New Classes of Complete Problems for the Second Level of the Polynomial Hierarchy

vorgelegt von
Dipl.-Math. oec. Berit Johannes

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Berichter:	Prof. Dr. James B. Orlin
	Prof. Dr. Rolf H. Möhring
Vorsitzender:	Prof. Dr. Fredi Tröltzsch

Tag der wissenschaftlichen Aussprache: 27. Juni 2011

Berlin 2011
D 83

Zusammenfassung

Eine wichtige Aufgabe der diskreten Mathematik besteht in der Kategorisierung von kombinatorischen Optimierungsproblemen nach ihrem Schwierigkeitsgrad. Die grundlegendsten und bekanntesten Komplexitätsklassen sind zweifelsohne P und NP. Auf P und NP baut sich die polynomielle Hierarchie auf, die aus vielen weiteren Komplexitätsklassen besteht, deren Probleme schwerer zu sein scheinen als die Probleme in P und NP. Die Komplexitätsklasse Σ_2^p liegt in dieser Hierarchie eine Stufe über NP und enthält all die Probleme, die durch einen nichtdeterministischen Algorithmus mit Hilfe eines NP-Orakels gelöst werden können. Im Gegensatz zu den Klassen P und NP erfreut sich die Klasse Σ_2^p geringerer Bekanntheit, was unter anderem daran liegen mag, dass sie naturgemäss komplizierter ist, und man bisher nur wenige natürliche Probleme kennt, die bezüglich dieser Klasse vollständig sind.

Der Hauptbeitrag der vorliegenden Arbeit liegt in der Entwicklung einer neuen Methode, die es erlaubt, ganze Klassen von Problemen, die in Σ_2^p liegen, als Σ_2^p-vollständig zu beweisen. Diese Technik benutzt bestimmte Eigenschaften bestehender Transformationen zwischen NP-vollständigen Problemen, um eine polynomielle Transformation von einem Σ_2^p-vollständigen Problem zu einer in Σ_2^p liegenden Variante eines anderen Problems zu erzeugen. Damit zeigt diese Arbeit nicht nur neue, durchaus natürliche und grundlegende, Σ_2^p-vollständige Probleme auf, sondern steuert ganze Familien von Problemen bei, die Σ_2^p-vollständig sind.

Zu dieser Art von Problemen gehören Varianten von in 0-1 Variablen formulierbaren Zulässigkeitsproblemen, bei denen man fragt, ob es für eine vorgegebene Teilmenge der Variablen eine 0-1 Zuweisung gibt, so dass für die restlichen Variablen keine 0-1 Belegung mehr möglich ist, die zu einer insgesamt zulässigen Lösung führen würde. Eine andere Klasse von Problemen betrifft Situationen, in denen die Zielfunktion vorerst nur in einem gewissen Rahmen vorgegeben ist. Dann geht es darum, die Instanz vor der eigentlichen Lösung des Problems zu verkleinern, indem man für einzelne Variablenbelegungen testet, ob sie überhaupt Teil einer optimalen Lösung sein können. Eine weitere Klasse von Σ_2^p-vollständigen Problemen ergibt sich aus Fragen nach der Existenz von Gegenbeispielen für Vermutungen im Zusammenhang mit NP-schweren Problemen. Ein Beispiel hierzu ist die Frage, ob eine Vermutung falsch ist, die besagt, dass alle Graphen mit einer bestimmten Eigenschaft einen Hamiltonischen Kreis haben. Zusätzliche Probleme und Problemklassen, die in dieser Arbeit als Σ_2^p-vollständig bewiesen werden, betreffen Fragestellungen wie den Versuch, eine beschränkte Anzahl der Variablen so zu belegen, dass es genau eine zulässige Erweiterung dazu gibt, oder die Aufgabe herauszufinden, ob ein gegebenes ganzzahliges Optimierungsproblem überflüssige Ungleichungen enthält.

Abstract

An important aspect of discrete optimization is to analyze the computational complexity of combinatorial optimization problems. The polynomial hierarchy provides a proper classification scheme for decision problems that appear to be harder than NP-complete. With P and NP at the bottom of the polynomial hierarchy, the next most interesting class is arguably Σ_2^p, which is the central topic of this thesis. The complexity class Σ_2^p lies one level above the class NP and contains all decision problems that can be solved efficiently by a nondeterministic algorithm that has access to an NP oracle. In contrast to the complexity classes P and NP, the class Σ_2^p has not yet attracted much attention, since it is inherently more intricate than the other classes and, on top of that, it has suffered from a scarcity of natural complete problems.

The main contribution of this thesis is the development of a new and powerful tool that uses established transformations between NP-complete problems to prove Σ_2^p-completeness for entire classes of problems. We show how and under which circumstances a polynomial time transformation from one NP-complete problem Π_1 to another NP-complete problem Π_2 can be used to derive a polynomial time transformation from an established Σ_2^p-complete problem corresponding to Π_1 to a version of Π_2 in Σ_2^p, which is then, as a consequence, Σ_2^p-complete as well. As a result, we will provide many families of complete problems for Σ_2^p, most of which are quite natural, some even basic.

This list of problems includes adversarial problems, which ask whether there is a 0-1 assignment to a specified subset of the variables so that it is not possible to complete this assignment to a feasible solution, no matter which values are assigned to the remaining variables. Another class concerns preprocessing problems. Given an instance of a combinatorial optimization problem with some flexibility for the objective function, we ask whether there exists an objective function such that a particular element becomes part of an optimal solution. We also establish that it is Σ_2^p-complete to decide whether there exists a counterexample to an NP-conjecture. An example of such a problem would be the question whether a graph with a certain property is Hamiltonian or not. More problems that are shown to be Σ_2^p-complete include defining set problems which ask whether for a given instance of a decision problem there is a partial solution of a certain size that can be completed to a feasible solution in a unique way, and the problem of deciding whether a given integer program has redundant constraints.

Contents

List of Figures

Chapter 1

Introduction

In this thesis we consider the complexity class Σ_2^p. This class lies one level above the class NP in the polynomial hierarchy and contains all decision problems that can be solved efficiently by a nondeterministic algorithm that uses an NP oracle. An NP oracle accepts as input an instance of some decision problem in NP and outputs the correct answer. Each call to the oracle is counted as one computational step.

The polynomial hierarchy gives means to find the proper classification of decision problems that appear to be harder than NP-complete. With P, NP, and coNP at the bottom of the polynomial hierarchy, the next most interesting class is arguably Σ_2^p. Classifying a problem as Σ_2^p-complete (instead of merely proving its NP-hardness), helps to appreciate its true difficulty, and to put it in the right place in the landscape of computational complexity.

The importance of a complexity class is often judged by the abundance of natural complete problems. In this regard, Σ_2^p is decidedly inferior to the class NP, which is known to contain a vast number of complete problems. Since complete problems are the most difficult problems in their class, they often serve as its representatives. If a complete problem is efficiently solvable, then so are all problems in its class. Com-

plete problems also are of use as test subjects for new algorithmic techniques and ideas aimed at understanding and tackling the inherent difficulties of the complexity class considered. Therefore, knowing a multitude of complete problems for any complexity class is very desirable.

While NP-completeness attempts to give us an understanding of which problems may not be solvable in (deterministic) polynomial time, Σ_2^p-completeness gives us a sense about which problems may not be solvable in (deterministic) polynomial time, even if one has access to an NP oracle.

Even though relatively few Σ_2^p-complete problems are currently known (compared to the huge number of NP-complete problems anyway), more and more problems are slowly but steadily added to the list of Σ_2^p-complete problems (see for example the compendium by Schaefer and Umans [SU02]). The first problem that was shown to be Σ_2^p-complete is the 2-ALTERNATING QUANTIFIED SATISFIABILITY problem (B_2), which is contained in the original paper by Meyer and Stockmeyer [MS72] on the polynomial hierarchy. An instance of (B_2) consists of two sets of Boolean variables, X and Y, and a Boolean expression E in disjunctive normal form, and the question is whether there exists a truth assignment to the variables in X such that for all truth assignments to the variables in Y, the expression E is satisfied. Equivalently, one can consider Boolean expressions E in conjunctive normal form and ask whether there is a truth assignment to the X variables such that for all truth assignments to the Y variables E is not satisfied. This defines the problem (B_2^{CNF}). Although the definition of the 2-ALTERNATING QUANTIFIED SATISFIABILITY problem may appear somewhat artificial, it is of great theoretical importance because it serves as the starting point for most Σ_2^p-completeness proofs.

Besides the theoretical interest in complete problems, there are Σ_2^p-complete prob-

lems of great practical relevance. The DNF MINIMIZATION problem, for example, has been studied by many researchers (see Coudert [Cou94] for a survey), since it was originally formulated by Quine [Qui52] in the fifties. Given a Boolean formula in disjunctive normal form and an integer K, DNF MINIMIZATION asks whether there is an equivalent formula in disjunctive normal form with at most K occurrences of literals. DNF MINIMIZATION and its relatives are important in VLSI-design, programmable logic array synthesis, multi-level logic synthesis, reliability analysis, and automated reasoning. Despite its relevance, it has only recently been shown to be Σ_2^p-complete by Umans [Uma00].

In this thesis we will populate the class Σ_2^p even further and provide many more complete problems, most of which are quite natural, some even basic. In fact, we will provide whole families of problems that are Σ_2^p-complete. In order to do so we provide a new and powerful tool that uses established transformations between NP-complete problems to prove Σ_2^p-completeness for entire classes of problems in Σ_2^p at once. This method can be applied to other complexity classes as well, and thus opens the door to finding many more complete problems for other complexity classes in the polynomial hierarchy, not only for Σ_2^p.

The first class of problems that we consider and for which we show Σ_2^p-completeness, is the class of adversarial problems. Any problem in this class is based on a combinatorial feasibility problem in NP that can be formulated with the help of 0-1 variables. An instance of the corresponding adversarial problem consists of the same set of variables and the same set of feasible solutions. However, the variables are divided into two sets X an Y, and we ask whether there is a 0-1 assignment to the variables in X so that it is not possible to complete this assignment to a feasible solution, no matter which values are assigned to the variables in Y. Obviously, the problem (B_2^{CNF}) belongs to the class of adversarial problems. The problem underlying (B_2^{CNF}) in the

first level of the hierarchy is the NP-complete SATISFIABILITY problem. In Chapter 2, we show how and under which conditions a polynomial time transformation from SATISFIABILITY to another problem Π can be used to derive a polynomial time transformation from (B_2^{CNF}) to the adversarial version of Π. As a consequence, the adversarial version of Π is Σ_2^p-complete.

Chapter 3 discusses the complexity of multilevel programming. In k-level programming, there is a hierarchy of k players, each controlling their own set of variables. The players act sequentially and choose their variables so as to optimize their own objective function, having complete knowledge about the objectives of the other players. We show that bilevel 0-1 integer programming is Σ_2^p-complete and trilevel linear programming is Σ_2^p-hard. While these results are not entirely new, we give different proofs that are based on the results obtained in the preceding chapter.

The next class of problems for which we establish Σ_2^p-completeness is the class of preprocessing problems (Chapter 4). Given an instance of a combinatorial optimization problem with some flexibility for the objective function, we ask whether there exists an objective function such that a particular element becomes part of an optimal solution. If, within the given wiggle room, there is no cost function that allows this specific element to be in an optimal solution, then this element can be eliminated from the problem instance — hence the term "preprocessing." This class of problems is of great practical interest in situations where the cost data is not completely specified in advance, since it potentially allows to reduce the size of instances, thereby enabling faster solutions later.

Next, in Chapter 5, we offer a possible explanation of why it often seems so difficult to disprove conjectures on graphs. A typical example would be conjectures on when a graph is Hamiltonian. We show that it is Σ_2^p-complete to find counterexamples. More

specifically, we establish Σ_2^p-completeness for finding counterexamples to conjectures related to the existence of objects whose existence is NP-complete to decide. Once again, we do so by pinpointing the properties that the NP-completeness proof would need to satisfy so that we can successfully modify it into a Σ_2^p-reduction.

In the final chapter we introduce two more problem classes. A defining set problem asks whether for a given instance of a decision problem there is a partial solution of a certain size, sometimes restricted to a subset of variables, that can be completed to a feasible solution in a unique way. A cost denying set problem is concerned with the existence of a partial solution (restricted to a subset of variables) of limited cost, which cannot be extended to a complete feasible solution. We show Σ_2^p-completeness for all problems in the class of defining set problems, and for all problems in the class of cost denying set problems. Furthermore, we provide additional Σ_2^p-completeness results that are based on results from previous chapters. These include constraint redundancy problems such as the minimum integer programming equivalence problem, which tries to detect constraints that can be deleted from an integer programming formulation without changing the solution space.

In the remainder of this introductory chapter, we introduce some notation, give a formal definition of Σ_2^p and the polynomial hierarchy, and provide a sketch of the history of the polynomial hierarchy.

1.1 Basic Definitions

We now introduce the basic concepts and notions that we will rely on in this thesis. If the reader desires a more comprehensive and formal introduction to algorithms and complexity, we recommend reading any good textbook on complexity theory, such as the one by Garey and Johnson [GJ79].

A *decision problem* Π consists of a set D_Π of instances and a subset $Y_\Pi \subseteq D_\Pi$ of yes-instances and asks whether a given instance is a yes-instance or not. It accepts only "yes" and "no" answers.

A *deterministic algorithm solves* a decision problem Π, if it halts for all input instances $I \in D_\Pi$, returns the answer "yes" if and only if $I \in Y_\Pi$, and returns the answer "no" otherwise. If the number of steps performed by the algorithm is bounded by a polynomial in the input size, then it is a *polynomial time deterministic algorithm.* P is the class of decision problems Π for which there is a polynomial time deterministic algorithm that solves Π.

A *nondeterministic algorithm* consists of two parts, the "guessing stage" and the "checking stage." It *solves* a decision problem Π if the following two properties hold for all instances $I \in D_\Pi$:

1. If $I \in Y_\Pi$, then there exists a certificate S that, when guessed for input I, will lead the algorithm to respond "yes" for I and S.

2. If $I \notin Y_\Pi$, then there exists no certificate S that, when guessed for input I, will lead the algorithm to respond "yes" for I and S.

A nondeterministic algorithm that solves a decision problem Π is said to operate in *polynomial time* if there exists a polynomial p such that, for every instance $I \in Y_\Pi$, there is some guess S that leads the deterministic checking stage to respond "yes" for I and S within time that is polynomial in the size of I. The class NP is the class of all decision problems that can be solved by polynomial time nondeterministic algorithms.

A *polynomial transformation* from a decision problem Π_1 to a decision problem Π_2 is a function $f : D_{\Pi_1} \to D_{\Pi_2}$ (with D_{Π_1} and D_{Π_2} being the set of input instances, and Y_{Π_1}

and Y_{Π_2} being the set of yes-instances of Π_1 and Π_2, respectively), that satisfies the following two conditions:

1. There is a polynomial time deterministic algorithm that computes f.
2. For all $I \in D_{\Pi_1}$, $I \in Y_{\Pi_1}$ if and only if $f(I) \in Y_{\Pi_2}$.

Polynomial transformations are *transitive*, that is, if there is a polynomial transformation from Π_1 to Π_2 and from Π_2 to Π_3, then there is a polynomial transformation from Π_1 to Π_3 as well.

A decision problem Π is *complete* for a complexity class $\mathcal{C}$ (with respect to polynomial transformability) if $\Pi \in \mathcal{C}$ and there is a polynomial transformation from Π' to Π for all $\Pi' \in \mathcal{C}$.

1.2 The Polynomial Hierarchy

There are several (equivalent) definitions of the complexity class Σ_2^p. We will exclusively rely on the original definition given by the inventors of the polynomial hierarchy, Meyer and Stockmeyer [MS72]. They defined Σ_2^p as the class of decision problems that are solvable by polynomial time nondeterministic algorithms with access to an oracle for some problem in NP. Recall that P is the class of decision problems that are solvable in polynomial time by a deterministic algorithm. Recall further that the class NP is the class of all decision problems Π that can be solved by polynomial time nondeterministic algorithms. In particular, if we are handed a certificate to a yes-instance I of the problem Π, we are able to verify in deterministic polynomial time that I is indeed a yes-instance. Now, if we are given a decision problem Π such that for any of its yes-instances there exists a certificate that can be checked by a polynomial time *nondeterministic* algorithm, then Π is member of the class Σ_2^p.

In other words, for problems in NP we are restricted to polynomial time deterministic algorithms for certificate verification, whereas for problems in Σ_2^p we may use polynomial time nondeterministic algorithms. That is, the problem of checking the certificate can itself be a problem in NP. If in turn we allow that the problem of checking a certificate is in Σ_2^p, we obtain the complexity class Σ_3^p. By allowing more and more powerful algorithms for certificate checking, we arrive at (supposedly) more and more complex classes of problems, resulting in the polynomial hierarchy.

This can be formalized as follows. Let $\mathcal{C}$ be a class of decision problems that can be solved by a particular class of algorithms. Let $\mathcal{D}$ be a class of decision problems that can be solved by a particular (potentially different) class of algorithms. For instance, P is the class of decision problems that can be solved by polynomial time deterministic algorithms, NP is the class of decision problems that can be solved by polynomial time nondeterministic algorithms, and Σ_2^p is the class of problems that can be solved by polynomial time nondeterministic algorithms that have access to an NP oracle. Then $\mathcal{C}^{\mathcal{D}}$ is defined to be the class of problems that can be solved by algorithms obtained from $\mathcal{C}$ which may, in addition, solve problems in $\mathcal{D}$ in one step (that is, they have access to a $\mathcal{D}$ oracle). The polynomial hierarchy is recursively defined by setting $\Sigma_0^p = \mathrm{P}$, and $\Sigma_k^p = \mathrm{NP}^{\Sigma_{k-1}^p}$ for all $k \geqslant 1$. Moreover, Π_k^p is defined to be the complement of Σ_k^p; that is, $\Pi_k^p = \mathrm{co}\Sigma_k^p$ for all $k \geqslant 0$. In particular, $\Sigma_0^p = \Pi_0^p = \mathrm{P}$, $\Sigma_1^p = \mathrm{NP}$, and $\Pi_1^p = \mathrm{coNP}$. Note that each class at each level includes all classes at previous levels; that is, $\Sigma_k^p \subseteq \Sigma_{k+1}^p$, $\Pi_k^p \subseteq \Pi_{k+1}^p$, $\Sigma_k^p \subseteq \Pi_{k+1}^p$, and $\Pi_k^p \subseteq \Sigma_{k+1}^p$. The polynomial hierarchy is defined as $\mathrm{PH} = \bigcup_{k \in \mathbb{N}} \Sigma_k$. Here $\mathbb{N}$ denotes the natural numbers including 0.

Each class in the polynomial hierarchy is closed under polynomial transformations. This means that for a class $\mathcal{C}$ in the hierarchy and two decision problems A and B with $B \in \mathcal{C}$, if there is a polynomial transformation from A to B, then $A \in \mathcal{C}$ as

well. This also implies that if B is a complete problem for Σ_k^p, then $\Sigma_{k+1}^p = \mathrm{NP}^B$. For instance, $\Sigma_2^p = \mathrm{NP}^{(3\mathrm{SAT})}$. In other words, if a decision problem is defined based on some oracle in $\mathcal{C}$, then we can assume that it is defined based on an oracle for a complete problem for $\mathcal{C}$.

Stockmeyer [Sto77] and Meyer and Stockmeyer [MS72, SM73] proved that each class in the polynomial hierarchy contains complete problems. More specifically, they showed that the k-ALTERNATING QUANTIFIED SATISFIABILITY problem (B_k) is Σ_k-complete. An instance of (B_k) is a well-formed Boolean expression E over a set of Boolean variables $X = \{x[i,j] : 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant m_i\}$ for some integers $m_1, m_2, \ldots, m_k \geqslant 0$. The question is whether the following expression is true:

$$\begin{aligned} &\exists x[1,1], \ldots, x[1,m_1] \\ &\quad \forall x[2,1], \ldots, x[2,m_2] \\ &\qquad \vdots \\ &\quad Qx[k,1], \ldots, Qx[k,m_k]E, \end{aligned}$$

where the quantifier Q is "$\exists$" if k is odd, and it is equal to "$\forall$" if k is even. For $k = 1$, the problem (B_1) equals the NP-complete problem SATISFIABILITY OF BOOLEAN EXPRESSIONS [GJ79, Problem LO7].

We will be referring many times to the problem (B_2^{CNF}), a close relative of (B_2), which is also Σ_2^p-complete. An instance of (B_2^{CNF}) consists of two sets of Boolean variables, X and Y, and a Boolean expression E in conjunctive normal form with exactly three literals per clause, and the question is whether there exists a truth assignment to the variables in X such that there is no truth assignment to the variables in Y that would help to satisfy E. Wrathall [Wra76] showed that the variant in which every clause has at most three literals is Σ_2^p-complete. Cook's [Coo71] simple transformation from (SAT) to (3SAT) can then be used to show Σ_2^p-completeness for (B_2^{CNF}).

Stockmeyer and Meyer [SM73] (see also [Sto77]) observed that the polynomial hierarchy is completely contained in the complexity class PSPACE, which is the class of decision problems solvable with polynomial space, since searching for a certificate can be done in polynomial space. In fact, they showed that the problem ALTERNATING QUANTIFIED SATISFIABILITY becomes PSPACE-complete if the number of alternations between the two quantifiers is unbounded.

Wrathall [Wra76] showed that if a PSPACE-complete problem is in Σ_k^p for some k, then the hierarchy collapses at that point, and $\Sigma_j^p = \Sigma_k^p = \text{PSPACE}$ for all $j \geqslant k$. If the hierarchy does not collapse to some finite level, then it does not contain complete problems: If there was a PH-complete problem A, then there must be a $k \geqslant 0$ with $A \in \Sigma_k^p$. Since any decision problem $A' \in \Sigma_{k+1}^p$ reduces to A, and all levels of the polynomial hierarchy are closed under polynomial transformations, this would imply that $A' \in \Sigma_k^p$ and thus $\Sigma_k^p = \Sigma_{k+1}^p$. This also implies that if PH = PSPACE, then the polynomial hierarchy would collapse because PSPACE does contain complete problems.

We have seen that every level of the polynomial hierarchy relies on the level below. It is therefore not surprising that an unexpected anomaly on any one level of the polynomial hierarchy would have significant repercussions for all levels that are further up in the polynomial hierarchy. Meyer and Stockmeyer [MS72], as well as Wrathall [Wra76] showed that if $\Sigma_k^p = \Pi_k^p$ for some level $k \geq 1$, then $\Sigma_j^p = \Pi_j^p = \Sigma_k^p$ for all $j \geqslant k$.

The biggest open question concerning the first level of the polynomial hierarchy is of course whether or not P = NP. Also unknown is whether NP = coNP. Open questions concerning higher levels of the polynomial hierarchy include:

(1) Does P $\neq$ NP imply $\Sigma_k^p \neq \Sigma_{k+1}^p$ for all k?

(2) Is $\Sigma_k^p \neq \Sigma_{k+1}^p$ for all $k \geqslant 0$?

(3) Is $\Sigma_k^p \neq \Pi_k^p$ for all $k \geqslant 1$?

(4) Is PH $\subsetneq$ PSPACE?

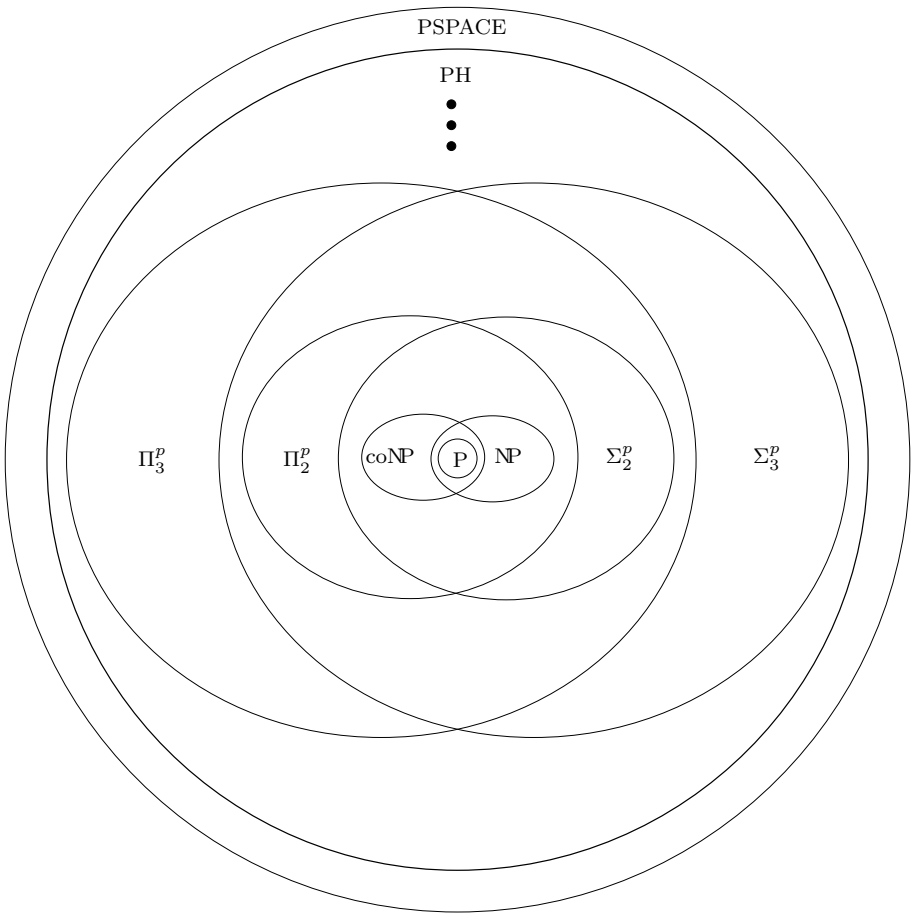

Figure 1-1: The polynomial hierarchy

Figure 1-1 depicts the polynomial hierarchy and the relationships between its components, assuming that the hierarchy does not collapse. Note that the definition of the polynomial hierarchy does not include PSPACE, but the polynomial hierarchy is contained in PSPACE. It is an open question whether PH=PSPACE.

1.3 Brief History of Σ_2^p

Turing [Tur36] pioneered the theory of computational complexity with the development of his theoretical computational model in the thirties. The field gained momentum in the early sixties with the key idea by Hartmanis and Stearns [HS65] to measure time and space as a function of the length of the input and with the publication of the first articles emphasizing the significance of polynomial time complexity. The class P itself was first mentioned by Cobham [Cob64]. He noted the wide variety of mathematical functions that can be computed in polynomial time and observed that the class of all such functions remains the same under many different models of computation. Edmonds [Edm65b] informally identified the term "good algorithm" with the notion of a polynomial time algorithm and, in another paper, [Edm65a], he introduced an informal description of nondeterministic polynomial time, the notion analogous to the complexity class NP, by proposing that a problem be said to have a "good characterization" if for every solution there exists "information" that can be used "with ease to verify with mathematical certainty" that it is indeed a solution. Around the same time several of the same notions of complexity were independently developed in the Soviet Union, by, among others, Barzdin and Trakhtenbrot, though these works became not known to the West until the seventies.

By the late sixties, a sizable class of quite relevant problems which resisted polynomial time solutions was widely recognized. NP-completeness captures the intrinsic difficulty of many of these problems and provides a method for proving that a combinatorial problem is computationally as difficult as any other problem in NP. The foundations for the theory of NP-completeness were laid in a paper by Cook [Coo71]. In this groundbreaking paper he emphasized the significance of "polynomial time reducibility," he focused the attention on the class NP of decision problems that can be solved in polynomial time by a nondeterministic algorithm, and, most importantly, he proved that one particular problem in NP, the SATISFIABILITY problem, has the

property that every other problem in NP can be polynomially reduced to it. In other words, he had identified the first NP-complete problem. Meanwhile Levin [Lev73] independently showed NP-completeness for a different problem. In particular, Cook and Levin established the notion that if there is a polynomial time algorithm for certain problems in NP, then there is one for every problem in NP. The question whether every problem in NP has an efficient algorithm has resisted the efforts of many researchers ever since, and is known as the P versus NP problem, one of the most central open questions at the intersection of theoretical computer science and mathematics.

With a first complete problem at hand, Karp [Kar72] quickly produced many more NP-complete problems, thus showing beyond any doubt the broad applicability of this concept. In his highly influential work he presented several key techniques to prove NP-completeness using polynomial transformations from problems previously known to be NP-complete. It set up a general framework for proving NP-completeness results and established several useful gadgets for such proofs. Among the problems which he proved to be NP-complete are CLIQUE, INDEPENDENT SET, SET COVER, VERTEX COVER, and the TRAVELING SALESMAN problem. In the same paper he also notes that "a polynomial-bounded version of Kleenes Arithmetic Hierarchy [Rogers (1967)] becomes trivial if P = NP," thus already hinting at the polynomial hierarchy, which was later formally defined by Meyer and Stockmeyer [MS72].

Since these fundamental works, theoretical computer science has continued to seek a precise understanding of how efficiently computational problems may be solved. For any one problem, the goal is to establish an upper bound in the form of an algorithm, and a lower bound, by assigning the problem to a complexity class, thereby showing that all algorithms with a certain computational power are incapable (under certain assumptions) of solving the problem.

While numerous decision problems have been proved NP-complete, some decision problems are outside NP and have escaped this classification. In the early 1970's, Meyer and Stockmeyer started to look beyond NP, wondering which problems may have a higher complexity and how to classify them. With the formal definition of the polynomial hierarchy they provided an extended classification scheme for the complexity of a wide range of decision problems. The problem that led Meyer and Stockmeyer to define the polynomial hierarchy was the problem MINIMAL, which asks whether a given Boolean expression Φ is minimal. That is, is there no equivalent expression Φ' with fewer literals? Since one presumably cannot check efficiently whether two Boolean expressions are equivalent, the problem MINIMAL and its complement appear to lie beyond NP, but they seem unlikely to be PSPACE-complete. (It is in fact still unresolved whether the problem MINIMAL is complete for the second level of the polynomial hierarchy.) But, if we are given an oracle for testing the equivalence of Boolean formulas, then we can solve the complement problem of MINIMAL in nondeterministic polynomial time. This suggests a "hierarchy" of classes, and the polynomial hierarchy was born.

Meyer and Stockmeyer [MS72], as well as Wrathall [Wra76] gave several different characterizations of the polynomial hierarchy and described conditions under which it would collapse. Meyer and Stockmeyer [MS72] also provided a complete problem for every level of the polynomial hierarchy. They proved that for any level k of the hierarchy, (B_k), the k-ALTERNATING QUANTIFIED SATISFIABILITY problem, which was introduced in Section 1.1, is Σ_k-complete. While the definition of (B_k) may seem rather artificial, it opens the door to the identification of many more complete problems for various levels of the polynomial hierarchy, as did, for the class NP, Cook's Theorem [Coo71] on the NP-completeness of the SATISFIABILITY problem.

Later, Stockmeyer [Sto77] proposed a variant of the problem MINIMAL, called DNF MINIMIZATION, in which both formulae, Φ and Φ', are required to be in disjunctive normal form. Early on this problem had been conjectured to be complete for the class Σ_2^p, and its Σ_2^p-completeness was eventually proved by Umans [Uma00]. In the same paper, Umans defined more than a dozen other natural optimization problems and showed their Σ_2^p-completeness. His work contributed significantly to the acceptance and importance of the complexity class Σ_2^p, which had previously suffered from the scarcity of natural complete problems. Another problem that had long been known to be in Σ_2^p and a candidate for Σ_2^p-completeness is MINIMUM EQUIVALENT EXPRESSION, which is another variant of MINIMAL. This open question was recently settled by Buchfuhrer and Umans [BU11], who showed that it is indeed Σ_2^p-complete. MINIMUM EQUIVALENT EXPRESSION is the problem of deciding for a given Boolean formula that uses the Boolean operations *or*, *and*, and *negate* only, and an integer K, whether there is an equivalent Boolean formula, using *or*, *and*, and *negate* only, of size at most K. Here the size is measured again in terms of the number of occurrences of literals in the formula. However, the computational complexity of the version of MINIMUM EQUIVALENT EXPRESSION that allows the use of the Boolean operations *or*, *and*, *negate*, and *implies* is still open.

One of the first natural complete problems in the second level of the hierarchy is INTEGER EXPRESSION INEQUIVALENCE, which appeared in the original papers introducing the polynomial hierarchy [Sto77, MS72]. A couple of problems related to INTEGER EXPRESSION INEQUIVALENCE were later shown to be complete for the second level of the hierarchy by Wagner [Wag86]. Moreover, Sagiv and Yannakakis [SY80] proved that deciding whether two monotonic relational expressions are equivalent is Π_2^p-complete, and Huynh [Huy84] shows that deciding whether two context-free grammars each with only one terminal letter generate the same language is Π_2^p-complete. In a similar spirit, Lin [Lin95] showed that a problem related to PATTERN MATCHING

(and program optimization) is Σ_2^p-complete. Ko and Tzeng [KT91] exhibited three Σ_2^p-complete problems that belong to the realm of computational learning theory. The problems are PATTERN CONSISTENCY, which asks whether for two sets of words there is a pattern that matches each word in one of the sets but no word in the other, GRAPH RECONSTRUCTION, asking whether for two sets of graphs there is a graph G such that all graphs in the first set are isomorphic to a subgraph of G but none of the graphs in the second set is, and, finally, GENERALIZED 3-CNF CONSISTENCY, which asks whether for two sets of Boolean formulas there is a 3-CNF formula which is consistent with each formula from the first set but with no formula from the second set.

More Σ_2^p-complete problems can be found in [SU02], which is a list of complete problems for the lower levels in the polynomial hierarchy compiled by Schaefer and Umans. Moreover, in the next five chapters we present many more (classes of) Σ_2^p-complete problems.

Chapter 2

Adversarial Problems

The 2-ALTERNATING QUANTIFIED SATISFIABILITY problem (B_2) plays the same role for Σ_2^p which SATISFIABILITY has played for NP. Specifically, it is the first problem in its class that was shown to be complete [MS72]. A close relative of (B_2) is (B_2^{CNF}), which is also Σ_2^p-complete [Wra76]. An instance of (B_2^{CNF}) consists of two sets of Boolean variables, X and Y, and a Boolean expression E over X and Y in conjunctive normal form containing three variables per clause, and the question is whether there exists a truth assignment to the variables in X such that there is no truth assignment to the variables in Y that would satisfy E. The problem (B_2^{CNF}) may therefore be regarded as a game with two players, in which the first player controls the variables in X, and the second player controls the variables in Y. The X-player goes first and tries to find a truth assignment for her variables that prevents the Y-player from finding a truth assignment for his variables such that E is satisfied. In this sense, (B_2^{CNF}) belongs to the class of "adversarial problems," where the X-player assumes the role of the adversary.

In general, any problem in NP that can be formulated with the help of 0-1 variables can be turned into an adversarial problem: partition the set of variables into two sets and give each player control over one of these sets. The player who moves first wants

to fix the values of her variables in a way that makes a feasible solution impossible, no matter which values the second player might choose for his variables. In the literature, problems of this kind tend to have a leading "$\exists\forall$", or "EA" in the problem name.

It is not difficult to establish membership in Σ_2^p for a wide variety of adversarial problems. Completeness, however, is a different matter, and would normally require a separate proof for each individual adversarial problem. Instead, we will now develop a general theory that implies Σ_2^p-completeness for many adversarial problems at once.

Our technique exploits certain properties of existing polynomial transformations used for showing NP-completeness of the underlying non-adversarial problem. Most polynomial transformations f from an NP-complete problem Π_1 to an NP-complete problem Π_2 involve a direct one-to-one mapping g between the set Z_1 of variables of an instance $I_1 \in \Pi_1$ and a subset Z_2' of the variables of the instance $I_2 = f(I_1) \in \Pi_2$. This mapping typically enables us to deduce a solution S_1 for I_1 from a solution S_2 for I_2, and vice versa. In particular, the 0-1 assignment to the variables in the set Z_1 and the subset Z_2' is maintained: given a solution S_1 for I_1, there is a solution S_2 for I_2 such that the value $S_1(z)$ of a variable $z \in Z_1$ is identical to the value $S_2(g(z))$ of its image in Z_2', and vice versa. (In the entire thesis, we use the following convention: A "solution S" and a "0-1 assignment S" are both 0-1 vectors. A solution is always a feasible solution. A 0-1 assignment may be infeasible.)

We will show that, whenever there is a polynomial transformation between any two NP-complete problems with this characteristic, one can modify this transformation in a generic way to obtain a transformation between the two corresponding adversarial problems. For example, for all polynomial transformations from (3SAT) to an NP-complete problem Π that have this property, we obtain a polynomial transformation that reduces (B_2^{CNF}) to the adversarial version of Π. If the adversarial version of Π is

in Σ_2^p, it then follows that it is in fact Σ_2^p-complete.

Since it appears to be the case that most transformations between NP-complete problems involve such a mapping, or can be tweaked slightly so that they do, this tool is widely applicable and allows us to prove Σ_2^p-completeness for a great number of adversarial problems.

2.1 A Generic Class of Adversarial Problems

We begin with a formal definition of the basic Σ_2^p-complete problem (B_2^{CNF}) and its underlying NP-complete non-adversarial problem (3SAT).

PROBLEM: 3-SATISFIABILITY (3SAT)
INSTANCE: Set U of variables, Boolean expression E over U in conjunctive normal form with exactly 3 variables in each clause.
QUESTION: Is there a 0-1 assignment for the variables in U that satisfies E?

PROBLEM: (B_2^{CNF})
INSTANCE: Sets X and Y of variables, Boolean expression E over X and Y in conjunctive normal form with exactly 3 variables in each clause.
QUESTION: Is there a 0-1 assignment for X so that there is no 0-1 assignment for Y such that E is satisfied?

We now formally introduce the concept of an ADVERSARIAL PROBLEM (ADV). We start by formulating its underlying non-adversarial problem (NON-ADV), which is a combinatorial feasibility problem in NP.

PROBLEM: COMBINATORIAL FEASIBILITY PROBLEM (NON-ADV)
INSTANCE: Finite ground set Z of variables, (implicitly given) set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible 0-1 assignments for the variables in Z.
QUESTION: Is $\mathcal{F}$ non-empty?

In most cases, Z would correspond to the set of nodes or the set of edges in a graph or the Boolean variables in a satisfiability problem. We have a 0-1 variable for each such node or edge or Boolean variable, and Z formally denotes the set of these variables. Also, note that $\mathcal{F}$ is typically of exponential size (in $|Z|$). We therefore assume that its elements are not listed explicitly because this would render the problem trivial. Instead $\mathcal{F}$ is implicitly given, specified by some property, and any specific instantiation of $\mathcal{F}$ leads to a specific feasibility problem. For instance, $\mathcal{F}$ may specify the set of all Hamiltonian cycles in a graph, resulting in the Hamiltonian cycle problem.

Given an instance of (NON-ADV), if we divide the variables in Z into two disjoint sets, we obtain an instance of the corresponding ADVERSARIAL PROBLEM.

PROBLEM: ADVERSARIAL PROBLEM (ADV)
INSTANCE: Disjoint ground sets X and Y of 0-1 variables with $X \cup Y = Z$, (implicitly given) set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible 0-1 assignments for the variables in Z.
QUESTION: Is there a 0-1 assignment S_X for the variables in X such that for all 0-1 assignments S_Y for the variables in Y, the combined 0-1 assignment (S_X, S_Y) for all variables in Z is not in $\mathcal{F}$?

In the subsequent discussion, we will use (NON-ADV) (and (ADV)) as placeholders for some *specific* (adversarial) combinatorial feasibility problem, in which the set $\mathcal{F}$ has been specified beforehand. In this sense, (NON-ADV) and (ADV) specify a class of problems, parameterized by $\mathcal{F}$.

The following theorem details the properties of a polynomial transformation between (3SAT) and a combinatorial feasibility problem (NON-ADV) that are sufficient in order to deduce Σ_2^p-completeness of the corresponding adversarial problem (ADV). For ease of exposition, we introduce the following notation. For a function g that maps elements from a set U to elements of a set Z we define $g(U_1)$ with $U_1 \subseteq U$ as the set $g(U_1) = \{z \in Z : \text{there exists a } u \in U_1 \text{ with } g(u) = z\}$. In other words, $g(U_1)$ is the image of U_1 under g.

Theorem 2.1.1. *Let (ADV) be an* ADVERSARIAL PROBLEM. *Let $(NON\text{-}ADV)$ denote the corresponding non-adversarial problem. We assume that $(NON\text{-}ADV)$ is in NP. Let f be a polynomial transformation from $(3SAT)$ to $(NON\text{-}ADV)$ that satisfies the following property. If U is the ground set of an instance $I_{(3SAT)}$ of $(3SAT)$ and Z is the ground set of the instance $f(I_{(3SAT)})$ of $(NON\text{-}ADV)$, then there is a subset Z' of Z and a bijective function $g : U \to Z'$ such that:*

(1) *If S^U is a solution of $I_{(3SAT)}$, then the 0-1 assignment $S_{Z'}$ to the variables in Z' with $S_{Z'}(z) = S^U(g^{-1}(z))$ for all $z \in Z'$ can be extended to a 0-1 assignment S of all variables in Z such that S is in $\mathcal{F}$.*

(2) *If S is a solution of $f(I_{(3SAT)})$, then the 0-1 assignment S^U with $S^U(x) = S(g(x))$ for all $x \in U$ represents a solution to $I_{(3SAT)}$.*

Then there is a polynomial transformation from (B_2^{CNF}) to the adversarial problem (ADV); in particular, (ADV) is Σ_2^p-complete.

Proof. Since (NON-ADV) is in NP, it is straightforward to see that (ADV) is in Σ_2^p.

Consider an instance $I_{(B_2^{CNF})}$ of the problem (B_2^{CNF}) with disjoint variable sets X^U and Y^U. Let $I_{(3\text{SAT})}$ be the corresponding instance of (3SAT) with the same set $U =$

$X^U \cup Y^U$ of variables and the same set C of clauses, and hence the same Boolean formula E.

Let f be a polynomial transformation with the properties described in the theorem. Let g be the associated bijective function. Let $I_{(\text{NON-ADV})} = f(I_{(\text{3SAT})})$ be the instance of the problem (NON-ADV) resulting from applying the transformation f to $I_{(\text{3SAT})}$. Splitting the variables in the set Z of instance $I_{(\text{NON-ADV})}$ into the disjoint sets X and Y, with $X = g(X^U)$ and $Y = Z \setminus X$ results in an instance $I_{(\text{ADV})}$ of the problem (ADV), for which the set $\mathcal{F}$ is identical to the set of feasible solutions in $I_{(\text{NON-ADV})}$.

We need to show that $I_{(B_2^{CNF})}$ is a yes-instance if and only if $I_{(\text{ADV})}$ is a yes-instance.

So assume that $I_{(B_2^{CNF})}$ is a yes-instance and consider a solution $S^U_{X^U}$ of $I_{(B_2^{CNF})}$. We create a solution S_X for $I_{(\text{ADV})}$ by setting $S_X(z) = S^U_{X^U}(g^{-1}(z))$ for all $z \in X$. Suppose that S_X is not a solution for $I_{(\text{ADV})}$. Then, there is a 0-1 assignment S_Y for the variables in Y such that (S_X, S_Y) is in $\mathcal{F}$. We now define a 0-1 assignment $S^U_{Y^U}$ for the variables in Y^U of instance $I_{(B_2^{CNF})}$ by setting $S^U_{Y^U}(x) = S_Y(g(x))$ for all $x \in Y^U$. By Property (2), $S^U = (S^U_{X^U}, S^U_{Y^U})$ is a 0-1 assignment for the variables in U that satisfies E, and $S^U(x) = S^U_{X^U}(x)$ for all $x \in X^U$. In other words, $S^U_{X^U}$ can be extended by $S^U_{Y^U}$ to a satisfying truth assignment S^U for E. Hence, $S^U_{X^U}$ is not a solution to $I_{(B_2^{CNF})}$, which is a contradiction. Consequently, S_X is a solution of $I_{(\text{ADV})}$, and $I_{(\text{ADV})}$ is a yes-instance.

Assume now that $I_{(B_2^{CNF})}$ is a no-instance. Hence, for every 0-1 assignment $S^U_{X^U}$ of the variables in X^U there is a 0-1 assignment $S^U_{Y^U}$ for the variables in Y^U such that E is satisfied. Consider the corresponding instance $I_{(\text{ADV})}$ and an arbitrary 0-1 assignment S_X for the variables in X. We have to argue that S_X can be extended to a 0-1 solution S of all variables in $Z = X \cup Y$. Let $S^U_{X^U}$ be the corresponding

assignment of the variables in X^U of $I_{(B_2^{CNF})}$ with $S_{X^U}(x) = S_X(g(x))$ for all $x \in X^U$. Let $S(z) = S^U_{Y^U}(g^{-1}(z))$ for all $z \in g(Y^U)$. By now we have an assignment for every variable in $g(U)$, which corresponds to the set of variables Z' in instance $I_{(\text{ADV})}$. According to Property (1), this assignment can be extended to all variables in Z to yield a solution in $\mathcal{F}$. As a consequence, $I_{(\text{ADV})}$ is a no-instance as well. This completes the proof. □

Figure 2-1 illustrates the way in which the function g relates the sets of variables of the instances of the two problems (3SAT) and (NON-ADV) in Theorem 2.1.1. The sets U and $g(U)$ are highlighted. Notice that the (NON-ADV) instance can contain other variables besides the ones in $g(U) = Z'$. The sets Z_0 and Z_1 denote the sets of variables that have value 0 or 1, respectively, in a given solution of the (NON-ADV) instance. Notice also that in any solution of (3SAT), the variables in $U_1 \subseteq U$, that is those having value 1, are mapped into the set Z'_1, the subset of variables in $g(U)$ that have value 1 in the corresponding solution of the instance of (NON-ADV). Similarly, the variables having value 0 (in U_0) are mapped into the set Z'_0, the subset of variables in $g(U)$ that have value 0 in the corresponding solution of the instance of (NON-ADV). In other words, u and $g(u)$ have the same value.

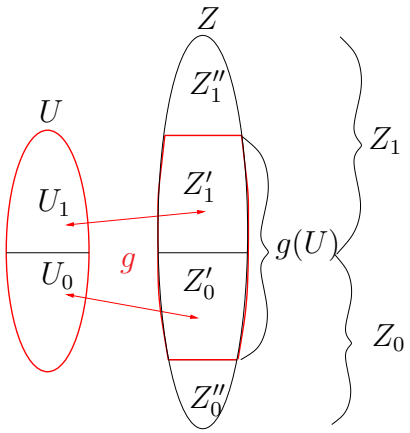

Figure 2-1: Bijective function g between two problems in NP

It should be clear from the proof of Theorem 2.1.1 that (3SAT) can be replaced by any other combinatorial feasibility problem for which the corresponding adversarial problem is Σ_2^p-complete. We call a polynomial transformation f from one combinatorial feasibility problem to another which has the properties described in Theorem 2.1.1 *value preserving*. Note that value preserving transformations are transitive. That is, if we have a transformation f_1 transforming an NP-complete problem Π_1 to a problem Π_2, and a transformation f_2 transforming Π_2 to a problem Π_3, and the problems Π_1, Π_2 and Π_3 as well as the functions f_1 and f_2 satisfy the respective conditions of Theorem 2.1.1, then the composition $f_2 \circ f_1$ of f_1 and f_2, transforming Π_1 to Π_3, satisfies Properties 1 and 2 as well. The consequences of this fact are summarized in the following theorem.

Theorem 2.1.2. *Let Π_1^{adv}, Π_2^{adv}, and Π_3^{adv} be three adversarial problems in Σ_2^p. Let Π_1, Π_2, and Π_3 denote the corresponding non-adversarial problems, respectively. Let f_1 be a polynomial transformation from Π_1 to Π_2, and let f_2 be a polynomial transformation from Π_2 to Π_3. For $i = 1, 2$, we assume that f_i satisfies the following property. There is a bijective function $g_i : Z_i \to Z'_{i+1}$ mapping the ground set Z_i in an instance I_i of Π_i to a subset Z'_{i+1} of the ground set Z_{i+1} in an instance I_{i+1} of Π_{i+1} such that:*

(1) *If S^i is a solution to I_i, then the 0-1 assignment $S^{i+1}_{Z'_{i+1}}$ to the variables in Z'_{i+1} with $S^{i+1}_{Z'_{i+1}}(z) = S^i(g_i^{-1}(z))$ for all $z \in Z'_{i+1}$ can be extended to a 0-1 assignment S^{i+1} of all variables in Z_{i+1} such that S^{i+1} is feasible.*

(2) *If S^{i+1} is a solution of $f_i(I_i)$, then the 0-1 assignment S^i with $S^i(x) = S^{i+1}(g_i(x))$ for all $x \in Z_i$ represents a feasible solution of I_i.*

Then there is a polynomial transformation from the adversarial problem Π_1^{adv} to the adversarial problem Π_3^{adv}. In particular, if Π_1^{adv} is Σ_2^p-complete, then so is Π_3^{adv}.

Proof. The correctness follows from two applications of the proof of Theorem 2.1.1. □

Figure 2-2 illustrates how the functions g_1 and g_2 of the transformations f_1 and f_2 connect the sets of variables of the instances of the problems Π_1, Π_2, and Π_3 as in Theorem 2.1.2.

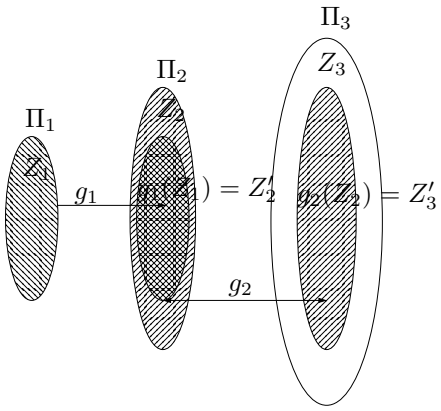

Figure 2-2: Transitive combination of two value preserving polynomial transformations

2.2 Adversarial 3-Dimensional Matching

At this point we want to illustrate the applicability of Theorem 2.1.1 by example of the ADVERSARIAL 3-DIMENSIONAL MATCHING problem. Later, in Section 2.3, we will demonstrate an application of Theorem 2.1.2. The ADVERSARIAL 3-DIMENSIONAL MATCHING problem is defined as follows.

PROBLEM: ADVERSARIAL 3-DIMENSIONAL MATCHING (ADV-3DM)
INSTANCE: Three disjoint sets A, B, and D of size q each, two disjoint sets $M_X \subseteq A \times B \times D$ and $M_Y \subseteq A \times B \times D$.
QUESTION: Is there a subset $M_X^\star$ of M_X so that there is no subset $M_Y^\star$ of M_Y which

would make $M_X^\star \cup M_Y^\star$ a 3-dimensional matching of $A \times B \times D$?

Note that (ADV-3DM) is in Σ_2^p. By exhibiting an explicit transformation from (B_2^{CNF}), McLoughlin [McL84] showed that the ADVERSARIAL 3-DIMENSIONAL MATCHING problem is Σ_2^p-complete. It turns out, however, that the same result follows rather easily from Theorem 2.1.1. The corresponding non-adversarial problem is the 3-DIMENSIONAL MATCHING problem, which belongs to Karp's original list of NP-complete problems [Kar72]. It is defined as follows.

PROBLEM: 3-DIMENSIONAL MATCHING (3DM)
INSTANCE: Three disjoint sets A, B, and D of size q each, set $M \subseteq A \times B \times D$.
QUESTION: Is there a subset $M^\star$ of M such that $M^\star$ represents a 3-dimensional matching of $A \times B \times D$ (that is, $|M^\star| = q$ and no two elements of $M^\star$ agree in any coordinate)?

To see that we can use Theorem 2.1.1 to infer Σ_2^p-completeness for (ADV-3DM), let us recall the transformation between (3SAT) and (3DM) used to show NP-completeness of (3DM) by Karp [Kar72] (see also [GJ79]). Let $I_{(3SAT)}$ be an instance of (3SAT), let U be the set of variables, and let C be the set of clauses. For each variable $u_i \in U$ we introduce $|C|$ elements $b_i^1, \ldots, b_i^{|C|}$ for the set B, another $|C|$ elements $d_i^1, \ldots, d_i^{|C|}$ for the set D, another $|C|$ elements $a_i^1, \ldots, a_i^{|C|}$ for the set A, and yet another $|C|$ elements $\bar{a}_i^1, \ldots, \bar{a}_i^{|C|}$, again for the set A. For each variable $u_i \in U$ we also introduce $2|C|$ ordered triples from these elements for the set M as follows: $U_i = \{(\bar{a}_i^j, b_i^j, d_i^j) : j \in \{1, \ldots, |C|\}\}$, and $\bar{U}_i = \{(a_i^j, b_i^{j+1}, d_i^j) : j \in \{1, \ldots, |C| - 1\}\} \cup \{(a_i^{|C|}, b_i^1, d_i^{|C|})\}$. For each clause $c_j \in C$, $j = 1, \ldots, |C|$, we introduce two elements $r_j \in B$ and $s_j \in D$. Moreover, for each positive literal u_i in c_j, we introduce the ordered triple (a_i^j, r_j, s_j). For each negated literal $\bar{u}_i$ in c_j we introduce the ordered triple $(\bar{a}_i^j, r_j, s_j)$. We combine the triples for each clause c_j to the set C_j.

As the elements b_i^j and d_i^j, $j = 1, \ldots, |C|$, do not occur in any triples outside of $U_i \cup \bar{U}_i$, any 3-dimensional matching $M' \subseteq M$ contains $|C|$ many triples from $U_i \cup \bar{U}_i$. Moreover, because of the way U_i and $\bar{U}_i$ are defined, these triples are either all in U_i or all in $\bar{U}_i$ (see Figure 2-3 for an illustration). This is used to establish the following correspondence to solutions of $I_{(3\text{SAT})}$: $u_i = 1$ if all triples in U_i are chosen, or $u_i = 0$ if all triples in $\bar{U}_i$ are chosen.

As r_j and s_j do not appear outside C_j, any 3-dimensional matching M' contains exactly one triple from each C_j. As this triple can only contain a_i^j (or $\bar{a}_i^j$) if it is not part of the triples chosen to cover $U_i \cup \bar{U}_i$, it points out a literal that satisfies the clause c_j.

To complete the construction of the instance $I_{(3\text{DM})}$ of (3DM) and to match the unused elements of the form a_i^j or $\bar{a}_i^j$, Karp introduced a "garbage collection" component G with the elements $g_k \in B$ and $\bar{g}_k \in D$, $1 \leqslant k \leqslant |C|(|U|-1)$, and another set of triples $G = \{(a_i^j, g_k, \bar{g}_k), (\bar{a}_i^j, g_k, \bar{g}_k) : 1 \leqslant k \leqslant |C|(|U|-1), 1 \leqslant j \leqslant |C|, 1 \leqslant i \leqslant |U|\}$. To summarize, we have

$$\begin{aligned} A &= \{a_i^j : i = 1, \ldots, |U|, j = 1, \ldots, |C|\} \\ &\quad \cup \{\bar{a}_i^j : i = 1, \ldots, |U|, j = 1, \ldots, |C|\} \ , \end{aligned}$$

$$\begin{aligned} B &= \{b_i^j : i = 1, \ldots, |U|, j = 1, \ldots, |C|\} \\ &\quad \cup \{r_j : j = 1, \ldots, |C|\} \\ &\quad \cup \{g_k : k = 1, \ldots, |C|(|U|-1)\} \ , \end{aligned}$$

$$\begin{aligned} D &= \{d_i^j : i = 1, \ldots, |U|, j = 1, \ldots, |C|\} \\ &\quad \cup \{s_j : j = 1, \ldots, |C|\} \\ &\quad \cup \{\bar{g}_k : k = 1, \ldots, |C|(|U| - 1)\} \ , \end{aligned}$$

$$M = \bigcup_{i=1}^{|U|} \bar{U}_i \cup \bigcup_{i=1}^{|U|} U_i \cup \bigcup_{j=1}^{|C|} C_j \cup G \ .$$

In particular, $q = 2|C||U|$. Notice that, given a matching M', the elements a_i^j or $\bar{a}_i^j$ that are not part of $(U_i \cup \bar{U}_i) \cap M'$ or $C_j \cap M'$ will be matched with a pair $(g_k, \bar{g}_k)$. Since there are exactly $|C|(|U| - 1)$ many unmatched elements a_i^j, $\bar{a}_i^j$, they can be covered by the elements in G.

Let us point out how Theorem 2.1.1 applies to the transformation f from 3-SATISFIABILITY to 3-DIMENSIONAL MATCHING that we just described. The set U is the set of variables in a (3SAT) instance $I_{(3SAT)}$. The set Z is (indexed by) the set M of triples in the corresponding (3DM) instance $I_{(3DM)} = f(I_{(3SAT)})$. The set Z' of $I_{(3DM)}$ consists of the first triple in each of the sets U_i, $i = 1, \ldots, |U|$, that is, $Z' = \{(\bar{a}_i^1, b_i^1, d_i^1) : i = 1, \ldots, |U|\}$. The bijective function g maps the elements in U to the elements in Z'. In other words, for each variable $u_i \in U$ of $I_{(3SAT)}$ we have $g(u_i) = (\bar{a}_i^1, b_i^1, d_i^1)$ and thus $Z' = g(U)$. Now, if we have a solution $S_{(3SAT)}$ for $I_{(3SAT)}$, then there is a solution $S_{(3DM)}$ for $I_{(3DM)}$ such that $S_{(3DM)}(z) = S_{(3SAT)}(g^{-1}(z))$ for all $z \in Z'$, that is, if a variable u_i has the value 1 in $S_{(3SAT)}$, then its image $g(u_i)$, which is the triple $(\bar{a}_i^1, b_i^1, d_i^1)$, will belong to the 3-dimensional matching $S_{(3DM)}$. Furthermore, if we have a solution $S_{(3DM)}$ for $I_{(3DM)}$, then the 0-1 assignment $S_{(3SAT)}$ with $S_{(3SAT)}(u_i) = S_{(3DM)}(g(u_i))$ for all $i = 1, \ldots, |U|$ is a solution for $I_{(3SAT)}$.

Since all conditions of Theorem 2.1.1 are met by the problem (ADV-3DM) and the transformation f from (3SAT) to (3DM), we have the following corollary, which

implies McLoughlin's result.

Corollary 2.2.1. ADVERSARIAL 3-DIMENSIONAL MATCHING *is* Σ_2^p*-complete.*

Let us illustrate the transformation from (3SAT) to (3DM) with the help of the following example.

Example 2.2.2. *Let an instance of (3SAT) be given by* $U = \{u_1, u_2, u_3, u_4\}$ *and* $E = (u_1 \vee u_2 \vee u_3) \wedge (u_2 \vee \bar{u}_3 \vee u_4)$*. As a possible solution we provide* $u_1 = 1$*,* $u_2 = 0$*,* $u_3 = 0$*,* $u_4 = 1$*.*

The following Figure 2-3 illustrates the transformation from (3SAT) to (3DM) using Example 2.2.2. The garbage collection components have only been outlined for the first variable u_1. A triangle of edges represents a triple. The triples with dashed edges point out a solution of the (3DM) instance that corresponds to the sample solution of the underlying (3SAT) instance from Example 2.2.2.

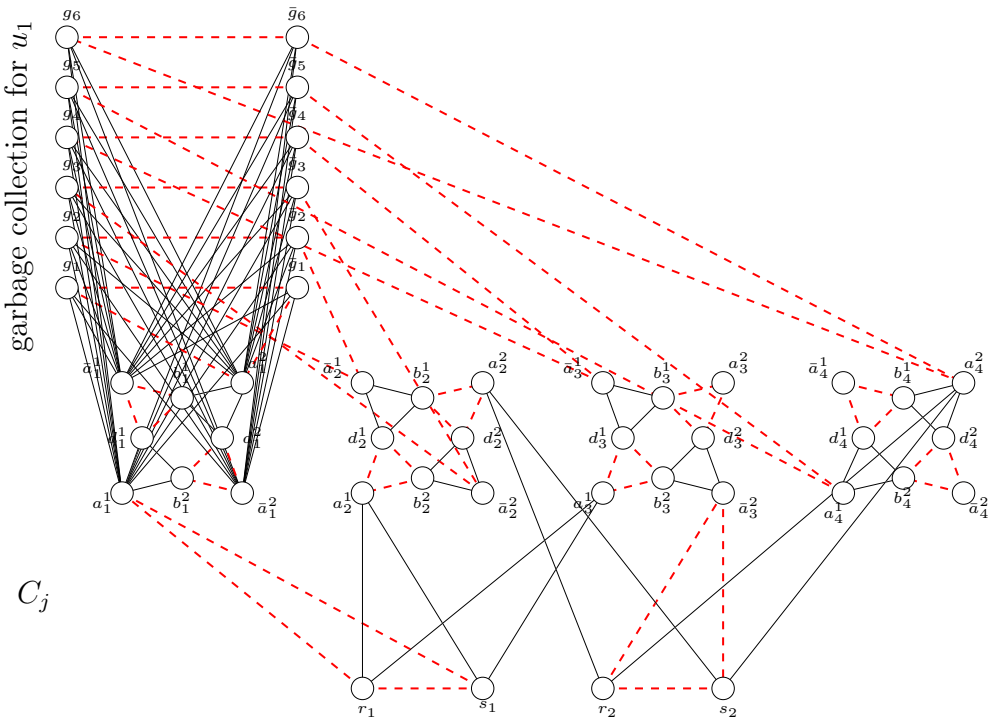

Figure 2-3: Transformation from (3SAT) to (3DM)

Figure 2-4 shows the application of Theorem 2.1.1 to the ADVERSARIAL 3-DIMENSIONAL MATCHING problem.

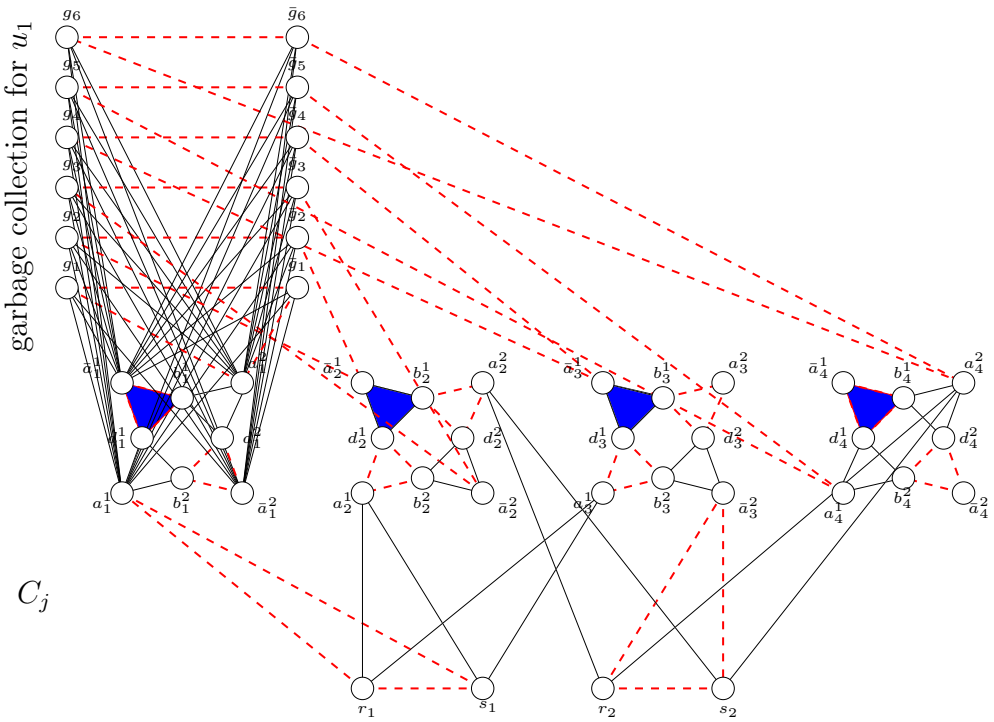

Figure 2-4: Application of Theorem 2.1.1 to (ADV-3DM)

The set M of triples corresponds to the set Z in Theorem 2.1.1. The highlighted triples belong to the set Z', which contains all triples in the image of g. Notice that in order for g to be a bipartite function, we cannot map each variable u_i in U to all triples in U_i. This however does not pose a problem, since, as we observed earlier, either all triples in any given U_i are part of a solution M', or none. Hence, we pick the first triple of each set U_i and let it be the indicator of whether or not $u_i = 1$ in the corresponding solution of $I_{(3\text{SAT})}$. Thus, for each variable $u_i \in U$ of $I_{(3\text{SAT})}$ we have $g(u_i) = (\bar{a}_i^1, b_i^1, d_i^1)$. Consequently we have $Z' = g(U) = \{(\bar{a}_i^1, b_i^1, d_i^1), i = 1, \ldots, |U|\}$. All triples with dashed edges belong to the solution of the matching problem. The triples that are filled form the set $Z' = g(U)$. Among those we recognize that the first triple and the last triple, that is, $\{\bar{a}_1^1, b_1^1, d_1^1\}$ and $\{\bar{a}_4^1, b_4^1, d_4^1\}$, correspond to a variable in U with

value 1 in the solution (u_1 and u_4), whereas the other two correspond to a variable with value 0 (u_2 and u_3).

2.3 Adversarial Partition Problem

One of Karp's feats was to establish a connection between set problems and number problems. In particular, he provided an explicit polynomial transformation from the 3-DIMENSIONAL MATCHING problem to the PARTITION problem, which established NP-completeness of PARTITION [Kar72]. As this transformation satisfies the conditions of Theorem 2.1.2, it allows us to also link set problems and number problems within Σ_2^p.

The ADVERSARIAL PARTITION problem (ADV-PRT) is defined as follows.

PROBLEM: ADVERSARIAL PARTITION (ADV-PRT)
INSTANCE: Two disjoint finite sets X and Y and a positive integer $\ell(n)$ for each $n \in N$ where $N = X \cup Y$.
QUESTION: Is there a subset $X^\star$ of X so that there is no subset $Y^\star$ of Y such that $\sum_{n \in X^\star \cup Y^\star} \ell(n) = \sum_{n \in (X \cup Y) \setminus (X^\star \cup Y^\star)} \ell(n)$?

The corresponding non-adversarial problem is, of course, the PARTITION problem:

PROBLEM: PARTITION (PRT)
INSTANCE: Finite set N, and a positive integer $\ell(n)$ for each $n \in N$.
QUESTION: Is there a subset $N^\star$ of N such that $\sum_{n \in N^\star} \ell(n) = \sum_{n \in N \setminus N^\star} \ell(n)$?

Obviously, the ADVERSARIAL PARTITION problem (ADV-PRT) belongs to Σ_2^p. We will

use the fact that ADVERSARIAL 3-DIMENSIONAL MATCHING is Σ_2^p-complete, which we established in Section 2.2, to show Σ_2^p-completeness for the ADVERSARIAL PARTITION problem.

Consider the following polynomial transformation from 3-DIMENSIONAL MATCHING to PARTITION, suggested by Karp [Kar72]. Let $A = \{a_1, \ldots, a_q\}$, $B = \{b_1, \ldots, b_q\}$, $D = \{d_1, \ldots, d_q\}$, and $M = \{m_1, \ldots, m_k\} \subseteq A \times B \times D$ be an arbitrary instance of (3DM). Let α, β, and δ denote the functions that represent the subscripts for the first, second, and third component of a triple in M. That is, $m_i = (a_{\alpha(i)}, b_{\beta(i)}, d_{\delta(i)})$, $i = 1, \ldots, k$. We create an instance of (PRT) as follows. Let $N = \{n_i : 1 \leqslant i \leqslant k+2\}$. Let $p = \lceil \log_2(k+1) \rceil$. The element n_i in N is associated with the triple m_i in M, for all $i = 1, \ldots, k$. We specify its size $\ell(n_i)$ via its binary representation. There are $3pq$ bits in total, which are divided into $3q$ "zones" of p bits each. See Figure 2-5 for an illustration. The q left-most zones correspond to the elements in A, in the order a_1 to a_q, the next q zones correspond to the elements of B, in the same order, and the q right-most zones correspond to the elements of D, again in that order. The binary representation of $\ell(n_i)$ has 0's everywhere except for the right-most bit in the zones associated with $a_{\alpha(i)}$, $b_{\beta(i)}$, and $d_{\delta(i)}$, where it is 1. This translates to $\ell(n_i) = 2^{p(3q-\alpha(i))} + 2^{p(2q-\beta(i))} + 2^{p(q-\delta(i))}$. It is important to note that the sum of all entries in any zone over all elements $n_1, \ldots n_k$ does not exceed $k = 2^p - 1$. Thus, when we consider $\sum_{n \in N'} \ell(n)$ for any subset $N' \subseteq \{n_i : 1 \leqslant i \leqslant k\}$, the binary representation of the sum of all entries in any zone stays within that zone. Hence, if we define L via the binary representation that has a 1 in the right-most bit of every zone and 0's everywhere else (that is, $L = \sum_{j=0}^{3q-1} 2^{pj}$), then any subset $N' \subseteq \{n_i : 1 \leqslant i \leqslant k\}$ satisfies $\sum_{n \in N'} \ell(n) = L$ if and only if $M' = \{m_i : n_i \in N'\}$ is a 3-dimensional matching for M.

It remains to define n_{k+1} and n_{k+2}. Let $\ell(n_{k+1}) = 2\sum_{i=1}^{k} \ell(n_i) - L$ and $\ell(n_{k+2}) = \sum_{i=1}^{k} \ell(n_i) + L$. Suppose that there is a subset $N' \subseteq N$ such that $\sum_{n \in N'} \ell(n) =$

$\sum_{n\in N\setminus N'} \ell(n)$. Then $\sum_{n\in N'} \ell(n) = \sum_{n\in N\setminus N'} \ell(n) = 2\sum_{i=1}^{k} \ell(n_i)$, and either N' or $N \setminus N'$ contains n_{k+1}, but not n_{k+2}. It follows that the remaining elements of that set form a subset of $\{n_i : 1 \leqslant i \leqslant k\}$ whose sizes sum up to L, and hence that subset corresponds to a 3-dimensional matching in M. On the other hand, if $M' \subseteq M$ is a 3-dimensional matching, then the set $\{n_i : m_i \in M'\} \cup \{n_{k+1}\}$ defines a solution to the PARTITION instance.

We will argue now that this transformation satisfies all assumptions associated with the transformation f_2 in Theorem 2.1.2, where Π_1 corresponds to (3SAT), Π_2 is the 3-DIMENSIONAL MATCHING problem, Π_3 represents the PARTITION problem, and f_1 stands for the value preserving polynomial transformation from (3SAT) to 3-DIMENSIONAL MATCHING that we discussed in Section 2.2. The function g_2 maps each triple m_i in an instance of (3DM) to the element n_i of the corresponding PARTITION instance ($i = 1, \ldots, k$). The preceding discussion of the transformation from 3-DIMENSIONAL MATCHING to PARTITION also showed how a solution to an instance of the 3-DIMENSIONAL MATCHING problem can be extended to a solution of the associated PARTITION instance (Property (1)). For Property (2) we need to be slightly more careful. However, without loss of generality, we may assume that if the PARTITION instance has a solution, then n_{k+1} is part of it. It follows that the preimage of the other elements in that set is a solution to the (3DM) instance.

We obtain the following corollary.

Corollary 2.3.1. ADVERSARIAL PARTITION *is* Σ_2^p*-complete.*

We illustrate the value preserving transformation f_2 from (3DM) to PARTITION in Figure 2-5 below. The figure is based on Example 2.2.2; that is, the (3DM) instance itself resulted from a transformation from the (3SAT) instance of Example 2.2.2. To maintain readability, only the garbage collection components for the (3DM) problem

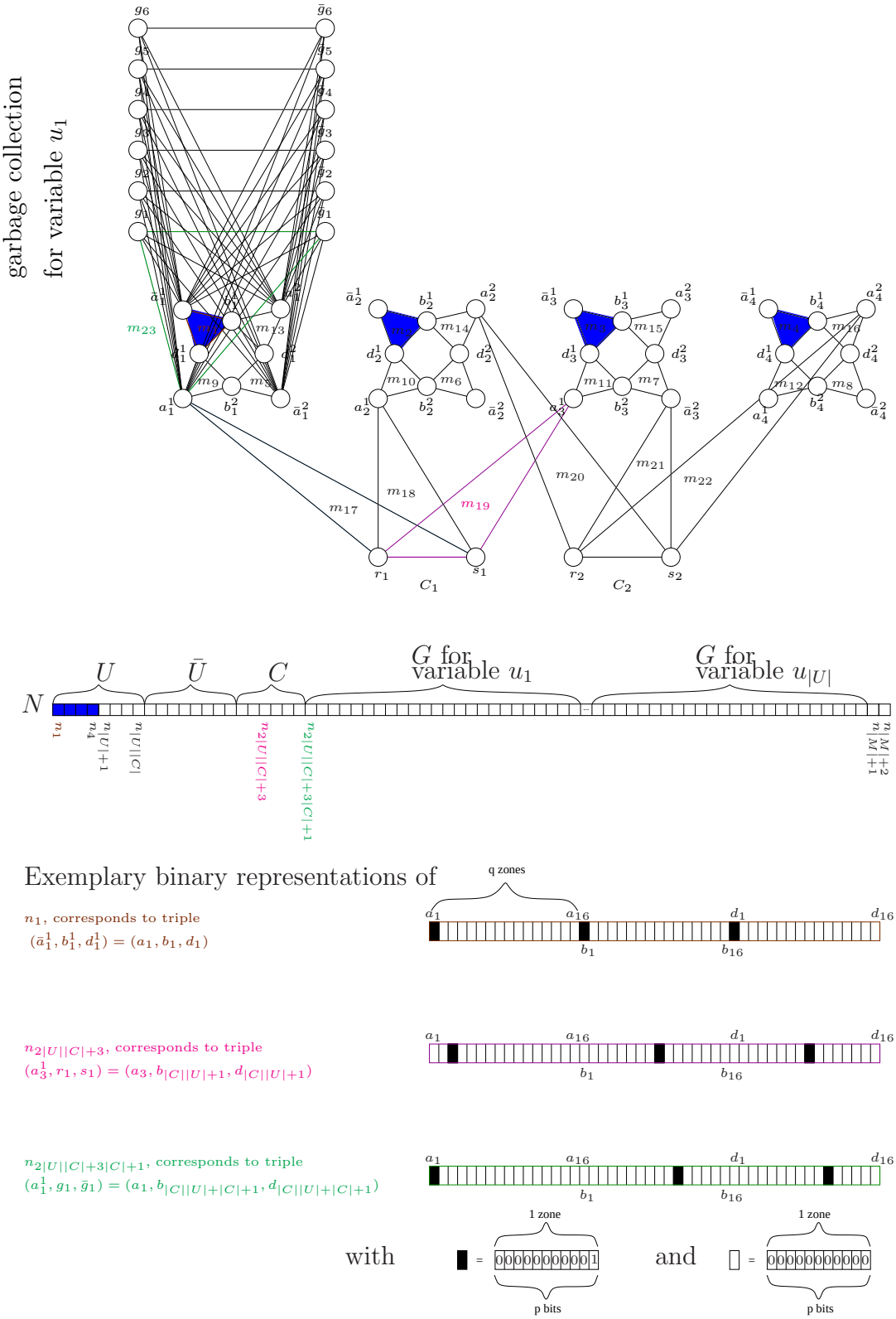

Figure 2-5: Transformation from (3DM) to (PRT)

corresponding to the first variable u_1 of its underlying (3SAT) instance are depicted. Likewise, the binary representations for only a few numbers in the partition instance are sketched. The triples in the set M, as well as the numbers in the set N, are sorted as described in the transformation above. The highlighted triples in the (3DM) graph belong to the set $Z_2' = g_1(U)$, and the numbers with filled boxes belong to the set $g_2(Z_2')$.

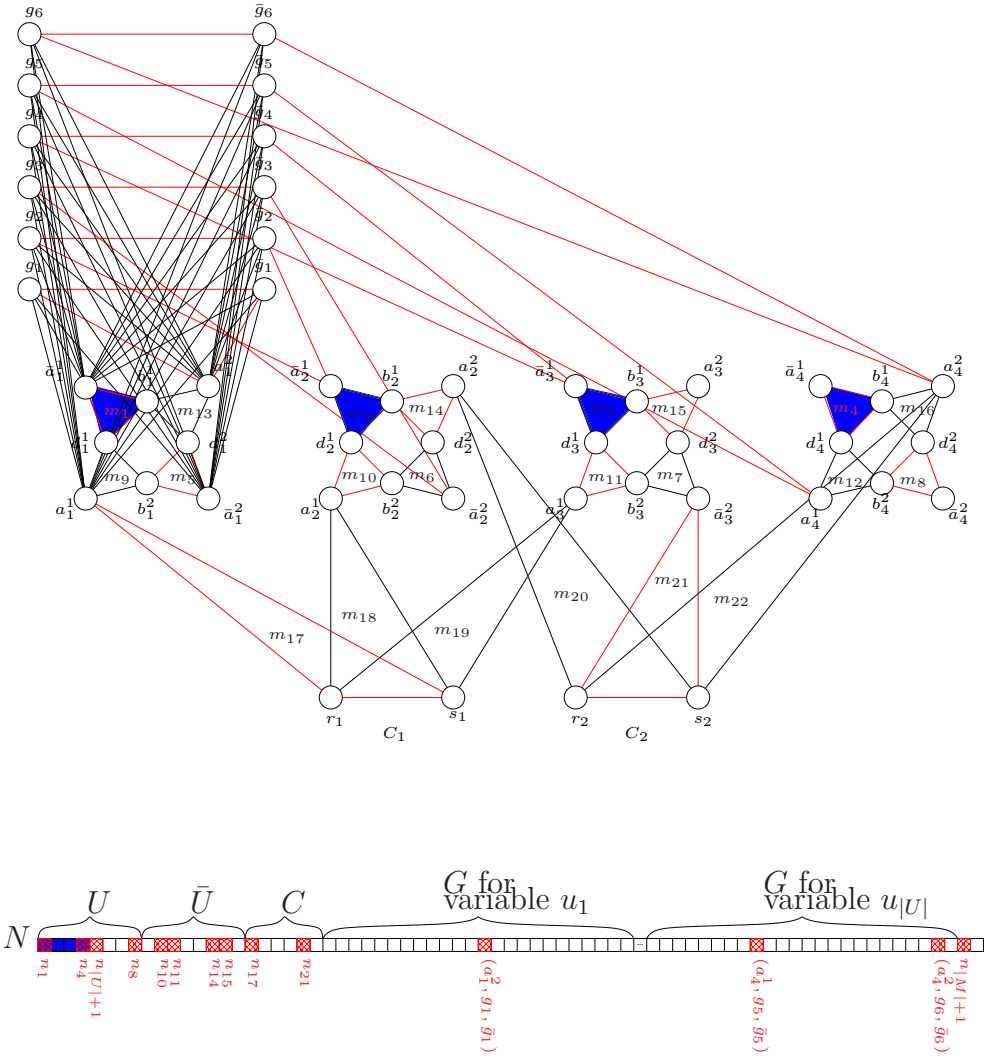

Figure 2-6: Link between the solutions of the (3DM) and the (PRT) instance

Figure 2-6 shows the correspondence between the solution of the (3DM) instance and the partition instance, based on the sample solution in Example 2.2.2. The triples and numbers (pictured as squares) highlighted with bold lines are part of the corresponding solution. The completely filled objects correspond to the image and preimage of the bijective function g.

Chapter 3

Multilevel Programming

Multilevel programming captures the hierarchical structure that is inherent in many decision making processes and provides the means to formulate the problem of coordinating the decisions made at different levels in this hierarchy.

In multilevel mathematical programming, each decision maker controls some variables. Lower-level decision makers may have different objectives than higher-level decision makers. Their objective functions and therefore their (rational) reactions to upper level actions are perfectly known. The highest-level decision maker fixes the values of the variables under her control first, followed by the next highest, and so on. The highest level decision maker optimizes the objective of the highest level of the hierarchical organization, knowing that the lower-level managers will optimize their own objectives in choosing the values of their variables after the leader's decision variables are set. Decisions of lower-level managers are not dictated by their superiors, but are influenced by upper level managers through the values chosen for their control variables. Thus, higher-level decision makers, in making their choices of decision variables, can incorporate the anticipated responses of the lower-level decision makers.

Formally, multilevel optimization problems are mathematical programs which have a

subset of their variables constrained to be an optimal solution of other mathematical programs parameterized by the remaining variables. When these other programs are ordinary mathematical programs we are dealing with bilevel programming. Trilevel programs result when these other programs are themselves bilevel programs. By extending this idea it is possible to define multilevel programs with any number of levels. Of course, the complexity of these problems increases significantly with the number of levels. Multilevel programming problems were first defined and studied by Bracken and McGill [BM73, BM74].

Multilevel optimization is an active and well documented discipline within the field of mathematical programming, see, for example, the bibliography of Vicente and Calamai [VC94]. Applications of multilevel programming are manifold since its structure facilitates the formulation of a number of practical problems that involve a hierarchy of autonomous, and possibly conflictual, decision makers. This includes applications in transportation [Mig95], such as network design problems [LeB73, Mar86, BABBL92, BABB88], toll setting in transportation networks [LMS98], vehicle routing [MMP07], management of hazardous materials [KV04], applications in management, such as coordination of multidivisional firms [Bar83], location of logistics distribution centers [SGW08], competitive location planning [Fis04], network facility location with delivered price competition [MFT92], revenue management [CMS03], and credit allocation [CKR71], social and agricultural policies [CFAM81], electric utility planning [HN92], engineering [FTL99, FTL01, HLDS04, PMSP88], and many others. It can be argued that most managerial decisions are of a multilevel nature, in the sense that they impact systems with some degree of autonomy and conflicting objectives. For more detailed descriptions of applications and further references we refer the reader to [CMS07].

3.1 Bilevel 0-1 Integer Programming

Bilevel programming is an example of a two stage, noncooperative game in which the first player can influence, but not control the actions of the second. It involves two nested optimization problems, an upper one and a lower one. Both problems share the feasible region $\mathcal{F} = \{(x, y) \geqslant 0 : Ax + By \leqslant b\}$, with A and B being integer matrices, and b being an integer vector. The upper decision maker, also called the *leader*, has control over the variables x and makes a decision first, knowing the objective function of the lower decision maker. Then the lower decision maker, also called the *follower*, sets his variables y based on the value of x and according to his own objective function. This basic leader/follower strategy was originally proposed for a duopoly by von Stackelberg [vS34]. Stackelberg games are applicable in particular to policy problems, where policy makers on top influence the decisions of individuals and companies. For instance, a government can impose taxes and set quotas and rationing, to which individuals and companies respond by adjusting their production, and pricing. For a survey of bilevel programming we refer to Ben-Ayed [BA93].

Formally, a bilevel linear program (BLP) is of the following form:

$$\begin{aligned} &\min_{x \geqslant 0} c_1 x + d_1 y \\ &\text{where } y \text{ solves} \quad \max_{y \geqslant 0} d_2 y \\ &\qquad\qquad\quad \text{such that } Ax + By \leqslant b \end{aligned}$$

Here, A and B are integer matrices, and c_1, d_1, d_2, and b are integer vectors of appropriate dimensions. A bilevel integer program (BIP) requires in addition that the variables x and y only assume integer values, a bilevel 0-1 integer program only allows the value 0 and 1 for its variables. A solution of a bilevel linear or integer program is specified by the values of the decision variables of the upper decision maker.

PROBLEM: BILEVEL 0-1 INTEGER PROGRAMMING (BiIP)
INSTANCE: Bilevel 0-1 integer program, integer K.
QUESTION: Is there a 0-1 solution to the bilevel 0-1 integer program such that the objective function value of the leader is at most K?

Jeroslow [Jer85] showed that BILEVEL LINEAR PROGRAMMING is NP-hard. A few years later, Bard [Bar98] and Ben-Ayed and Blair [BAB90] presented shorter proofs. Hansen, Jaumard, and Savard [HJS92] showed that BILEVEL LINEAR PROGRAMMING is strongly NP-hard. More proofs for related hardness results for BILEVEL LINEAR PROGRAMMING have appeared in [Bla92, DWW95, Den98, TKW98].

In fact, Jeroslow showed that MULTILEVEL LINEAR PROGRAMMING with ℓ players is $\Sigma^p_{\ell-1}$-hard, and multilevel integer programming with ℓ players is Σ^p_ℓ-hard. Thus, the complexity of a multilevel programming problem increases by one level in the polynomial hierarchy if either a new player is added, or if we switch from linear to integer constraints.

Σ^p_2-completeness of BILEVEL 0-1 INTEGER PROGRAMMING follows almost immediately from the Σ^p_2-completeness of (B^{CNF}_2). We include the details here because our proof may be a bit more explicit than Jeroslow's original proof.

Theorem 3.1.1. BILEVEL 0-1 INTEGER PROGRAMMING *is* Σ^p_2*-complete.*

Proof. BILEVEL 0-1 INTEGER PROGRAMMING is in Σ^p_2, since we can guess a solution x and then solve the lower optimization problem, which is a 0-1 integer programming problem.

As mentioned before, we will present a polynomial transformation from (B_2^{CNF}) to (BiIP). So let us consider an instance $I_{(B_2^{CNF})}$ of (B_2^{CNF}). The (BiIP) instance contains three kinds of variables, x, y, and z. The x variables correspond to the set X of Boolean variables in $I_{(B_2^{CNF})}$, and the y variables correspond to the set Y in $I_{(B_2^{CNF})}$. In the (BiIP) instance the x variables belong to the upper decision maker, whereas the lower decision maker controls the y and z variables.

For each clause c_j in the set C of clauses in the Boolean expression of $I_{(B_2^{CNF})}$ we introduce a variable z_j and a constraint requiring the sum of the variables corresponding to the literals of that clause to be greater than or equal to z_j. If the variable x_i (or y_i) appears as a positive literal in c_j, then we add x_i (or y_i) for that literal; if it occurs in negated form, however, we add $1 - x_i$ (or $1 - y_i$). For example, the clause $c_j = (x_1 \vee \overline{x}_2 \vee y_1)$ would result in the constraint $x_1 + (1 - x_2) + y_1 \geqslant z_j$. In addition to the clause constraints we also have constraints that ensure that all variables can only assume the value 0 or 1. Thus, if a clause c_j is satisfied, the corresponding variable z_j can assume the value 1, otherwise it has to be 0.

The objective function of the upper player is $\min_x \sum_{j=1}^{|C|} z_j$. The objective function of the lower player is $\max_{y,z} \sum_{j=1}^{|C|} z_j$. Hence, the objective function value of either player is the number of satisfied clauses, and the lower player tries to maximize the number of satisfied clauses, whereas the upper player tries to minimize it. Furthermore, the objective function value of the upper player is less than $|C|$ if and only if $I_{(B_2^{CNF})}$ is a yes-instance. Hence, (BiIP) is Σ_2^p-complete. □

3.2 Trilevel Linear Programming

When adding one hierarchy to a bilevel linear programm, we obtain a trilevel linear program. Hence, the overall set of variables is divided into three subsets, and each

of the three players has control over exactly one of these sets. Every one of the three players has his or her own linear objective function, which depends on all or a subset of the variables. The first player has to fix the values of her variables first, then the second player chooses her values, and at last the third player fixes his variables. Again, it is the objective function value of the first player that matters. Formally, a trilevel linear program LP^3 is of the following form:

$$
\begin{array}{lll}
\min\limits_{x \geqslant 0} c_1 x + d_1 y + e_1 z & & \\
\text{where } y, z \text{ solve} & \min\limits_{y \geqslant 0} d_2 y + e_2 z & \\
& \text{where } z \text{ solves} & \min\limits_{z \geqslant 0} e_3 z \\
& & \text{such that } Ax + By + Cz \leqslant b
\end{array}
$$

Here, A, B, and C are integer matrices, and c_1, d_1, e_1, d_2, e_2, e_3, and b are integer vectors. The TRILEVEL LINEAR PROGRAMMING problem (TriLP) is thus defined as follows.

PROBLEM: TRILEVEL LINEAR PROGRAMMING (TriLP)
INSTANCE: Trilevel linear program, $K \in \mathbb{Q}$.
QUESTION: Is there a solution to the trilevel linear program such that the objective function value of the leader is at most K?

We will show in the following that the ADVERSARIAL PARTITION problem can be formulated as a BILEVEL 0-1 INTEGER PROGRAMMING problem, which in turn can be transformed in polynomial time to a TRILEVEL LINEAR PROGRAMMING problem. This leads to the following theorem.

Theorem 3.2.1. TRILEVEL LINEAR PROGRAMMING *is* Σ_2^p*-hard.*

Proof. We use ideas from Ben-Ayed and Blair [BAB90] who proved that BILEVEL LINEAR PROGRAMMING is NP-hard. They transform a particular integer program into a bilevel linear program and then show that the bilevel linear program, if feasible, always has an optimal feasible solution that is integer. In the same fashion, we first find a suitable Σ_2^p-complete problem, ADVERSARIAL PARTITION, that can be modeled as a bilevel 0-1 integer program and then use their method to transform this bilevel 0-1 integer program into a trilevel linear program. We can show that this trilevel linear program also always has an optimal feasible solution that is integer. But first recall the ADVERSARIAL PARTITION problem, which we showed to be Σ_2^p-complete in Section 2.3.

PROBLEM: ADVERSARIAL PARTITION (ADV-PRT)
INSTANCE: Two disjoint finite sets X and Y, and a positive integer $\ell(a)$ for each $a \in X \cup Y$.
QUESTION: Is there a subset $X^\star$ of X so that there is no subset $Y^\star$ of Y with $\sum_{a\in X^\star \cup Y^\star} \ell(a) = \sum_{a\in (X\cup Y)\setminus(X^\star\cup Y^\star)} \ell(a)$?

The ADVERSARIAL PARTITION problem can be modeled as a trilevel linear program. To build up to it consider first the bilevel 0-1 integer program IP^2 that represents the ADVERSARIAL PARTITION problem. Let $L = \sum_{a\in X\cup Y} \ell(a)/2$.

$$\begin{aligned} &\min_{x\in\{0,1\}^X} \sum_{a\in X} x_a\ell(a) + \sum_{a\in Y} y_a\ell(a) \\ &\quad \text{where } y \text{ solves} \qquad \max_{y\in\{0,1\}^Y} \sum_{a\in X} x_a\ell(a) + \sum_{a\in Y} y_a\ell(a) \\ &\qquad\qquad\qquad\qquad \text{such that} \qquad \sum_{a\in X} x_a\ell(a) + \sum_{a\in Y} y_a\ell(a) \leqslant L \end{aligned}$$

If in an instance $I_{(\text{ADV-PRT})}$ of the ADVERSARIAL PARTITION problem $\sum_{a\in X} x_a\ell(a) > L$, then $I_{(\text{ADV-PRT})}$ is trivial to solve. Thus we can assume that $\sum_{a\in X} x_a\ell(a) \leqslant L$. Furthermore, if $I_{(\text{ADV-PRT})}$ is a yes-instance, then $\sum_{a\in X^\star} x_a\ell(a) \neq L$ since otherwise this

would yield a $Y^\star$ such that $\sum_{a\in X^\star\cup Y^\star} \ell(a) = L$, and thus a partition of $X \cup Y$.

Lemma 3.2.2. *Let $I_{(ADV\text{-}PRT)}$ be an instance of the* ADVERSARIAL PARTITION *problem with $\sum_{a\in X} x_a\ell(a) \leqslant L$. Let IP^2 be the corresponding bilevel 0-1 integer program. The optimal objective function value of the X-player of IP^2 is smaller than L if and only if $I_{(ADV\text{-}PRT)}$ is a yes-instance.*

Proof. Assume that $I_{(\text{ADV-PRT})}$ is a yes-instance, that is, there is an $X^\star \subseteq X$ such that there is no subset $Y^\star$ of Y with $\sum_{a\in X^\star\cup Y^\star} \ell(a) = L$. Consider the 0-1 assignment for IP^2 with $x_a = 1$ for all $a \in X^\star$ and $x_a = 0$ for all $a \in X\setminus X^\star$. Based on this assignment for X, the Y-player tries to maximize his objective function. If he succeeded to obtain $\sum_{a\in X} x_a\ell(a) + \sum_{a\in Y} y_a\ell(a) = L$, then the corresponding set $X^\star \cup Y^\star$ with $a \in Y^\star$ if $y_a = 1$ and $a \in Y \setminus Y^\star$ if $y_a = 0$ for $a \in Y$ defines a partition for $I_{(\text{ADV-PRT})}$, which contradicts our assumption. Thus, $\sum_{a\in X} x_a\ell(a) + \sum\limits_{a\in Y} y_a\ell(a) < L$.

Now assume that $I_{(\text{ADV-PRT})}$ is a no-instance, that is, for all $X^\star \subseteq X$ there exists $Y^\star \subseteq Y$ such that $\sum_{a\in X^\star\cup Y^\star} \ell(a) = L$. Consider any 0-1 assignment for the x variables of IP^2. Let $X^\star$ be the corresponding subset of X in $I_{(\text{ADV-PRT})}$ such that $a \in X^\star$ if $x_a = 1$ in X and $a \in X \setminus X^\star$ if $x_a = 0$ in X. Hence, for $X^\star$ there is a $Y^\star \subseteq Y$ such that $\sum_{a\in X^\star\cup Y^\star} \ell(a) = L$. It follows that there is a 0-1 assignment for the y variables in IP^2 with $y_a = 1$ if $a \in Y^\star$ and $y_a = 0$ if $a \in Y \setminus Y^\star$. Moreover, $\sum_{a\in X} x_a\ell(a) + \sum\limits_{a\in Y} y_a\ell(a) = L$. This concludes the proof. □

Hence, IP^2 delivers yet another way of proving that (BiIP) is Σ_2^p-complete. We will now transform IP^2 to a trilevel linear program LP^3 by introducing a new player for the third level and new variables s over which he has control. These s-variables function as slack variables, which capture the distance of the values of the x-variables

and y-variables to their respective nearest integer.

Let $L_{\max} = \max_{a \in X \cup Y} \ell(a)$. Let $M = L_{\max}^2$. LP^3 is defined as follows:

$$\min_x \sum_{a \in X} x_a \ell(a) + \sum_{a \in Y} y_a \ell(a) + M \sum_{a \in X} s_a + M \sum_{a \in Y} s_a$$

where y solves

$$\max_y \sum_{a \in X} x_a \ell(a) + \sum_{a \in Y} y_a \ell(a) - M \sum_{a \in X} s_a - M \sum_{a \in Y} s_a$$

where s solves

$$\max_s \sum_{a \in X} s_a + \sum_{a \in Y} s_a$$

such that

$$\sum_{a \in X} x_a \ell(a) + \sum_{a \in Y} y_a \ell(a) \leqslant L$$

$$s_a \leqslant x_a \quad \text{and} \quad s_a \leqslant 1 - x_a \quad \text{for all } a \in X$$
$$s_a \leqslant y_a \quad \text{and} \quad s_a \leqslant 1 - y_a \quad \text{for all } a \in Y$$

$$x_a \geqslant 0 \quad \text{and} \quad x_a \leqslant 1 \quad \text{for all } a \in X$$
$$y_a \geqslant 0 \quad \text{and} \quad y_a \leqslant 1 \quad \text{for all } a \in Y$$

We use $s(x)$ and $s(y)$ to refer to the s variables associated with x and y, respectively. Note that the constraints and the objective function of the third player imply that $s_a = \min\{x_a, 1 - x_a\}$ for all $a \in X$ and $s_a = \min\{y_a, 1 - y_a\}$ for all $a \in Y$, no matter

which values x and y may have. We may therefore restrict ourselves to solutions of this kind and specify the values of x and y only. That is, whenever we consider a feasible solution (x, y), it is implicitly assumed that (x, y) represents $(x, y, s(x), s(y))$, where $(s(x), s(y))$ are chosen as described above. We claim that for every feasible solution (x, y) which contains at least one fractional component in x or y, there is a feasible solution $(x^\star, y^\star)$ that contains at least one less fractional component than (x, y) and for which the objective function value is the same or better for both the X-player and the Y-player. As a consequence, the optimal solution of LP^3 is integer.

Lemma 3.2.3. *Let* $f(x, y) = \sum_{a \in X} x_a \ell(a) + \sum_{a \in Y} y_a \ell(a)$. *Let* $g(s(x), s(y)) = \sum_{a \in X} s_a + \sum_{a \in Y} s_a$. *Let* $L_{\max} = \max_{a \in X \cup Y} \ell(a)$. *Let* $M = L_{\max}^2$. *Let* (x, y) *be a feasible solution of* LP^3 *that contains at least one fractional component. Then there is a solution* $(x^\star, y^\star)$ *of* LP^3 *with more integer components in* $(x^\star, y^\star)$ *than in* (x, y), *and*

(1) $f(x, y) + Mg(s(x), s(y)) \geqslant f(x^\star, y^\star) + Mg(s(x^\star), s(y^\star))$,

(2) $f(x, y) - Mg(s(x), s(y)) \leqslant f(x^\star, y^\star) - Mg(s(x^\star), s(y^\star))$.

Moreover, the solution is feasible. In particular,

(3) $f(x^\star, y^\star) \leqslant L$.

Proof. Without loss of generality we assume that $L_{\max} \geqslant 2$ since any instance of the ADVERSARIAL PARTITION problem with $\ell(a) = 1$ for all $a \in X \cup Y$ is trivial to solve.

We consider the following cases.

(a) If (x, y) contains a fractional component z_k with $0 < z_k \leqslant 1 - \frac{1}{L_{\max}}$, then let $(x^\star, y^\star)$ be (x, y) with $z_k^\star$ changed to 0.

(b) If (x, y) contains two fractional components z_j and z_k, with $1 - \frac{1}{L_{\max}} < z_k < 1$ and $1 - \frac{1}{L_{\max}} < z_j < 1$, then choose $z_j^\star$ and $z_k^\star$ such that $z_j^\star \ell(j) + z_k^\star \ell(k) = z_j \ell(j) + z_k \ell(k)$ and $z_j^\star + z_k^\star \geqslant z_j + z_k$, and $z_j^\star = 1$ or $z_k^\star = 1$.

(c) If (x, y) contains only one fractional component z_k, and that fractional component z_k is greater than $1 - \frac{1}{L_{\max}}$, then let $(x^\star, y^\star)$ be (x, y) with $z_k^\star$ changed to 1.

We show that for each of the cases (a), (b), and (c), the objective function value for the X-player does not increase (which corresponds to statement (1)), the objective function value for the Y-player does not decrease (see statement (2)), and the main constraint (equaling statement (3)) still holds, when changing the solution (x, y) to $(x^\star, y^\star)$, as described.

Consider case (a).
Statement (1) is easily seen since $z_k \ell(k) + Ms(z_k) > 0$. For statement (2) we want to show that $z_k \ell(k) - Ms(z_k) \leq 0$. If $z_k \leqslant 1/2$ we have $z_k = s(z_k)$, because our solution maximizes $\sum_{a \in X} s_a + \sum_{a \in Y} s_a$. Since $\ell(k) \leqslant L_{\max} < L_{\max}^2 = M$, the statement is true in this case. If $z_k > 1/2$ we have $s(z_k) = 1 - z_k$, again because of optimality for $s(x)$ and $s(y)$, and hence $z_k \ell(k) - Ms(z_k) \leqslant z_k L_{\max} - L_{\max}^2(1 - z_k) \leqslant (1 - \frac{1}{L_{\max}})(L_{\max} + L_{\max}^2) - L_{\max}^2 = -1 < 0$. The feasibility (statement (3)) is not disturbed by changing z_k to 0.

Consider now case (b).
For statement (1) it needs to be shown that $z_j \ell(j) + z_k \ell(k) + Ms(z_j) + Ms(z_k) \geqslant z_j^\star \ell(j) + z_k^\star \ell(k) + Ms(z_j^\star) + Ms(z_k^\star)$. Since $z_j^\star \ell(j) + z_k^\star \ell(k) = z_j \ell(j) + z_k \ell(k)$ it remains to show that $s(z_j) + s(z_k) \geqslant s(z_j^\star) + s(z_k^\star)$. Let's assume without loss of generality that $z_j^\star = 1$. Since $z_j > 1 - 1/L_{\max} \geqslant 1/2$ and because of the optimality of $s(x)$ and $s(y)$, we have $s(z_j) = 1 - z_j$. Since $z_j^\star + z_k^\star \geqslant z_j + z_k$ it follows that $s(z_j) \geqslant z_k - z_k^\star$.

Also, $s(z_j^\star) = 0$. Consequently, $s(z_j) + s(z_k) \geqslant z_k - z_k^\star + 1 - z_k = 1 - z_k^\star \geqslant s(z_k^\star)$, and the statement holds. Statement (2) is in this case identical to the statement (1) and therefore true. Statement (3) holds naturally since $z_j^\star \ell(j) + z_k^\star \ell(k) = z_j \ell(j) + z_k \ell(k)$.

Now consider case (c).
To prove statement (1) we need to show that $z_k \ell(k) + Ms(z_k) > \ell(k)$. We observe that with $z_k > 1 - \frac{1}{L_{\max}}$, and, thus, $s_k = 1 - z_k$, $M = L_{\max}^2$, and $\ell(k) \leqslant L_{\max}$, this inequality can be reduced to $L_{\max} > 1$, which is true. Statement (2) holds since $z_k < 1 = z_k^\star$ and hence $z_k \ell(k) - Ms(z_k) < z_k^\star \ell(k)$. For statement (3) note that $\sum_{a \in X \cup Y} z_a^\star \ell(a) < \sum_{a \in X \cup Y} z_a \ell(a) + 1 \leqslant L + 1$. Since $\sum_{a \in X \cup Y} z_a^\star \ell(a)$ and L are integer, we conclude that $\sum_{a \in X \cup Y} z_a^\star \ell(a) \leqslant L$. □

We continue with the proof of Theorem 3.2.1. Notice that the objective function values strictly improve for both the X-player and the Y-player in case (a) and (c). Only in case (b) it could remain the same. However, no fractional feasible solution $(x^\star, y^\star)$ can be transformed into an integer feasible solution by just using the method of case (b), since at least one component remains fractional when using just method (b). Therefore we have to use at least once the method of case (a) or (c) and thus achieve that the objective function values for both the X-player and the Y-player strictly improve when turning a fractional solution into an integer one by the methods described in cases (a), (b), and (c). Since each player acts selfishly according to their own objective function, the optimal integer solution will indeed be assumed by LP^3.

Thus, we showed how one can model any instance of the problem ADVERSARIAL PARTITION by a trilevel linear program LP^3, which, as we have shown with Lemma 3.2.3, has an optimal integer solution. Since ADVERSARIAL PARTITION is Σ_2^p-complete, see Section 2.3, it follows that TRILEVEL LINEAR PROGRAMMING is Σ_2^p-hard. □

Chapter 4

Preprocessing Problems

Preprocessing problems are closely related to the class of partial inverse optimization problems. Given a combinatorial optimization problem, the corresponding inverse optimization problem is to find a minimal adjustment of the cost function such that a given feasible solution becomes optimal. A partial inverse optimization problem is more general than an inverse optimization problem in that only some part of a feasible solution is given. One wants to find a smallest modification of the cost function such that there is an optimal solution that is compatible with the given partial solution. In preprocessing problems, we are given only one element and the goal is to find out whether this element can be part of an optimal solution to the given optimization problem. One usually assumes that the objective function is not entirely known, but is chosen from some known universe. Here, we consider the variant in which individual bounds for each component of the objective function are specified beforehand. Thus, given an instance of a combinatorial optimization problem $\min_{S \in \mathcal{F}} \sum_{j \in S} c_j$, where $\mathcal{F}$ is a collection of subsets of a ground set Z, the question is, given an element $e \in Z$, and vectors ℓ and u, is there a cost vector c' with $\ell \leqslant c' \leqslant u$ and a solution $S' \in \mathcal{F}$ such that $e \in S'$, and S' is optimal with respect to $\min_{S \in \mathcal{F}} \sum_{j \in S} c'_j$? (Assuming ℓ and u are integer, it is not difficult to see that there exists an integer cost vector c' for which S' is optimal, if there exists such a cost vector at all. We therefore assume throughout

this chapter that ℓ, u, and c' are integer.) Of course, if the answer is no, then e may be safely eliminated and the problem can be simplified. It is therefore not surprising that preprocessing problems have a variety of practical applications. For instance, they are of particular interest to real time optimization, where problems may include many variables and where it is important to solve problems quickly. If the feasible region of the optimization problem and some bounds for the cost vector are known in advance, but the exact coefficients of the cost vector will only become known later, preprocessing can eliminate variables from the problem that will not be part of an optimal solution. This enables faster solutions of the problem at a time when the precise costs become eventually known.

A comprehensive survey on inverse optimization was compiled by Heuberger [Heu04]. The concept of partial inverse optimization was introduced by Orlin [Orl], who also gives a number of complexity results. Lai and Orlin [LO03] derived some initial results on preprocessing problems. However, so far no preprocessing problem was shown to be Σ_2^p-complete. In this chapter, we will show that the preprocessing problems that can be associated with NP-complete problems are Σ_2^p-complete.

4.1 Preprocessing Satisfiability

We begin by establishing Σ_2^p-completeness of the preprocessing problem associated with 3-SATISFIABILITY. For this purpose, recall the Σ_2^p-complete 2-ALTERNATING QUANTIFIED SATISFIABILITY problem.

PROBLEM: 2-ALTERNATING QUANTIFIED SATISFIABILITY (B_2^{CNF})
INSTANCE: Sets X and Y of variables, Boolean expression E over $X \cup Y$ in conjunctive normal form with exactly 3 variables in each clause.
QUESTION: Is there a truth assignment for X such that there is no truth assignment

for Y so that E would be satisfied?

The preprocessing version of (3SAT) is defined as follows:

PROBLEM: PREPROCESSING 3SAT (PP 3SAT)
INSTANCE: Set U of Boolean variables, Boolean expression E over U in conjunctive normal form with exactly three variables in each clause, bounds $\ell(z)$ and $u(z)$ for each variable $z \in U$, variable $z^\star \in U$.
QUESTION: Does there exist a satisfying truth assignment S_U for U with $S_U(z^\star) = 1$ and a cost vector c on U with $\ell(z) \leqslant c(z) \leqslant u(z)$ for all $z \in U$ such that S_U minimizes $\sum_{z \in U} c(z)S(z)$ over all satisfying truth assignments S?

To show Σ_2^p-completeness for (PP 3SAT) we take a small detour via the problem PREPROCESSING 4SAT (PP 4SAT), which differs from (PP 3SAT) only in that every clause may contain up to four variables.

Theorem 4.1.1. *(PP 4SAT) is Σ_2^p-complete.*

Proof. First observe that the problem (PP 4SAT) lies in Σ_2^p. To see that, consider the MIN COST SATISFIABILITY problem which, provided a Boolean formula E, a linear cost function c on the set of variables, and an integer K, asks whether there is a satisfying truth assignment for E of total cost at most K. The MIN COST SATISFIABILITY problem is obviously in NP. Therefore we can, for an instance of (PP 4SAT), guess a truth assignment S_U and a cost vector c, and verify via the MIN COST SATISFIABILITY problem whether S_U is minimal with respect to c. Checking that c is within its bounds and that $S_U(z^\star) = 1$ is trivial.

We now present a polynomial transformation from (B_2^{CNF}) to (PP 4SAT). Let $I_{(B_2^{CNF})}$ be a (B_2^{CNF}) instance with variable sets X and Y and Boolean expression $E_{(B_2^{CNF})}$. We

construct an instance $I_{(\text{PP 4SAT})}$ of (PP 4SAT) as follows. We add a new variable $z^\star$ to the set of variables in $I_{(B_2^{CNF})}$ to obtain the new set of variables $U = X \cup Y \cup \{z^\star\}$. We add the positive literal $z^\star$ to each clause in $E_{(B_2^{CNF})}$, resulting in the Boolean expression $E_{(\text{PP 4SAT})}$ in conjunctive normal form with exactly 4 variables per clause. We set the bounds on the costs for the variables in $U = X \cup Y \cup \{z^\star\}$ as follows. For every variable $x \in X$ we set $\ell(x) = -1$ and $u(x) = 1$. For every variable $y \in Y$ we set $\ell(y) = u(y) = 0$, and for the variable $z^\star$ we set $\ell(z^\star) = u(z^\star) = 1$. Clearly, the transformation from (B_2^{CNF}) to (PP 4SAT) is polynomial.

Consider the case that $I_{(B_2^{CNF})}$ is a yes-instance, that is, there is a truth assignment S_X for the variables in X such that there is no truth assignment for the variables in Y for which $E_{(B_2^{CNF})}$ would be satisfied. Let S_X be such a truth assignment for the variables in X of $I_{(B_2^{CNF})}$. Let $S_{(\text{PP 4SAT})}$ be a truth assignment for the variables in $I_{(\text{PP 4SAT})}$ such that $S_{(\text{PP 4SAT})}(z^\star) = 1$ and $S_{(\text{PP 4SAT})}(x) = S_X(x)$ for all $x \in X$. Consider the following cost vector c. Let $c(x) = -1$ for all $x \in X$ with $S_{(\text{PP 4SAT})}(x) = 1$. Let $c(x) = 1$ for all $x \in X$ with $S_{(\text{PP 4SAT})}(x) = 0$. Let $c(y) = 0$ for all $y \in Y$. And finally let $c(z^\star) = 1$.

We claim that $S_{(\text{PP 4SAT})}$ is a satisfying truth assignment for $I_{(\text{PP 4SAT})}$ that is minimal with respect to c. Clearly, since $S_{(\text{PP 4SAT})}(z^\star) = 1$, the Boolean expression $E_{(\text{PP 4SAT})}$ is satisfied. The cost of $S_{(\text{PP 4SAT})}$ is $-|\{x \in X : S_{(\text{PP 4SAT})}(x) = 1\}| + 1$, due to the cost of -1 for each variable in X with value 1 in $S_{(\text{PP 4SAT})}$, and the cost of 1 of having $S_{(\text{PP 4SAT})}(z^\star) = 1$. Changing the value of any variable in X will increase the cost by at least 1. Therefore, in order to reduce the costs, we can only try to change the truth assignment of $z^\star$ in $S_{(\text{PP 4SAT})}$ to 0. As a consequence, $E_{(\text{PP 4SAT})}$ is not automatically satisfied anymore by the assignment of $z^\star$. In other words, with $S_{(\text{PP 4SAT})}(z^\star) = 0$, the Boolean expression $E_{(\text{PP 4SAT})}$ is satisfied if and only if $E_{(B_2^{CNF})}$ is satisfied. Since $I_{(B_2^{CNF})}$ is a yes-instance, it follows that it is not possible to complement the truth assignments for the variables in $X \cup \{z^\star\}$ with truth

assignments for the variables in Y such that $E_{(\text{PP 4SAT})}$ is satisfied. Thus, if we want $S_{(\text{PP 4SAT})}(z^\star) = 0$, we cannot keep the truth assignment of S_X for the variables in X. As we observed above, changing the truth assignment for variables in X would result in a cost increase of at least 1, which would hence offset any savings from setting $S_{(\text{PP 4SAT})}(z^\star)$ to 0. Therefore, $S_{(\text{PP 4SAT})}$ is optimal with respect to the costs c and $I_{(\text{PP 4SAT})}$ is a yes-instance as well.

Consider now the case that $I_{(B_2^{CNF})}$ is a no-instance, that is, for each truth assignment for the variables in X, there is a truth assignment for the variables in Y such that the expression $E_{(B_2^{CNF})}$ is satisfied. Let c be a cost vector with $\ell(z) \leq c(z) \leq u(z)$ for all $z \in U$, and let $S_{z^\star=1}$ be an arbitrary satisfying truth assignment for $E_{(\text{PP 4SAT})}$ such that $S_{z^\star=1}(z^\star) = 1$. Moreover, let $S_{z^\star=0}$ be a satisfying truth assignment for $E_{(\text{PP 4SAT})}$ such that $S_{z^\star=0}(x) = S_{z^\star=1}(x)$ for all $x \in X$ and $S_{z^\star=0}(z^\star) = 0$. Note that $S_{z^\star=0}$ exists because every truth assignment for the variables in X can be extended to the variables in Y such that $E_{(B_2^{CNF})}$ is satisfied.

Clearly, the costs of $S_{z^\star=0}$ are smaller than the costs of $S_{z^\star=1}$ since the truth assignment for the variables in X are identical in both assignments, the costs coefficients for the nodes in Y are 0 due to the tight lower and upper bounds on the costs, and the cost of having $S_{z^\star=1}(z^\star) = 1$ is 1 versus the cost of 0 for having $S_{z^\star=0}(z^\star) = 0$. It follows that for any cost function c within the bounds ℓ and u, there is no truth assignment giving the value 1 to the variable $z^\star$ that is optimal with respect to c. In other words, $I_{(\text{PP 4SAT})}$ is a no-instance as well. □

Now we can add one small step to transform (PP 4SAT) to (PP 3SAT) to prove Σ_2^p-completeness for the problem (PP 3SAT).

Theorem 4.1.2. *(PP 3SAT) is Σ_2^p-complete.*

Proof. We can use the same argument as in the preceding proof to show that (PP 3SAT) is in Σ_2^p. Now, let $I_{(\text{PP 4SAT})}$ be an instance of (PP 4SAT) with the Boolean expression $E_{(\text{PP 4SAT})}$. We introduce a new variable z_i for each clause C_i in $E_{(\text{PP 4SAT})}$, and we replace each clause $C_i = (u_1 \vee u_2 \vee u_3 \vee u_4)$ in $E_{(\text{PP 4SAT})}$ with the two clauses $(u_1 \vee u_2 \vee z_i)$ and $(u_3 \vee u_4 \vee \overline{z}_i)$, a transformation first proposed by Cook [Coo71]. The bounds for the costs of the original variables remain the same, the new variables receive 0 as lower and upper bounds on their costs and $z^\star$ remains $z^\star$, completing this straightforward transformation. It is easy to see that a yes-instance of (PP 4SAT) corresponds to a yes-instance of (PP 3SAT) and a no-instance of (PP 4SAT) corresponds to a no-instance of (PP 3SAT). □

Note that the proof of Theorem 4.1.1 implies that (PP 3SAT) remains Σ_2^p-complete even if the underlying (3SAT) instance is known to be satisfiable. In other words, we could have formulated (PP 3SAT) for satisfiable (3SAT) instances only.

4.2 A Generic Class of Preprocessing Problems

We are now going to show that not only SATISFIABILITY problems have a preprocessing version which is Σ_2^p-complete. In fact, many NP-hard problems possess a corresponding preprocessing version, which, if certain conditions are met, is Σ_2^p-complete.

One such problem would be, for example, the preprocessing version of VERTEX COVER, which we will discuss in detail in Section 4.3. The input of the preprocessing version of VERTEX COVER, (PP-VC), consists of a graph $G = (V, E)$, a specific node $v^\star \in V$, bounds $\ell(v)$ and $u(v)$ for each node v in V, and an integer K. The question is whether there is a cost vector c on the nodes of G with $\ell(v) \leqslant c(v) \leqslant u(v)$ for each $v \in V$ and a vertex cover of size at most K that contains the node $v^\star$ and that is minimal with respect to c. The problem (PP-VC) lies in Σ_2^p, because given a cost vector c and a vertex cover V', one can solve an instance of the cardinality constrained

min-cost vertex cover problem, which is in NP, to verify the optimality of the solution (c, V'). By exploiting certain properties of the polynomial transformation from 3-SATISFIABILITY to VERTEX COVER, we will derive a polynomial transformation from (PP 3SAT) to (PP-VC), which implies that the latter problem is Σ_2^p-complete.

As mentioned before, the 3-SATISFIABILITY problem can once again be replaced with any other NP-complete problem Π_1 which possesses a suitable Σ_2^p-complete version. Similarly, VERTEX COVER can be replaced with any other suitable NP-complete problem Π_2 for which there is a corresponding preprocessing problem, and for which there is a value preserving polynomial transformation from Π_1 to Π_2. That is, we can prove Σ_2^p-completeness for a wide variety of preprocessing problems at once, by making use of the properties of value preserving polynomial transformations employed in the NP-completeness proofs of the underlying non-preprocessing problems, similarly to what we did for adversarial problems in Chapter 2. Informally, we can say that it again suffices if the polynomial transformation involves a mapping that allows us, given a solution to one of the instances, to easily read off the solution of the other instance, and vice versa. Formally, we recall that a polynomial transformation f from a combinatorial feasibility problem Π_1 to a combinatorial feasibility problem Π_2 is value preserving if it has the following properties: If U is the set of variables of an instance I of Π_1 and Z is the set of variables of the corresponding instance $f(I)$ of Π_2, then there is a subset $Z' \subseteq Z$ and a bijective function $g : U \to Z'$ such that:

(1) If S^U is a solution of I, then the 0-1 assignment $S_{Z'}$ with $S_{Z'}(z) = S^U(g^{-1}(z))$ for all $z \in Z'$ can be extended to a 0-1 assignment S of all variables in Z such that S is a feasible solution for $f(I)$.

(2) If S is a solution of $f(I)$, then the 0-1 assignment S^U with $S^U(x) = S(g(x))$ for all $x \in U$ represents a solution to I.

We will first illustrate our result on value preserving polynomial transformations orig-

inating from (3SAT). Recall that the preprocessing version of (3SAT) is (PP 3SAT), which was proved to be Σ_2^p-complete in the previous section. We use (NON-PP) to denote the generic non-preprocessing problem to which there is a value preserving polynomial transformation from the 3-SATISFIABILITY problem. It is the same combinatorial feasibility problem as in Chapter 2.

PROBLEM: COMBINATORIAL FEASIBILITY PROBLEM (NON-PP)
INSTANCE: Finite ground set Z of variables, (implicitly given) set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible 0-1 assignments for the variables in Z.
QUESTION: Is $\mathcal{F}$ non-empty?

We want to show Σ_2^p-completeness of the preprocessing version that corresponds to the problem (NON-PP), which is defined as follows.

PROBLEM: PREPROCESSING PROBLEM (PP)
INSTANCE: Same ground set Z as in (NON-PP), bounds $\ell(z)$ and $u(z)$ for each element $z \in Z$, special element $z^\star \in Z$, same set $\mathcal{F}$ of feasible solutions as for (NON-PP).
QUESTION: Is there a feasible solution $S \in \mathcal{F}$ with $S(z^\star) = 1$ and a cost vector c on Z with $\ell(z) \leqslant c(z) \leqslant u(z)$ for all $z \in Z$, such that S minimizes c over all solutions in $\mathcal{F}$?

Theorem 4.2.1. *Let (PP) be a* PREPROCESSING PROBLEM. *Let (NON-PP) denote the corresponding* COMBINATORIAL FEASIBILITY PROBLEM. *Assume that (NON-PP) lies in NP. Let f be a value preserving polynomial transformation from (3SAT) to (NON-PP). Then there is a polynomial transformation from (PP 3SAT) to (PP). In particular, (PP) is Σ_2^p-complete.*

Proof. As we may assume that the cost vector c takes on integer values, and since (NON-PP) is in NP, it follows that (PP) is in Σ_2^p.

So consider an instance $I_{(\text{PP 3SAT})}$ of the problem (PP 3SAT). Let $I_{(\text{3SAT})}$ be the corresponding instance of the underlying problem (3SAT) with the set U of variables. Let $u^\star \in U$ be the special element. Lower and upper bounds on the cost coefficients are denoted by ℓ and u, as usual.

Let f be the value preserving polynomial transformation from (3SAT) to (NON-PP). Let $I_{(\text{NON-PP})} = f(I_{(\text{3SAT})})$ be the corresponding instance. Let g be the bijective function as described in the definition of the value preserving polynomial transformation f. Let $Z' = g(U)$. For each $z \in Z'$ let $\ell(z) = \ell(g^{-1}(z))$ and $u(z) = u(g^{-1}(z))$. For every $z \in Z \setminus Z'$ set $\ell(z) = u(z) = 0$. Taking the instance $I_{(\text{NON-PP})}$ and adding the values ℓ and u for each $z \in Z$ as lower and upper bounds for the costs on the elements of Z, and $z^\star = g(u^\star)$ as the special element in Z, leads to an instance $I_{(\text{PP})}$ of the problem (PP).

Assume that $I_{(\text{PP 3SAT})}$ is a yes-instance and consider a solution (c, S) of $I_{(\text{PP 3SAT})}$. Thus, S is a solution of $I_{(\text{3SAT})}$ and minimal with respect to the cost vector c. Also, $S(u^\star) = 1$. We create a solution $(c_{(\text{PP})}, S_{(\text{PP})})$ for $I_{(\text{PP})}$ as follows. Let $S_{Z'}(z) = S(g^{-1}(z))$ for all $z \in Z'$. According to the Condition (1) on f there is an extension of $S_{Z'}$ to the remaining variables in Z resulting in a 0-1 assignment $S_{(\text{PP})}$ such that $S_{(\text{PP})}$ represents a solution to $I_{(\text{NON-PP})}$. Let $c_{(\text{PP})}(z) = c(g^{-1}(z))$ for all $z \in Z'$. For all other $z \in Z \setminus Z'$ we set $c_{(\text{PP})}(z) = 0$. Obviously, since $\ell(z) = c_{(\text{PP})}(z) = u(z) = 0$ for all $z \in Z \setminus Z'$, the costs for the elements in $Z \setminus Z'$ lie within their bounds. For all $z \in Z'$ we have $\ell(z) = \ell(g^{-1}(z))$ and $u(z) = u(g^{-1}(z))$ and $c_{(\text{PP})}(z) = c(g^{-1}(z))$. Hence, $\ell(z) = \ell(g^{-1}(z)) \leqslant c(g^{-1}(z)) = c_{(\text{PP})}(z)$, and $u(z) = u(g^{-1}(z)) \geqslant c(g^{-1}(z)) = c_{(\text{PP})}(z)$. Since $S(u^\star) = 1$ we have that $S_{(\text{PP})}(z^\star) = 1$ as well. Because (c, S) is a

solution of instance $I_{(\text{PP 3SAT})}$, we know that S is minimal with respect to the cost vector c. The cost of $S_{(\text{PP})}$ with respect to the cost vector $c_{(\text{PP})}$ is $\sum_{z\in Z} c_{(\text{PP})}(z)S_{(\text{PP})}(z) = \sum_{z\in Z'} c_{(\text{PP})}(z)S_{(\text{PP})}(z) = \sum_{z\in Z'} c(g^{-1}(z))S(g^{-1}(z)) = \sum_{u\in U} c(u)S(u)$, and thus equal to the cost of S. Suppose that $S_{(\text{PP})}$ is not minimal with respect to the costs $c_{(\text{PP})}$ in instance $I_{(\text{PP})}$. Then, there is another solution $\overline{S}_{(\text{PP})}$ in $\mathcal{F}$ that has smaller costs with respect to $c_{(\text{PP})}$. With Condition (2) we can apply the function g^{-1} and obtain the 0-1 assignment $\overline{S}$ with $\overline{S}(u) = \overline{S}_{(\text{PP})}(g(u))$ for all $u \in U$ as a solution for $I_{(\text{3SAT})}$. The costs c of $\overline{S}$ are $\sum_{u\in U} c(u)\overline{S}(u) = \sum_{u\in U} c_{(\text{PP})}(g(u))\overline{S}_{(\text{PP})}(g(u))$, and hence equal the costs of the solution $(c, \overline{S}_{(\text{PP})})$ of instance $I_{(\text{PP})}$. Since $\overline{S}_{(\text{PP})}$ has smaller costs than $S_{(\text{PP})}$ with respect to $c_{(\text{PP})}$, it follows that $\overline{S}$ has smaller costs than S with respect to c. This however contradicts the optmality of S. Hence, $S_{(\text{PP})}$ is minimal with respect to $c_{(\text{PP})}$. To summarize, we have that $S_{(\text{PP})}(z^\star) = 1$, the costs $c_{(\text{PP})}$ lie within their bounds, that is, $\ell(z) \leqslant c_{(\text{PP})}(z) \leqslant u(z)$ for all $z \in Z$, $S_{(\text{NON-PP})}$ is in $\mathcal{F}$, and it is minimal with respect to $c_{(\text{PP})}$. Instance $I_{(\text{PP})}$ is therefore a yes-instance for (PP).

Consider now the case that $I_{(\text{PP 3SAT})}$ is a no-instance of (PP 3SAT). Suppose that the corresponding instance $I_{(\text{PP})}$ is a yes-instance of (PP). Let $(c_{(\text{PP})}, S_{(\text{PP})})$ be a solution to $I_{(\text{PP})}$. Hence, $S_{(\text{PP})}$ is a solution to the corresponding instance of the non-preprocessing problem (NON-PP) and, by Condition (2) on f, there is a 0-1 assignment S with $S(u) = S(g(u))$ for all $u \in U$ that is a solution for $I_{(\text{3SAT})}$. Let the cost vector c for the variables in U be as follows. For every $u \in U$ we set $c(u) = c_{(\text{PP})}(g(u))$. Thus, for all $u \in U$ we have $\ell(u) = \ell(g(u)) \leqslant c_{(\text{PP})}(g(u)) = c(u)$ and $u(u) = u(g(u)) \geqslant c_{(\text{PP})}(g(u)) = c(u)$. Since $z^\star = g(u^\star)$, we have $S(u^\star) = S_{(\text{PP})}(z^\star) = 1$. It remains to show that S is minimal with respect to c. The cost of $S_{(\text{PP})}$ with respect to $c_{(\text{PP})}$ is $\sum_{z\in Z} c_{(\text{PP})}(z)S_{(\text{PP})}(z) = \sum_{z\in Z'} c_{(\text{PP})}(z)S_{(\text{PP})}(z) = \sum_{z\in Z'} c(g^{-1}(z))S(g^{-1}(z)) = \sum_{u\in U} c(u)S(u)$. Suppose that there is another 0-1 assignment $\overline{S}$ that satisfies E, but has smaller cost with respect to c than S. Consider any corresponding 0-1 assignment $\overline{S}_{(\text{PP})}$ with $\overline{S}_{(\text{PP})}(z) = \overline{S}(g^{-1}(z))$ for all $z \in Z'$. The cost $c_{(\text{PP})}$ of $\overline{S}_{(\text{PP})}$

is $\sum_{z\in Z} c_{(\mathrm{PP})}(z)\overline{S}_{(\mathrm{PP})}(z) = \sum_{z\in Z'} c_{(\mathrm{PP})}(z)\overline{S}_{(\mathrm{PP})}(z) = \sum_{z\in Z'} c(g^{-1}(z))\overline{S}(g^{-1}(z)) = \sum_{u\in U} c(u)\overline{S}(u) < \sum_{u\in U} c(u)S(u)$, which contradicts our assumption that S_{PP} is minimal with respect to $c_{(\mathrm{PP})}$ for $I_{(\mathrm{PP})}$.

Since (PP 3SAT) is Σ_2^p-complete, the result follows. □

We noted earlier (see Section 2.1) that value preserving polynomial transformations are transitive. That is, if we have a value preserving polynomial transformation f_1 transforming a problem Π_1 to a problem Π_2, and a value preserving polynomial transformation f_2 transforming the problem Π_2 to a problem Π_3, then the composition $f_2 \circ f_1$ of the two functions f_1 and f_2, transforming the problem Π_1 to the problem Π_3, is value preserving as well. Consequently, if Π_1 is Σ_2^p-complete, then the preprocessing version of problem Π_3 is Σ_2^p-complete as well. This observation has been formalized in the context of adversarial problems in Section 2.1.

As we have mentioned before, it seems that the majority of polynomial transformations between two NP-complete combinatorial feasibility problems seem to be, or can be modified just slightly to become value preserving, implying Σ_2^p-completeness for a great number of preprocessing problems.

We want to exemplify this statement in the following by exploring the application of Theorem 4.2.1 to the preprocessing version of a few fundamental problems in NP, including VERTEX COVER, 3-DIMENSIONAL MATCHING, and HAMILTONIAN CYCLE.

4.3 Preprocessing Vertex Cover

The VERTEX COVER problem is one of the fundamental problems in combinatorial optimization and was among the first for which Karp [Kar72] showed NP-completeness.

It is defined as follows.

PROBLEM: VERTEX COVER (VC)
INSTANCE: Graph $G = (V, E)$, positive integer K.
QUESTION: Does G have a vertex cover V' of size K or less, that is, is there a subset V' of V of size at most K such that for every edge in E at least one of its end nodes is contained in V'?

The preprocessing version of the VERTEX COVER problem is defined as follows:

PROBLEM: PREPROCESSING VERTEX COVER (PP-VC)
INSTANCE: Graph $G = (V, E)$, positive integer K, bounds $\ell(v)$ and $u(v)$ for each node $v \in V$, node $v^\star \in V$.
QUESTION: Does there exist a vertex cover V' of size K or less for G that includes the node $v^\star$, and a cost vector c on V with $\ell(v) \leqslant c(v) \leqslant u(v)$ for all $v \in V$ such that V' minimizes c over all vertex covers of size K or less (including the ones not containing $v^\star$)?

Theorem 4.3.1. *(PP-VC) is Σ_2^p-complete.*

Proof. Let us first verify by hand that (PP-VC) is in Σ_2^p. Assuming that we have guessed a solution (V', c) to a yes-instance of (PP-VC), we need to check whether the set V' is a vertex cover of size at most K that contains the node $v^\star$ and that is minimal with respect to c among all vertex covers of size K or less. This question can be answered in nondeterministic polynomial time since the problem of deciding whether there exists a vertex cover of size at most K with cost at most some integer J is in NP.

With Theorem 4.2.1 in hand, it remains to show that the polynomial transformation from (3SAT) to VERTEX COVER by Karp [Kar72] (which is also described in [GJ79])

is value preserving. This transformation works as follows.

Let $I_{(3SAT)}$ be an instance of (3SAT), with variable set U and clause set C. For each variable u in U we introduce two nodes u and $\bar{u}$ and an edge $\{u, \bar{u}\}$ between them. Node u is called the "true-setting" node of the Boolean variable u, and $\bar{u}$ is called the "false-setting" node of u. We combine the true-setting nodes of all variables to the set V_U, and all false-setting nodes of all variables to $V_{\bar{U}}$. The set of edges between nodes in V_U and $V_{\bar{U}}$ is called E_U.

For each clause $c_i \in C$ we introduce a triangle named C_i with three nodes and three edges. We call the set of all triangle edges E_C and the set of all triangle nodes V_C. Furthermore, for each non-negated literal of a variable u in a clause c_i we introduce an edge from a node of the corresponding triangle C_i to the true-setting node u in V_U. Similarly, for each negated literal of a variable u in a clause c_i we introduce an edge from a node of C_i to the false-setting node $\bar{u}$ in $V_{\bar{U}}$. The edges between a node in V_C and a node in $V_U \cup V_{\bar{U}}$ form the set E_{CU}. We choose a different node of C_i for each literal in the clause, so that in the end each node in C_i has degree 3, for all $i \in \{1, \ldots, |C|\}$. Finally, let $K = 2|C| + |U|$. This completes the formal description of Karp's polynomial transformation from (3SAT) to (VC).

Note that every vertex cover must contain at least one node from each pair of nodes in $V_U \cup V_{\bar{U}}$ to cover the edges in E_U. Furthermore, each vertex cover must contain at least two nodes from each C_i, $i = 1, \ldots, |C|$, in order to cover the edges in E_C. Hence, each vertex cover must contain at least K nodes. The claim is that the so-constructed graph has a vertex cover of size K if and only if $I_{(3SAT)}$ is a yes-instance. This can be seen as follows.

Assume that $I_{(3SAT)}$ is a yes-instance and consider some satisfying truth assignment.

We define a corresponding vertex cover V' of size K as follows. For each variable u that has value 1 in the truth assignment, we include the true-setting node $u \in V_U$ in V'. For each variable u that has value 0 in the truth assignment, we include the corresponding false-setting node $\bar{u} \in V_{\bar{U}}$ in V'. This ensures that all edges in E_U are covered. Moreover, for each triangle C_i, $i \in \{1, \ldots, |C|\}$, at least one of its three incident edges in E_{CU} is covered as well, since each clause contains at least one true literal. We also include two nodes from each triangle, making sure that any remaining uncovered edges in E_{CU} are covered. Of course, all edges in E_C are now covered as well. Hence, V' represents a vertex cover, and it is of size K.

Conversely, if we have a vertex cover V' of size K, we can use it to derive a satisfying truth assignment S. Note that V' contains exactly two nodes from each triangle C_i, $i = 1, \ldots, |C|$, and one node from each node pair in $V_U \cup V_{\bar{U}}$. We set a Boolean variable u in $I_{(\text{PP 3SAT})}$ to 1 if the true-setting node u is in V', and we set it to 0 if the false-setting node $\bar{u}$ is in V'. Observe that in the triangle corresponding to a clause c_i, two of its three incident edges in E_{CU} can be covered by nodes from C_i, yet the last one has to be covered by a node u in $V_U \cup V_{\bar{U}}$. This implies that the clause c_i is satisfied since u corresponds to a literal in c_i that is satisfied by our truth assignment.

It remains to argue that this transformation f is value preserving. To see this, we define a function $g : U \to V_U$ by mapping the Boolean variable $u \in U$ to its true-setting node $u \in V_U$. This is obviously a one-to-one mapping. Moreover, given a satisfying truth assignment for an instance of (3SAT), we have seen that there is a vertex cover of size at most K that contains exactly those nodes in V_U whose corresponding Boolean variable equals 1. On the other hand, any vertex cover of size K contains a subset of V_U that yields a satisfying truth assignment if its corresponding Boolean variables are set to 1. Thus, f is indeed value preserving. We can therefore

invoke Theorem 4.2.1 to conclude that (PP-VC) is Σ_2^p-complete. □

We illustrate the transformation described in the proof of Theorem 4.3.1 in Figure 4-1 below. We base it on Example 2.2.2 introduced on Page 36.

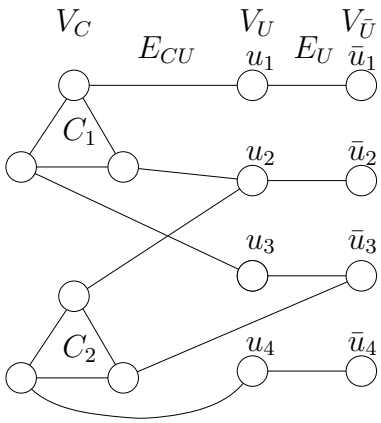

Figure 4-1: Transformation from (3SAT) to (VC)

The solution provided in Example 2.2.2 corresponds to the set of nodes highlighted in Figure 4-2, which represents a vertex cover for the instance depicted in Figure 4-1.

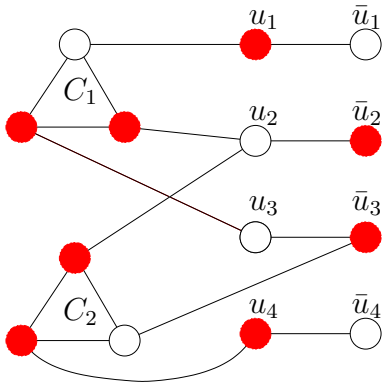

Figure 4-2: Example solution for the (VC) instance of Figure 4-1

Figure 4-3 illustrates Theorem 4.3.1 and the transformation from (3SAT) to the VERTEX COVER problem.

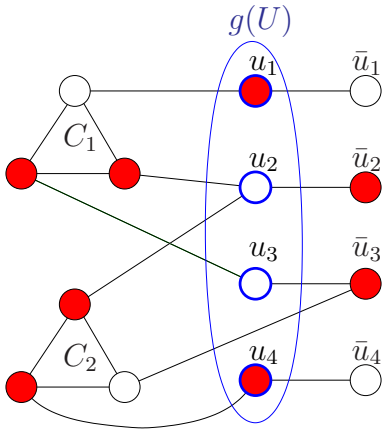

Figure 4-3: Application of Theorem 4.3.1 to (PP-VC)

The filled nodes all belong to the solution of the vertex cover instance. The other, non-filled nodes represent the nodes that are not part of the proposed vertex cover. The nodes with a thick border within the bounding ellipse are the ones belonging to $g(U)$, that is, they map directly via the function g to and from U, the set of variables in the (3SAT) instance. Hence, a satisfiability solution can be read off solely by looking at the nodes within that set. The filled nodes within the bounding ellipse correspond to variables in U of the (3SAT) instance that have value 1 in the proposed solution, and the other nodes within the frame correspond to variables in U of the (3SAT) instance that have value 0.

4.4 Preprocessing 3-Dimensional Matching

Another basic NP-complete problem is the 3-DIMENSIONAL MATCHING problem (3DM). In Section 2.2 we showed that the adversarial version of (3DM) is Σ_2^p-complete. Theorem 4.2.1 is directly applicable to 3-DIMENSIONAL MATCHING, resulting in Σ_2^p-completeness of its preprocessing version.

PROBLEM: PREPROCESSING 3-DIMENSIONAL MATCHING (PP-3DM)
INSTANCE: Three disjoint sets A, B, and D of size q each, set $M \subseteq A \times B \times D$, bounds $\ell(v)$ and $u(v)$ for each element $v \in M$, element $v^\star \in M$.
QUESTION: Is there a 3-dimensional matching $M^\star$ that contains $v^\star$, and a cost vector c on M with $\ell(v) \leqslant c(v) \leqslant u(v)$ for all $v \in M$ such that $M^\star$ minimizes c over all 3-dimensional matchings in M?

In Section 2.2 we gave a detailed description of the polynomial transformation from (3SAT) used to prove NP-completeness of the 3-DIMENSIONAL MATCHING problem. In particular, we argued that this transformation is value preserving. We obtain the following corollary.

Corollary 4.4.1. *(PP-3DM) is Σ_2^p-complete.*

4.5 Preprocessing Hamiltonian Cycle

The next problem that we consider in order to illustrate the broad applicability of Theorem 4.2.1 is the NP-complete problem HAMILTONIAN CYCLE. As with most NP-complete problems considered thus far, it is part of Karp's original list of NP-complete problems [Kar72].

PROBLEM: HAMILTONIAN CYCLE (HC)
INSTANCE: Graph $G = (V, E)$.
QUESTION: Is there a Hamiltonian cycle $E^\star$ for G, that is, a subset $E^\star$ of E such that the edges in $E^\star$ form a cycle in G which visits each node in V exactly once?

The preprocessing version of (HC) is the following.

PROBLEM: PREPROCESSING HAMILTONIAN CYCLE (PP-HC)
INSTANCE: Graph $G = (V, E)$, bounds $\ell(e)$ and $u(e)$ for each edge $e \in E$, edge $e^\star \in E$.
QUESTION: Is there a Hamiltonian cycle $E^\star$ for G that includes the edge $e^\star$, and a cost vector c on E with $\ell(e) \leqslant c(e) \leqslant u(e)$ for all $e \in E$ such that $E^\star$ minimizes $\sum_{e \in H} c(e)$ over all Hamiltonian cycles H in G?

The PREPROCESSING HAMILTONIAN CYCLE problem is in Σ_2^p, since the problem of deciding, given a graph G and an integer J, whether there is a Hamiltonian cycle in G of cost at most J, is in NP, and can be used to verify a certificate of (PP-HC) consisting of the integer costs c and an optimal Hamiltonian cycle $E^\star$ containing $e^\star$.

As already mentioned after the proof of Theorem 4.2.1, to establish Σ_2^p-completeness of (PP-HC), the polynomial transformation to (HC) does not have to originate from (3SAT). Instead, we will use the well-known polynomial transformation from VERTEX COVER to HAMILTONIAN CYCLE, originally presented by Karp [Kar72] and later refined by Garey and Johnson [GJ79]. This transformation works as follows: Consider an arbitrary instance of (VC), consisting of a graph $G = (V, E)$ and an integer $K \leqslant |V|$. The corresponding graph for the Hamiltonian cycle problem has $K + 12|E|$ many nodes. There are K so-called "selector nodes," which help to ensure that the vertex cover contains at most K nodes. The main gadget is a subgraph that is built for each edge $e = \{u, v\} \in E$ and which is constructed in such a way that each Hamiltonian cycle will "cover" at least one of the end nodes of e. It is usually called a "cover-testing component" and consists of twelve nodes, which we call (u_i, e) and (v_i, e), for $i = 1, \ldots, 6$. There is an edge between (u_i, e) and (u_{i+1}, e) and between (v_i, e) and (v_{i+1}, e), for all $i = 1, \ldots, 5$. There are four additional edges, namely between (u_1, e) and (v_3, e), between (v_1, e) and (u_3, e), between (u_4, e) and (v_6, e), and between (v_4, e) and (u_6, e). Any cover-testing component is connected to the rest of the graph via the nodes (u_1, e), (v_1, e), (u_6, e), and (v_6, e) only. Figure 4-4 depicts the

structure of a cover-testing component.

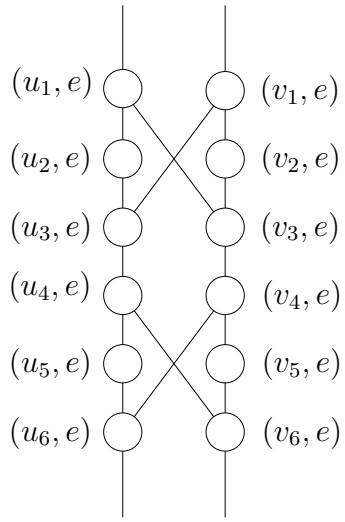

Figure 4-4: Cover-testing component used in the transformation from VERTEX COVER to HAMILTONIAN CYCLE

It is easy to check that any Hamiltonian cycle has only three ways of traversing a cover-testing component. All three possibilities are illustrated in Figure 4-5. The first alternative is the consecutive use of the edges $\{(u_1, e), (u_2, e)\}$, $\{(u_2, e), (u_3, e)\}$, $\{(u_3, e), (u_4, e)\}$, $\{(u_4, e), (u_5, e)\}$, and $\{(u_5, e), (u_6, e)\}$ as well as the consecutive use of $\{(v_1, e), (v_2, e)\}$, $\{(v_2, e), (v_3, e)\}$, $\{(v_3, e), (v_4, e)\}$, $\{(v_4, e), (v_5, e)\}$, and $\{(v_5, e), (v_6, e)\}$. This corresponds to including both nodes u and v in the vertex cover. The second option is to traverse the nodes of the cover-testing component as follows: (u_1, e), (u_2, e), (u_3, e), (v_1, e), (v_2, e), (v_3, e), (v_4, e), (v_5, e), (v_6, e), (u_4, e), (u_5, e), (u_6, e). This corresponds to making the node u, but not v part of the cover. The third possibility is symmetric to the previous one and corresponds to selecting v, but not u: the nodes are traversed in the order (v_1, e), (v_2, e), (v_3, e), (u_1, e), (u_2, e), (u_3, e), (u_4, e), (u_5, e), (u_6, e), (v_4, e), (v_5, e), (v_6, e).

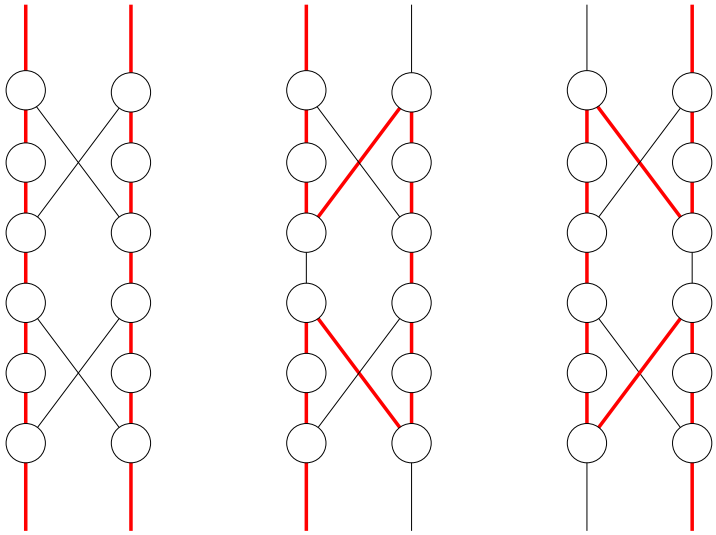

Figure 4-5: The three ways in which a Hamiltonian cycle can traverse a cover-testing component

The cover-testing components are connected as follows. For each node u of the original vertex cover graph G, choose an arbitrary order of its incident edges. If $\delta(u)$ denotes the set of edges incident to u, let $e^u_1, e^u_2, \ldots, e^u_{|\delta(u)|}$ be this order. For $i = 1, \ldots, |\delta(u)| - 1$, there is an edge between (u_6, e^u_i) and (u_1, e^u_{i+1}). In addition there are edges between (u_1, e^u_1) and all selector nodes as well as between $(u_6, e^u_{|\delta(u)|})$ and all selector nodes. This concludes the description of the polynomial transformation from VERTEX COVER to HAMILTONIAN CYCLE. Figure 4-6 provides an illustration of the construction. It is based on the vertex cover instance of Figure 4-1. However, the picture does not show all edges incident to selector nodes, but a sufficient subset thereof. The picture also presents a sample solution to the Hamiltonian cycle instance (by way of highlighted edges). It corresponds to the solution given in Figure 4-2 of the underlying VERTEX COVER problem.

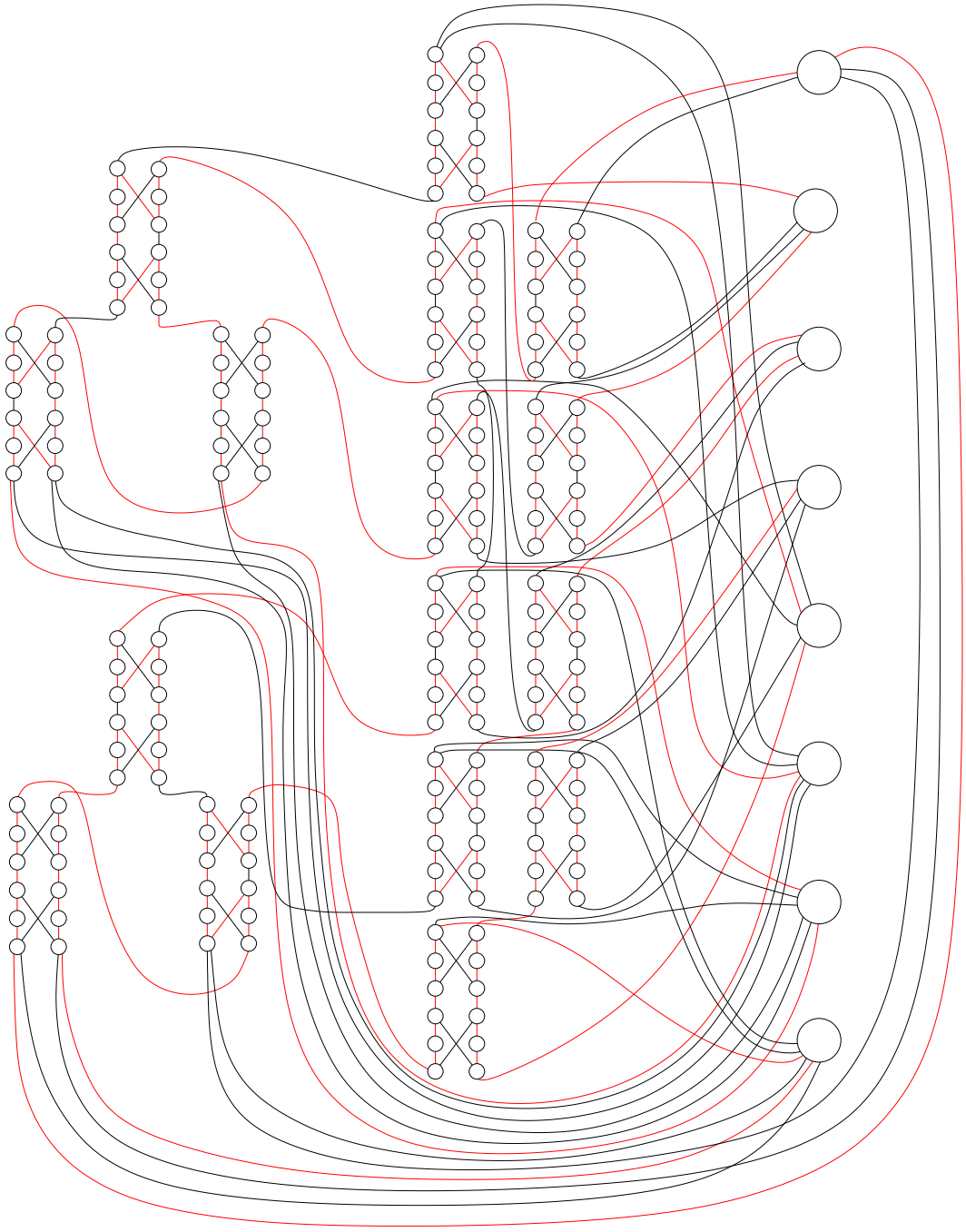

Figure 4-6: Transformation from (VC) to (HC)

It remains to identify a bijective function g with Properties (1) and (2) to show that this polynomial transformation is value preserving. For this, it is best to describe in

detail how a solution to the resulting HAMILTONIAN CYCLE instance corresponds to a vertex cover in the original graph G, and vice versa. So consider a Hamiltonian cycle and trace its path from one of the selector nodes. By construction, it will first visit a node (u_1, e_1^u) for some node $u \in V$. It will then run, in one of two ways (recall Figure 4-5), through the cover-testing component of e_1^u, and from (u_6, e_1^u) it will connect to (u_1, e_2^u), and so on until it reaches $(u_6, e_{|\delta(u)|}^u)$, from where it will visit another selector node. The cycle continues in this way until all selector nodes have been included. Each node $u \in V$ that is "visited" in this way becomes part of the cover. On the other hand, if we are given a vertex cover of size at most K, we may assume that it has size exactly K and we can form a Hamiltonian cycle in exactly the same way. If the vertex cover contains both end nodes of an edge, its cover-testing component is traversed as in the left-most option of Figure 4-5; otherwise, if $e = \{u, v\}$ and only u is in the cover, the cover-testing component of e is traversed according to the second option in Figure 4-5. Although it appears natural to define $g(u)$ by mapping $u \in V$ to the edge between the selector node and (u_1, e_1^u), this would not work because it is a priori not known which selector node will connect to (u_1, e_1^u) or whether there is such a connection at all. Instead, we use the edge $\{(v_3, e_1^u), (v_4, e_1^u)\}$ (where we assumed that $e_1^u = \{u, v\}$) as the indicator: a quick check confirms that it is part of the Hamiltonian cycle if and only if u belongs to the vertex cover. Hence, we define $g(u) = \{(v_3, e_1^u), (v_4, e_1^u)\}$. From our preceding discussion, it follows that g satisfies Properties (1) and (2), so the polynomial transformation from (HC) to (VC) is value preserving.

Corollary 4.5.1. *(PP-HC) is Σ_2^p-complete.*

4.6 Beyond Value Preserving Transformations

The preceding discussion indicates that it may sometimes be difficult to identify the "right" function g to prove that a polynomial transformation is indeed value preserving. It turns out that one can relax the Conditions (1) and (2) a bit without altering the results. In the context of PREPROCESSING HAMILTONIAN CYCLE, for instance, we would allow a node u of the original vertex cover graph to be mapped to the set of edges between (u_1, e_1^u) and all selector nodes. If we do this, any Hamiltonian cycle contains at most one edge from each $g(u)$, and it does contain such an edge if and only if the corresponding vertex cover contains u. Moreover, for every two nodes $u, v \in V$, $u \neq v$, the sets $g(u)$ and $g(v)$ are disjoint. We can show that these properties suffice for Theorem 4.2.1 to remain true. This can be formalized as follows.

Let Π_1 and Π_2 be two NP-complete combinatorial feasibility problems. Given an instance I_1 of Π_1 and an instance I_2 of Π_2, we use Z_1 and Z_2 to denote their respective ground sets of variables. Let f be a polynomial transformation from Π_1 to Π_2. We say that f is *good enough* if there is a function $g : Z_1 \to 2^{Z_2}$ mapping elements of Z_1 to subsets of Z_2 that has the following properties:

(1) If S_1 is a feasible 0-1 solution for I_1, then there is a feasible 0-1 solution S_2 for I_2 such that $\sum_{z_2 \in g(z_1)} S_2(z_2) = S_1(z_1)$ for all $z_1 \in Z_1$.

(2) If S_2 is a feasible 0-1 solution for I_2, then there is a feasible 0-1 solution S_1 for I_1 such that $S_1(z_1) = \sum_{z_2 \in g(z_1)} S_2(z_2)$ for all $z_1 \in Z_1$.

In particular, $\sum_{z_2 \in g(z_1)} S_2(z_2) \in \{0, 1\}$ for all $z_1 \in Z_1$.

With these adjustments to the original Conditions (1) and (2), the proof of Theorem 4.2.1 carries over to this more general class of polynomial transformations. Moreover, these modifications can also be applied to Theorem 2.1.1, and to Theorems 6.1.2 and 6.2.1 below.

Chapter 5

Counterexamples to NP-Conjectures

A research direction with a long tradition in algorithmic graph theory is to give practical characterizations of graph properties that are, in general, NP-complete to decide. One typical example of such a property is the question of whether a graph G is *Hamiltonian*, that is, whether G has a Hamiltonian cycle.

A property $\mathcal{P}$ of a graph G is a sufficient condition for G to be Hamiltonian, if for every graph G with property $\mathcal{P}$, the graph G is Hamiltonian. Numerous sufficient conditions for Hamiltonian graphs are already known, and many more have been proposed.

One example of a sufficient condition for an undirected graph G to be Hamiltonian is Fan's Theorem [Fan84], which says that if G is 2-connected and if, for all pairs of nodes of distance two, one of the nodes connects to at least half of all nodes in G, then G is Hamiltonian. Another example is a theorem by Duffus, Gould and Jacobson [DGJ82], which states that if G is a 2-connected graph that does not contain a claw or a net, then G is Hamiltonian.

One famous conjecture that has been refuted was made by [Tai84], who conjectured that every 3-connected planar cubic graph has a Hamiltonian cycle. The counterexample by Tutte [Tut46] has 46 nodes, but smaller counterexamples have been found later.

There are many conjectures of sufficient conditions that are yet to be proved right or wrong. Open conjectures include one by Thomassen [Tho86], who suggests that every 4-connected line graph is Hamiltonian, and another one by Barnette [Bar69], who believes that every planar 3-connected 3-regular bipartite graph is Hamiltonian. Barnette's conjecture has been open since the late sixties, and computational experiments have shown that there is no counterexample with fewer than 86 nodes [HMM85]. A list of quite a few computer generated examples of open (and solved) conjectures is maintained by DeLaVina [DeL10].

We will concern ourselves in this section with the computational complexity of refuting conjectures concerning NP-complete graph properties, such as the existence of a Hamiltonian cycle. How hard is it, in a complexity theoretic sense, to find a counterexample to a conjecture that suggests a sufficient condition for a graph to have a property that is in general NP-complete to decide? For example, what is, in general, the computational complexity of finding a counterexample to a conjecture that suggests a sufficient condition for a graph to be Hamiltonian?

5.1 Hamiltonian Cycle Conjectures

We start by focusing on conjectures concerning the Hamiltonicity of graphs, simply to have a more concrete setting. Let $\mathcal{P}$ be some graph property (such as being a 4-connected line graph). Then Conjecture $Conj(\mathcal{P}, \mathrm{HC})$ claims that every undirected graph with property $\mathcal{P}$ has a Hamiltonian cycle. In principle, we want to study the

following problem:

INSTANCE: Property $\mathcal{P}$.
QUESTION: Is there a counterexample to conjecture $Conj(\mathcal{P}, \text{HC})$?

However, before we can analyze its computational complexity, we need to address an important technical detail. That is, we need to formalize the input size of an instance, which formally consists of a property $\mathcal{P}$. Implicitly, however, an instance is composed of a family of graphs, encoded by $\mathcal{P}$. Currently, $\mathcal{P}$ describes all graphs with this property, and such graphs can be of very different size. We will handle this by relating the class of graphs that form the instance to a counterexample problem to an instance I of some decision problem Π. That is, every instance $I \in \Pi$ leads to a property $\mathcal{P}_I$, and the graphs with property $\mathcal{P}_I$ are all of size polynomial in the input size of I. Viewed differently, Π plays the role of an infinite index set, each element I of Π corresponds to a graph property $\mathcal{P}_I$, and $\mathcal{P}_I$ defines a class of graphs that are all of similar size. We also assume that it can be checked in time polynomial in the input size of I whether a given graph whose size is polynomial in the input size of I has property $\mathcal{P}_I$. We can now formally state the problem that we will consider:

PROBLEM: COUNTEREXAMPLE HAMILTONIAN CYCLE (CExplHC)
INSTANCE: Property $\mathcal{P}_I$.
QUESTION: Is there a counterexample to conjecture $Conj(\mathcal{P}_I, \text{HC})$?

The problem COUNTEREXAMPLE HAMILTONIAN CYCLE is in Σ_2^p. If the answer is "yes," we can exhibit a graph G with a number of nodes that is polynomial in the input size, check that it has property $\mathcal{P}_I$, and solve an instance of the HAMILTONIAN CYCLE problem to verify that G is not Hamiltonian. In fact, COUNTEREXAMPLE HAMILTONIAN CYCLE belongs to the hardest problems in Σ_2^p.

Theorem 5.1.1. COUNTEREXAMPLE HAMILTONIAN CYCLE *is* Σ_2^p*-complete.*

Proof. We will construct a polynomial transformation from the Σ_2^p-complete problem (B_2^{CNF}) to (CExplHC). This transformation will itself be based on a polynomial transformation from (3SAT) to the HAMILTONIAN CYCLE problem. We obtain the transformation from (3SAT) to HAMILTONIAN CYCLE by combining the transformations from 3-SATISFIABILITY to VERTEX COVER (see Section 4.3) and from VERTEX COVER to HAMILTONIAN CYCLE (Section 4.5). As we will make use of the special structure of the resulting HAMILTONIAN CYCLE instances and slightly modify it for our purposes, let us briefly consider what happens when the two transformations are combined.

Let $I_{(3SAT)}$ be an instance of (3SAT) with the set C of clauses and the set U of variables. Let $I_{(VC)}$ denote the instance of (VC) that results when we apply to $I_{(3SAT)}$ the polynomial transformation from (3SAT) to (VC). Recall that $I_{(VC)}$ consists of a special graph (see Figure 4-1 for an example) in which we have a triangle (that is a K_3) for each clause, where each node corresponds to one of the three literals of that clause. We also have pairs of nodes u and $\bar{u}$ that are connected by an edge, for each variable $u \in U$. Depending on whether a variable u occurs positively or negatively in a clause, there is an edge from u or $\bar{u}$ to its corresponding node in the clause. Moreover, $K = 2|C| + |U|$, and it is clear from the graph structure that any vertex cover has to pick u or $\bar{u}$. It follows that the HAMILTONIAN CYCLE instance $I_{(HC)}$, which arises from applying to $I_{(VC)}$ the polynomial transformation from VERTEX COVER to HAMILTONIAN CYCLE, has $2|C| + |U|$ many selector nodes, of which we earmark $|U|$ many for the variables. More precisely, we will call the *variable selector nodes* $v_1, \ldots, v_{|U|}$, and the remaining *clause selector nodes* are denoted by $c_{11}, c_{12}, c_{21}, c_{22}, \ldots, c_{|C|1}, c_{|C|2}$. The other main ingredient of the reduction from VERTEX COVER to HAMILTONIAN CYCLE are the cover-testing components. Because

of the origin of the vertex cover instance, we can associate these components with elements of the (3SAT) instance as well. Recall that $I_{(\mathrm{VC})}$ has three different types of edges: edges of the form $\{u, \bar{u}\}$, sets of three edges forming a clause triangle, and edges between the clause triangles and the variable nodes. For each edge, there is a cover-testing component. So we have one cover-testing component for each variable $u \in U$, corresponding to the edge $\{u, \bar{u}\}$ in $I_{(\mathrm{VC})}$. Furthermore, consider a literal ℓ_{jk}, $k \in \{1, 2, 3\}$, in a clause c_j, $j \in \{1, \ldots, |C|\}$. Let u be the variable associated with ℓ_{jk}. The first cover-testing component introduced for a literal ℓ_{jk} is the one for the edge between the node for literal ℓ_{jk} and the node for literal $\ell_{j(k+1)}$ (or between ℓ_{j3} and ℓ_{j1} if $k = 3$). The second cover-testing component introduced in connection with the literal ℓ_{jk}, $k \in \{1, 2, 3\}$, in clause c_j, $j \in \{1, \ldots, |C|\}$, corresponds to the edge between the node for ℓ_{jk} and $\bar{u}$ if the literal ℓ_{jk} of the variable u is negated, and to the edge between ℓ_{jk} and u, otherwise. Hence, the total number of cover-testing components introduced is $6|C| + |U|$.

The various cover-testing components are connected in the same way as described in the original transformation from VERTEX COVER to HAMILTONIAN CYCLE (see Section 4.5). Finally, we need to connect the selector nodes with the chains of cover-testing components, and this is where we refine the original transformation for our purposes. In the original transformation, Karp [Kar72] (see also [GJ79]) connected every chain at both ends with every single selector node. Since we want to define a particular bijective function g between the sets of variables in $I_{(\mathrm{3SAT})}$ and a subset of variables in $I_{(\mathrm{HC})}$ with the usual Properties (1) and (2) of a value preserving transformation, we only introduce a subset of the connecting edges, as follows. Recall that we introduced two chains of cover-testing components for each variable u in U, one for the true-setting node u in $I_{(\mathrm{VC})}$, and one for the false-setting node $\bar{u}$ in $I_{(\mathrm{VC})}$. We connect the start nodes of the two chains, corresponding to the variable u_1, to the first variable selector node v_1. We connect the end nodes of the two chains corre-

sponding to the variable u_i to the variable selector node v_{i+1}, for all $i = 1, \ldots, |U|-1$. We connect the start nodes of the chains corresponding to the variable u_{i+1} to the variable selector node v_{i+1}, for all $i = 1, \ldots, |U| - 1$. We connect the end nodes of the chains corresponding to the variable $u_{|U|}$ to the first clause selector node c_{11}.

To cover the clause nodes, we connect the start node of each chain corresponding to a literal in clause c_j to the clause selector nodes c_{j1} and c_{j2}, for all $j \in \{1, \ldots, |C|\}$, we connect the end node of each chain corresponding to a literal in clause c_j to the clause selector nodes c_{j2} and $c_{(j+1)1}$, for all $j \in \{1, \ldots, |C| - 1\}$, and finally we connect the end node of the chains corresponding to a literal in the last clause $c_{|C|}$ to the variable selector node v_1.

It can be easily checked that this transformation does indeed map yes-instances into yes-instances and no-instances into no-instances.

The following Figure 5-1 presents as an example the instance of (HC) that results when using this transformation from (3SAT) to (HC) on the (3SAT) instance of Example 2.2.2. This figure also presents a sample solution to the Hamiltonian cycle instance (by way of highlighted edges). It corresponds to the solution of Figure 4-2 of the underlying VERTEX COVER problem.

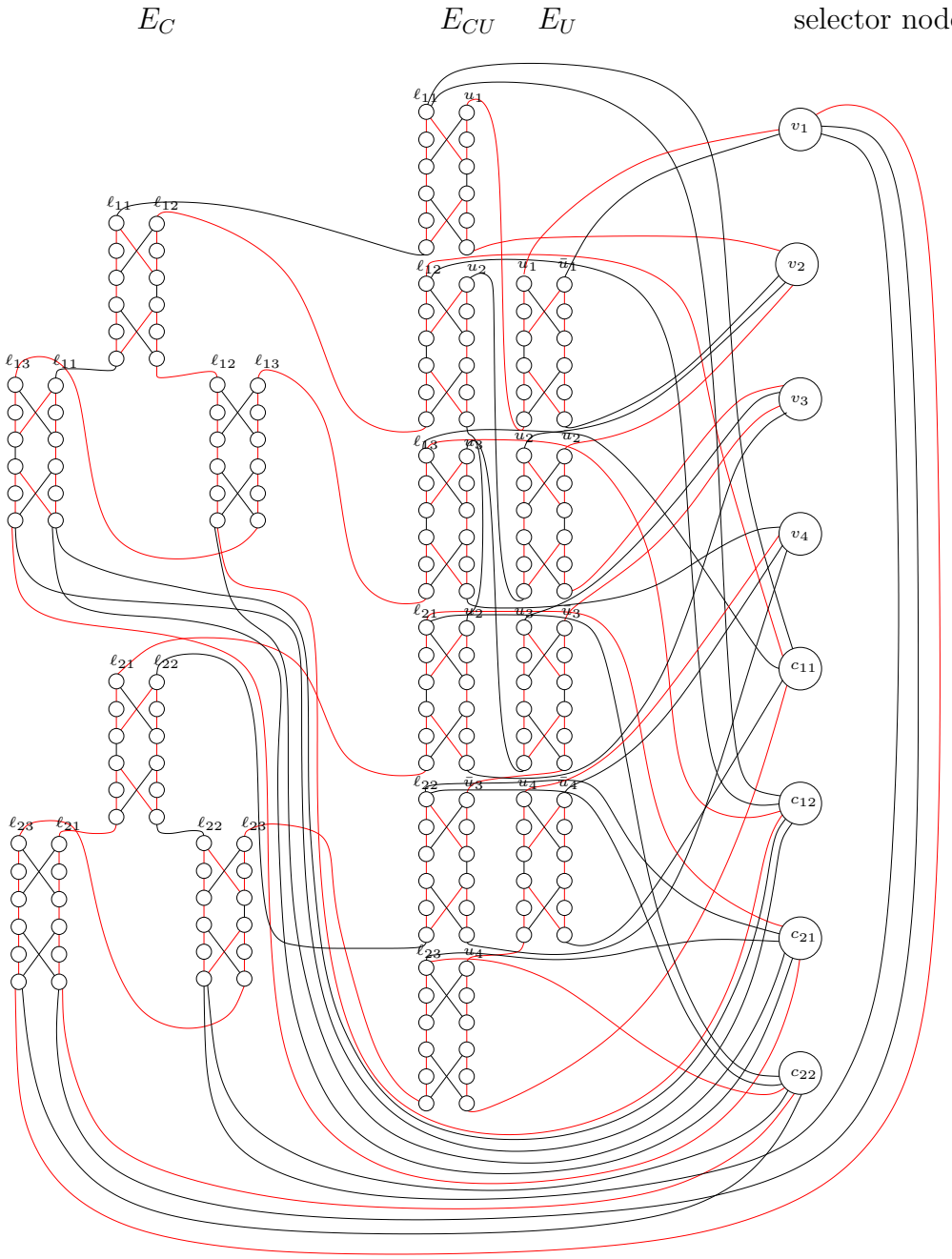

Figure 5-1: Transformation from (3SAT) to (HC) via (VC)

We now define a bijective function g between the set of variables of $I_{(3SAT)}$ and a subset of edges in $I_{(HC)}$. For a Boolean variable $u_i \in U$ we define $g(u_i)$ as the edge that connects the start node of the chain that corresponds to the true-setting vari-

able u_i in $I_{(\mathrm{VC})}$ with the variable selector node v_i, for $i = 1, \ldots, |U|$. Notice that this transformation from (3SAT) to (HC) with this function g satisfies the conditions of Theorem 2.1.1.

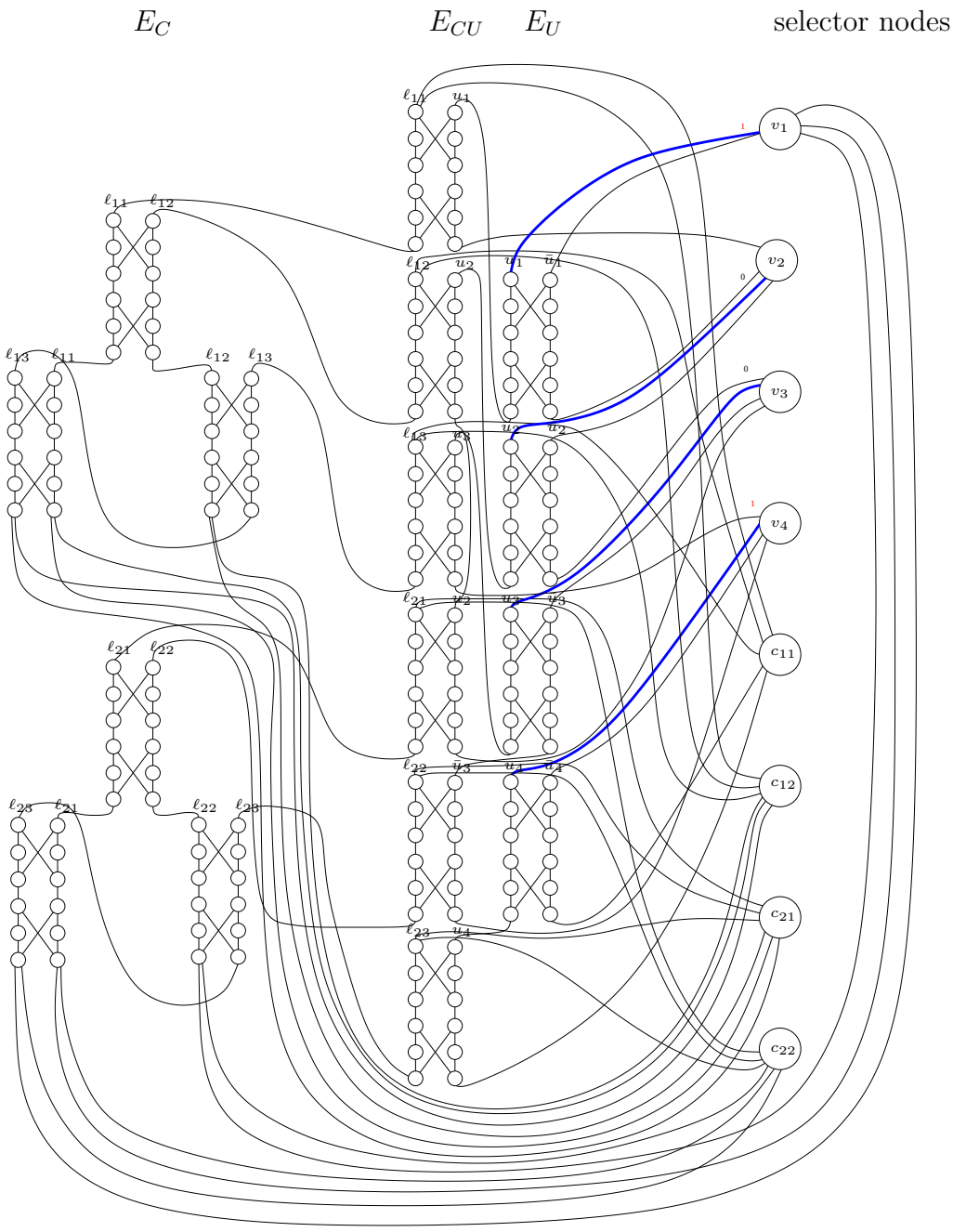

Figure 5-2: Function g in the transformation from (3SAT) to (HC)

Figure 5-2 highlights the set $g(U)$ in our HAMILTONIAN CYCLE instance.

Now that we have completed the description of the value preserving polynomial transformation from (3SAT) to HAMILTONIAN CYCLE, we can proceed with presenting our polynomial transformation from (B_2^{CNF}) to (CExplHC). So given an instance $I_{(B_2^{CNF})}$, with variable set $U = X \cup Y$, we need to define a graph property $\mathcal{P}_{I_{(B_2^{CNF})}}$. We first apply the previously discussed transformation from (3SAT) to HAMILTONIAN CYCLE to the (3SAT) instance $I_{(3SAT)}$ that underlies $I_{(B_2^{CNF})}$. Let $I_{(HC)}$ be the resulting HAMILTONIAN CYCLE instance. Let V_X denote the set of variable selector nodes that correspond to a variable that is part of the subset X of Boolean variables in $I_{(B_2^{CNF})}$. Hence, $|V_X| = |X|$. Let E_X denote the set of edges in $I_{(HC)}$ that connect the start node of a chain that corresponds to the true-setting node of a variable in X with a variable selector node in V_X. Let $\bar{E}_X$ denote the set of edges that connect the start node of a chain that corresponds to the false-setting node of a variable in X with a variable selector node in V_X.

We now create a class $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(HC))}$ of graphs as follows. A graph G is in the class $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(HC))}$ if it contains all nodes from $I_{(HC)}$ and all edges from $I_{(HC)}$ with the following exception: for every variable selector node v in V_X, either its incident edge $e \in E_X$ or its incident edge $\overline{e} \in \bar{E}_X$ is part of G, but not both. Thus, by enumerating over all possibilities to include either e or $\overline{e}$ for each $v \in V_X$, the class $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(HC))}$ contains $2^{|X|}$ many graphs. We say that a graph has property $\mathcal{P}_{I_{(B_2^{CNF})}}$ if it is part of $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(HC))}$. That is, it can be derived from $I_{(B_2^{CNF})}$ in the described way. Note that one can check in polynomial time whether a given graph has this property.

This completes the formal description of our polynomial transformation from (B_2^{CNF}) to (CExplHC). It remains to show that yes-instances of (B_2^{CNF}) correspond to yes-

instances of (CExplHC) and vice versa.

If $I_{(B_2^{CNF})}$ is a yes-instance, then there is a truth assignment $S_{(B_2^{CNF})}$ for its variables in X such that there is no assignment for its remaining variables in Y so that the combined truth assignment satisfies its Boolean expression E. Thus, there is a graph G in $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(\mathrm{HC}))}$, which does not contain a Hamiltonian cycle and hence constitutes a counterexample to the conjecture that every graph with property $\mathcal{P}_{I_{(B_2^{CNF})}}$ is Hamiltonian. One such graph G is the one that includes the edge e_i if and only if $S_{(B_2^{CNF})}(u_i) = 1$ for all $u_i \in X$. If, however, $I_{(B_2^{CNF})}$ is a no-instance, that is, for each truth assignment for the X-variables there is an assignment for the Y-variables such that the combined truth assignment is feasible, then every graph in $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(\mathrm{HC}))}$ has a Hamiltonian cycle and $D_{(\mathcal{P}_{I_{(B_2^{CNF})}},(\mathrm{HC}))}$ does not contain a counterexample to the conjecture $Conj(\mathcal{P}_{I_{(B_2^{CNF})}}, \mathrm{HC})$.

It follows that (CExplHC) is Σ_2^p-complete. □

Given this result, it may be less surprising that for many conjectures about Hamiltonian graphs, finding a counterexample has been elusive. Furthermore, it may suggest that disproving a conjecture might often be harder than commonly believed and deserves, in many cases, equal merits as proving the correctness of a conjecture.

5.2 Counterexamples to NP-Conjectures

Our analysis is of course not limited to conjectures concerning Hamiltonian cycles. We now prove Σ_2^p-completeness for a whole variety of counterexample problems, in a similar fashion to previous sections on other generic classes of Σ_2^p-complete problems, such as the preprocessing problems in Section 4.2, or the adversarial problems in Section 2.1.

We consider once more combinatorial feasibility problems in NP, defined as follows.

PROBLEM: COMBINATORIAL FEASIBILITY PROBLEM IN NP (NP)
INSTANCE: Ground set Z of 0-1 variables, (implicitly given) set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible solutions.
QUESTION: Is $\mathcal{F}$ non-empty?

For example, in the previous section, Z represented the set of edges of a graph G, and $\mathcal{F}$ represented the set of all Hamiltonian cycles in G, encoded as 0-1 vectors. Our conjecture $Conj(\mathcal{P}_I, NP)$ is now based on the subset $D_{(\mathcal{P}_I,NP)}$ of instances of (NP) that share property $\mathcal{P}_I$. Note that we will again assume that $D_{(\mathcal{P}_I,NP)}$ can be encoded succinctly, by specifying the instance I of (B_2^{CNF}) from which it is derived, for example.

The conjecture $Conj(\mathcal{P}_I, NP)$ states that every instance in $D_{(\mathcal{P}_I,NP)}$ has a feasible solution, that is, every instance in $D_{(\mathcal{P}_I,NP)}$ is automatically a yes-instance of (NP). Based on the problem (NP) defined above, we define the problem of finding a counterexample to an NP-conjecture as follows.

PROBLEM: COUNTEREXAMPLE TO NP-CONJECTURE (CExplNP)
INSTANCE: Property $\mathcal{P}_I$.
QUESTION: Is there a counterexample to conjecture $Conj(\mathcal{P}_I, NP)$?

It will be helpful to read the following theorem and proof with the proof of Theorem 2.1.1 for adversarial problems in mind, which was discussed in Section 2.1. In order to prove Σ_2^p-completeness for the class of adversarial problems, we showed, under certain conditions, the existence of a polynomial transformation from (B_2^{CNF}) to the adversarial version of a combinatorial feasibility problem. We now expand on this transformation to map instances of (B_2^{CNF}) to instances of (CExplNP). Given

an instance I of (B_2^{CNF}), every element of $D_{(\mathcal{P}_I,NP)}$ will correspond to exactly one 0-1 assignment for the variables in X of the instance I. The question for the corresponding adversarial problem was whether the instance is feasible, meaning, is there a 0-1 assignment for the variables in X such that there is no feasible extension to the variables in Y? Now the question is whether the set $D_{(\mathcal{P}_I,NP)}$ of input instances for the problem of COUNTEREXAMPLE TO NP-CONJECTURE does contain an infeasible element. The property $\mathcal{P}_I$, which all these input instances have in common, is the fact that they all can be derived from I in the way described.

Theorem 5.2.1. *Let (NP) be a combinatorial feasibility problem which has the following properties.*

(a) *(NP) is in NP.*

(b) *There is a value preserving polynomial transformation f from (3SAT) to (NP).*

(c) *If some of the variables in an instance $I_{(NP)}$ of (NP) are fixed a priori to either 0 or 1, the resulting instance is still of type (NP). That is, one can create in polynomial time an instance $I'_{(NP)}$ of (NP) such that $I'_{(NP)}$ is a yes-instance if and only if $I_{(NP)}$ is a yes-instance.*

Then there is a polynomial transformation from (B_2^{CNF}) to (CExplNP). In particular, (CExplNP) is Σ_2^p-complete.

Proof. The main idea of the proof is the following. Given an instance I of (B_2^{CNF}), we will construct a set $D_{(\mathcal{P}_I,NP)}$ of instances of (NP) which collectively form an instance $I_{(\text{CExplNP})}$ of (CExplNP) such that I is a yes-instance if and only if $D_{(\mathcal{P}_I,NP)}$ contains a no-instance. Property $\mathcal{P}_I$ itself is then defined by the specific way in which the instances of $D_{(\mathcal{P}_I,NP)}$ are derived from the original (B_2^{CNF}) instance I.

Consider an instance I of (B_2^{CNF}) with the set U of Boolean variables partitioned into X and Y. Let E be the Boolean expression in conjunctive normal form with three Boolean variables per clause. Let $I_{(3\text{SAT})}$ be the corresponding (3SAT) instance. That is, $I_{(3\text{SAT})}$ has the same set U of Boolean variables, $U = X \mathring{\cup} Y$, and the same set of clauses, and hence the same Boolean formula E.

Let $I_{(NP)} = f(I_{(3\text{SAT})})$ be the instance of the combinatorial feasibility problem (NP) that results when we apply the transformation f to $I_{(3\text{SAT})}$. This defines the ground set Z and the set of feasible solutions $\mathcal{F}$. Let g be the bijective function that is part of f, with the properties described in the definition of a value preserving polynomial transformation.

We define $D_{(\mathcal{P}_I,NP)}$, the set of instances of (NP) that satisfy property $\mathcal{P}_I$ as follows. Let $S_1, S_2, \ldots, S_{2^{|X|}}$ denote all possible 0-1 assignment to the variables in $g(X)$. With each S_j we associate an instance I'_j of $D_{(\mathcal{P}_I,NP)}$. This is done by fixing the variables of instance $I_{(NP)}$ according to S_j, which results in an instance I_j. Assumption (c) guarantees that I_j can be transformed (if needed) into an equivalent instance I'_j of (NP).

To complete the construction of instance $I_{(\text{CExplNP})}$, we say that an instance of (NP) has property $\mathcal{P}_I$ if it can be derived from the (B_2^{CNF}) instance I by way of the construction that we just described. In other words, $D_{(\mathcal{P}_I,NP)}$ is the set of all these instances, by definition. As a consequence, if $I_{(\text{CExplNP})}$ is a yes-instance, then there is an element in $D_{(\mathcal{P}_I,NP)}$ of size polynomial in that of $I_{(\text{CExplNP})}$, which is a counterexample to the NP-conjecture. Checking that it is indeed a counterexample involves solving an instance of (NP), which is a problem in NP. Therefore, (CExplNP) is in Σ_2^p.

It remains to show that I is a yes-instance if and only if $I_{(\text{CExplNP})}$ is a yes-instance.

Let us first assume that I is a yes-instance. Consider a solution S_X of I. This means that for the 0-1 assignment S_X to the variables in X there is no assignment S_Y for the variables in Y such that the combined assignment satisfies E. By the properties of a value preserving transformation, there is an index $j \in \{1, 2, \ldots, 2^{|X|}\}$ such that $S_X(x) = S_j(g(x))$ for all $x \in X$. Moreover, there is no way of extending S_j to a solution $S \in \mathcal{F}$. This implies that I_j and hence I'_j is a no-instance, so it does represent a counterexample to conjecture $Conj(\mathcal{P}_I, NP)$. This makes $I_{(\text{CExplNP})}$ a yes-instance as well.

We now assume that $I_{(\text{CExplNP})}$ is a yes-instance. This means that there is an instance $I'_j \in D_{(\mathcal{P}_I, NP)}$, for some $j \in \{1, 2, \ldots, 2^{|X|}\}$, such that I'_j is a no-instance. Therefore, I_j is a no-instance as well and any 0-1 assignment to the variables in $Z \setminus g(X)$ paired with the 0-1 values assigned to the variables in $g(X)$ according to the definition of S_j results in an infeasible solution. It now follows from the properties of a value preserving transformation that there is a 0-1 assignment S_X to the variables in X, namely $S_X(x) = S_j(g(x))$, so that there is no 0-1 assignment S_Y to the variables in Y that would make (S_X, S_Y) a satisfying truth assignment for E. Hence, I is a yes-instance. This completes the proof. □

Note that the proof implies that we could have stated Theorem 5.2.1 in slightly more general terms. Instead of requiring that every instance of (NP) and every possible way of fixing some of its variables to 0 or 1 has an equivalent (NP) instance, it would be sufficient to make this assumption for the instances resulting from the transformation from (B_2^{CNF}) only. In fact, in this relaxed form, Theorem 5.2.1 would subsume Theorem 5.1.1, which established the Σ_2^p-completeness of the COUNTEREXAMPLE HAMILTONIAN CYCLE problem. There we were able to force the use of a certain edge (that is, set its corresponding variable to 1) by deleting the only other

possible edge a Hamiltonian cycle could take. Sometimes, however, it may be slightly less straightforward to ensure that Assumption (c) (or its relaxed version) is satisfied. The next section contains such an example.

5.3 Counterexample to Vertex Cover Conjectures

Recall the VERTEX COVER problem, which is NP-complete.

PROBLEM: VERTEX COVER (VC)
INSTANCE: Graph $G = (V, E)$, integer K.
QUESTION: Does G have a vertex cover of size at most K?

Thus, the ground set of variables corresponds to the set V of nodes in G. The set of feasible solutions $\mathcal{F}$ corresponds to the set of all subsets $V^\star$ of nodes such that $V^\star$ is a vertex cover for G and $|V^\star| \leqslant K$. Thus, the VERTEX COVER problem may be regarded as one particular combinatorial feasibility problem.

The conjecture $Conj(\mathcal{P}_I, VC)$ is now based on the problem (VC) and states that, given some property $\mathcal{P}_I$, every VERTEX COVER instance with property $\mathcal{P}_I$ has a vertex cover of at most a given size. Hence, the corresponding COUNTEREXAMPLE TO VERTEX COVER CONJECTURE problem (CExplVC) is defined as follows.

PROBLEM: COUNTEREXAMPLE TO VERTEX COVER CONJECTURE (CExplVC)
INSTANCE: Property $\mathcal{P}_I$.
QUESTION: Is there a counterexample to conjecture $Conj(\mathcal{P}_I, VC)$?

Recall the transformation from the 3-SATISFIABILITY problem (3SAT) to the VERTEX COVER problem, which was discussed in Section 4.3. As already pointed out in

Section 4.3, this transformation is polynomial and value preserving. We describe in the following why Assumption (c) is satisfied as well.

It suffices to argue that we can transform any vertex cover instance in which certain nodes are "enforced" (the corresponding variable is fixed to 1) and in which others are "forbidden" (the corresponding variable is fixed to 0) into a regular vertex cover instance. This is achieved by the following algorithm.

1. For each node whose value is fixed to 1 do: Remove this node and its incident edges. Reduce the value of K by one.

2. For each node whose value is fixed to 0 do: Remove this node and fix the value of each adjacent node to 1. (If an adjacent node was supposed to have the value 0, stop. The instance is a no-instance.)

3. If there are nodes whose value is fixed to 1 that have not been removed yet, goto step 1. Otherwise, stop. Return the remaining instance.

It is not difficult to show that the resulting instance is indeed a VERTEX COVER instance, and it is a yes-instance if and only if the original instance (with the fixed variables) is a yes-instance.

It follows that we may apply Theorem 5.2.1 to the VERTEX COVER problem.

Corollary 5.3.1. *Problem (CExplVC),* COUNTEREXAMPLE TO VERTEX COVER CONJECTURE, *is* Σ_2^p*-complete.*

Chapter 6

More Σ_2^p-Complete Problems

In this chapter we collect a number of additional problems which turn out to be Σ_2^p-complete as well.

6.1 Cost Denying Set Problems

A rather straightforward extension of the results in Section 2 is the addition of the class of cost denying set problems to the family of Σ_2^p-complete problems. In a COST DENYING SET problem we are given a set of 0-1 variables, an integer cost vector on these variables, as well as an integer K, and we want to know whether there is a partial 0-1 assignment of cost at most K that cannot be extended to a feasible 0-1 assignment for all variables. We start this section by establishing Σ_2^p-completeness for the cost denying set problem that is based on the fundamental 3-SATISFIABILITY problem.

6.1.1 Cost Denying 3-Satisfiability

The COST DENYING 3-SATISFIABILITY problem is defined as follows.

PROBLEM: COST DENYING 3SAT (CD SAT)
INSTANCE: Set U of Boolean variables, integer cost vector c on U, Boolean expression E over U in conjunctive normal form with three variables in each clause, integer K.
QUESTION: Is there a partial truth assignment S for a subset of variables in U of cost at most K such that there is no extension of S to a full truth assignment for all variables in U such that E is satisfied?

Theorem 6.1.1. COST DENYING 3SAT *is* Σ_2^p*-complete.*

Proof. It is not hard to see that (CD SAT) belongs to Σ_2^p. We give a polynomial transformation from (B_2^{CNF}) to prove completeness. Let $I_{(B_2^{CNF})}$ be an instance of (B_2^{CNF}). We transform $I_{(B_2^{CNF})}$ into an instance $I_{(\text{CD SAT})}$ of the COST DENYING 3SAT problem as follows. Let the set U of variables of $I_{(\text{CD SAT})}$ be the same as in $I_{(B_2^{CNF})}$, that is, $U = X \cup Y$, and let the Boolean expression E of $I_{(\text{CD SAT})}$ be the same as in $I_{(B_2^{CNF})}$. Let $K = |X|$. Let $c(u) = 1$ if $u \in X$ and $c(u) = K + 1$ if $u \in Y$.

Assume that $I_{(B_2^{CNF})}$ is a yes-instance. Let S_X denote a truth assignment for the X variables in (B_2^{CNF}) such that there is no truth assignment for Y that satisfies E. Let S be a partial truth assignment for $I_{(\text{CD SAT})}$ such that $S(x) = S_X(x)$ for all $x \in X$. Since $K = |X|$, the cost of S is at most K. Since there is no way to extend S_X to satisfy E, there is no extension of S to satisfy E either. Thus, $I_{(\text{CD SAT})}$ is a yes-instance as well.

Assume that $I_{(\text{CD SAT})}$ is a yes-instance. Let S be a partial solution for $I_{(\text{CD SAT})}$ with cost at most K and such that there is no extension of S that satisfies E. Since the cost of S does not exceed K, all variables that received a value in S must be part of X. Let $S_X(x) = S(x)$ for all x that received a value in S. Let $S_X(x) = 0$ for all

other $x \in X$. Since there is no extension of S that would satisfy E, there is also no extension of S_X that would satisfy E. Consequently, $I_{(B_2^{CNF})}$ is a yes-instance as well. □

6.1.2 Generic Cost Denying Set

In this section we formally define the class of cost denying set problems. As was the case for other problem classes, every problem in it is based on a combinatorial feasibility problem.

PROBLEM: COST DENYING SET (CDS)
INSTANCE: Set Z of 0-1 variables, integer cost vector c for Z, implicitly given set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible solutions, integer K.
QUESTION: Is there a partial 0-1 assignment S of cost at most K such that there is no extension of S to a full 0-1 assignment that belongs to $\mathcal{F}$?

Thus, an instance of COST DENYING SET is derived from an instance of (NP) by keeping the sets Z and $\mathcal{F}$, and by adding an integer cost vector c and an integer K.

Instead of using the just established Σ_2^p-completeness of the COST DENYING 3SAT problem, we will base the theorem and proof of Σ_2^p-completeness for the problem COST DENYING SET again on the problem (B_2^{CNF}), as this helps to make the proof very similar to the proof of Theorem 6.1.1.

Theorem 6.1.2. *Let (CDS) be a cost denying set problem. Let (NP) be its underlying combinatorial feasibility problem in NP. Let f be a value preserving polynomial transformation from (3SAT) to (NP). Then there is a polynomial transformation from (B_2^{CNF}) to (CDS), and (CDS) is Σ_2^p-complete.*

Proof. It is straightforward to see that (CDS) is in Σ_2^p.

Consider an instance $I_{(B_2^{CNF})}$ of the problem (B_2^{CNF}). Let $I_{(3SAT)}$ be the corresponding instance of the underlying problem (3SAT) with the same set U of variables and the same set of clauses. Let f be a polynomial transformation with the properties described in the theorem. Let $I_{(NP)} = f(I_{(3SAT)})$ be the instance of (NP) resulting from applying the transformation f to $I_{(3SAT)}$. Let g be the bijective function that corresponds to f as described in the definition of a value preserving polynomial transformation. We define an instance $I_{(CDS)}$ of (CDS) based on $I_{(NP)}$ and by setting $K = |X|$, $c(g(x)) = 1$ for all $x \in X$, and $c(z) = K + 1$ for all $z \in Z \setminus g(X)$.

Assume that $I_{(B_2^{CNF})}$ is a yes-instance and consider a solution S^X of $I_{(B_2^{CNF})}$. We create a solution $S^{(CDS)}$ for $I_{(CDS)}$ from S^X by setting $S^{(CDS)}(g(x)) = S^X(x)$ for all $x \in X$. The cost of $S^{(CDS)}$ is at most K. Suppose that $S^{(CDS)}$ is not a solution for $I_{(CDS)}$, that is, there is a 0-1 assignment S for the variables in $Z \setminus g(X)$ such that $S^{(CDS)}$ together with S represents a feasible solution in $\mathcal{F}$. We need to show that in this case S^X is not a solution of $I_{(B_2^{CNF})}$ either, that is, there is a truth assignment S^Y for the variables in Y of $I_{(B_2^{CNF})}$ such that (S^X, S^Y) satisfies E. We define a 0-1 assignment S^Y for the variables in Y of instance $I_{(B_2^{CNF})}$ by setting $S^Y(u) = S(g(u))$. Due to the properties of f we thus have a 0-1 assignment $S^{(B_2^{CNF})} = (S^X, S^Y)$ for the variables in U such that the expression E is satisfied, and $S^{(B_2^{CNF})}(x) = S^X(x)$ for all $x \in X$. Hence, the assignment S^X does not represent a solution for $I_{(B_2^{CNF})}$ as assumed, since this assignment can be extended by S^Y to a satisfying truth assignment $S^{(B_2^{CNF})}$ for E. Hence we arrived at a contradiction and $S^{(CDS)}$ is in fact a solution for $I_{(CDS)}$ if S^X is a solution for $I_{(B_2^{CNF})}$.

Suppose now that $I_{(B_2^{CNF})}$ is a no-instance. Hence, for every 0-1 assignment S^X for the variables in X there is a 0-1 assignment S^Y for the variables in Y such that E

is satisfied. Consider the corresponding instance $I_{(\mathrm{CDS})}$. For any given 0-1 assignment S^X for the variables in X consider the corresponding 0-1 assignment $S^{(\mathrm{CDS})}$ for the variables in $g(X)$, that is, $S^{(\mathrm{CDS})}(z) = S^X(g^{-1}(z))$. The cost of the assignment $S^{(\mathrm{CDS})}$ is at most K since only variables in $g(X)$ are part of this partial assignment. Since $c(z) > K$ for all $z \notin g(X)$, there is no possible solution to $I_{(\mathrm{CDS})}$ containing variables outside of $g(X)$. We want to show that $S^{(\mathrm{CDS})}$ can be extended to a 0-1 assignment S of all variables in Z. Let $S(z) = S^{(\mathrm{CDS})}(z)$ for all $z \in g(X)$ and $S(z) = S^Y(g^{-1}(z))$ for all $z \in g(Y)$. By now we have an assignment for every variable in $g(U)$. According to the properties of f, this assignment can be extended to all variables in Z to obtain an element of $\mathcal{F}$. As a consequence, $I_{(\mathrm{CDS})}$ is a no-instance as well. This completes the proof. □

6.2 Defining Set Problems

In this section we consider problems that ask whether for a given instance there is a partial solution of prescribed size such that there is exactly one way to complete this partial assignment to a full solution. As pointed out by Hatami and Maserrat [HM05], this notion is related to that of a defining set. Let $\mathcal{F}$ be a family of sets. For a set $S \in \mathcal{F}$, a set $D \subseteq S$ is a *defining set*, for $(\mathcal{F}, S)$, sometimes also refered to as *critical* or *forcing set* in the literature, if S is the only element in $\mathcal{F}$ that contains D as a subset. Defining sets have been studied for various problem types, such as vertex colorings [MNZ97], perfect matchings [AMM04, AHM04, Har93], dominating sets [CGRV97], block designs [Gra90], geodetics [CZ99], orientations [CHSW94, Pas98], Latin squares [BR99, CCS84], and logic [HM05].

The following problem is an example of a "defining partial solution problem." Hatami and Maserrat [HM05] showed that it is Σ_2^p-complete.

PROBLEM: MINIMUM 3SAT DEFINING SET (DEF-SAT)
INSTANCE: Set U of variables, Boolean expression E over U in conjunctive normal form with exactly 3 variables in each clause, integer K.
QUESTION: Does E have a defining set D of size at most K? That is, is there a truth assignment S_D for at most K of the variables in U such that there is a unique extension of S_D to a complete truth assignment S such that E is satisfied?

Here (and henceforth) "defining set" is used for a subset of variables whose values have been fixed to either 0 or 1.

We will once again make use of existing transformations between NP-complete problems to show the existence of polynomial transformations between (DEF-SAT) and the defining set version for a variety of problems in NP. In particular, we will show Σ_2^p-completeness for the following class of defining set problems.

PROBLEM: MINIMUM DEFINING PARTIAL SOLUTION PROBLEM (DEF)
INSTANCE: Disjoint sets X and Y of variables with $X \mathring{\cup} Y = Z$, implicitly given set $\mathcal{F} \subseteq \{0,1\}^Z$ of feasible 0-1 assignments to the variables in Z, integer K.
QUESTION: Is there a subset $X' \subseteq X$ of size at most K and a partial 0-1 assignment $S_{X'}$ to the variables in X' such that $S_{X'}$ can be extended to a feasible solution, and all feasible solutions $S_Z^1, S_Z^2 \in \mathcal{F}$ with $S_Z^1(x) = S_Z^2(x) = S_{X'}(x)$ for all $x \in X'$, satisfy $S_Z^1(x) = S_Z^2(x)$ for all $x \in X \setminus X'$?

Note that, in contrast to the problem MINIMUM 3SAT DEFINING SET, the definition of the DEFINING PARTIAL SOLUTION PROBLEM imposes a partition of the variables into two sets, and only one of these sets may be used to specify the defining set. We will illustrate the necessity of this partition later in this chapter with the help of the MINIMUM DEFINING VERTEX COVER problem, the defining set version of the VER-

TEX COVER problem. The need to have the set of variables partitioned arises because (value preserving) polynomial transformations from 3-SATISFIABILITY to other problems in NP often introduce extra variables, in addition to the variables that result from the bijective function g itself; see, for example, the transformation from (3SAT) to (VC) in Section 4.3.

Theorem 6.2.1. *Let (DEF) be a* MINIMUM DEFINING PARTIAL SOLUTION PROBLEM. *Let (NON-DEF) denote the corresponding underlying combinatorial feasibility problem in NP. Let f be a value preserving polynomial transformation from (3SAT) to (NON-DEF). Then there is a polynomial transformation from (DEF-SAT) to (DEF); in particular, (DEF) is Σ_2^p-complete.*

Proof. We first argue that (DEF) is in Σ_2^p. Suppose we are given, for some yes-instance of (DEF), a defining set X' with a partial solution $S_{X'}$, a unique extension of $S_{X'}$ to a partial solution S_X of all variables in X, and an extension of S_X to a solution S_Z of all variables in Z. It is easily checked that the size of X' is at most K, that S_Z is in $\mathcal{F}$, that S_X is an extension of $S_{X'}$, and that S_Z is an extension of S_X. It remains to be verified that the extension S_X of $S_{X'}$ is unique. In order to do that we can ask, for each variable x in $X \setminus X'$, whether there is a solution in $\mathcal{F}$ that assigns value $(1 - S_X(x))$ to x. Since (NON-DEF) is in NP, this is a problem in NP as well. If the answer is "no" for all variables in $X \setminus X'$, then S_X is a unique extension of $S_{X'}$ and our instance is indeed a yes-instance. Hence, (DEF) is in Σ_2^p.

Consider now an instance $I_{(\text{DEF-SAT})}$ of the problem (DEF-SAT). Let $I_{(\text{3SAT})}$ be the corresponding instance of the underlying problem (3SAT). Let $I = f(I_{(\text{3SAT})})$ be an instance of the problem (NON-DEF) that is obtained by applying the polynomial transformation f to the instance $I_{(\text{3SAT})}$. This provides us with the set Z of variables and its subsets $X = g(U)$ and $Y = Z \setminus X$ for the corresponding instance $I_{(\text{DEF})}$ of

(DEF). The set $\mathcal{F}$ in $I_{(\text{DEF})}$ is the same as in $I_{(\text{NON-DEF})}$ and the integer K in $I_{(\text{DEF})}$ is the same as in $I_{(\text{DEF-SAT})}$. Let g be the bijective function that is part of f, as described in the definition of value preserving polynomial transformations.

Assume that $I_{(\text{DEF-SAT})}$ is a yes-instance and consider a defining set D of size at most K of $I_{(\text{DEF-SAT})}$. Let S_D denote the corresponding partial assignment to D. Let D_1 denote the set of variables in U that are 1 in S_D, and let D_0 be the set of variables in U that are 0 in S_D.

We create a partial solution S' for at most K variables in X for instance $I_{(\text{DEF})}$ as follows. Let $S'(x) = 1$ for all $x \in g(D_1)$, and let $S'(x) = 0$ for all $x \in g(D_0)$. Since $g(u_1) \neq g(u_2)$ for all $u_1, u_2 \in U$, $u_1 \neq u_2$, and $|D| \leqslant K$, S' is well defined and assigns a 0-1 value to at most K variables in X. We claim that S' is a solution of $I_{(\text{DEF})}$.

Observe first that there is an extension S_Z of S' such that $S_Z \in \mathcal{F}$. This follows from the fact that D is a defining set for $I_{(\text{DEF-SAT})}$ and from Property 1 of value preserving transformations. Suppose now that there are two different extensions, S_X^1 and S_X^2, of S' to the remaining variables in X that are each extendable to a full solution in $\mathcal{F}$. We use S_X^1 and S_X^2 to create two truth assignments S^1 and S^2 for the variables in U that satisfy E. Let S^1 be defined by setting $S^1(u) = S_X^1(g(u))$ and let S^2 be defined by setting $S^2(u) = S_X^2(g(u))$, for all $u \in U$. Hence, $S^1 \neq S^2$, but $S^1(u) = S^2(u)$ for all $u \in D$ since $S_X^1(x) = S'(x) = S_X^2(x)$ for all $x \in g(D)$. This implies, by Property (2) of value preserving transformations, that there is more than one extension of S_D that satisfies E, which contradicts the definition of D as a defining set of $I_{(\text{DEF-SAT})}$. Hence, $I_{(\text{DEF})}$ is a yes-instance.

Now assume that $I_{(\text{DEF})}$ is a yes-instance. Let S' be a partial solution for K vari-

ables in X for $I_{(\text{DEF})}$, and let S_X be its unique extension to the remaining variables in X such that S_X can be extended to a full solution in $\mathcal{F}$. Let D be a subset of U such that for each variable $u \in D$ the variable $g(u)$ is among the at most K variables that are part of the partial solution S'. Since g is bijective we have $|D| \leqslant K$. We define S_D from S' by setting $S_D(u) = S'(g(u))$ for all $u \in D$. Since $X = g(U)$ we can define the truth assignment S for $I_{(\text{DEF-SAT})}$ as $S(u) = S_X(g(u))$. To show that S is a unique extension of S_D satisfying E, we suppose the opposite. Let $S_1^{(\text{DEF-SAT})}$ and $S_2^{(\text{DEF-SAT})}$ be each a satisfying truth assignment for E such that the value for the variables in D equal their values in S_D and $S_1^{(\text{DEF-SAT})} \neq S_2^{(\text{DEF-SAT})}$. Hence, according to the definition of f, there are two solutions S_Z^1 and S_Z^2 for $I_{(\text{NON-DEF})}$ such that both S_Z^1 and S_Z^2 are in $\mathcal{F}$, for each variable $x \in X$ we have $S_Z^1(x) = S_1^{(\text{DEF-SAT})}(g^{-1}(x))$ and $S_Z^2(x) = S_2^{(\text{DEF-SAT})}(g^{-1}(x))$, and for all $u \in D$ we have $S_Z^1(g(u)) = S_Z^2(g(u)) = S_D(u)$. Since $S_1^{(\text{DEF-SAT})} \neq S_2^{(\text{DEF-SAT})}$ there must be a variable $u' \in U \setminus D$ such that $S_1^{(\text{DEF-SAT})}(u') \neq S_2^{(\text{DEF-SAT})}(u')$. It follows that $S_Z^1(x') \neq S_Z^2(x')$ for the variable $x' = g(u')$. Since $x' \in X$ we have a contradiction in that there is more than one extension S_X of S' to the remaining variables of X that can be extended to a full solution S_Z in $\mathcal{F}$. We conclude that $I_{(\text{DEF-SAT})}$ is a yes-instance if and only if $I_{(\text{DEF})}$ is a yes-instance. □

6.2.1 Minimum Defining Vertex Cover

Let us illustrate Theorem 6.2.1 on the example of the VERTEX COVER problem, or rather the defining set version thereof, which is defined as follows.

PROBLEM: MINIMUM DEFINING VERTEX COVER (DEF-VC)
INSTANCE: Graph G with disjoint sets of nodes X and Y, integers J and K.
QUESTION: Is there an assignment of 0-1 values to at most J nodes in X such that this assignment can be extended to exactly one assignment of 0-1 values to all nodes

in X such that this assignment can then be extended to a 0-1 solution that represents a vertex cover of G of size at most K?

Recall the transformation from the 3-SATISFIABILITY problem (3SAT) to the VERTEX COVER problem, which we reviewed in Section 4.3. As pointed out there, the transformation is value preserving. Therefore, this transformation is the foundation of the transformation from the MINIMUM 3SAT DEFINING SET problem to the MINIMUM DEFINING VERTEX COVER problem. This transformation introduces, for each Boolean variable u in an instance $I_{(3SAT)}$ of (3SAT), two nodes, the true-setting node u and the false-setting node $\bar{u}$, for the corresponding VERTEX COVER instance $I_{(VC)}$. All true-setting nodes are combined in the set V_U, and g satisfies $g(U) = V_U$. In particular, a solution of an instance $I_{(DEF\text{-}VC)}$ of the MINIMUM 3SAT DEFINING SET problem, if it exists, can be read off the values of these variables in V_U in a solution for the associated instance $I_{(DEF\text{-}VC)}$ of the MINIMUM DEFINING VERTEX COVER problem. Therefore it is important to allow the solution only to choose variables from the set V_U to form a defining set for $I_{(DEF\text{-}VC)}$.

The following example, illustrated in Figure 6-1 shows that the transformation is incorrect if the nodes of the instance of the VERTEX COVER problem are not partitioned and the defining set was allowed to include nodes outside of $X = V_U$. Figure 6-1 shows the vertex cover example of Figure 4-1 in Section 4.3 with $K = 8$ and the set X of nodes encircled. This instance does have a defining set of size at most 3 (it does in fact have four defining sets of size 3), as per the definition of the problem (DEF-VC). One possible defining set $D = \{x_1, x_3, x_4\}$ with $x_1 = 1$, $x_3 = 1$, and $x_4 = 0$, is indicated in Figure 6-1 by nodes depicted as filled circles (value 1), and nodes depicted as circles without fill but with a thick border (value 0). Note that if we allotted D to be completed not only within X but the entire graph G, then there is more than one possible way to extend D to a vertex cover of G of size at most 8. Also, if the

defining set was not restricted to the nodes in X, then there is no defining set of at most 3 nodes.

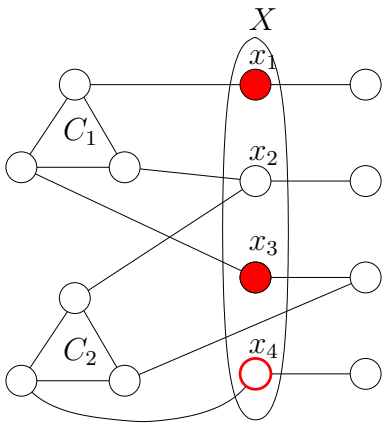

Figure 6-1: Transformation from (B_2^{CNF}) to (DEF-VC)

6.3 Constraint Redundancy Problems

Another interesting class of problems in Σ_2^p is the class of constraint redundancy problems. Imagine an integer programming problem with a multitude of constraints. It is sometimes beneficial to be able to formulate the same optimization problem with only a small subset of these constraints, if possible. However, we show that it is Σ_2^p-complete to answer the question whether there is a set of constraints of at least a given size, that can be removed without altering the solution space. We start by looking at the SATISFIABILITY problem. If the Boolean expression is given in disjunctive normal form, determining whether it is possible to delete a certain number of terms without changing the set of solutions is Σ_2^p-complete [Uma00]. This result leads directly to the Σ_2^p-completeness of the MAXIMUM CLAUSE DELETION problem, which arises if the Boolean formula is given in conjunctive normal form.

The Σ_2^p-complete MAXIMUM TERM DELETION problem is defined as follows.

PROBLEM: MAXIMUM TERM DELETION (DNF-Del)
INSTANCE: Set U of variables, Boolean expression $E_{\text{(DNF-Del)}}$ over U in disjunctive normal form with the set T of terms, integer K.
QUESTION: Is it possible to delete at least K terms from $E_{\text{(DNF-Del)}}$ such that the resulting expression $E'_{\text{(DNF-Del)}}$ is equivalent to $E_{\text{(DNF-Del)}}$?

And here is the formal definition of the MAXIMUM CLAUSE DELETION problem.

PROBLEM: MAXIMUM CLAUSE DELETION (CNF-Del)
INSTANCE: Set U of variables, Boolean expression $E_{\text{(CNF-Del)}}$ over U in conjunctive normal form with the set C of clauses, integer K.
QUESTION: Is it possible to delete at least K clauses from $E_{\text{(CNF-Del)}}$ such that the resulting expression $E'_{\text{(CNF-Del)}}$ is equivalent to $E_{\text{(CNF-Del)}}$?

Theorem 6.3.1. MAXIMUM CLAUSE DELETION *is* Σ_2^p*-complete.*

Proof. The equivalence of two Boolean expressions E and E' can be checked by solving the SATISFIABILITY problem $\neg(E \to E')(E' \to E)$ once. Hence, (CNF-Del) is in Σ_2^p. We will describe a transformation that maps instances $I_{\text{(DNF-Del)}}$ of MAXIMUM TERM DELETION (DNF-Del) to instances $I_{\text{(CNF-Del)}}$ of (CNF-Del).

Let $E_{\text{(CNF-Del)}}$ be the negation of the expression $E_{\text{(DNF-Del)}}$, that is, every "or"-operator in $E_{\text{(DNF-Del)}}$ is replaced with an "and"-operator, every "and"-operator in $E_{\text{(DNF-Del)}}$ is replaced with an "or"-operator, every positive literal in $E_{\text{(DNF-Del)}}$ is replaced with its negated form, and every negative literal is replaced with its positive form. Hence, the resulting Boolean expression $E_{\text{(CNF-Del)}}$ is in conjunctive normal form and, if a

truth assignment makes the expression $E_{\text{(CNF-Del)}}$ true, then the same truth assignment renders the corresponding expression $E_{\text{(DNF-Del)}}$ false, and vice versa.

As a consequence we have that if $I_{\text{(DNF-Del)}}$ is a yes-instance, that is, there is a set of K terms (or more) that can be removed from $E_{\text{(DNF-Del)}}$ such that the resulting expression $E'_{\text{(DNF-Del)}}$ is equivalent to $E_{\text{(DNF-Del)}}$, then removing the corresponding set of K (or more) clauses from $E_{\text{(CNF-Del)}}$ yields an expression $E'_{\text{(CNF-Del)}}$ which is equivalent to $E_{\text{(CNF-Del)}}$. Hence, $I_{\text{(CNF-Del)}}$ is a yes-instance as well.

If $I_{\text{(CNF-Del)}}$ is a yes-instance, then there is a set of (at least) K clauses that can be removed from $E_{\text{(CNF-Del)}}$ such that the resulting expression $E'_{\text{(CNF-Del)}}$ is equivalent to $E_{\text{(CNF-Del)}}$. When removing the corresponding terms from $E_{\text{(DNF-Del)}}$ we obtain $E'_{\text{(DNF-Del)}}$ which is the negated expression of $E'_{\text{(CNF-Del)}}$. Since $E'_{\text{(CNF-Del)}}$ is equivalent to $E_{\text{(CNF-Del)}}$, it follows that $E'_{\text{(DNF-Del)}}$ is equivalent to $E_{\text{(DNF-Del)}}$, which implies that $I_{\text{(DNF-Del)}}$ is a yes-instance as well. □

Using the Σ_2^p-completeness of (CNF-Del) as a starting point, we can easily show that the constraint redundancy problem for integer programs is Σ_2^p-complete as well.

PROBLEM: INTEGER PROGRAMMING EQUIVALENCE (IPEqu)
INSTANCE: 0-1 integer program with set Y of constraints, integer K.
QUESTION: Is there a subset Y' of Y with $|Y'| \leqslant |Y| - K$ such that the integer program with the set Y' as constraints has the same set of 0-1 solutions as the integer program with the set Y as constraints?

Theorem 6.3.2. INTEGER PROGRAMMING EQUIVALENCE *is* Σ_2^p*-complete.*

Proof. We start out by showing that (IPEqu) is in Σ_2^p. Let (IP) denote an instance of (IPEqu) with the full set Y of constraints. Let Y' be our certificate, and let (IP)$'$

be the integer program with the set Y' of constraints. We first check whether (IP) is infeasible. If it is, we check whether (IP)$'$ is infeasible as well. This can be done in nondeterministic polynomial time. So we may assume that (IP) is feasible. Obviously, any feasible solution of (IP) is also a feasible solution of (IP)$'$. Without loss of generality, we assume that all constraints are "$\leqslant$" constraints. For every constraint $y \in Y \setminus Y'$ we build an integer program that maximizes the left-hand side of y over the constraint set Y'. If the optimal value of each integer program of this type is less than or equal to the right-hand side of the corresponding constraint y, then the sets of feasible solutions of (IP) and (IP)$'$ are identical. Checking the certificate can therefore be done via a polynomial time nondeterministic algorithm. Hence, (IPEqu) is in Σ_2^p.

We will transform an instance $I_{\text{(CNF-Del)}}$ of (CNF-Del) to an instance $I_{\text{(IPEqu)}}$ of (IPEqu) by modeling the Boolean formula E as a 0-1 integer program. The integer program (IP) for our instance $I_{\text{(IPEqu)}}$ has the same set U of variables as the instance $I_{\text{(CNF-Del)}}$. For every clause in $I_{\text{(CNF-Del)}}$ we introduce a constraint in which the sum of its variables has to be at least one, where a positive literal appears as is, and a negated literal appears as one minus the variable. For example, the clause $(\overline{x_1} \vee x_2)$ will be transformed into the constraint $(1-x_1)+x_2 \geqslant 1$. Hence, the set Y contains as many constraints as the set C contains clauses, and the number of variables in $I_{\text{(IPEqu)}}$ is identical to the number of variables in U. The resulting integer program (IP) models the CNF formula E of (CNF-Del) perfectly, that is, (IP) is feasible if and only if E is satisfiable and a 0-1 solution to (IP) represents a truth assignment satisfying E, and a truth assignment for E represents a solution to (IP).

It is now straightforward to see that $I_{\text{(CNF-Del)}}$ is a yes-instance if and only if $I_{\text{(IPEqu)}}$ is a yes-instance as well. If $I_{\text{(CNF-Del)}}$ is a yes-instance, then there is a set of K clauses that can be removed from E such that the resulting expression E' is equivalent to E. When

removing the corresponding constraints from the integer program (IP) in $I_{(\text{IPEqu})}$ we obtain a new integer program (IP)$'$ which models the expression E', that is, a 0-1 assignment to the variables of the (IP) represents a satisfying truth assignment for E' if and only if it is a solution to (IP)$'$. Therefore, if E' is equivalent to E, then (IP)$'$ is equivalent to (IP).

The same arguments apply for the case in which $I_{(\text{IPEqu})}$ is a yes-instance, implying that $I_{(\text{CNF-Del})}$ must be a yes-instance as well. □

Bibliography

[AHM04] P. Afshani, H. Hatami, and E.S. Mahmoodian. On the spectrum of the forced matching number of graphs. *Australasian Journal of Combinatorics*, 30:147–160, 2004.

[AMM04] P. Adams, M. Mahdian, and E.S. Mahmoodian. On the forced matching number of graphs. *Discrete Mathematics*, 281:1–12, 2004.

[BA93] O. Ben-Ayed. Bilevel linear programming. *Computer & Operations Research*, 20:485–501, 1993.

[BAB90] O. Ben-Ayed and C.E. Blair. Computational difficulties of bilevel linear programming. *Operations Research*, 38:556–559, 1990.

[BABB88] O. Ben-Ayed, D.E. Boyce, and C.E. Blair. A general bilevel linear programming formulation of the network design problem. *Transportation Research B*, 22:311–318, 1988.

[BABBL92] O. Ben-Ayed, C.E. Blair, D.E. Boyce, and L.J. LeBlanc. Construction of a real-world bilevel linear programming model of the highway network design problem. *Annals of Operations Research*, 34:219–254, 1992.

[Bar69] D.W. Barnette. Conjecture 5. In W.T. Tutte, editor, *Recent Progress in Combinatorics: Proceedings of the Third Waterloo Conference on Combinatorics*. New York: Academic Press, 1969.

[Bar83] J.F. Bard. Coordination of a multidivisional organization through two levels of management. *OMEGA*, 11:457–468, 1983.

[Bar98] J.F. Bard. Practical bilevel optimization: Algorithms and applications. In *Nonconvex Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 1998.

[Bla92] C.E. Blair. The computational complexity of multi-level linear programming. *Annals of Operations Research*, 34:13–19, 1992.

[BM73] J. Bracken and J.M. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21:37–44, 1973.

[BM74] J. Bracken and J.M. McGill. A method for solving mathematical problems with nonlinear programs in the constraints. *Operations Research*, 22:1097–1101, 1974.

[BR99] J.A. Bate and G.H.J. Van Rees. The size of the smallest strong critical set in a Latin square. *Ars Combinatoria*, 53:73–83, 1999.

[BU11] D. Buchfuhrer and C. Umans. The complexity of Boolean formula minimization. *Journal of Computer and System Sciences*, 77:142–153, 2011.

[CCS84] C.J. Colbourn, M.J. Colbourn, and D.R. Stinson. The computational complexity of recognizing critical sets. In *Graph Theory, Singapore 1983*, pages 248–253. Springer Berlin, 1984.

[CFAM81] W. Candler, J. Fortuny-Amat, and B. McCarl. The potential role of multilevel programming in agricultural economics. *American Journal of Agricultural Economics*, 63:521– 531, 1981.

[CGRV97] G. Chartrand, H. Gavlas, and F. Harary R.C. Vandell. The forcing domination number of a graph. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 25:161–174, 1997.

[CHSW94] G. Chartrand, F. Harary, M. Schultz, and C.E. Wall. Forced orientation numbers of a graph. *Congress Numerantium*, 100:183 –191, 1994.

[CKR71] R.G. Cassidy, M.J.L. Kirby, and W.M. Raike. Efficient distribution of resources through three levels of government. *Management Science*, 17:462–473, 1971.

[CMS03] J-P. Côté, P. Marcotte, and G. Savard. A bilevel modelling approach to pricing and fare optimization in the airline industry. *Journal of Revenue and Pricing Management*, 2:23–47, 2003.

[CMS07] B. Colson, P. Marcotte, and G. Savard. An overview of bilevel optimization. *Annals of Operations Research*, 153:235–256, 2007.

[Cob64] A. Cobham. The intrinsic computational difficulty of functions. In *Proceedings of the 1964 Congress for Logic, Mathematics, and Philosophy of Science*, pages 24–30, 1964.

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. The Association for Computing Machinery, New York, 1971.

[Cou94] O. Coudert. Two-level logic minimization: An overview. *Integration, the VLSI Journal*, 17:97–140, 1994.

[CZ99] G. Chartrand and P. Zhang. The forcing geodetic number of a graph. *Discussiones Mathematicae Graph Theory*, 19:45–58, 1999.

[DeL10] E. DeLaVina. Written on the wall II (conjectures of graffiti.pc). `http://cms.dt.uh.edu/faculty/delavinae/research/wowII/`, August 2010.

[Den98] X. Deng. Complexity issues in bilevel linear programming. In A. Migdalas, P.M. Pardalos, and P. Värbrand, editors, *Multilevel Op-*

timization: Algorithms and Applications, pages 149–164. Kluwer Academic Publishers, Dordrecht, 1998.

[DGJ82] D. Duffus, R.J. Gould, and M.S. Jacobson. Forbidden subgraphs and the Hamiltonian theme. In *Proceedings of the 4th International Conference on the Theory and Applications of Graphs*, pages 297–316, 1982.

[DWW95] X. Deng, Q. Wang, and S. Wang. On the complexity of linear bilevel programming. In *Proceedings of the First International Symposium on Operations Research with Applications*, pages 205–212. Beijing, China, 1995.

[Edm65a] J. Edmonds. Minimum partition of a matroid into independent subsets. *Journal of Research of the National Bureau of Standards*, 69B:67–72, 1965.

[Edm65b] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Fan84] Geng-Hua Fan. New sufficient conditions for cycles in graphs. *Journal of Combinatorial Theory, Series B*, 37:221–227, 1984.

[Fis04] K. Fischer. Sequential discrete p-facility models for competitive location planning. *Annals of Operations Research*, 111:253–270, 2004.

[FTL99] M.C. Ferris and F. Tin-Loi. On the solution of a minimum weight elastoplastic problem involving displacement and complementarity constraints. *Computer Methods in Applied Mechanics and Engineering*, 174:107–120, 1999.

[FTL01] M.C. Ferris and F. Tin-Loi. Limit analysis of frictional block assemblies as a mathematical program with complementarity constraints. *International Journal of Mechanical Sciences*, 43:209–224, 2001.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York, 1979.

[Gra90] K. Gray. On the minimum number of blocks defining a design. *Bulletin of the Australian Mathematical Society*, 41:97–112, 1990.

[Har93] F. Harary. Three new directions in graph theory. In M.Kilp and U. Nummert, editors, *Proceedings of the First Estonian Conference on Graphs and Applications*, pages 15–19. University of Tartu, 1993.

[Heu04] C. Heuberger. Inverse combinatorial optimization: A survey on problems, methods, and results. *Journal of Combinatorial Optimization*, 8:329–361, 2004.

[HJS92] P. Hansen, B. Jaumard, and G. Savard. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on Scientific and Statistical Computing*, 13:1194–1217, 1992.

[HLDS04] J. Herskovits, A. Leontiev, G. Dias, and G. Santos. Contact shape optimization: A bilevel programming approach. *International Journal of Structural and Multidisciplinary Optimization*, 20:214–221, 2004.

[HM05] H. Hatami and H. Maserrat. On the computational complexity of defining sets. *Discrete Applied Mathematics*, 149:101–110, 2005.

[HMM85] D.A. Holton, B. Manvel, and B.D. McKay. Hamiltonian cycles in cubic 3-connected bipartite planar graphs. *Journal of Combinatorial Theory*, 38:279–297, 1985.

[HN92] B.F. Hobbs and S.K. Nelson. A nonlinear bilevel model for analysis of electric utility demand-side planning issues. *Annals of Operations Research*, 34:255–274, 1992.

[HS65] J. Hartmanis and R.E. Stearns. On the computational complexity of algorithms. *Transactions of the American Mathematical Society*, 117:285–305, 1965.

[Huy84] Dung T. Huynh. Deciding the inequivalence of context-free grammars with 1-letter terminal alphabet is Σ_2^p-complete. *Theoretical Computer Science*, 33:305–326, 1984.

[Jer85] R. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical Programming*, 32:146–164, 1985.

[Kar72] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[KT91] K.-I Ko and W.-G. Tzeng. Three Σ_2^p-complete problems in computational learning theory. *Computational Complexity*, 1:269–310, 1991.

[KV04] B.Y. Kara and V. Verter. Designing a road network for hazardous materials transportation. *Transportation Science*, 38:188–196, 2004.

[LeB73] L.J. LeBlanc. *Mathematical Programming Algorithms for Large Scale Network Equilibrium and Network Design Problems*. PhD thesis, Northwestern University, Evanston, Illinois, 1973.

[Lev73] L. Levin. Universal sorting problems. *Problemi Peredaci Informacii*, pages 115–116, 1973.

[Lin95] C.-L. Lin. Optimizing TRIEs for ordered pattern matching is /spl pi//sub 2//sup p/-complete. In *Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT'95)*, pages 238–245, Los Alamitos, CA, USA, 1995. IEEE Computer Society.

[LMS98] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44:1608–1622, 1998.

[LO03] T.C. Lai and J.B. Orlin. The complexity of preprocessing. Research Report of the Sloan School of Management, MIT, 2003.

[Mar86] P. Marcotte. Network design problem with congestion effects: A case of bilevel programming. *Mathematical Programming*, 34:23–36, 1986.

[McL84] A. McLoughlin. The complexity of computing the covering radius of a code. *IEEE Transactions on Information Theory*, 30:800–804, 1984.

[MFT92] T. Miller, T.L. Friesz, and R.L. Tobin. Heuristic algorithms for delivered price spatially competitive network facility location problems. *Annals of Operations Research*, 34:177–202, 1992.

[Mig95] A. Migdalas. Bilevel programming in traffic planning: Models, methods and challenge. *Journal of Global Optimization*, 7:381–405, 1995.

[MMP07] Y. Marinakis, A. Migdalas, and P.M. Pardalos. A new bilevel formulation for the vehicle routing problem and a solution method using a genetic algorithm. *Journal of Global Optimization*, 38:555–580, 2007.

[MNZ97] E.S. Mahmoodian, R. Naserasr, and M. Zaker. Defining sets in vertex colorings of graphs and latin rectangles. *Discrete Mathematics*, 167-168:451–460, 1997.

[MS72] A.R. Meyer and L.J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *13th Annual Symposium on Switching and Automata Theory*, pages 125–129. IEEE Computer Society, 1972.

[Orl] J.B. Orlin. Partial inverse optimization problems. Working paper, Sloan School of Management, MIT.

[Pas98] D. Pascovici. On the forced unilateral orientation number of a graph. *Discrete Mathematics*, 187:171–183, 1998.

[PMSP88] P.D. Panagiotopoulos, E.S. Mistakidis, G.E. Stavroulakis, and O.K. Panagouli. Multilevel optimization methods in mechanics. In A. Migdalas, P. Pardalos, and P. Värbrand, editors, *Multilevel Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht, 1988.

[Qui52] W. V. Quine. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59:521–531, 1952.

[SGW08] H. Sun, Z. Gaoa, and J. Wua. A bi-level programming model and solution algorithm for the location of logistics distribution centers. *Applied Mathematical Modelling*, 32:610–616, 2008.

[SM73] L.J. Stockmeyer and A.R. Meyer. Word problems requiring exponential time (preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, pages 1–9, 1973.

[Sto77] L.J. Stockmeyer. The polynomial time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[SU02] M. Schaefer and C. Umans. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33:32–49, September 2002.

[SY80] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *Journal of the ACM*, 27:633–655, 1980.

[Tai84] P. G. Tait. Listing's Topologie. *Philosophical Magazine*, 17:30–46, 1884.

[Tho86] C. Thomasson. Reflections on graph theory. *Journal of Graph Theory*, 10:309–324, 1986.

[TKW98] Dudás T., B. Klinz, and G.J. Woeginger. The computational complexity of multi-level bottleneck programming problems. In A. Migdalas, P.M. Pardalos, and P. Värbrand, editors, *Multilevel Optimization: Algorithms and Applications*, pages 165–179. Kluwer Academic Publishers, Dordrecht, 1998.

[Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2:230–265, 1936.

[Tut46] W. T. Tutte. On Hamiltonian circuits. *Journal of the London Mathematical Society*, 21:98–101, 1946.

[Uma00] C. Umans. *Approximability and Completeness in the Polynomial Hierarchy.* PhD thesis, U.C. Berkeley, 2000.

[VC94] L.N. Vicente and P.H. Calamai. Bilevel and multilevel programming: A bibliography review. *Journal of Global Optimization*, 5:291–306, 1994.

[vS34] H. von Stackelberg. *Marktform und Gleichgewicht (Market and Equilibrium).* Verlag von Julius, Wien, 1934.

[Wag86] K.W. Wagner. The complexity of combinatorial problems with succinct input representation. *Acta Informatica*, 23:325–356, 1986.

[Wra76] C. Wrathall. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science*, 3:23–33, 1976.