning a weight to each edge of G, and a subset $S \subseteq V(G)$ of so called terminal vertices.

SOLUTION: A Steiner tree, i.e., a subtree of G that includes all the vertices in S.

MEASURE: The sum of the weights of the edges in the subtree.

The nonterminal vertices of G are called Steiner vertices. The problem MINST is approximable with factor $(1 + \frac{ln3}{2}) \approx 1.55$ [RZ05] and is APX-complete [BP89].

A Steiner tree instance G can be transformed to an equivalent Steiner tree instance $\Delta(G)$ that satisfies the triangle inequality as follows. For every edge e of G we search for a shortest path in G that connects the two endpoints of e and modify the weight of e to be the weight of this shortest path. Let $\Delta(G)$ denote the resulting graph, which clearly satisfies the triangle inequality. Now a Steiner tree solution T for $\Delta(G)$ can be transformed into a Steiner tree solution for G of at most the same weight by expanding every edge e of T into the shortest paths between the endpoints of e in G.

In this thesis we study the problem of reoptimization when changing the weight of single edges. The corresponding modification functions inc and dec are defined as usual. But, the decrease of a single edge in G might lead to the decrease of several edges in the graph $\Delta(G)$, as there could be several new shortest paths in G between the endpoints of some edges. Since we are mainly interested in very *small* modifications of the original instance we also consider a modification function dec_Δ that does not allow an edge decrease in G that would lead to multiple edge decreases in $\Delta(G)$. The respective modification function dec_Δ is defined as follows

$$dec_\Delta(G,(e,i)) := \begin{cases} dec(G,(e,i)), & \text{if } \Delta(G) \text{ and } \Delta(dec(G,(e,i))) \text{ differ} \\ & \text{in the weight of at most one edge,} \\ \text{trivial instance}, & \text{otherwise.} \end{cases}$$

A modification function inc_Δ can be defined in the same way.

The problem of *reoptimizing* MINST-instances has already been discussed in literature. In [EMP07] it is shown, that the problem MOD_{adv}MINST is $\frac{3}{2}$-approximable (the added vertex may be a terminal or a Steiner vertex). In [BHK+] the modification consists of changing the type of a single vertex from being a Steiner vertex to being a nonterminal vertex, and vice versa, and a 3/2-approximation is given for this problem. Just recently, in [BBH+08] these last results have been improved to 1.33 for the augmentation of the terminal set and 1.4 for restricting the terminal set. Also in [BBH+08] are the following results.

Theorem 10.9 ([BBH+08]).

1. MOD_{inc}MINST *is* $\frac{4}{3}$*-approximable and*

2. $\mathrm{MOD}_{dec_\Delta}\textsc{MinST}$ *is* 1.3*-approximable.*

We complement these results, by giving lower bounds on the reoptimizability of $\textsc{MinST}$ when the modification is dec_Δ or inc_Δ.

Theorem 10.10. *Unless* $\mathrm{P} = \mathrm{NP}$, $(dec_\Delta, \textsc{MinST})$ *has no* FPTAS.

Proof. Assume that $(dec_\Delta, \textsc{MinST})$ has an FPTAS A. We show that SAT $\in \mathrm{P}$ under this assumption. In a nutshell, the proof is as follows: We will construct in a series of transformations from a formula F a $\textsc{Clique}$-instance G', from G' a VC-instance $\overline{G'}$, and from $\overline{G'}$ a $\textsc{MinST}$-instance G^o that has exactly one optimum solution. We will then modify G^o to G^m such that containment of F in SAT can be decided with help of the FPTAS A.

Let $F = \{C_1, ..., C_m\}$ be a Boolean formula. We assume the reader to be familiar with the construction and the notations from Section 9.5, in particular with the subsection 'Selected solutions as hint'. Application of the reduction function f from [Kar72], which is described in Section 9.5, gives a $\textsc{Clique}$-instance $f(F)$. Let $v_1, ..., v_n$ be the vertices of $f(F)$. To $f(F)$ we add another vertex z that is connected to all vertices from $f(F)$. Also we add a separated complete graph over $m + 1$ new vertices $u_1, ..., u_{m+1}$. The resulting graph G'

- has an $m + 1$-clique $C^* = \{u_1, ..., u_{m+1}\}$,
- has no $m + 2$-clique,
- has another $m + 1$-clique if and only if F is satisfiable, and
- every such additional $m+1$-clique contains the vertex z and does not contain any of the vertices $\{u_1, ..., u_{m+1}\}$.

We leave a formal proof of these simple facts to the reader. Using the standard reduction for $\textsc{Clique} \leq^p_m$ VC (see [GJ79] for example) we obtain from G' a graph $\overline{G'} = (V(G'), \overline{E'})$ such that

(P1) $\overline{G'}$ has a vertex cover $D^* := \{v_1, ..., v_n, z\}$ of size $m' := |V(G')| - (m + 1)$,

(P2) $\overline{G'}$ has no vertex cover of size $m' - 1$,

(P3) $\overline{G'}$ has another vertex cover of size m' if and only if F is satisfiable, and

(P4) every such additional vertex cover of size m' does not contain the vertex z and contains the vertices $\{u_1, ..., u_m\}$.

See Figure 10.1 for an illustration of the graph $\overline{G'}$. Let q denote the number of edges of $\overline{G'}$, i.e, $q := |\overline{E'}|$.

From $\overline{G'}$ we construct a Steiner tree instance G^o with weight function w in the following way. Let $e_1, ..., e_q$ denote the edges in $\overline{E'}$. W.l.o.g. let $e_j = \{z, u_j\}$, $1 \leq j \leq m+1$.

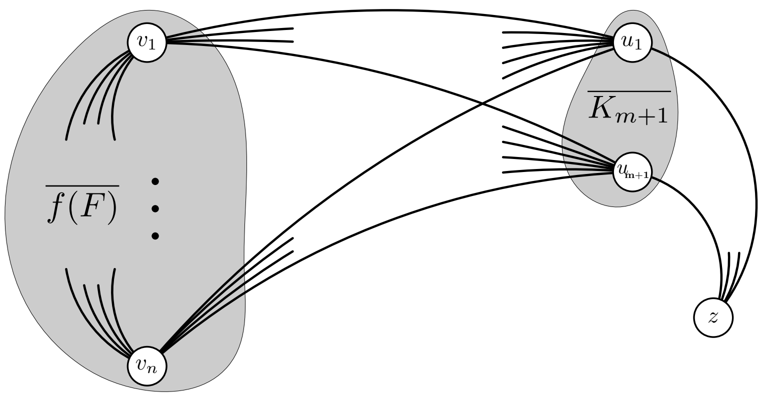

Figure 10.1: The VC-instance $\overline{G'}$.

We also assume that $e_{m+1+j} = \{u_1, v_j\}$, $1 \leq j \leq n$. Every edge e_j, $2 \leq j \leq q$, is replaced by a length-2 path in that the middle vertex is a newly introduced terminal vertex t_j and the two weights $6q + 1$ and $6q + 2$ are assigned to the both edges of the path. Thereby, we assign the smaller value $6q + 1$ to the edge that is incident to one of the vertices $v_1, ..., v_n$ or z and the value $6q + 2$ to the other edge. In case that both of the two edges of the path are incident to a vertex $v_1, ..., v_n$ the two weights $6q + 1$ and $6q + 2$ are allocated arbitrarily between these two edges. The edge $e_1 = \{z, u_1\}$ is also replaced by such a path, but $w(\{z, t_1\}) = 8q + 1$ and $w(\{t_1, u_1\}) = 10q + 1$. Also, an additional terminal vertex a is added. The terminal vertices mentioned are the only terminal vertices of G^o, i.e., $S := \{t_j : 1 \leq j \leq q\} \cup \{a\}$. Beside the already specified edges, the weight of the *other* edges is given by

$$w(\{u,v\}) := \begin{cases} 6q, & \text{if } u = a \text{ and } v \text{ is a Steiner vertex,} \\ 12q+2, & \text{if } u \in S \setminus \{a\} \text{ and } v \text{ is a Steiner vertex,} \\ 12q, & \text{if } u \text{ and } v \text{ are Steiner vertices,} \\ 12q, & \text{if } u = a \text{ and } v \in S \setminus \{a\}, \\ 12q+2, & \text{if } u, v \in S \setminus \{a\}. \end{cases}$$

See also Figure 10.2 for an illustration of G^o.

Given a vertex cover D of $\overline{G'}$ we construct a Steiner tree T for G^o by the following algorithm.

```
Algorithm $InducedSteinerTree(D)$
Input: A vertex cover $D$ for $\overline{G'}$
Output: A Steiner tree $T$ for $G^o$
begin
  1. Choose as Steiner vertices in $T$ the vertices in $D$.
  2. For each terminal vertex $t_j$ in $S \setminus \{a\}$ connect $t_j$ to some Steiner vertex of $T$
     via an edge of weight at most $10q + 1$. If two such connections are possible,
```

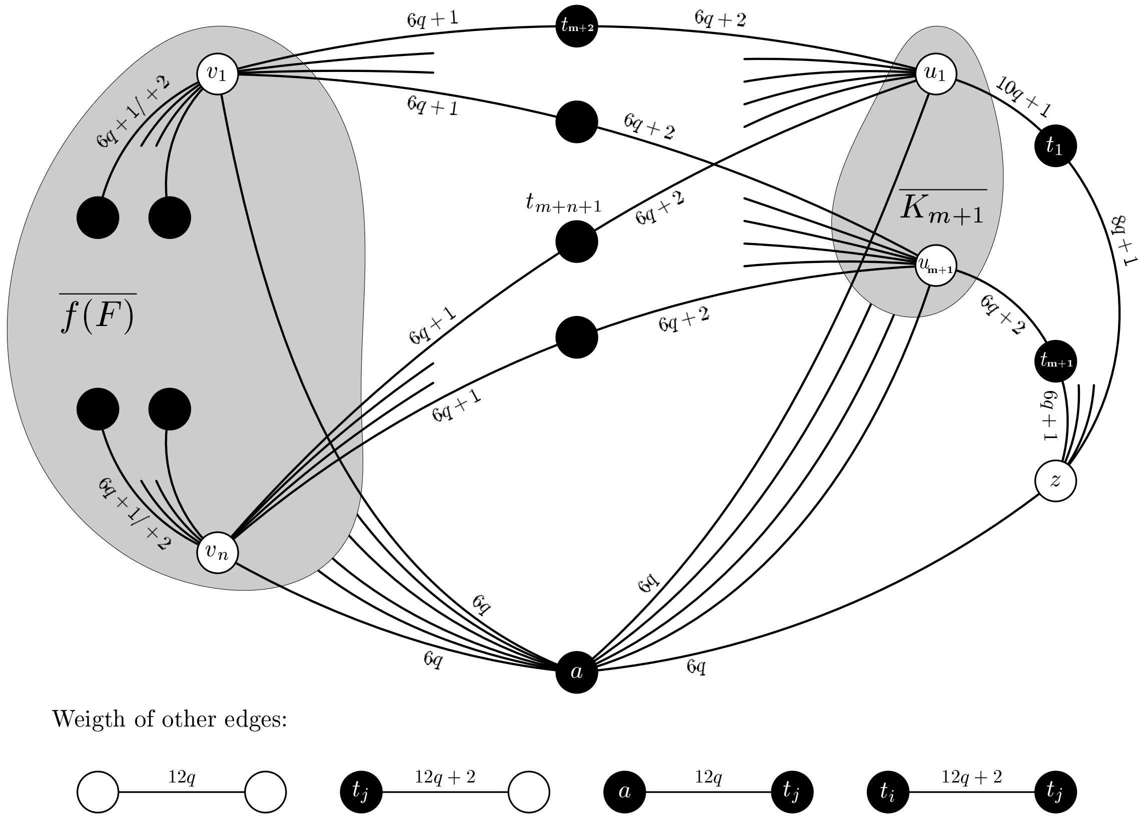

Figure 10.2: The MINST-instance G that is constructed from $\overline{G'}$. Only edges with weight at most $10q + 1$ are shown. White vertices are Steiner vertices, black vertices are terminal vertices.

```
        then choose the cheaper one.
    3. For each Steiner vertex $w \in D$ connect $w$ to $a$ by the edge $\{w, a\}$.
end;
```

We claim that $T^o_{opt} := InducedSteinerTree(D^*)$ is the sole optimum solution in G^o (see Claim (iv) at the end of this proof). We modify G^o by decreasing the weight of the edge $\{t_1, u_1\}$ from $10q + 1$ to $6q + 1$. Note that the original graph and the modified graph both satisfy the triangle inequality, thus dec_Δ is applicable. We claim that the modified graph G^m has a solution T^m with $cost(G^m, T^m) < cost(G^o, T^o_{opt}) - q$ if and only if $\overline{G'}$ has a size m' vertex cover D other than D^* (see Claim (v) at the end of this proof).

Let $\varepsilon := 1/(cost(G^o, T^o_{opt}) + 1)$. The input for our FPTAS A consists of the original graph G^o, its sole optimum solution T^o_{opt}, the modification that turns G^o into G^m, and the error bound ε. Let T^A denote the output of A. Note that $opt(G^m) \leq cost(G^o, T^o_{opt})$,

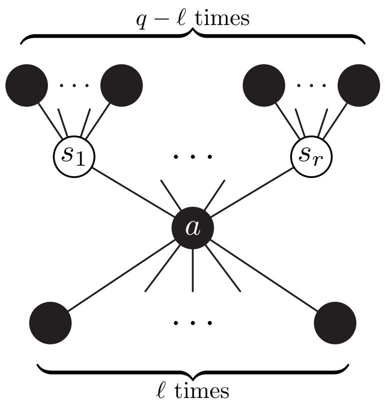

Figure 10.3: A Steiner tree in normal form.

since T^o_{opt} only uses unmodified edges. Now,

$$\begin{aligned} cost(G^m, T^A) &\le (1+\varepsilon)opt(G^m) \\ &\le opt(G^m) + \frac{1}{cost(G^o, T^o_{opt}) + 1} \cdot cost(G^o, T^o_{opt}) \\ &< opt(G^m) + 1. \end{aligned}$$

In other words, A always yields an optimum solution for G^m. Thus, given T^A it is easy to decide whether there exists a Steiner tree T^m with $cost(G^m, T^m) < cost(G^o, T^o_{opt}) - q$, which is the case if and only if $\overline{G'}$ has a size m' vertex cover D other than D^*. The latter is the case if and only if F is satisfiable by (P2). The assertion SAT $\in$ P follows from the fact that all our reductions are computable in polynomial time and our FPTAS A runs in polynomial time as well, since $cost(G^o, T^o_{opt})$, and therefore also $1/\varepsilon$, is bounded by a polynomial in q (see also proof of Claim (iv)).

We conclude by giving proofs for the postponed Claims(iv) and (v). In order to prove these claims, we first establish some auxiliary results.

Claim (i): *A Steiner tree T in G^o or G^m can be transformed to a Steiner tree T' that consists of the same vertices as T, has the same cost (or less) as T, and only uses edges of weight at most $10q+1$ or edges of the form $\{t_j, a\}$. Additionally, we can choose T' such that every terminal vertex in T' has degree 1. We say that T' has normal form (see also Figure 10.3).*

Proof of Claim: First, we show how an edge $\{u, v\}$ with cost greater than $10q+1$ in T can be locally substituted by edges of weight at most $10q+1$ without increasing the weight of the Steiner tree.

Let $\{u, v\}$ be an edge of weight greater than $10q+1$, or equivalently, an edge of weight at least $12q$. The vertex a is a vertex of T, since it is a terminal vertex. By deleting the edge $\{u, v\}$ from T we obtain a forest with two components — one of the components contains u, the other one contains v. If a and u are in the same component the edge $\{a, v\}$ (of weight at most $12q$) is added to connect the two components to a Steiner tree. Otherwise, if a and v are in the same component, we add the edge $\{a, u\}$ (of weight at most $12q$). Thus we may assume that T only contains the above mentioned types of edges.

Next we show how to assure that every terminal vertex in T' has degree 1. This is again done by local transformations on vertices that violate this property. Let t_j be a terminal vertex with degree at least two. Let u and v be two vertices adjacent to t_j. Since we may assume that T only uses the above mentioned normal-form-edges it suffices to consider the following two cases.

Case 1: $u = a$ and v is a Steiner vertex**:** Removal of the edge $\{t_j, v\}$ of weight at least $6q + 1$ and addition of the edge $\{a, v\}$ of weight $6q$ decreases the degree of t_j by one.

Case 2: u and v are Steiner vertices**:** Deletion of $\{u, t_j\}$ (of weight at least $6q + 1$) results in a forest with two components. If u and a belong to the same component we add the edge $\{a, v\}$. Otherwise, if v and a belong to the same component, we add the edge $\{a, u\}$. This reduces the degree of t_j by one.

Repeated application of this procedure leads to the desired tree T'.

Claim (ii): *Let T be a Steiner tree in normal form (see Claim (i)), $s_1, ..., s_r$ be the Steiner vertices of T, and $\{a, t_{i_1}\}, ..., \{a, t_{i_\ell}\}$ be the edges of weight $12q$ in T. Then T induces a vertex cover of size $r + \ell$ in $\overline{G'}$ that uses the vertices $s_1, ..., s_r$. Additionally, for every edge $\{a, t_j\}$ of weight $12q$ in T the vertex for the vertex cover may be arbitrarily chosen from e_j.*

Proof of Claim: This fact is obvious by the construction of G^o from $\overline{G'}$.

Claim (iii): *Let T be a Steiner tree in normal form for G^o (see Claim (i)), r be the number of Steiner vertices in T, ℓ be the number of edges of weight $12q$ in T, and*

$$cost(G^o, T) \leq q(6q + 1) + m' \cdot 6q + 2q. \tag{1}$$

Then $r + \ell = m'$. The same statement holds for the graph G^m.

Proof of Claim: Let T, r, l, and $cost(G^o, T)$ be as above. Note that $r + \ell \geq m'$, since otherwise we could construct from T a vertex cover of size $m' - 1$ for $\overline{G'}$ (see Claim (ii)), which contradicts (P2). Thus, it suffices to show that $r + \ell \leq m'$. Assume to the contrary that $r + \ell \geq m' + 1$. Then

$$\begin{aligned} cost(G^o, T) &\geq (q - \ell)(6q + 1) + r \cdot 6q + \ell \cdot 12q \\ &= q(6q + 1) + 6q(r + \ell) - \ell \\ &\geq q(6q + 1) + 6q(m' + 1) - \ell \\ &> q(6q + 1) + m' \cdot 6q + 2q, \end{aligned}$$

since $4q > \ell$. This contradicts (1) and thus $r + \ell = m'$.

Claim (iv): *T^o_{opt} is the only optimum solution in G^o.*

Proof of claim: First, we examine the structure of the solution T^o_{opt}, which is given by $T^o_{opt} := InducedSteinerTree(D^*)$. Note that since $z \in D^*$ the algorithm $Induced$-$SteinerTree$ connects in its second step t_1 to z via the edge of weight $8q + 1$ and all of the vertices $t_2, ..., t_{m+1}$ to z via an edge of weight $6q + 1$. Since $v_1, ..., v_n \in D^*$ it also connects all of the remaining terminal vertices $t_{m+1}, ..., t_q$ to a Steiner vertex via an edge of weight $6q + 1$. In the third step of the algorithm m' edges of weight $6q$ are added. In summary we get $cost(G^o, T^o_{opt}) = q(6q + 1) + m' \cdot 6q + 2q$.

Let T'_{opt} be an optimum Steiner tree in G^o with $cost(G^o, T'_{opt}) \leq cost(G^o, T^o_{opt})$ other than T^o_{opt}. We may assume that T'_{opt} is in normal form (see Claim (i)). Let r' denote the number of Steiner vertices in T'_{opt} and ℓ' denote the number of weight-$12q$ edges in T'_{opt}. By application of Claim (iii) we have that $r' + \ell' = m'$.

First, we show that z is contained in T'_{opt}. Assume to the contrary, that z is no Steiner vertex in T'_{opt}. Then, there are only two possibilities to make t_1 a vertex of degree 1 in the normal form tree T'_{opt}.

Case 1: $\{a, t_1\}$ **is contained in** T'_{opt}**:** In this case, the edge $\{a, t_2\}$ is no edge in T'_{opt}, since otherwise we could obtain a better tree than T'_{opt} by deleting $\{a, t_1\}$ and $\{a, t_2\}$ and adding the edges $\{a, z\}$,$\{z, t_1\}$ and $\{z, t_2\}$, which contradicts optimality of T'_{opt}. Since z is no Steiner vertex the only possibility to connect t_2 to the tree is to use the edge $\{t_2, u_2\}$. Hence, u_2 is a Steiner vertex of T'_{opt}. But then, T'_{opt} induces a vertex cover of size $r' + \ell' = m'$ in $\overline{G'}$ that uses the vertices u_1 and z. This contradicts (P1) or (P4).

Case 2: $\{u_1, t_1\}$ **is contained in** T'_{opt}**:** Since $w(\{u_1, t_1\}) = 10q + 1$ we have

$$\begin{aligned} cost(G^o, T'_{opt}) &\geq (q - \ell' - 1)(6q + 1) + r'6q + \ell'12q + (10q + 1) \\ &= q(6q + 1) + 6q(r' + \ell') - \ell' + 4q \\ &= q(6q + 1) + m' \cdot 6q - \ell' + 4q \\ &= cost(G^o, T^o_{opt}) + 2q - \ell' \\ &> cost(G^o, T^o_{opt}), \end{aligned}$$

since $2q > \ell'$. This contradicts our initial assumption that $cost(G^o, T'_{opt}) \leq cost(G^o, T^o_{opt})$.

Thus we have shown that z is contained in T'_{opt}. Next, observe that none of the vertices $\{u_1, ..., u_m\}$ may be contained in T'_{opt}. Otherwise T'_{opt} would induce a vertex cover of size $r' + \ell' = m'$ that contains both of z and u_j for some $j \leq m$ (see Claim (ii)). This is impossible since (P1) and (P4). In particular, u_1 is no Steiner vertex of T'_{opt}.

Also, none of the edges $\{a, t_{m+1+j}\}$, $1 \leq j \leq n$, is contained in T'_{opt}, since such an edge would also induce a forbidden vertex cover with z and u_1 (see Claim (ii)).

Consequently, T'_{opt} has no other choice but to use the edges $\{t_{m+1+j}, v_j\}$, $1 \leq j \leq n$. Thus, all the vertices $v_1, ..., v_n$ are Steiner vertices in T'_{opt}.

In consequence, T'_{opt} and T^o_{opt} use the same Steiner vertices. Also, since T'_{opt} is in normal form, T'_{opt} uses m' edges of weight $6q$ to connect its Steiner vertices to a. Just as in T^o_{opt} the terminal vertices $t_1, ..., t_q$ are connected to the Steiner vertices of T'_{opt} via edges of weight $6q + 1$ (or $8q + 1$ for t_1), otherwise T'_{opt} is not optimum. Since only one such edge of weight $6q + 1$ (or $8q + 1$ for t_1) exists for every terminal vertex t_j, $1 \leq j \leq q$, we conclude that $T^o_{opt} = T'_{opt}$.

Claim (v): *There exists a Steiner tree of size at most $cost(G^o, T^o_{opt}) - q - 1$ in G^m if and only if there exists a size m' vertex cover in $\overline{G'}$ other than D^*.*

Proof of claim: Note that we have already shown in Claim (iv) that $cost(G^o, T^o_{opt}) = q \cdot (6q + 1) + m' \cdot 6q + 2q$.

To show sufficiency, let T be a Steiner tree with $cost(G^m, T) < q\cdot(6q+1)+m'\cdot 6q+q$. We assume that T is given in normal form (see Claim (i)). Let r denote the number of Steiner vertices in T and ℓ denote the number of terminal vertices that are directly connected to a. By Claim (ii) we have $l + r = m'$.

We show that T induces a vertex cover of size m' that uses the vertex u_1. Therefore, it suffices to show that u_1 is a Steiner vertex of T (see Claim(ii)). Assume to the contrary that u_1 is no Steiner vertex in T. Then, there are only two possibilities how the terminal vertex t_1 is connected to T.

Case 1: $\{a, t_1\}$ is contained in T**:** We may assume that the edge $\{a, t_2\}$ is no edge in T, since otherwise we could obtain a better tree than T by deleting $\{a, t_1\}$ and $\{a, t_2\}$ and adding the edges $\{a, z\}$,$\{z, t_1\}$, and $\{z, t_2\}$. Thus, either $\{z, t_2\}$ or $\{t_2, u_2\}$ is an edge of T. In either case, we can construct from T a vertex cover of size $l + r = m'$ for $\overline{G'}$ that includes z and one of the vertices u_1 or u_2. This contradicts (P1) or (P4).

Case 2: $\{z, t_1\}$ is contained in T**:** Since $w\{z, t_1\} = 8q + 1$ we have

$$\begin{aligned} cost(G^m, T) &\geq (q - \ell - 1)(6q + 1) + r \cdot 6q + \ell \cdot 12q + (8q + 1) \\ &= q(6q + 1) + 6q(r + \ell) - \ell + 2q \\ &= q(6q + 1) + m' \cdot 6q - \ell + 2q \\ &\geq q(6q + 1) + m' \cdot 6q + q, \end{aligned}$$

since $l \leq q$. This contradicts our initial assumption that $cost(G^m, T) < q(6q + 1) + m' \cdot 6q + q$.

Thus, u_1 is a Steiner vertex of T. By Claim (iii) the Steiner tree T induces a vertex cover of size $l + r = m'$ for $\overline{G'}$ that uses the vertex u_1, therefore being unlike D^*.

To show necessity, let $D \neq D^*$ be a vertex cover of size m' for $\overline{G'}$. By (P4) the vertex cover D contains the vertex u_1. Thus,

$$\begin{aligned} cost(G^m, InducedSteinerTree(D)) &\leq (q-1)(6q+2) + (6q+1) + m' \cdot 6q \\ &= q(6q+1) + m' \cdot 6q + q - 1. \end{aligned}$$

□

The last proof can be adapted to show that $(inc_\Delta, \text{MINST})$ is unlikely to have a FPTAS.

Corollary 10.11. *Unless* $\text{P} = \text{NP}$, $(inc_\Delta, \text{MINST})$ *has no* FPTAS*.*

Proof. The construction of G^o follows the proof of Theorem 10.10, with the exception that $w(\{z, t_1\}) = 6q + 1$ and $w(\{t_1, u_1\}) = 8q + 1$. The modification consists of the increase of the weight of $\{z, t_1\}$ from $6q + 1$ to $10q + 1$. The rest of the proof is similar. □

10.4 MINIMUMMAXIMALINDEPENDENTSET (MINMAXIS)

We already mentioned in Section 7 that many optimization problems have an approximation with an absolute error of 1 or 2. We gave two examples of such problems, namely MINVC and MINMAXMATCH. But do all unweighted[1] optimization problems have such constant absolute error reoptimizations of 1 or 2?

We answer this question to the negative, by giving the rare example of an unweighted problem that has no absolute error approximation in the reoptimization case (unless $\mathrm{P} = \mathrm{NP}$). Namely, we show this for the problem MINIMUMMAXIMALINDEPENDENTSET, or short MINMAXIS. This problem is also known as MINIMUMINDEPENDENTDOMINATINGSET (see [ACG+99]) and is defined as follows:

PROBLEM: MINMAXIS

INSTANCE: An (undirected) graph $G = (V, E)$.

SOLUTION: A maximal independent set in G, that is, a set $V' \subseteq V$ of mutually non-adjacent vertices such that the introduction of an additional vertex destroys the non-adjacency property.

MEASURE: $|V'|$.

Unless $\mathrm{P} = \mathrm{NP}$, the problem MINMAXIS has no factor $|V|^{1-\varepsilon}$-approximation, for any $\varepsilon > 0$ [Hal93]. We show, that the same non-approximability result also holds when *re-optimizing* MINMAXIS, whereas the considered modifications are the deletion and the insertion of a single edge. We first prove this for the modification function ad, which adds a single edge.

Theorem 10.12. *Unless* $\mathrm{P} = \mathrm{NP}$, $(ad, \text{MINMAXIS})$ *has no* $|V|^{1-\varepsilon}$*-approximation, for any* $\varepsilon > 0$.

Proof. Let $(ad, \text{MINMAXIS})$ be $|V|^{1-\varepsilon}$-approximable for some $0 < \varepsilon < 1$ via an algorithm A. We show that $\mathrm{SAT} \in \mathrm{P}$ under this assumption.

In the course of the proof we use a construction from [Hal93] that, given a formula $F = \{C_1, ..., C_m\}$ over variables $x_1, ..., x_n$, constructs a graph $f(F)$ as follows. The graph $f(F)$ has for each variable x_i two vertices labelled x_i and $\neg x_i$. Also, for every clause C_j, $1 \leq j \leq m$, it contains a certain number t of vertices $y_1^j, ..., y_t^j$. We fix the number t later in the proof. For the moment, just assume that t is large, e.g., $t = 2n$. The edges of $f(F)$ are $\{x_i, \neg x_i\}$, for all $i \leq n$, and $\{l, y_1^j\}, ..., \{l, y_t^j\}$ for all literals l of the clause C_j, $1 \leq j \leq m$. The graph $f(F)$ has some interesting properties.

1. For any fixed j, all the vertices $y_1^j, ..., y_t^j$ share the same neighborhood. Thus, if one of $y_1^j, ..., y_t^j$ is contained in a maximal independent set then all of $y_1^j, ..., y_t^j$ are contained.

[1]Obviously, optimization problems in which the cost depends on some weight function can have a constant error of at least k, k constant, by multiplying the weights with a large enough natural number.

2. If $F \in$ SAT then $f(F)$ has a maximal independent set of size n. To see this, let β be a satisfying assignment of F. Note that the independent set given by choosing all vertices x_i with $\beta(x_i) = 1$ and all vertices $\neg x_i$ with $\beta(x_i) = 0$, $1 \leq i \leq n$, is maximal, since all clauses are 'covered' by some literal.

3. If $t > n$ the independent sets from 2. are minimum. For a proof, note that a minimum maximal matching cannot contain a vertex of type y_ℓ^j (see 1.). But is has to contain one of the vertices x_i or $\neg x_i$ for each j in order to be maximal.

4. If $F \notin$ SAT then a minimum maximal independent set S has size at least t. No matter what combination of x_i- and $\neg x_i$-vertices are chosen, there must be at least one clause C_j that contains none of the corresponding literals. Consequently, all of $y_1^j, ..., y_t^j$ have to be in the resulting maximal independent set.

In constructing an original graph G^o with exactly one optimum solution, we use an auxiliary graph gadget H_s. For any $s \in \mathbb{N}$, the gadget H_s consist of a vertex z, which is the only vertex that may be connected to other vertices outside H_s, and s additional vertices $b_1, ..., b_s$, which are solely connected to z. The gadget H_s is useful for making specific vertices costly. In detail, if another vertex v is connected to the gadget H_s then together with v also the vertices $b_1, ..., b_s$ are in a maximal matching. Thus, a maximal matching that contains v has cost at least $s + 1$.

We are now prepared to define an original graph G^o that will be the input for our $|V|^{1-\varepsilon}$-approximation algorithm A. Let $F = \{C_1, ..., C_m\}$ be a formula over variables $x_1, ..., x_n$. From F we construct a CNF-formula

$$\begin{array}{lccccccccc} F' := & (\neg a \vee C_1) & \wedge & (\neg a \vee C_2) & \wedge & \dots & \wedge & (\neg a \vee C_m) & \wedge \\ & (a \vee x_1) & \wedge & (a \vee x_2) & \wedge & \dots & \wedge & (a \vee x_n). & \end{array}$$

Note that F' has exactly one satisfying assignment β with $\beta(a) = 0$, namely $\beta(x_i) = 1$ for all variables x_i, $1 \leq i \leq n$. Also, F' has a satisfying assignment β with $\beta(a) = 1$ if and only if $F \in$ SAT. Our starting point in defining G^o is the graph $f(F')$, to which we add the two gadgets H_2 and H_t. We connect the gadget H_2 to the vertex a. This concludes the construction of G^o. For an illustration of G^o see Figure 10.4.

Note that G^o has a solution of size $n + 3$, namely $S = \{x_1, ..., x_n, \neg a, z, z'\}$. If $t \geq n + 2$ this solution S is a unique minimum. To see this let $S' \neq S$ be a minimum solution in G^o.

Case 1: $a \in S'$**:** Then S' also contains b_1 and b_2. Furthermore, S' has to contain at least one of $z', b'_1, ..., b'_t$. Also, S' does not contain a vertex y_ℓ^j, since this would lead to a suboptimum solution of size $t + 4$ ($\geq n + 6$). Hence, by maximality of S', one of x_i or $\neg x_i$ is contained in S' for every $i \leq n$. In total this leads to a solution of size $n + 4$. Thus $|S'| > |S|$ and S' is no optimum.

Case 2: $a \notin S'$**:** In order to obtain a *minimum* maximal independent set on G^o it is best to choose the vertices z and z' to be in the cover. Also no vertex y_ℓ^j is part of S', otherwise S' has size at least $t + 2 \geq n + 4$, which is clearly not optimum.

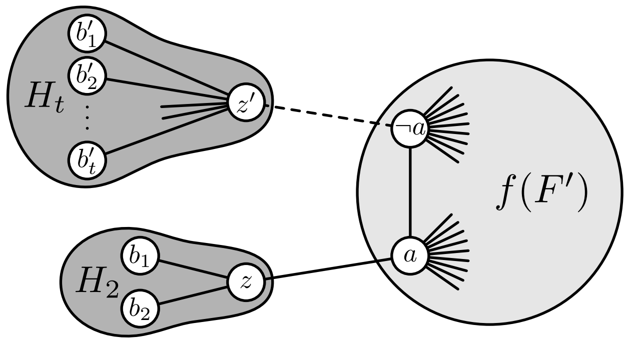

Figure 10.4: The MinMaxIS-instances G^o and G^m. The dashed line is present in G^m, but missing in G^o.

> If no y^j_ℓ vertex is used in S' then one of x_i or $\neg x_i$, $1 \leq i \leq n$, and the vertex $\neg a$ have to be part of S' by maximality of S'. Since $S' \neq S$ there exists an index j with $\neg x_j \in S'$. This implies a satisfying assignment β for F' with $\beta(a) = 0$ and $\beta(x_j) = 0$, a contradiction to the construction of F'.

This shows that if t is large enough then G^o has exactly one optimum solution.

We modify G^o by connecting the gadget H_t to $\neg a$. The resulting graph is G^m. Intuitively, this modification prevents the vertex $\neg a$ to be in a small maximal independent set. Thus, G^m has a small solution if and only if there exists an assignment for F' that has $\beta(a) = 1$, or equivalently $F \in$ SAT. We now elaborate this sketch more formally.

If $t \geq n + 4$ then

$$opt(G^m) \begin{cases} \leq n+4, & \text{if } F \in \text{SAT}, \\ > t, & \text{if } F \notin \text{SAT}. \end{cases}$$

For a proof of this claim, let $F \in$ SAT. Thus, F' has a satisfying assignment β with $\beta(a) = 1$. Consequently,

$$S := \{a, b_1, b_2, z'\} \cup \{x_i : \beta(x_i) = 1\} \cup \{\neg x_i : \beta(x_i) = 0\}$$

is a maximal independent set in G^m. For the case $F \notin$ SAT we argue as follows. Assume to the contrary that S' is a solution of G^m of size at most t. Thus, S' does not contain any y^j_ℓ vertex. Also, the vertex $\neg a$ is not contained in S', since otherwise also $b'_1, ..., b'_t$ are contained. The assignment

$$\beta(x_i) = \begin{cases} 1, & \text{if } x_i \in S, \\ 0, & \text{if } \neg x_i \in S, \end{cases}$$

yields a satisfying assignment for F' with $\beta(a) = 1$, a contradiction to $F \notin$ SAT.

In a last step, we now show how to choose t in our construction of G^o such that existence of a $|V|^{1-\varepsilon}$-approximation for $(ad, \text{MinMaxIS})$ yields that SAT $\in$ P. We aim to choose t large enough such that after $|V|^{1-\varepsilon}$-approximating an optimum solution in G^m, we can distinguish between the cases $F \in \text{SAT}$ and $F \notin \text{SAT}$. Let S' be such an approximated solution. If $F \in \text{SAT}$ and $t \geq n+4$ then

$$|S'| \leq \mathit{opt}(G^m) \cdot |V(G^m)|^{1-\varepsilon} \leq (n+4) \cdot ((m+1)t + 2(n+1)+4)^{1-\varepsilon}.$$

If $F \notin \text{SAT}$ then trivially $|S'| > t$, since this is a bound on the size of an optimum solution in this case. We are able to distinguish the cases $F \in \text{SAT}$ and $F \notin \text{SAT}$ by the size of $|S'|$ if

$$(n+4) \cdot ((m+1)t + 2(n+1)+4)^{1-\varepsilon} \leq t. \tag{2}$$

We claim that if $t \geq max\{\left[(n+4)[(m+1)^{1-\varepsilon}+1)]\right]^{1/\varepsilon}, 2(n+1)+4\}$ then (2) holds. From the first condition

$$t \geq \left[(n+4)[(m+1)^{1-\varepsilon}+1)]\right]^{1/\varepsilon}$$

it follows that

$$t^\varepsilon \geq (n+4)[(m+1)^{1-\varepsilon}+1)].$$

Consequently

$$t \geq (n+4) \cdot [(m+1)^{1-\varepsilon}+1] \cdot t^{1-\varepsilon},$$

or equivalently,

$$t \geq (n+4)[(m+1)^{1-\varepsilon} \cdot t^{1-\varepsilon} + t^{1-\varepsilon}].$$

Since $t \geq 2(n+1)+4$ we conclude that

$$t \geq (n+4)[((m+1)\cdot t)^{1-\varepsilon} + (2(n+1)+4)^{1-\varepsilon}].$$

Since $f(a+b) \leq f(a)+f(b)$ for every continuous, concave function f with $f(0) \geq 0$ (and in particular for the function $f(n) = n^{1-\varepsilon}$) the assertion (2) follows.

Since t is polynomial in the size of F, we conclude that G^o is polynomial-time computable. □

The ideas of the last proof are also helpful for showing

Theorem 10.13. *Unless* $\text{P} = \text{NP}$, $(rm, \text{MinMaxIS})$ *has no* $|V|^{1-\varepsilon}$*-approximation, for all* $\varepsilon \geq 0$.

Proof. Using the assumption that $(rm, \text{MinMaxIS})$ is $|V|^{1-\varepsilon}$-approximable we show that $\text{SAT} \in \text{P}$. Thereby we assume that the given formula $F = \{C_1, ..., C_m\}$ is over variables $x_1, ..., x_n$, that no clause of F contains an unnegated literal L and the corresponding negated literal $\neg L$ at the same time, and that the assignment β with $\beta(x_i) = 1$, $1 \leq i \leq n$, is no satisfying assignment for F.

Similar to the proof of Theorem 10.12 we start with the formula F'. We construct a graph G^o by composing the graph $f(F')$ with the two gadgets H_t and H_{t-1}. Furthermore, the gadget H_t is connected to a and H_{t-1} is connected to $\neg a$. An illustration of G^o as well as the labels of the vertices of the gadgets are depicted in Figure 10.5.

We claim that if $t > n$ then $\{\neg a, b_1, ..., b_{t-1}, z', x_1, ..., x_n\}$ is the sole optimum solution of G^o. To see this let $S' \neq S$ be a minimum solution of G^o.

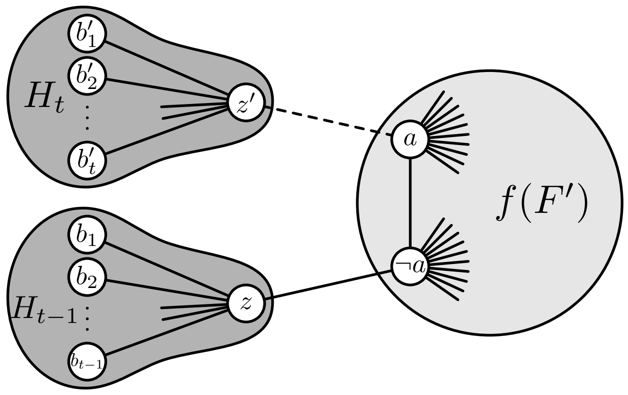

Figure 10.5: The MINMAXIS-instances G^o and G^m when removing an edge. The dashed line is present in G^o, but missing in G^m.

Case 1: $a \in S'$**:** Then S' also contains $b'_1, ..., b'_t$ and z. In order to avoid choosing a vertex y^j_ℓ it also has to contain one of x_i or $\neg x_i$ for each variable x_i. In total this leads to a solution of size $t + n + 2$. Thus $|S'| > |S|$ and S' is no optimum.

Case 2: $\neg a \in S'$**:** Then S' also contains $b_1, ..., b_{t-1}$ and the vertex z'. Furthermore, S' does not contain a vertex y^j_ℓ, otherwise S' was not optimum. Consequently, one of x_i or $\neg x_i$, $1 \leq i \leq n$, has to be part of S' by maximality of S'. Since $S' \neq S$ there exists an index j with $\neg x_j \in S'$. This implies a satisfying assignment β for F' with $\beta(a) = 0$ and $\beta(x_j) = 0$, a contradiction to the construction of F'.

Case 3: $\{\neg a, a\} \cap S' = \emptyset$**:** Thus z and z' are part of S'. Note that S' does not contain all of the vertices $y^j_1, ..., y^j_t$ and $y^{j'}_1, ..., y^{j'}_t$ for different numbers j, j', since otherwise $|S'|$ is too big.

Assume for the moment that S' contain vertices of type y^j_ℓ for *exactly* one natural number j. By our assumptions, no two literal x_i and $\neg x_i$ are contained in the corresponding clause C_j of F'. Thus, for every $i \leq n$, we can choose one vertex among x_i and $\neg x_i$ to be in the set S'. Hence, $|S'| = t + n + 2$, which is clearly no optimum.

By the former arguments, we may assume that S' does not contain any vertex of type y^j_ℓ at all. Then, all the vertices $x_1, ..., x_n$ have to be part of S' — otherwise, if $x_i \notin S'$ for some $i \leq n$ then all the vertices $y^p_1, ..., y^p_t$ that correspond to the clause $C_p = (a \vee x_i)$ in F' would be in S'. Recall that by our assumptions the assignment β with $\beta(x_i) = 1$, $1 \leq i \leq n$, is no satisfying assignment for F. Let C_k be a clause in F' that is not satisfied by this last assignment β. Then, the vertices $y^q_1, ...y^q_t$ that correspond to the clause $C_q = (C_k \vee \neg a)$ of F' have to be contained in S', a contradiction.

We modify G^o to G^m by deleting the edge $\{z', a\}$. Now,

$$opt(G^m)\begin{cases} \leq n+4, & \text{if } F \in \text{SAT}, \\ > t, & \text{if } F \notin \text{SAT}. \end{cases}$$

The rest of the proof is similar to the proof of Theorem 10.12. □

Index

Bibliography

[ABS03] Claudia Archetti, Luca Bertazzi, and M. Grazia Speranza. Reoptimizing the traveling salesman problem. *Networks*, 42(3):154–159, 2003. 3, 4, 68

[ACG$^+$99] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Potasi. *Complexity and Approximation*. Springer-Verlag, Berlin, 1999. 66, 67, 69, 70, 71, 142

[AEMP06] Giorgio Ausiello, Bruno Escoffier, Jérôme Monnot, and Vangelis Th. Paschos. Reoptimization of minimum and maximum traveling salesman's tours. In *Algorithm theory—SWAT 2006*, volume 4059 of *Lecture Notes in Comput. Sci.*, pages 196–207. Springer, Berlin, 2006. 3, 4, 68, 71, 73, 128, 131

[AK06] V. Arvind and Piyush P. Kurur. Graph isomorphism is in SPP. *Inform. and Comput.*, 204(5):835–852, 2006. 59

[BBH$^+$08] Davide Bilò, Hans-Joachim Böckenhauer, Juraj Hromkovič, Richard Královič, Tobias Mömke, Peter Widmayer, and Anna Zych. Reoptimization of steiner trees. In *SWAT '08: Proceedings of the 11th Scandinavian workshop on Algorithm Theory*, pages 258–269, Berlin, Heidelberg, 2008. Springer-Verlag. 78, 133

[BC94] Daniel Pierre Bovet and Pierluigi Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall International Series in Computer Science. Prentice Hall International, New York, 1994. 110

[BDG90] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural Complexity. II*, volume 22 of *EACTS Monographs on Theoretical Computer Science*. Springer-Verlag, Berlin, 1990. 6

[BDG95] José Luis Balcázar, Josep Díaz, and Joaquim Gabarró. *Structural complexity. I*. Texts in Theoretical Computer Science. An EACTS Series. Springer-Verlag, Berlin, second edition, 1995. 6, 39

[BFH$^+$07] Hans Joachim Böckenhauer, Luca Forlizzi, Juraj Hromkovič, Joachim Kneis, Joachim Kupke, Guido Proietti, and Peter Widmayer. On the approximability of TSP on local modifications of optimally solved instances. *Algorithmic Oper. Res.*, 2(2):83–93, 2007. 71, 75

[BG82] Andreas Blass and Yuri Gurevich. On the unique satisfiability problem. *Inform. and Control*, 55(1-3):80–88, 1982. 12

[BHK$^+$] Hans Joachim Böckenhauer, Juraj Hromkovič, R. Kralovic, Tobias Mömke, and P. Rossmanith. Reoptimization of Steiner trees: changing the terminal set. *Theoretical Computer Science*, to appear. 133

[BHMW08] Hans Joachim Böckenhauer, Juraj Hromkovič, Tobias Mömke, and Peter Widmayer. On the hardness of reoptimization. In *Proc. of the 34th International Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2008)*, volume 4910 of *Lecture Notes in Comput. Sci.*, pages 50–65. Springer, Berlin, 2008. 3, 68, 69, 71, 74

[BLS84] Ronald V. Book, Timothy J. Long, and Alan L. Selman. Quantitative relativizations of complexity classes. *SIAM J. Comput.*, 13(3):461–487, 1984. 13

[BP89] Marshall Bern and Paul Plassmann. The Steiner problem with edge lenghts 1 and 2. *Inform. Process. Lett.*, 32(4):171–176, 1989. 133

[Cai07] Jin-Yi Cai. $\mathrm{S}_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *J. Comput. System Sci.*, 73(1):25–35, 2007. 14

[Can96] Ran Canetti. More on BPP and the polynomial-time hierarchy. *Inform. Process. Lett.*, 57(5):237–241, 1996. 14

[CCHO05] Jin-Yi Cai, Venkatesan T. Chakaravarthy, Lane A. Hemaspaandra, and Mitsunori Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inform. and Comput.*, 198(1):1–23, 2005. 14

[CD97] Marco Cadoli and Francesco M. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997. 83

[CDLS02] Marco Cadoli, Francesco M. Donini, Paolo Liberatore, and Marco Schaerf. Preprocessing of intractable problems. *Inform. and Comput.*, 176(2):89–120, 2002. 81, 83, 84, 85, 87, 88

[Che03] Hubie Chen. Inverse NP problems. In *Mathematical foundations of computer science 2003*, volume 2747 of *Lecture Notes in Comput. Sci.*, pages 338–347. Springer, Berlin, 2003. 11, 17

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem, 1976. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh. 73, 74, 78

[CN07] Zhi-Zhong Chen and Takayuki Nagoya. Improved approximation algorithms for metric MaxTSP. *J. Comb. Optim.*, 13(4):321–336, 2007. 78, 131

[Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd STOC*, pages 151–158, 1971. 10, 38, 39

[DK00] Ding-Zhu Du and Ker-I Ko. *Theory of Computational Complexity*. Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience, New York, 2000. 21, 39

[DS05] Irit Dinur and Shmuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005. 69, 78

[EGIN97] David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997. 3

[EMP07] Bruno Escoffier, Martin Milanič, and Vangelis Paschos. Simple and fast reoptimizations for the steiner tree problem, 2007. Lamsade Technical Report No. 245, LAMSADE, Université Paris-Dauphin. 3, 68, 133

[FK99] J. Feigenbaum and S. Kannan. Dynamic graph algorithms. In *in 'Handbook of Discrete and Combinatorial Mathematics' (Kenneth H. Rosen)*, pages 583–591. CRC, 1999. 3

[Gab74] H.N. Gabow. Implementation of algorithms for maximum matching on nonbipartite graphs, 1974. Ph.D. Thesis, Stanford University. 128

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A Guide to the Theory of NP-completeness, A Series of Books in the Mathematical Sciences. 3, 23, 24, 48, 58, 105, 112, 122, 124, 126, 134

[GNW90] Thomas Gundermann, Nasser Ali Nasser, and Gerd Wechsung. A survey on counting classes. In *Fith Annual Structure in Complexity Theory Conference (Barcelona 1990)*, pages 140–153, Los Alamitos, CA, 1990. IEEE Comput. Soc. Press. 12

[Gol05] Oded Goldreich. Bravely, moderately: A common theme in four recent results. *Electronic Colloquium on Computational Complexity (ECCC)*, (098), 2005. 4, 5

[GPR98] Mathhew L. Ginsberg, Andrew J. Parker, and Amitabha Roy. Supermodels and robustness. In *AAAI-98/IAAI-98 Proceedings (Madison, WI, 1998)*, pages 334–339, Cambridge, MA, 1998. MIT Press. 4

[Gre98] Harvey J. Greenberg. An annotated bibliography for post-solution analysis in mixed integer programming and combinatorial optimization. In *Advances in computational and stochastic optimization, logic programming, and heuristic search*, volume 9 of *Oper. Res./Comput. Sci. Interfaces Ser.*, pages 97–147. Kluwer Acad. Publ., Boston, MA, 1998. 4

[Gro04] André Große. On relation classes and solution relations, 2004. Ph.D. Thesis, Friedrich-Schiller University, Jena. 13

[Hal93] Magnús M. Halldórsson. Approximating the minimum maximal independence number. *Inform. Process. Lett.*, 46(4):169–172, 1993. 78, 142

[Hen00] Monika R. Henzinger. Improved data structures for fully dynamic biconnectivity. *SIAM J. Comput.*, 29(6):1761–1815 (electronic), 2000. 3

[HK01] Monika R. Henzinger and Valerie King. Maintaining minimum spanning forests in dynamic graphs. *SIAM J. Comput.*, 31(2):364–374 (electronic), 2001. 3

[HK08] Harald Hempel and Madlen Kimmritz. Persistent computations of turing machines. In *CIAA '08: Proceedings of the 13th international conference on Implementation and Applications of Automata*, pages 171–180, Berlin, Heidelberg, 2008. Springer-Verlag. 4

[HMRS98] Harry B. Hunt, III, Madhav V. Marathe, Venkatesh Radhakrishnan, and Richard E. Stearns. The complexity of planar counting problems. *SIAM J. Comput.*, 27(4):1142–1167 (electronic), 1998. 99, 100

[HNOS96] Lane A. Hemaspaandra, Ashish V. Naik, Mitsunori Ogihara, and Alan L. Selman. Computing solutions uniquely collapses the polynomial hierarchy. *SIAM J. Comput.*, 25(4):697–708, 1996. 46

[Kar72] Richard M. Karp. Redcibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y. 1972)*, pages 85–103. Plenum, New York, 1972. 9, 106, 109, 134

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between nonuniform and uniform complexity classes. In *STOC'80: Proceedings of the twelfth annual ACM symposium on Theory of computing*, pages 302–309, New York, 1980. ACM. 13

[KMW98] S. Khanna, R. Motwani, and R. H. Wilson. On certificates and lookahead in dynamic graph problems. *Algorithmica*, 21(4):377–394, 1998. 3

[Kos98] Sven Kosub. Persistent computations, 1998. Technical Report No. 217, Julius-Maximilians-Universität Würzburg, Institut für Informatik. 4

[Krü05] Michael Krüger. Weitere Hamiltonkreise in Graphen und inverse Hamiltonkreisprobleme, 2005. Master Thesis, Friedrich-Schiller-University, Jena. 116

[Krü08] Michael Krüger. On the complexity of alternative solutions, 2008. Ph.D. Thesis, Friedrich-Schiller University, Jena. 17

[Lib98a] P. Liberatore. Compilation of intractable problems and its application to artificial intelligence, 1998. Ph.D. Thesis, Dipartimento di Informatica e Sistemistica, Università di Roma ”La Sapienza”. 83

[Lib98b] Paolo Liberatore. On non-conservative plan modification. In *European Conference on Artificial Intelligence*, pages 518–519, 1998. 4

[Lib01] Paolo Liberatore. Monotonic reductions, representative equivalence, and compilation of intractable problems. *J. ACM*, 48(6):1091–1125, 2001. 83

[Lib04] Paolo Liberatore. The complexity of modified instances. *arXiv.org*, cs/0402053, http://arxiv.org/abs/cs/0402053, 2004. 2, 4, 17, 24, 28, 30, 36, 81, 89, 110

[LL78] Nancy Lynch and Richard J. Lipton. On structure preserving reductions. *SIAM J. Comput.*, 7(2):119–126, 1978. 31

[LP86] L. Lovász and M.D. Plummer. *Matching Theory*. North-Holland, Amsterdam, 1986. Annals of Discrete Mathematics 29. 128

[LvdPSvdV98] Marek Libura, Edo S. van der Poort, Gerard Sierksma, and Jack A. A. van der Veen. Stability aspects of the traveling salesman problem based on k-best solutions. *Discrete Appl. Math.*, 87(1-3):159–185, 1998. 4

[MSVT94] Peter Bro Miltersen, Sairam Subramanian, Jeffrey Scott Vitter, and Roberto Tamassia. Complexity models for incremental computation. *Theoret. Comput. Sci.*, 130(1):203–236, 1994. 3, 15

[NK92] Bernhard Nebel and Jana Koehler. Plan modification versus plan generation: A complexity-theoretic perspective. (RR-92-48):15, 1992. 4

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994. 6, 8

[PS76] Christos H. Papadimitriou and Kenneth Steiglitz. Some complexity results for the traveling salesman problem. In *Eighth Annual ACM Symposium on Theory of Computing (Hershey, Pa., 1976)*, pages 1–9. Assoc. Comput. Mach., New York, 1976. 113

[PS82] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall Inc., Englewood Cliffs, N.J., 1982. 113, 114

[PY93] Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Math. Oper. Res.*, 18(1):1–11, 1993. 73

[Rog67] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill Book Co., New York, 1967. 11

[RS98] Alexander Russell and Ravi Sundaram. Symmetric alternation captures BPP. *Comput. Complexity*, 7(2):152–162, 1998. 14

[RZ05] Gabriel Robins and Alexander Zelikovsky. Tighter bounds for graph Steiner tree approximation. *SIAM J. Discrete Math.*, 19(1):122–134 (electronic), 2005. 78, 133

[Sch78] Thomas J. Schaefer. The complexity of satisfiability problems. In *Conference Record of the Tenth Annual ACM Symposium on Theory of Computing (San Diego, Calif., 1978)*, pages 216–226. ACM, New York, 1978. 99

[Sch87] Uwe Schöning. Graph isomorphism is in the low hierarchy. In *STACS 87 (Passau, 1987)*, volume 247 of *Lecture Notes in Comput. Sci.*, pages 114–124. Springer, Berlin, 1987. 59

[Sch97] Markus W. Schäffter. Scheduling with forbidden sets. *Discrete Appl. Math.*, 72(1-2):155–166, 1997. Models and algorithms for planning and scheduling problems (Loveno di Menaggio, 1993). 3, 4, 68

[Ser84] A. I. Serdyukov. An algorithm with an estimate for the traveling salesman problem of the maximum. *Upravlyaemye Sistemy*, 25:80–86, 1984. 78, 128, 129

[Sim75] J. Simon. On some central problems in computational complexity, 1975. Technical Report, Cornell University, Ithaca NY, TR75-224. 31

[SK96] Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *J. ACM*, 43(2):193–224, 1996. 83

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time: preliminary report. In *Fifth Annual ACM Symposium on Theory of Computing (Austin, Tex., 1973)*, pages 1–9. Assoc. Comput. Mach., New York, 1973. 11

[Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theoret. Comput. Sci.*, 3(1):1–22 (1977), 1976. 11

[TK00] Mikkel Thorup and David R. Karger. Dynamic graph algorithms with applications. In *Algorithm theory—SWAT 2000 (Bergen)*, volume 1851 of *Lecture Notes in Comput. Sci.*, pages 1–9. Springer, Berlin, 2000. 3

[Tor04] Jacobo Torán. On the hardness of graph isomorphism. *SIAM J. Comput.*, 33(5):1093–1108 (electronic), 2004. 59

[UN96] N. Ueda and T. Nagao. NP-completeness results for NONOGRAM via parsimonious reductions, 1996. Technical Report, TR96-0008. 121

[VHW99] Stan Van Hoesel and Albert Wagelmans. On the complexity of postoptimality analysis of $0/1$ programs. *Discrete Appl. Math.*, 91(1-3):251–263, 1999. 4

[Wec00] Gerd Wechsung. *Vorlesungen zur Komplexitätstheorie*. Teubner- Texte zur Informatik, Band. 32. B.G. Teubner, Stuttgart, Leipzig, Wiesbaden, 2000. 39

[Wes96] Douglas B. West. *Introduction to Graph Theory*. Prentice-Hall Inc., Upper Saddle River, NJ, 1996. 7

[WS07] Volker Weber and Thomas Schwentick. Dynamic complexity theory revisited. *Theory Comput. Syst.*, 40(4):355–377, 2007. 3

[WW86] Klaus Wagner and Gerd Wechsung. *Computational Complexity*, volume 21 of *Mathematics and its Applications (East European Series)*. D. Reidel Publishing Co., Dordrecht, 1986. 6, 8, 39

[Yap83] Chee-K. Yap. Some consequences of nonuniform conditions on uniform classes. *Theoret. Comput. Sci.*, 26(3):287–300, 1983. 15

[YG80] M. Yannakakis and F. Gavril. Edge dominating sets in graphs. *SIAM J. Appl. Math.*, 38(3):364–372, 1980. 70

Acknowledgements

I thank everybody who supported me during my work on this thesis. First of all, I want to thank my supervisor Harald Hempel for suggesting to work in this field and the excellent guidance during my time as PhD student and the work on my diploma thesis. He was a steady source for advice and inspiration, but also for encouragement.

Special thanks go to my colleague and friend Michael Krüger who had a helping hand, ear, or brain whenever it was needed. His contribution to this work, and in particular to the acknowledgements, cannot be valued high enough. The door to his office was open whenever something had to be discussed, examined, explained, or proofread, which was an enormous help especially in tough times.

Furthermore, my thanks go to all members of our “lunch group”. The interesting, entertaining or enlightening lunch talks with Harald, Micha, Marcus, Dave, Karsten, Madlen, Sylvia, and all the friends I might have forgotten here, were a welcome change from staring at the monitor.

I also express my thanks to Michael Jäger who abused his universities online-access to provide me with a lot of the needed papers. However, many of those papers would have never come to my attention if my colleagues Hannes Moser, Jiong Guo and Nadja Betzler had not pointed me to them.

Last, but not least, I am very grateful to all of my friends and family members. Without their steady support, this thesis would have been impossible.

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe.

Jena, den 22.07.2008

Tobias Berg

Lebenslauf

Persönliche Daten

Name:	Tobias Berg
Geburtsdatum:	28. Dezember 1978
Geburtsort:	Jena
Staatsangehörigkeit:	deutsch
Familienstand:	ledig

Schulausbildung

1985-1991	Besuch der Polytechnischen Oberschule Berthold-Brecht in Jena
1991-1997	Besuch des Carl-Zeiss-Gymnasiums mit math.-nat.-techn. Spezialklassen Jena, Leistungsfächer Mathematik und Informatik, mit Abschlussnote: 1.1

Zivildienst

1997-1998	Ableisten des Zivildienstes am Klinikum der Friedrich-Schiller-Universität Jena (FSU)

Studium

1998-2005	Studium der Informatik an der FSU Jena
April 2005	Abschluss des Studium der Informatik mit der Diplomarbeit zum Thema ”Complexity of Inverse Problems”
Okt. 2005	Studium der Mathematik und der Informatik auf Lehramt Gymnasium an der FSU Jena

Akademische Laufbahn

April 2006	Doktorand am Institut für Informatik der FSU und Beginn des Promotionsvorhabens ”Complexity of Modified Instances” gefördert durch ein Graduiertenstipendium des Landes Thüringen