

Computational Complexity of Problems in Robust, Bilevel and Online Optimization

Von der Fakultät für Informatik der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Christoph Manfred Grüne, Master of Science

aus Leverkusen, Deutschland

Berichter : Professor Dr. Peter Rossmanith
Professor Dr. Marc Goerigk

Tag der mündlichen Prüfung: 07.10.2025

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek online verfügbar.

Abstract

This thesis deals with the complexity of robust, bilevel, and online optimization. In these areas, we are concerned with min-max (respectively multi-stage) problems from the popular areas of network interdiction, most vital vertex problems, min-max regret optimization, two-stage adjustable problems, recoverable robustness, and online optimization. Although these areas have been well-researched for more than two decades, and one would naturally expect most of the problems to be complete for lower levels of the polynomial hierarchy or polynomial space, almost no hardness results are known. We address this issue by designing two reduction frameworks with the goal of analyzing the common substructure of combinatorial problems that include some form of uncertainty, which induces similar complexity behavior.

The reduction frameworks work as follows. We first provide a general definition of a combinatorial problem, which is also called the nominal problem. Based on this definition of a nominal problem, we are able to derive definitions for more general variations of that problem that additionally model some kind of uncertainty. The general idea is now to examine the reductions between the nominal problems and determine if additional properties hold that help us to directly derive reductions for the more general variations. Accordingly, we provide a definition of such a reduction and show that these properties are already fulfilled by existing reductions or slight modifications of them. We apply these frameworks to nominal problems that are NP-complete and derive complexity results for completeness in the polynomial hierarchy and in PSPACE.

The first framework is concerned with *subset search problems* (or in short SSP), thus it is named SSP framework. A subset search problem is a combinatorial problem that consists of instances \mathcal{I} that contain a *universe* \mathcal{U} together with additional information. A solution $S \in \mathcal{S}$ is a subset of the universe, i.e. $S \subseteq \mathcal{U}$. The corresponding SSP reductions are usual reductions with an additional injective function that maps the universe of the one problem into that of the other. Indeed, many existing reductions on the NP-level fulfill this SSP property. Based on this notion, we are able to “upgrade” the SSP reductions between the nominal problems to reductions between the problem versions in the areas of robust and bilevel optimization. Concretely, we apply this framework to the areas of interdiction, min-max regret optimization, two-stage adjustable robustness, and recoverable robustness, and prove Σ_2^p - and Σ_3^p -completeness for the corresponding problems if the nominal problem is NP-complete.

The second framework considers combinatorial problems, which can be described by an underlying universe U and nested relations R over the universe. The corresponding reduction is termed universe gadget reduction. The special property of a universe gadget reduction from problem A to B is that a gadget is created for every universe element and every relational element from any of the relations of problem A . These gadgets consist of the universe and relation elements of the problem B and are disjoint. Based on this reduction, we are able to show Σ_3^p -completeness for variations of recoverable robust problems and PSPACE-completeness for several online graph problems with a map.

Zusammenfassung

Diese Arbeit befasst sich mit der Komplexität von robuster, zweistufiger und Echtzeit-Optimierung. In diesen Gebieten beschäftigen wir uns mit min-max (beziehungsweise mehrstufigen) Problemen aus den beliebten Themenbereichen von Netzwerk-Untersagung, Wichtigste-Knoten-Probleme, Min-Max-Bedauern-Optimierung, Zwei-Phasen-Anpassbarkeit-Optimierung, wiederherstellbar-robuster Optimierung und Echtzeit-Optimierung. Obwohl diese Gebiete seit mehr als zwei Jahrzehnten erforscht werden und man natürlicherweise die meisten der Probleme als vollständig für die unteren Stufen der polynomiellen Hierarchie oder polynomiellen Platz einschätzt, sind kaum exakte Vollständigkeitsresultate bekannt. Wir adressieren diese Lücke, indem wir zwei Reduktionsrahmenstrukturen entwickeln, um die gemeinsamen Strukturen von kombinatorischen Problemen zu analysieren, die eine Art von Unsicherheit enthalten, die wiederum ein ähnliches Verhalten in ihrer Komplexität induzieren.

Die Reduktionsrahmenstrukturen funktionieren wie folgt. Zuerst definieren wir kombinatorische Probleme im Allgemeinen, diese nennen wir auch nominale Probleme. Basierend auf dieser Definition des nominalen Problems können wir Definitionen für die allgemeineren Varianten ableiten, die bestimmte Arten von Unsicherheiten modellieren. Die übergeordnete Idee ist nun, die Reduktionen zwischen den nominalen Problemen auf gemeinsame Eigenschaften zu untersuchen, die uns helfen, Reduktionen zwischen den allgemeineren Varianten herzuleiten. Demzufolge definieren wir solch eine Art von Reduktion und zeigen, dass diese Eigenschaften bereits von existierenden Reduktionen oder geringen Modifikationen dieser erfüllt werden. Wir wenden die Rahmenstrukturen auf NP-vollständige nominale Probleme an und leiten Komplexitätsresultate für die Vollständigkeit in der polynomiellen Hierarchie oder der Komplexitätsklasse PSPACE ab.

Die erste Rahmenstruktur beschäftigt sich mit Teilmengensuchproblemen (abgekürzt TSP), wir nennen die Rahmenstruktur daher TSP-Rahmenstruktur. Ein Teilmengensuchproblem ist ein kombinatorisches Problem, das aus Instanzen \mathcal{I} besteht, die wiederum ein Universum \mathcal{U} zusammen mit weiteren Informationen enthalten. Eine Lösung $S \in \mathcal{S}$ ist eine Teilmenge des Universums, d.h. $S \subseteq \mathcal{U}$. Die entsprechenden TSP-Reduktionen sind gewöhnliche Reduktionen zusammen mit einer zusätzlichen injektiven Funktion, die das Universum des einen Problems in das des anderen Problems abbildet. Tatsächlich erfüllen viele existierende Reduktionen auf der NP-Ebene genau diese TSP-Eigenschaft. Basierend auf diesem Begriff können wir die TSP-Reduktionen zwischen den nominalen Problemen zu Reduktionen zwischen den Problemversionen im Gebiet von robuster und zweistufiger Optimierung aufrüsten. Wir wenden diese Rahmenstruktur konkret auf die Gebiete von Untersagung, Min-Max-Bedauern-Optimierung, Zwei-Phasen-Anpassbarkeit-Optimierung, und wiederherstellbar-robuster Optimierung an und zeigen Σ_2^P - und Σ_3^P -Vollständigkeit für die entsprechenden Probleme, wenn das nominale Problem NP-vollständig ist.

In der zweiten Rahmenstruktur geht es um kombinatorische Probleme, die mit einem unterliegenden Universum und verschachtelten Relationen über dem Universum beschrieben werden

können. Die entsprechende Reduktion nennen wir Universum-Apparat-Reduktion. Die besondere Eigenschaft einer Universum-Apparat-Reduktion von einem Problem A zu einem Problem B ist, dass ein Apparat für jedes Universumselement und jedes Relationselement des Problems A erzeugt wird. Diese Apparate bestehen aus den Universums- und Relationselementen des Problems B und sind disjunkt. Basierend auf dieser Reduktion sind wir fähig, Σ_3^P -Vollständigkeit für Variationen von wiederherstellbar-robuste Problemen und PSPACE-Vollständigkeit von Echtzeit-Graph-Problemen mit einer Karte zu zeigen.

Declaration of Authorship

I, Christoph Grüne, declare that this thesis and the work presented therein are my own and have been generated by me as the result of my own original research. I do solemnly swear that:

- (1) This work was done completely or mainly while in candidature for the doctoral degree at this faculty and university;
- (2) Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated;
- (3) Where I have consulted the published work of others or myself, this is always clearly attributed;
- (4) Where I have quoted from the work of others or myself, the source is always given. This thesis is entirely my own work, with the exception of such quotations;
- (5) I have acknowledged all major sources of assistance;
- (6) Where the thesis is based on work done by myself jointly with others, I have made clear what was done by others and what I have contributed myself;
- (7) Parts of this work have been published before. A detailed list can be found in Section 1.2.

Aachen, 31st of October in 2025

Acknowledgments

I would like to thank all the supervisors who accompanied and supported me along my doctoral journey, which was unfortunately marked by uncertainty and changes due to various events. First, I want to express my gratitude to Christina Büsing and Arie Koster, with whom I was able to start my doctoral work. Due to a change in topic, Gerhard Woeginger took me on, for which I was very thankful, as this is not something one can take for granted. Following Gerhard's illness and passing, I was supported by Walter Unger, who had to guide the chair through uncertain times once again, and Peter Rossmanith, who welcomed me as his new supervisee. I especially want to thank these two for making everything possible during this time of uncertainty and for their inimitable support. Lastly, I would also like to thank Martin Hofer, who supported me in the final phase of my doctoral work so that I could complete it.

Besides my supervisors, I found a mentor in Janosch, who shared his knowledge patiently in many discussions, both in scientific and teaching contexts. A special thanks goes out to Lasse, who has been another mentor to me. Collaborating with him on numerous shared papers has been a truly rewarding experience, and I always find joy in working together. Additionally, I am grateful for the many opportunities we've had to meet across the globe to discuss our ideas. I also want to thank Anto for introducing me to the high art of server administration. I learned a lot from him at ungodly hours, which certainly was not something one could take for granted. My gratitude extends to Tom, with whom I have been friends since the beginning of our studies. Alongside co-authoring papers together, we have also supervised our foundational lecture on Computability and Complexity several times. Although it was no children's party, it came quite close. Furthermore, I would like to thank Daniel for proofreading parts of my dissertation. My thanks go out as well to all other colleagues and friends who accompanied me on this journey. An important building block of success were also the good souls behind the scenes who organized a lot and made bureaucracy bearable: Birgit, Erika, and Helen.

I would like to extend my thanks to those individuals with whom I have expanded the boundaries of knowledge through many scientific discussions. Thanks go out to Marc Goerigk for reviewing this work alongside my supervisor, Peter. I would like to thank co-authors not previously mentioned: Stephan Zieger, Dorothee Henke, Eva Rotenberg, James B. Orlin, and Femke Pfaue. Additionally, I would like to thank James B. Orlin for his helpful feedback on parts of this thesis. At this point, I also want to say a few words about Gerhard: Although I never had the chance to research alongside him due to his illness, retrospectively, I know he had a significant influence on this dissertation as he inspired many people and brought them together in unexpected ways. This work likely would never have come into existence without his unconscious contributions.

Last but definitely not least, I'd like to thank my family: Heike, Axel, Heinz, Sylke, and Stefan - who have supported me over the years through both beautiful times and challenging circumstances. Without you, all this would not have been possible!

Contents

1	Introduction	13
1.1	Thesis Overview	14
1.2	Bibliographic Note	15
2	Theoretical Foundations	17
2.1	Sets	17
2.2	Graphs	17
2.3	Logic	18
2.4	Complexity Theory	19
2.5	Robust Optimization	27
2.6	Online Optimization	31
I	Subset Search Problem Framework	33
3	Subset Search Problem Framework	35
3.1	Introduction	35
3.2	Framework	41
3.3	SSP-NP-complete Problems	46
4	Minimum Cost Interdiction	47
4.1	Introduction	47
4.2	Interdiction Problems	49
4.3	Combinatorial Interdiction Problems	50
4.4	A Meta-Reduction for Combinatorial Interdiction Problems	50
4.5	Adapting the Meta-Reduction to the Cost Version	53
4.6	The Meta-Reduction is also an SSP reduction	53
5	Min-Max Regret Optimization	57
5.1	Introduction	57
5.2	Min-Max Regret Problems with Interval Uncertainty	58
5.3	Simplifying the Structure: Restricted Interval Min-Max Regret Problems	59
5.4	A Meta-Reduction from Restricted Interval Min-Max Regret Problems	62
5.5	Adapting the Meta-Reduction to the Interval Min-Max Regret Version	63
5.6	The Meta-Reduction is an SSP reduction	65

6	Two-Stage Adjustable Robustness	67
6.1	Introduction	67
6.2	Two-Stage Adjustable Problems	68
6.3	Combinatorial Two-Stage Adjustable Problems	69
6.4	A Meta-Reduction for Combinatorial Two-Stage Adjustable Problems	70
6.5	Adapting the Meta-Reduction to the Cost Version	72
6.6	The Meta-Reduction is an SSP reduction	73
7	Recoverable Robustness	75
7.1	Introduction	75
7.2	Recoverable Robust Problems	76
7.3	An SSP Framework for Recoverable Robust Problems	79
7.4	The Issue of Transitivity: Preserving the Blow-up Gadget	88
8	Minimum Cardinality Interdiction	95
8.1	Introduction	95
8.2	Minimum Cardinality Interdiction Problems	96
8.3	Invulnerability Reductions for Various Problems	101
8.4	Cases where the meta-theorem does not apply	108
9	SSP Reduction Compendium	115
10	Conclusion	137
II	Universe Gadget Reduction Framework	139
11	Universe Gadget Reduction Framework	141
11.1	Introduction	141
12	Recoverable Robustness	147
12.1	Introduction	147
12.2	Combinatorial Problem Framework	148
12.3	Recoverable Robust Problems and the Polynomial Hierarchy	152
12.4	Classes of Recoverable Robust Problems	156
13	Online Graph Problems	181
13.1	Introduction	181
13.2	Online Vertex Subset Game Framework	182
13.3	Gadget Reductions	183
13.4	A Reduction Framework for Online Vertex Subset Games	185
13.5	Applying the Framework to Vertex Cover	190
13.6	More Vertex Subset Problems	196
14	Conclusion	203

Chapter 1

Introduction

In everyday life, we encounter a manifold of situations in which we need to guarantee an efficient usage of resources. In the field of optimization, these situations are encountered by introducing corresponding models and employing algorithms to solve for the best possible resource allocation. Optimization is applied in various fields such as infrastructure planning and maintenance, automation, production planning, finance, and many areas of research. In the real world, however, uncertainty plays a key role. What do we do if there is a faulty sensor or a traffic jam on the road on which we need to travel? Then we would like to be able to encounter this uncertainty by employing corresponding strategies. The field of optimization under uncertainty is a subfield of optimization in which those strategies are developed and different forms of uncertainty are studied. In this thesis, we are concerned with three different areas in optimization under uncertainty: Robust optimization, bilevel optimization, and online optimization.

Robust optimization deals with problems that include a measure of robustness to model uncertainty within an optimization problem. This uncertainty is typically modeled by an uncertainty set that contains different scenarios that may occur. The goal is now to find a solution that is immune to all scenarios. That is, no matter in which way the uncertainty realizes itself, there is a good solution that can be used. In order to analyze the worst-case behavior, one can model this problem setting also as a game between a decision maker who plays against an adversary: The decision maker wants to compute a solution of minimal costs, while the adversary tries to maximize the costs for the decision maker. Accordingly, these problems typically have a min-max optimization goal. We can also extend this structure to min-max-min problems or, in general, to multi-stage problems. Popular robustness concepts are interdiction, min-max regret, recoverable robustness, and many more.

Bilevel problems are related to robust optimization in the sense that in a bilevel problem, one problem is embedded into the other. Therefore, the min-max structure also applies here. In a bilevel problem, we have a leader who controls the decisions on the outer problem, and a follower who controls the decisions on the inner problem. Each of them is assigned an optimization goal. Both optimization goals, however, do not necessarily oppose each other, in contrast to robust optimization. We can interpret this situation again as a game between the leader and the follower: The leader first decides on a solution to the outer problem while incorporating the possible decisions of the follower in the inner problem. Then, given the decisions of the leader, the follower decides on a solution to the inner problem. Popular concepts are Stackelberg games, toll setting problems, defense settings (related to interdiction), and many more.

Online problems also deal with a form of uncertainty: The input is not known upfront but is revealed piecewise instead. More precisely, during the decision process, more and more information is revealed, and irrevocable decisions have to be made for each new piece of information.

The uncertainty thus results from not knowing (parts of) the instance at the start and in the process of the decision-making. This again can be modeled as a two-player game between the online decision maker and an adversary, where the online decision maker wants to maximize its performance and the adversary wants to minimize it.

In this thesis, we analyze the complexity of problems encountered in the areas of robust, bilevel, and online optimization. The main tool to solve problems in combinatorial optimization is Mixed Integer Programming (MIP). In the last decades, efficient algorithms were developed to solve problems if they can be modeled as a polynomial-sized (or compact) MIP. Modeling a problem as MIP means that there is an efficient algorithm, a so-called reduction, that constructs a compact model of the problem. The concept of a reduction is at the heart of the field of computational complexity theory. In this field, we are interested in classifying problems by the amount of resources that are needed to solve these problems. If an efficient reduction from a problem A to MIP exists, we know that any MIP-solving algorithm can also solve problem A . Vice versa, if we are, however, able to show that such a reduction cannot exist, then we know that we need to employ more sophisticated strategies to encounter such a problem A . This enables us to assess whether and how problem A can be solved. In order to classify the problems from the areas of robust, bilevel, and online optimization, we use their common substructure of the underlying uncertainty to show that a multitude of problems is complete for the lower levels of the polynomial hierarchy or PSPACE. This is done by introducing a framework that “upgrades” the reductions between the nominal problems to reductions between the robust, bilevel, or online version of the problem. This induces (under the assumption that $\text{NP} \neq \Sigma_2^P$) that efficient reductions to compact MIP models are in general not available. Thus, new tools need to be developed to encounter the problems from robust, bilevel, and online optimization.

1.1 Thesis Overview

In this thesis, we present two reduction frameworks to classify the complexity of problems from robust, bilevel, and online optimization. We divide the two frameworks into two parts because of their slightly different nature. We start by giving an introduction to the theoretical foundations that are necessary to understand the content of the thesis. In the following chapters, we present the *subset search problem framework* and the *universe gadget reduction framework*:

In Part I, we present the subset search problem framework, or SSP framework for short. A subset search problem is a combinatorial problem that is composed of a ground set \mathcal{U} of combinatorial elements. All solutions S to such a problem can be defined by a subset of elements of that ground set $S \subseteq \mathcal{U}$. We first introduce the SSP framework by presenting the fundamental ideas and the corresponding reductions between the problems in Chapter 3. In the following chapters, we apply the framework to various fields in robust and multi-stage optimization. The goal for all the areas is to show completeness for the lower levels of the polynomial hierarchy if the nominal problem is NP-complete. We start by showing that minimum cost interdiction problems with discrete budgeted uncertainty are Σ_2^P -complete in Chapter 4. In Chapter 5, we apply the SSP framework to min-max regret problems with interval uncertainty and show their Σ_2^P -completeness. We then proceed with two-stage adjustable robustness with discrete budgeted uncertainty in Chapter 6 and show their Σ_3^P -completeness. For the following two fields of recoverable robustness and minimum cardinality interdiction, it is not directly possible to use the standard SSP framework. In Chapter 7, we present an extension to SSP reductions, which we call blow-up SSP reductions. By using the additional blow-up property, we are able to show that recoverable robust problems with discrete budgeted uncertainty are Σ_3^P -complete for a wide range of natural distance measures between the solutions. In Chapter 8, we return to interdiction problems, albeit in a more general form: minimum cardinality interdiction with discrete budgeted

uncertainty. For this, we reuse the Σ_2^P -completeness of minimum cost interdiction problems and present a complementary reduction, so-called invulnerability reductions, to show that the more general minimum cardinality interdiction problems are also Σ_2^P -complete. In Chapter 9, we present all problems and reductions in a compendium-style list. At last, we conclude Part I in Chapter 10.

In Part II, we present the universe gadget reduction framework. This framework is an earlier and more technically involved framework than the SSP framework. We first present the fundamental definitions and the concept of a gadget reduction in Chapter 11. We then apply the framework to recoverable robust problems with elemental uncertainty in Chapter 12 and show their Σ_3^P -completeness. In Chapter 13, we use a version of universe gadget reductions to show that online problems with a map based on popular graph problems are PSPACE-complete for the neighborhood reveal model. At last, we conclude Part II in Chapter 14.

1.2 Bibliographic Note

This thesis is based on the following five research papers in different stages of publication. All of the papers are joint work with the respective co-authors.

- ▶ Christoph Grüne and Lasse Wulf. On the complexity of recoverable robust optimization in the polynomial hierarchy. In Pawel Gawrychowski, Filip Mazowiecki, and Michal Skrzypczak, editors, 50th International Symposium on Mathematical Foundations of Computer Science, MFCS 2025, August 25-29, 2025, Warsaw, Poland, volume 345 of LIPIcs, pages 52:1–52:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025. [GW25c]
The full version is available on arXiv. [GW24]
- ▶ Christoph Grüne and Lasse Wulf. Completeness in the polynomial hierarchy for many natural problems in bilevel and robust optimization. In Nicole Megow and Amitabh Basu, editors, Integer Programming and Combinatorial Optimization - 26th International Conference, IPCO 2025, Baltimore, MD, USA, June 11-13, 2025, Proceedings, volume 15620 of Lecture Notes in Computer Science, pages 256–269. Springer, 2025. [GW25a]
The full version is available on arXiv. [GW23]
- ▶ Christoph Grüne and Lasse Wulf. The Complexity of Blocking All Solutions. ArXiv: CoRR, abs/2502.05348, 2025. [GW25b]
- ▶ Christoph Grüne. The Complexity Classes of Hamming Distance Recoverable Robust Problems. In José A. Soto and Andreas Wiese, editors, LATIN 2024: Theoretical Informatics - 16th Latin American Symposium, Puerto Varas, Chile, March 18-22, 2024, Proceedings, Part I, volume 14578 of Lecture Notes in Computer Science, pages 321–335. Springer, 2024. [Grü24]
The full version is available on arXiv. [Grü22]
- ▶ Janosch Fuchs, Christoph Grüne, and Tom Janßen. The Complexity of Online Graph Games. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024, Cochem, Germany, February 19-23, 2024, Proceedings, volume 14519 of Lecture Notes in Computer Science, pages 269–282. Springer, 2024. [FGJ24]
The full version is available on arXiv. [FGJ22]

Further Publications. During his doctoral studies, the author also contributed to the following publications. These publications are not part of the thesis.

- ▶ Janosch Fuchs, Christoph Grüne, and Tom Janßen. The Complexity of Graph Exploration Games. In Rastislav Kráľovic and Vera Kurková, editors, SOFSEM 2025: Theory and Practice of Computer Science - 50th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2025, Bratislava, Slovak Republic, January 20-23, 2025, Proceedings, Part II, volume 15539 of Lecture Notes in Computer Science, pages 17–30. Springer, 2025 [FGJ25]
The full version is available on arXiv. [FGJ23]
- ▶ Christoph Grüne and Stephan Zieger. Solving the Dial-a-Ride Problem for Railway Traffic by Means of Heuristics. In Cornel Klein, Matthias Jarke, Jeroen Ploeg, Markus Helfert, Karsten Berns, and Oleg Gusikhin, editors, Smart Cities, Green Technologies, and Intelligent Transport Systems - 11th International Conference, SMARTGREENS 2022, and 8th International Conference, VEHITS 2022, Virtual Event, April 27-29, 2022, Revised Selected Papers, volume 1843 of Communications in Computer and Information Science, pages 93–133. Springer, 2022. [GZ22b]
- ▶ Christoph Grüne and Stephan Zieger. Demand-responsive Scheduling in Railway Transportation. In Jeroen Ploeg, Markus Helfert, Karsten Berns, and Oleg Gusikhin, editors, Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2022, Online Streaming, April 27-29, 2022, pages 239–248. SCITEPRESS, 2022. [GZ22a]

Further Projects. The author of this thesis also launched the related project “reductions.network: A compendium of reductions” [GP25]. This project is a database for problems and reductions of various complexity classes. This database is visualized via an interactive graph. The vertices are the problems, and the arcs are the reductions. By clicking on the corresponding object, one can obtain further information. The author of this thesis supervised the following theses, which contributed to this project.

- ▶ Shubham Verma. A Survey on SSP and Parsimonious Reductions and on NP-Complete Problems. Master’s thesis. RWTH Aachen University, 2025.
- ▶ Yin He. Survey on Inapproximability Results for Optimization Problems under PCP Theorem and Unique Games Conjecture. Bachelor’s thesis. RWTH Aachen University, 2025.
- ▶ Yaman Faour. Extending Reductions.Network: A Survey of Parameterized Complexity. Bachelor’s thesis. RWTH Aachen University, 2025.
- ▶ Celina Janet Bartlett. A Compendium of Subset Search Problems and Reductions relating to the Parsimonious Property. Bachelor’s thesis. RWTH Aachen University, 2025. [Bar25]
- ▶ Femke Pfaue. Exploring the Reductions Between SSP-NP-complete Problems and Developing a Compendium Website Displaying the Results. Bachelor’s thesis. RWTH Aachen University, 2024. [Pfa24]

Chapter 2

Theoretical Foundations

In this chapter, we present all the necessary mathematical foundations and notations to understand the content of this thesis. The audience of this thesis is most likely familiar with the notation that we use throughout the thesis, or it is explained in the corresponding context. Accordingly, one might skip this chapter and come back if needed.

2.1 Sets

When referring to sets, we denote sets by uppercase letters A, B and elements with lower case letters a, b . Furthermore, we denote a set of sets by calligraphic letters \mathcal{A}, \mathcal{B} .

The empty set is denoted by \emptyset , the set of all natural numbers by $\mathbb{N} = \{1, 2, \dots\}$. If we refer to the natural numbers including 0, we denote them by \mathbb{N}_0 . Moreover, the set of all integers is denoted by \mathbb{Z} , the set of all rational numbers by \mathbb{Q} , and the set of all real numbers by \mathbb{R} . If we consider a restriction of a set, we denote it by a subscript, for example \mathbb{R}_+ for all positive real numbers. We denote the element relation by $a \in A$ and the subset relation by $A \subseteq B$.

We use the following set operations. We denote the union of two sets by $A \cup B$, the union of all sets in \mathcal{A} by $\bigcup_{A \in \mathcal{A}} A$, and the union of all sets A_1, \dots, A_n by $\bigcup_{i=1}^n A_i$. We denote the set difference by $A \setminus B$. An intersection of two sets A and B is denoted by $A \cap B$. A set A is disjoint from a set B if $A \cap B = \emptyset$. The disjoint union of two sets is denoted by $A \dot{\cup} B$. For the Cartesian product, we use $A \times B$ and the k -ary Cartesian product A^k . The power set of a set A is denoted by 2^A . The symmetric difference of two sets A and B is defined by $A \Delta B = \{x \mid \text{either } x \in A \text{ or } x \in B\}$.

The cardinality $|A|$ of a finite set A is the number of elements contained in A . The cardinality of the empty set is $|\emptyset| = 0$ and of an infinite set is ∞ . A k -partition of a set A are k subsets A_1, A_2, \dots, A_k such that $A_1 \dot{\cup} A_2 \dot{\cup} \dots \dot{\cup} A_k = A$. We might also use weight functions for set elements, which are defined by $w : A \rightarrow \mathbb{R}$. For brevity, we also write $w(A') := \sum_{a \in A'} w(a)$ for the sum of the weights of all elements from $A' \subseteq A$.

2.2 Graphs

Graphs are mathematical objects that occur in many different optimization problems. The simplest form is an *undirected graph* which consists of a set of *vertices* V and a set of *edges* $E \subseteq 2^V$ with $|e| = 2$ for all $e \in E$. We denote an edge between vertex u and v always by a set $\{u, v\}$ to indicate that an edge is undirected. We also consider *directed graphs*. These also consist of vertices, but instead of edges, we call the relation between the vertices *arcs* $A \subseteq V^2$. This

relation is not symmetric, such that we denote arcs by pairs of vertices, e.g., for an arc between vertex u and v , we write (u, v) . If there is more than one edge $\{u, v\}$ or arc (u, v) between the same vertices u and v , we call this a *multi-edge* or respectively *multi-arc*. An edge $\{v, v\}$ or arc (v, v) from vertex v to the same vertex v is called *loop*. A graph without loops and multi-edges or multi-arcs is called simple. In this thesis, we assume graphs to be finite simple graphs.

A vertex $u \in V$ is *adjacent* to a vertex $v \in V$ if there is an edge $\{u, v\} \in E$ or an arc $(u, v) \in A$ that connects both vertices u and v . We then also call vertex v the *neighbor* of vertex u . An edge $e \in E$ is *incident* to a vertex $v \in V$ if $v \in e$. The same also holds vice versa: A vertex $v \in V$ is incident to an edge $e \in E$ if $v \in e$. We further call edges *adjacent* if they share an endpoint. We use the same definitions analogously for arcs.

The vertices, edges, or arcs may be associated with weights. We denote this by a weight function $w : X \rightarrow \mathbb{R}$ for $X \in \{V, E, A\}$. In order to associate the vertices V , edges E , or arcs A to a graph G , we might use the notation $V(G)$, $E(G)$, or $A(G)$.

A complete graph is a graph with a full edge respectively arc relation, i.e. $E = \{\{u, v\} \mid u, v \in V, u \neq v\}$ or $A = \{(u, v) \mid u, v \in V, u \neq v\}$. A *walk* in graph G is a sequence of edges or arcs r_1, \dots, r_k in G for which there is a sequence of vertices v_0, \dots, v_k in G such that $r_i = \{v_{i-1}, v_i\}$ for edges or respectively $r_i = (v_{i-1}, v_i)$ for arcs. A *path* from vertex $u \in V$ to vertex $v \in V$ is a walk that visits all vertices and edges or arcs respectively at most once. A graph is *connected* if there is a path between all vertices; otherwise, we call it *disconnected*. A *tour* in a graph G is a walk that contains all edges or arcs at least once.

2.3 Logic

In this thesis, we mainly use propositional logic. A propositional logic formula consists of *Boolean variables* and *logical operators*. A Boolean variable has the domain $\{0, 1\}$, which expresses whether the variable evaluates to true (1) or false (0). In order to denote a set of variables, we typically use X, Y, Z . Among the logical operators are $\neg, \wedge, \vee, \rightarrow, \leftarrow, \leftrightarrow, \oplus$. We denote the negation of a variable (and rarely a formula) $\neg x$ also by \bar{x} . We also call x the positive literal and \bar{x} the negative literal of variable $x \in X$. We usually denote the set of all literals by L . We typically denote propositional formulae by the small Greek letters φ and ψ . A propositional formula with free Boolean variables from the variable set X with $|X| = n$ is denoted by $\varphi(X)$ or $\varphi(x_1, \dots, x_n)$. If we want to partition the set X into sets Y, Z , we might also write $\varphi(Y, Z)$. An *assignment* of free Boolean variables is a function $\alpha : X \rightarrow \{0, 1\}$. We define the following semantics for the logical operators from above. A satisfying assignment α for formula φ , denoted by $\alpha \models \varphi$ as follows:

$\alpha \models x$	if	$\alpha(x) = 1,$
$\alpha \models \neg\varphi,$	if	$\alpha \not\models \varphi$
$\alpha \models \varphi \wedge \psi,$	if	$\alpha \models \varphi$ and $\alpha \models \psi$
$\alpha \models \varphi \vee \psi,$	if	$\alpha \models \varphi$ or $\alpha \models \psi$
$\alpha \models \varphi \rightarrow \psi,$	if	$\alpha \not\models \varphi$ or $\alpha \models \psi$
$\alpha \models \varphi \leftarrow \psi,$	if	$\alpha \models \psi \rightarrow \varphi$
$\alpha \models \varphi \leftrightarrow \psi,$	if	$\alpha \models \varphi \leftarrow \psi$ and $\alpha \models \varphi \rightarrow \psi$
$\alpha \models \varphi \oplus \psi,$	if	$\alpha \not\models \varphi \leftrightarrow \psi$

We also denote an assignment as a set of variables $X' \subseteq X$, where the induced assignment is defined by $\alpha(x) = 1$ if and only if $x \in X'$ (otherwise $\alpha(x) = 0$ for $x \notin X'$). We use an analogous definition also for literal sets. That is, the induced assignment by literal set $L' \subseteq L$ with the condition that either $x \in L'$ or $\bar{x} \in L'$ is $\alpha(x) = 1$ if and only if $x \in L'$ (otherwise $\alpha(x) = 0$ for $\bar{x} \in L'$). We call a formula φ satisfiable if a satisfying assignment α for φ exists.

There are different normal forms of propositional formulae. We use the *conjunctive normal form* or in short CNF. In a CNF, the formula consists of *clauses* C_1, \dots, C_m , which are *disjunctions* of literals, i.e. $(x_1 \vee \bar{x}_2 \vee x_3)$. The clauses are then *conjunct* to construct the formula φ , i.e. $\varphi = \bigwedge_{i=1}^m C_i$. A k -CNF is a formula in conjunctive normal form with the restriction that all clauses have length at most $|C_i| = k$ for all $C_i \in C$. The dual of a CNF is the *disjunctive normal form* or, in short, DNF. A DNF formula φ is a disjunction of conjunctions. Correspondingly, a k -DNF is a disjunction of conjunctions with length at most k .

We additionally take quantified propositional formulas into account. Besides Boolean variables and logical operators, in a quantified propositional formula, quantifiers are available. A quantifier is an \exists -quantifier or a \forall -quantifier. A quantifier is always associated with a non-empty variable set. If y is a variable and φ a formula, then $Qy\varphi(y, X)$ is a formula for $Q \in \{\exists, \forall\}$. We can define the semantics as follows for $\alpha : X \rightarrow \{0, 1\}$

$\alpha \models \exists y\varphi(y, X)$, if there is a $z \in \{0, 1\}$ such that $\alpha \models \varphi(z, X)$;

$\alpha \models \forall y\varphi(y, X)$, if for all $z \in \{0, 1\} : \alpha \models \varphi(z, X)$.

2.4 Complexity Theory

Computational complexity theory is a theory that tries to classify computational problems by the resource usage needed to solve them. A computational problem is considered to be a task that can be solved by a computer. The types of resources one can analyze are manifold: time, memory space, communication, etc. In this thesis, we first and foremost analyze problems in terms of their time and space complexity. A problem is considered to be hard to solve if any algorithm that correctly solves the problem needs significant amounts of resources, such as time or space.

Computational Problems. In order to be able to analyze computational problems, we need to define what we mean by the term “computational problem”. First of all, a problem has an input. For an input, we typically choose a word $w \in \{0, 1\}^*$ over the finite alphabet $\{0, 1\}$. We call this word w the instance of the problem. The output to an instance is again a word $o \in \{0, 1\}^*$. Accordingly, we can describe the problem as a function $f : \{0, 1\}^* \rightarrow \{0, 1\}^*$. The task of the computer is then to compute the output $f(w) \in \{0, 1\}^*$ to a given input $w \in \{0, 1\}^*$.

For simplicity, we limit ourselves to *decision problems* in this thesis. In a decision problem, we ask ourselves if the input w to the computer is either a *YES-instance* ($f(w) = 1$) or a *NO-instance* ($f(w) = 0$). We can then formally describe the problem as a subset of $\{0, 1\}^*$, which we also call a *language*.

Definition 2.1 (Language). A language is a set $L \subseteq \{0, 1\}^*$ such that $L = f^{-1}(1)$, where f is a given decision problem.

The language L is the set of all YES-instances of the problem. Consequently, the task of the computer is to answer the question: Is input w part of the language L ? From now on, we see language and the corresponding problem as equivalent and may use language and problem interchangeably. It is now clear what exactly a problem is and how the input and the output of a problem are defined. However, we still need a formal definition of a computation or algorithm.

Turing Machines. Turing machines are an abstract formal computational model [Tur37]. This model is a simple description of a machine that is able to perform all tasks that a computer is capable of. Thus, this model provides a simple mathematical ground to describe algorithms. From now on, we consider algorithms and Turing machines to be equivalent. A Turing machine has an infinite *tape* as its memory, which is composed of discrete cells each containing one symbol of the alphabet (0 or 1). Furthermore, a Turing machine has a *head* that points to exactly one of the cells. At last, a Turing machine has a state, which is part of a given finite state set. We denote the pair of the Turing machine's state, the head position, and the tape content as the *configuration* of the Turing machine.

The computation of a Turing machine works as follows. First, the input word is written on the tape before the computation. The Turing machine starts in the *start state*, which we denote by q_0 . The head of the Turing machine is on the cell of the first symbol of the input word. The combination of the tape content, state, and head position is the start configuration. A step of the Turing machine is performed by reading the symbol at the position of the head and the state the Turing machine is in. Based on the state and the symbol to which the head points, a new symbol is written to the position of the head, the head is moved one cell to the left, the right, or it stays at the same position, and the state is changed to a new state. Consequently, after one step, the Turing machine is in a new configuration. Such a step is defined by the *transition function* δ . The Turing machine halts if the final state \bar{q} is reached. The output is the symbol to which the head points to (either 0 or 1). The sequence of all the configurations reaching from the start configuration to the final configuration is the *computation path* of the Turing machine. Pertaining to this description, we can define the following formal definition of a Turing machine.

Definition 2.2 (Turing Machine, from [HMU07]). *A Turing machine is defined by a 7-tuple $(Q, \Sigma, \Gamma, B, q_0, \bar{q}, \delta)$ such that*

- ▶ Q is a finite set of states,
- ▶ $\Sigma = \{0, 1\}$ is the finite input alphabet,
- ▶ $\Gamma = \{0, 1, B\}$ is the finite tape alphabet,
- ▶ $q_0 \in Q$ is the start state,
- ▶ \bar{q} is the final state, and
- ▶ $\delta : (Q \setminus \{\bar{q}\}) \times \Gamma \rightarrow Q \times \Gamma \times \{R, L, N\}$ is the function of state transitions.

A Turing machine *accepts* an input word w if and only if it halts and outputs 1. On the other hand, a Turing machine *rejects* a word w if and only if it halts and outputs 0. The language $L(M)$ of a Turing machine M is the set of input words that the Turing machine accepts. We say that a Turing machine M *decides* a problem if for the language L of the problem holds: $L = L(M)$ and the Turing machine halts on all inputs.

With this formal definition of an algorithm, we are able to specify exactly what time and space as a resource mean. The time a Turing machine needs to solve a decision problem is the number of steps. The space a Turing machine needs to solve a decision problem is the number of tape cells that are used in the whole computation process.

In order to classify decision problems into complexity classes, we have to define the worst-case runtime and the worst-case space usage of a problem. We always measure the time and space in relation to the input size of the problem, where the input size of the word w is its length $|w|$. Accordingly, we define the worst-case runtime $t_M(n)$ of a Turing machine M to be the maximum number of steps over all possible inputs of length n . The worst-case runtime t_M of a Turing machine M is bounded by a function b if $\forall n \in \mathbb{N} : t_M(n) = O(b(n))$, where n is the length of the input word. Analogously, the worst-case space usage $s_M(n)$ of a Turing machine M is the

maximum number of tape cells used over all possible inputs of length n . The worst-case space usage s_M of a Turing machine M is bounded by a function b if $\forall n \in \mathbb{N} : s_M(n) = O(b(n))$, where n is the length of the input word.

On the basis of worst-case runtime and space usage, we can define several complexity classes of problems with regard to time and space complexity. In this thesis, we focus on the complexity classes:

- ▶ *Polynomial Time* (in short PTIME or just P),
- ▶ *Non-deterministic Polynomial Time* (in short NP),
- ▶ *Polynomial Time Hierarchy* (in short PH), and
- ▶ *Polynomial Space* (in short PSPACE).

2.4.1 The complexity class P

As the name already suggests, the class polynomial time is composed of all decision problems that are decidable by a Turing machine with a polynomial worst-case runtime.

Definition 2.3 (Polynomial Time). *The complexity class Polynomial Time (in short PTIME or P) consists of all languages $L \subseteq \{0, 1\}^*$ such that there is a Turing machine M with $L = L(M)$ and $\forall w \in \{0, 1\}^* : t_M(|w|) = |w|^{O(1)}$.*

The complexity class P is considered to be the class of all efficiently solvable problems. In other words, if there is a Turing machine that is able to decide a problem in polynomial time, we think of the problem as efficiently solvable. An example of a problem in P is the graph connectivity problem, which is defined as follows. It is possible to solve this problem via depth-first search, which runs in polynomial time.

GRAPH CONNECTIVITY

Input: Undirected graph $G = (V, E)$.

Output: Is the graph connected, i.e., is there a path between all vertex pairs $u, v \in V$?

2.4.2 The complexity class NP

Above, we have only considered deterministic Turing machines. A deterministic Turing machine is defined by a transition function δ , for which there is at most one possible transition based on the given state and symbol the head points to. In contrast, a non-deterministic Turing machine is a generalization of deterministic Turing machines in the sense that the transition function is relaxed to a transition relation. Consequently, we define the transition relation δ for non-deterministic Turing machines by $\delta \subseteq ((Q \setminus \{\bar{q}\}) \times \Gamma) \times (Q \times \Gamma \times \{R, L, N\})$. Thus, for each state and symbol pair, there are several options for the Turing machine to continue the computation. A valid computation path of a non-deterministic Turing machine is defined by the sequence of configurations starting at the start configuration with start state q_0 and ending in the final configuration with final state \bar{q} .

Because δ is a relation, the possible computation paths can be subsumed into a computation tree. We can define the computation tree by denoting the nodes as the configurations $Q \times \Gamma$ and the subsequent configurations as children defined by δ . The root of the tree is the start configuration with state q_0 . The leafs of the computation tree are all nodes from $\{\bar{q}\} \times \Gamma$. In comparison to deterministic Turing machines, where it is clear when the Turing machine accepts (reaching the final state and outputting 1), we have to redefine acceptance for non-deterministic Turing machines. We say that a non-deterministic Turing machine recognizes a

word w if there is at least one path in the computation tree that results in the final state and outputs 1. Consequently, the language $L(M)$ of a non-deterministic Turing machine M is the set of recognized words.

The runtime $r_M(w)$ of a non-deterministic Turing machine on input w is defined by the number of steps of the shortest accepting path of the computation tree of M on input w . If there is no accepting path, we define the runtime to be $r_M(w) = 0$ for convenience. Then, we can easily define the runtime $t_M(n)$ of the non-deterministic Turing machine M to be the maximum runtime $\max_{w \in \{0,1\}^n} r(w)$ of all input words of length n . We are now ready to define the complexity class NP, which is the non-deterministic counterpart of the complexity class P.

Definition 2.4 (Non-deterministic Polynomial Time). *The complexity class Non-deterministic Polynomial Time (in short NP) consists of all languages $L \subseteq \{0,1\}^*$ such that there is a non-deterministic Turing machine M with $L = L(M)$ and $\forall w \in \{0,1\}^* : t_M(|w|) = |w|^{O(1)}$.*

Besides the definition of non-deterministic Turing machines, there is also the possibility to equivalently define the class NP via a pair of a *certificate* and a *verifier*.

Definition 2.5 (Non-deterministic Polynomial Time (Certificate and Verifier)). *The complexity class Non-deterministic Polynomial Time (in short NP) consists of all languages $L \subseteq \{0,1\}^*$ such that there is a deterministic polynomial time Turing machine V , which we call the verifier, and a $m_1 = |w|^{O(1)}$ such that for all $w \in \{0,1\}^*$:*

$$w \in L \Leftrightarrow \exists y \in \{0,1\}^{m_1} : V(w, y) = 1.$$

Both definitions are equivalent because, given a non-deterministic Turing machine, any computation path through the computation tree can be encoded as a word (the certificate) representing the transition that is taken at every node of the computation tree. Then the deterministic Turing machine is able to simulate the corresponding path by using the certificate. On the other hand, a non-deterministic Turing machine is able to guess the certificate at the beginning of the computation. Of course, a non-deterministic Turing machine is able to simulate a deterministic computation (i.e., the verifier).

While the definition of this class might seem to be complicated and unnatural, many important real-world optimization problems are part of the class NP. To name a few examples: various scheduling problems, the traveling salesman problem, or knapsack. Since a function is always expressible as a relation, we have that $P \subseteq NP$.

While we consider polynomial time deterministic Turing machines to be efficient, this is not obviously the case for non-deterministic Turing machines. The problem is that non-determinism is not available to us to directly employ in a computation on a classical computer. Naively, we need to simulate the whole computation tree of a non-deterministic Turing machine by a deterministic Turing machine. Note that the description of a Turing machine is of constantly large size, with the result that the number of child nodes in the tree is always bounded by a constant Δ . In summary, at every node, we may have Δ subsequent configurations, while the computation tree is of at most polynomial depth. Consequently, $\Delta^{|w|^{O(1)}}$ many possible paths have to be simulated, resulting in an exponential number of steps. But is it possible that $P = NP$ and thus problems in NP are efficiently solvable? This is one of the unsolved Millennium Prize Problems. However, it is widely believed that $P \neq NP$. Is it somehow possible to separate problems from P and NP, if $P \neq NP$? The answer to this problem is a clear “Yes”. For this, we introduce the concept of a reduction.

Reductions. The concept of a reduction is at the heart of computational complexity theory. A reduction is an algorithm that maps the instances of one problem L_1 into instances of another

problem L_2 . If this reduction algorithm runs efficiently, i.e., in polynomial time, we can efficiently transform the instances of L_1 to instances of L_2 . If there were an efficient algorithm for L_2 , we would also be able to solve these transformed instances of L_1 . Consequently, we are able to solve problem L_1 efficiently by concatenating the efficient reduction from L_1 to L_2 with the efficient algorithm for L_2 . In other words, L_2 is at least as hard to solve as problem L_1 . Since the term efficient is used interchangeably, we define a polynomial time reduction as follows.

Definition 2.6 (Polynomial Time Reduction). *A polynomial time reduction from a language L_1 to a language L_2 is a function $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ such that*

- ▶ g is computable by a deterministic polynomial time Turing machine
- ▶ $w \in L_1$ if and only if $g(w) \in L_2$.

If there is a polynomial reduction from language L_1 to language L_2 , we write $L_1 \leq_p L_2$.

By using this technique, we are able to identify the hardest problems in NP. More precisely, if we have a deterministic polynomial time reduction from L_1 to L_2 and there is a deterministic polynomial time algorithm for L_2 , then L_1 is also solvable in deterministic polynomial time. Vice versa, if there is a language L^* , which is polynomially reducible from all languages L from the class NP, then it is at least as hard to solve as any other problem in NP. We conclude the following definition.

Definition 2.7 (NP-hardness). *A language L^* is NP-hard if $\forall L \in \text{NP} : L \leq_p L^*$*

We also define an NP-complete problem as follows.

Definition 2.8 (NP-completeness). *A language L^* is NP-complete, if it is in NP and NP-hard.*

Is there any NP-complete problem? Indeed, such a problem exists as Cook and Levin [Coo71, Tra84] have proven independently. The first problem shown to be NP-hard was SATISFIABILITY (or in short SAT), which is defined as follows.

SATISFIABILITY

Instances: A Boolean formula $\varphi(X)$ of variables X in CNF.

Output: Is there a satisfying assignment $\alpha : X \rightarrow \{0, 1\}$ for $\varphi(X)$?

In the following year, Karp [Kar72] presented 21 further NP-complete problems. At the end of the decade, Garey and Johnson published a compendium [GJ79] with hundreds of problems that were shown to be NP-complete. Among these are many important problems, such as the traveling salesman problem, various scheduling problems, knapsack, and many other routing problems. Solving all of those problems efficiently has a high impact on everyday life, e.g., for planning and maintaining infrastructure, optimizing and automating business processes, and conducting research in various areas.

The complement of NP. At last, we define the complexity class coNP. This class consists of all complements of languages that are contained in NP.

Definition 2.9 (Complement of NP). *The complexity class coNP is defined by*

$$\text{coNP} = \{L \mid (\{0, 1\}^* \setminus L) \in \text{NP}\}$$

Equivalently, coNP can be defined by an analogue of the certificate and verifier definition.

Definition 2.10 (Complement of NP (Certificate and Verifier)). *The complexity class coNP consists of all languages $L \subseteq \{0, 1\}^*$ such that there is a deterministic polynomial time Turing machine V and a $m_1 = |w|^{O(1)}$ such that for all $w \in \{0, 1\}^*$:*

$$w \in L \Leftrightarrow \forall y \in \{0, 1\}^{m_1} : V(w, y) = 1.$$

The canonical coNP-complete problem is TAUTOLOGY, which is defined as follows.

TAUTOLOGY

Instances: A Boolean formula $\varphi(X)$ of variables X in DNF.

Output: Are all assignments $\alpha : X \rightarrow \{0, 1\}$ for $\varphi(X)$ satisfying?

The problem TAUTOLOGY can be interpreted as the complement of the problem SATISFIABILITY because it asks whether a Boolean formula in DNF is satisfied by all possible assignments. Note that a negation of a formula in CNF is equivalent to a formula in DNF and vice versa. Furthermore, the negation of an \exists -quantifier results in a \forall -quantifier and vice versa. Thus, we can negate the input formula $\exists X \varphi(X)$, where $\varphi(X)$ is in CNF, to obtain the complement answer for the problem SATISFIABILITY. The resulting formula is $\neg \exists X \varphi(X)$ and thus $\forall X \neg \varphi(X)$, where $\neg \varphi(X)$ can be efficiently transformed into a DNF. This is equivalent to the question in TAUTOLOGY.

2.4.3 The Polynomial Hierarchy

The *polynomial hierarchy* is a generalization of the complexity class NP in terms of *oracle machines*, which was introduced by Stockmeyer [Sto76]. An oracle machine is a Turing machine that additionally has the power to ask an oracle. An oracle can be imagined as a black box that returns an answer in one computation step of the oracle machine. Accordingly, an oracle for problem A answers whether a word w is in A in a single computation step. Because the YES-answer for a problem A is equivalent to a NO-answer for its complement coA , inverting the answers for an oracle for A results in an oracle for coA . Consequently, an oracle machine with an oracle A has the same power as the same oracle machine with an oracle for coA . We use oracle machines in the context of P and NP to define the *polynomial hierarchy*. Here, the underlying idea is to answer the question of whether and how additional oracles help a (non-)deterministic Turing machine to solve more complex problems.

Definition 2.11 (Oracle Machine). *An oracle machine with oracle O is a Turing machine M^O that has an additional tape, which we call the oracle tape. Additionally, M^O has three additional states $q_?$, q_Y , q_N . M^O can ask the oracle O by writing the input to the oracle tape and going into state $q_?$. The oracle answers in one computation step by deleting the oracle tape and going into state q_Y if the input was a YES-instance and in q_N if the input was a NO-instance.*

With this definition, we can equip a Turing machine M with oracles of problems such as SATISFIABILITY, which we denote by M^{SAT} . Since we have defined P to be the class of all problems for which a polynomial time deterministic Turing machine exists, we can also equip these Turing machines with an oracle. The resulting class with an oracle for SAT is P^{SAT} . Because SAT is an NP-complete problem, P^{SAT} is the same class as P with a polynomial time non-deterministic Turing machine oracle. Conclusively, $P^{\text{NP}} = P^{\text{SAT}}$.

Now, we have everything to describe the Polynomial Hierarchy.

Definition 2.12 (Polynomial Hierarchy). *The Polynomial Hierarchy consists of the inductively defined families*

$$(\Delta_i^P)_{i \in \mathbb{N}_0}, (\Sigma_i^P)_{i \in \mathbb{N}_0}, (\Pi_i^P)_{i \in \mathbb{N}_0}.$$

The class P is the basis or the 0th level of the polynomial hierarchy. We correspondingly define $P = \Delta_0^P = \Sigma_0^P = \Pi_0^P$. For the following levels $i \in \mathbb{N}_{\geq 0}$, we use the classes P , NP , and $coNP$ to define

$$\begin{aligned} \Delta_{i+1}^P &= P^{\Sigma_i^P} \\ \Sigma_{i+1}^P &= NP^{\Sigma_i^P} \\ \Pi_{i+1}^P &= coNP^{\Sigma_i^P}. \end{aligned}$$

The polynomial hierarchy is defined by the union of all these classes, i.e. $PH = \bigcup_{i \in \mathbb{N}_0} \Delta_i^P \cup \Sigma_i^P \cup \Pi_i^P$

Consequently, the first level of the hierarchy consists of the classes $\Delta_1^P = P^P = P$, $\Sigma_1^P = NP^P = NP$, and $\Pi_1^P = coNP^P = coNP$. Note that an oracle for P does not add any power to a polynomial Turing machine because an addition of polynomial runtime still results in a polynomial runtime. Therefore, P , NP , and $coNP$ have the same power as if they are equipped with an oracle for P . Until this point, no new class has been defined. However, from the second level on, new classes are defined: $\Delta_2^P = P^{NP}$, $\Sigma_2^P = NP^{NP}$, and $\Pi_2^P = coNP^{NP}$. Because an oracle machine is at least as powerful as its oracle, the classes on one level trivially contain the classes of the lower levels. Thus, it holds that $PH = \bigcup_{i \in \mathbb{N}_0} \Delta_i^P = \bigcup_{i \in \mathbb{N}_0} \Sigma_i^P = \bigcup_{i \in \mathbb{N}_0} \Pi_i^P$. A visualization of the polynomial hierarchy and the inclusions of the classes can be found in Figure 2.1.

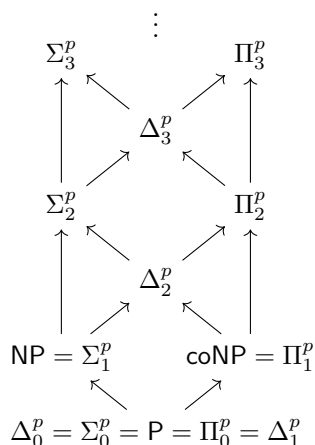


Figure 2.1: Visualization of the inclusion relations of the Polynomial Hierarchy

An alternative definition of the classes of the polynomial hierarchy in reference to the definition of NP over a pair of a certificate and a verifier can also be conducted.

Definition 2.13 (The Complexity Class Σ_p^k). We define the class Σ_p^k to contain all languages $L \subseteq \{0, 1\}^*$ such that there is a deterministic polynomial time Turing machine V and $m_1, \dots, m_k = |w|^{O(1)}$ such that for all $w \in \{0, 1\}^*$:

$$w \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} \dots Q y_k \in \{0, 1\}^{m_k} V(w, y_1, y_2, \dots, y_k) = 1,$$

where $Q = \exists$, if k is odd, and $Q = \forall$, else.

Definition 2.14 (The Complexity Class Π_k^p). We define the class Π_k^p to contain all languages $L \subseteq \{0, 1\}^*$ such that there is a deterministic polynomial time Turing machine V and $m_1, \dots, m_k = |w|^{O(1)}$ such that for all $w \in \{0, 1\}^*$, it holds $x \in L$ iff

$$w \in L \Leftrightarrow \forall y_1 \in \{0, 1\}^{m_1} \exists y_2 \in \{0, 1\}^{m_2} \dots Q_k y_k \in \{0, 1\}^{m_k} V(w, y_1, y_2, \dots, y_k) = 1,$$

where $Q = \forall$, if k is odd, and $Q = \exists$, else.

For the classes of the polynomial hierarchy, we are also able to establish the concept of hardness and completeness. We can again identify the hardest problems of each class by polynomial time reductions.

Definition 2.15 (Hardness and Completeness for Classes in the Polynomial Hierarchy). A language L^* is hard for class \mathcal{C} if $\forall L \in \mathcal{C} : L \leq_p L^*$. A language L^* is \mathcal{C} -complete if it is in \mathcal{C} and \mathcal{C} -hard.

Analogous to that SAT is the canonical NP-complete problem, the problem $\exists_1 \forall_2 \dots Q_k \text{SAT}$ is the canonical Σ_k^p -complete problem, where $Q = \exists$, if k is odd, and $Q = \forall$, else. For Π_k^p , the canonical problem is respectively $\forall_1 \exists_2 \dots Q_k \text{SAT}$, where $Q = \forall$, if k is even, and $Q = \exists$, else. Formally, the problem $Q_1 Q_2 \dots Q_k \text{SAT}$ is defined as follows.

k-QUANTIFIED SATISFIABILITY

Instances: A quantified Boolean formula $Q_1 X_1 Q_2 X_2 \dots Q_k X_k \varphi(X_1, \dots, X_k)$ over variable sets X_1, \dots, X_k with quantifiers $Q_1 Q_2 \dots Q_k \in \{\exists, \forall\}$.

Output: Is $Q_1 X_1 Q_2 X_2 \dots Q_k X_k \varphi(X_1, \dots, X_k)$ satisfiable?

We remark that we have defined k -QUANTIFIED SATISFIABILITY without demanding that the formula φ is in CNF. If the k -th quantifier is an \exists -quantifier, then the formula φ can be CNF but not in DNF. Vice versa, if the k -th quantifier is a \forall -quantifier, then the formula φ can be DNF but not in CNF. This is since CNF-SATISFIABILITY is NP-hard, while DNF-SATISFIABILITY is in P. Thus, the last quantifier collapses if the k -th quantifier is an \exists -quantifier and the formula is in DNF. Since a negated CNF results in a DNF, the inverse holds for coNP (DNF-TAUTOLOGY is hard and CNF-TAUTOLOGY is in P). For the lower levels of the polynomial hierarchy, many complete problems are known and are presented in the compendium of Schaefer and Umans [SU08].

2.4.4 The complexity class PSPACE

The last class that we present in this thesis is the class of polynomial space computation. Analogous to the class polynomial time P, the class PSPACE contains all problems that are solvable by a deterministic Turing machine in polynomial space.

Definition 2.16 (Polynomial Space). The complexity class *Polynomial Space* (in short PSPACE) consists of all languages $L \subseteq \{0, 1\}^*$ such that there is a deterministic Turing machine M with $L = L(M)$ and $\forall w \in \{0, 1\}^* : s_M(|w|) = |w|^{O(1)}$.

For this class, we can define the notion of hardness in completeness by reusing Definition 2.15. That is, a problem is PSPACE-hard if there is a polynomial time reduction from all problems in PSPACE. The canonical PSPACE-complete problem is TRUE QUANTIFIED BOOLEAN FORMULA (TQBF), which is a generalization of SAT and k -QUANTIFIED SAT, and is defined as follows.

TRUE QUANTIFIED BOOLEAN FORMULA

Instances: A quantified Boolean formula $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$ over variables x_1, \dots, x_n with quantifiers $Q_1 Q_2 \dots Q_n \in \{\exists, \forall\}$, and $\varphi(x_1, \dots, x_n)$ in CNF.

Output: Is $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \varphi(x_1, \dots, x_n)$ satisfiable?

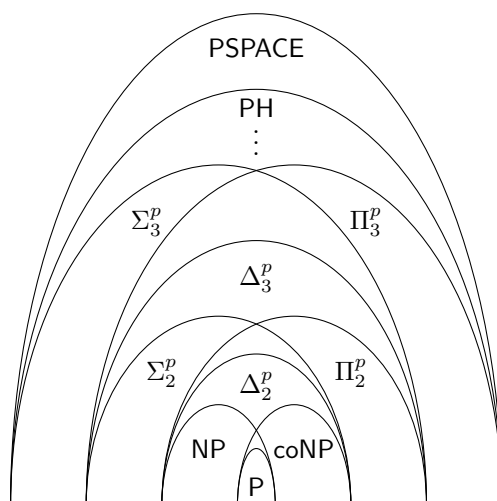


Figure 2.2: The relations of the presented complexity classes.

We remark that the difference to k -QUANTIFIED SAT is the number of quantifiers. While in k -QUANTIFIED SAT the number of quantifiers is a constant k , the number of quantifiers in TQBF is n , which is part of the input. Furthermore, each quantifier in TQBF is associated with exactly one variable in contrast to k -QUANTIFIED SAT where each quantifier is associated with a set of variables. This is not a restriction because a set of m quantified variables can be divided into m quantifiers with 1 variable.

Various examples of PSPACE-complete combinatorial games can be found in [FG87]. By the PSPACE-complete problem TQBF, we can see that all the classes presented above are part of PSPACE. The subset relation between the classes is visualized in Figure 2.2. While we do not know whether any of the subset relations are proper, it is widely believed that this is the case for all classes.

2.5 Robust Optimization

Robust optimization is a subarea of mathematical optimization. In this thesis, we are concerned with the complexity of robust optimization of combinatorial problems. Accordingly, we give a short introduction to combinatorial optimization and, based on this, to robust combinatorial optimization.

Combinatorial Optimization. Combinatorial optimization deals with finding an optimal solution from a finite and discrete set. Many combinatorial problems can be expressed in the following way. First, there is a ground set $\mathcal{U} = \{u_1, \dots, u_n\}$ of *combinatorial elements* that are the atoms of the problem. A solution is expressed by a vector $x \in \{0, 1\}^n$, where $x_i = 1$ if and only if u_i is part of the solution. Additionally, there is an *objective function* $f : \{0, 1\}^n \rightarrow \mathbb{Z}$. The goal is now to find a vector $x \in \{0, 1\}^n$ that *minimizes* the function f , i.e.

$$\min_{x \in \{0, 1\}^n} f(x).$$

For many practical problems, the objective function is a *linear* function. That is, it can be

described by a cost vector $c \in \mathbb{Z}^n$ with $f(x) = c^T x$. Furthermore, there is a set of *constraints*, which can be described by *linear inequalities*. Overall, this can be modeled by an *integer program*. An integer program consists of a matrix $A \in \mathbb{Z}^{m \times n}$, a cost vector $c \in \mathbb{Z}^n$, and a constraint bound vector $b \in \mathbb{Z}^m$. The goal is then to find the corresponding solution vector that fulfills all linear constraints as described by matrix A and the constraint bound vector b while minimizing the objective function f . Such an integer program is then fully described by

$$\min_{x \in \{0,1\}^n} \{c^T x \mid Ax \leq b\}$$

The decision version INTEGER PROGRAMMING is derived from the formulation above and is defined as follows.

INTEGER PROGRAMMING

Instances: A matrix $A \in \mathbb{Z}^{m \times n}$, a cost vector $c \in \mathbb{Z}^n$, a constraint bound vector $b \in \mathbb{Z}^m$, and a number $k \in \mathbb{N}$

Output: Is there $x \in \{0,1\}^n$ such that $c^T x \leq k$ subject to $Ax \leq b$?

This problem is an important NP-complete problem because problems in NP are intuitively describable as such integer programs. Indeed, mathematical optimization deals in a large part with understanding such programs, the geometric space of the polytope defined by A and b , and consequently the design of solvers. These solvers are deployed in various areas to solve the corresponding concrete problems by modeling them as an integer program. In order to analyze the complexity of such problems, we define a general decision version of the problems from above. Referring to the formulation by a linear objective function and linear inequalities, we name such problems *linear optimization problem* and define them as follows.

Definition 2.17 (Linear Optimization Problem, from [GW23]). *A linear optimization problem (or in short LOP problem) Π is a tuple $(\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$, such that*

- ▶ $\mathcal{I} \subseteq \{0,1\}^*$ is a language. We call \mathcal{I} the set of instances of Π .
- ▶ To each instance $I \in \mathcal{I}$, there is some
 - ▶ set $\mathcal{U}(I)$ which we call the universe associated to the instance I .
 - ▶ set $\mathcal{F}(I) \subseteq 2^{\mathcal{U}(I)}$ that we call the feasible solution set associated with the instance I .
 - ▶ function $d^{(I)} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ mapping each universe element e to its costs $d^{(I)}(e)$.
 - ▶ threshold $t^{(I)} \in \mathbb{Z}$.

For $I \in \mathcal{I}$, we define the solution set $\mathcal{S}(I) := \{S \in \mathcal{F}(I) : d^{(I)}(S) \leq t^{(I)}\}$ as the set of feasible solutions below the cost threshold. The instance I is a YES-instance, if and only if $\mathcal{S}(I) \neq \emptyset$.

We can define the corresponding optimization version of an LOP problem by $\min_{S \in \mathcal{F}(I)} d(S)$. As an example, we can formulate the problem INTEGER PROGRAMMING as a linear optimization problem, by defining the instances to be an encoding of matrix A , vectors b and c , and number k over $\{0,1\}$. The universe $\mathcal{U}(I)$ are the indices $i \in \{1, \dots, n\}$. We further define the feasible solutions by $\mathcal{F}(I) = \{\{i : x_i = 1\} \mid Ax \leq b\}$, the cost function $d^{(I)}(i) = c_i$, and threshold $t^{(I)} = k$.

Robust Optimization. In robust optimization, we want to solve a combinatorial problem as described above with the addition of possible uncertainty. In practical settings, uncertainty in the instance is a usual phenomenon. Even though we may encounter a faulty sensor in our production line or a traffic jam on our usual route, we still want to find good solutions so that we are able to produce further goods or reach our target destination in time.

A decision maker models a robust optimization problem by using a usual combinatorial problem, which we also call *nominal problem*. The uncertainty can now be modeled by an *uncertainty set* C^1 [BS03, BS04a, KZ16]. This uncertainty set contains *scenarios* that occur based on how the uncertainty materializes. If we take a nominal linear optimization problem $\min_{S \in \mathcal{F}(I)} d(S)$ as defined above, we model the robust problem by

$$\min_{S \in \mathcal{F}(I)} \max_{c \in C} c(S),$$

and calculate a solution S , which is immune to all possible scenarios in the uncertainty set C . Therefore, the choice of uncertainty plays an important role in the model. One might also imagine this problem as a game between two players: a decision maker and an adversary. While the decision maker wants to minimize its costs by choosing a corresponding solution $S \in \mathcal{F}(I)$, the adversary wants to do the opposite by choosing an appropriate uncertainty scenario $c \in C$.

Uncertainty Sets. In this thesis, we consider three different types of uncertainty: interval uncertainty, budgeted uncertainty, and elemental uncertainty. The first two types are defined over the costs of the ground set elements. Therefore, they are easily integrable into linear optimization problems. The third type originates from graph theory and models uncertainty over the existence of an element in the instance.

Interval uncertainty. We start with the intuitive concept of interval uncertainty [IS95, BS03]. Here, we assume the costs to be in an interval around the estimated costs. That is, if we expect the costs to be c_i for element i , we can define a lower bound \underline{c}_i and an upper bound \bar{c}_i to be certain that the costs lie within the interval. Formally, the interval uncertainty set can be defined by

$$C_I = \{c \in \mathbb{R}^n \mid \forall i \in \{1, \dots, n\} : c_i \in [\underline{c}_i, \bar{c}_i]\}.$$

This form is widely used in regret optimization [IS95, AL04] but also in many other kinds of robust optimization [KZ16].

Budgeted Uncertainty. Budgeted uncertainty was introduced by Bertsimas and Sim [BS04b]. This concept is popular in the area of robust optimization, especially because if the nominal problem is efficiently solvable, so is the robust problem efficiently solvable by decomposing it into a linear number of instances of the nominal problem [BS03]. In the literature, this form of uncertainty is also known under the name of Γ -*uncertainty* because of the parameter Γ , which specifies the budget of the adversary. Budgeted uncertainty comes in two flavors. The first one is *discrete budgeted uncertainty*, which we use in this thesis, and is defined as follows:

$$C_\Gamma = \{c \in \mathbb{Z}^n \mid \forall i \in \{1, \dots, n\} : c_i = \underline{c}_i + \delta_i(\bar{c}_i - \underline{c}_i), \delta_i \in \{0, 1\}, \sum_{i=1}^n \delta_i \leq \Gamma\}.$$

In other words, the adversary has a budget that he is able to distribute on the n elements by choosing at most Γ elements that receive increased costs of \bar{c}_i , and the rest staying at low costs of \underline{c}_i . The second flavor is *continuous budgeted uncertainty* in which the adversary can distribute the budget continuously onto the n elements, i.e., instead of $\delta_i \in \{0, 1\}$, we have $\delta_i \in [0, 1]$. In this thesis, we do not cover any results on this type of uncertainty.

Elemental uncertainty. Another form of uncertainty is elemental uncertainty. While budgeted uncertainty and interval uncertainty, as defined above, apply to the costs of the elements in the ground set, in elemental uncertainty, it is uncertain whether an element is part of the instance or not. Consequently, it is more difficult to describe this concept in general in the context of integer linear programming. However, in graph theory, the concept of elemental uncertainty is

¹Usually in literature, this set is denoted by U_Γ , however, we have already used the letter U in this thesis.

intuitive to define. For example, for a graph G , we can define the uncertainty set of budgeted vertex uncertainty with budget Γ as all instances in the set

$$\{G[U] \mid U = V(G) \setminus D, D \subseteq V(G), |D| \leq \Gamma\},$$

where $G[U]$ is the induced subgraph of G on vertex set U . Accordingly, one may ask the question: What happens if a vertex is removed from a flow network? In the area of network interdiction or most vital vertex problems, these questions are asked constantly [Yan78, Rut93, Woo93].

Multi-stage Robustness. An extension of robust optimization is multi-stage robust optimization. Because robust optimization tends to produce too conservative solutions, multi-stage robust optimization tries to mitigate the problem by introducing another stage of delayed decisions or a stage of recovery. Accordingly, the decision maker is able to divide the necessary decisions into two sets: the *here-and-now* or *first-stage* decisions, which the decision maker has to make before any form of uncertainty is realized, and *wait-and-see* or *second-stage* decisions, which the decision maker is able to postpone until the uncertainty is realized. Thus, the decision maker is able to react to the uncertainty in a reasonable way. This approach can, of course, be generalized to more than two stages of decision-making. Two important fields of multi-stage robust optimization are *two-stage adjustable robustness* and *recoverable robustness*.

In two-stage adjustable robustness [BGGN04], the decision maker first selects a subset S_1 of combinatorial elements to be part of the solution. Then, the uncertainty is realized and subsequently the decision maker is able to complete its solution by choosing a complementary set S_2 to the set S_1 such that $S_1 \cup S_2$ is a feasible solution. We can thus formulate the problem as follows, based on an LOP problem: Given a first stage cost function $c_1 : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and second stage cost functions $\underline{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and $\bar{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ as well as an uncertainty parameter $\Gamma \in \mathbb{Z}$, we want to solve

$$\min_{S_1 \subseteq \mathcal{U}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \subseteq \mathcal{U}(I) \setminus S_1 \\ S_1 \cup S_2 \in \mathcal{F}(I)}} c_1(S_1) + c_2(S_2).$$

On the other hand, in recoverable robust optimization [LLMS09], the decision maker has to compute a full solution S_1 to the problem. Then again, the uncertainty is realized. Now, the decision maker is able to recover from the solution S_1 by choosing a solution S_2 , which is not too far away from solution S_1 according to some predefined distance measure. We can thus formulate the problem as follows by using again the definition of an LOP problem: Let $\text{dist}(\cdot, \cdot)$ be a distance measure between two solutions. Given three cost functions $c_1 : \mathcal{U}(I) \rightarrow \mathbb{Z}$, $\underline{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and $\bar{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ as well as an uncertainty parameter $\Gamma \in \mathbb{N}_0$ and a recoverability parameter $\kappa \in \mathbb{N}_0$, we want to compute

$$\min_{S_1 \in \mathcal{F}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \in \mathcal{F}(I) \\ \text{dist}(S_1, S_2) \leq \kappa}} c_1(S_1) + c_2(S_2).$$

Note that for both two-stage adjustable robustness and recoverable robustness, the same uncertainty sets can be used. Specifically in this thesis, we use discrete budgeted uncertainty for both robustness concepts. The main difference between these two concepts is how the solution can be chosen. In two-stage adjustable robustness, the solution is partly chosen in the first stage and is then completed in the second stage after the uncertainty has been realized. In contrast, in recoverable robustness, we want to compute a complete solution for the first stage, which is then adaptable to a different complete solution after the uncertainty has been realized. The two complete solutions, however, need to be not too far away from each other according to a distance measure.

2.6 Online Optimization

The area of online algorithms or online optimization deals with optimization problems, where the information is not known beforehand. In this setting, the instance is revealed piecewise to the decision maker, and the decision maker has to decide irrevocably on an action.

An intuitive first example is the Treasure Hunt Problem [KKKS15]. Here, the decision maker lands on an island on which a treasure is hidden. Usually, the place where a treasure is hidden is marked with a cross; however, the decision maker has no map to find the exact location. Thus, the decision maker has to travel along the paths to get to the marked location. But what is the fastest way to do so? The problem is that the decision maker has no idea how the path network on the island looks like. Thus, possible trails are revealed only if the decision maker stands in front of the crossing, where the respective trail begins.

In order to analyze the worst-case performance of the decision maker, we can utilize an adversary who constructs the network dynamically in such a way that the travel time of the decision maker is maximized. The adversary knows exactly how the decision maker behaves and is thus able to perfectly react to the actions of the decision maker by introducing crossings or paths, i.e., the adversary is in full control of the network and constructs it as he likes. If the decision maker travels along a path, the adversary prolongs the path and hides the treasure on a different path accordingly. Then, after reaching the end of the path, the decision maker has to travel back to the start, and then he will find the treasure on the other path, which is also way shorter. Conclusively, the decision maker lost much time in comparison to the optimal solution. Indeed, no matter which (deterministic) strategy the decision maker employs, it is not constant competitive [KKKS15].

We can summarize that the adversary constructs the instance piece by piece, while the decision maker has to make an irrevocable decision on each newly constructed piece of the instance. A visualization of the corresponding general online model can be found in Section 2.6. This setting is highly asymmetric in favor of the adversary. Thus, for most decision problems, the adversary is able to abuse the imbalance of power to prevent the decision maker from finding a solution that is close to the optimal one.

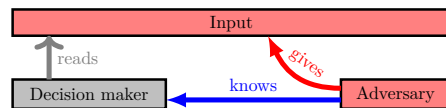


Figure 2.3: The general model of worst-case analysis of online computation visualizing the relation between the online algorithm and the adversary.

To overcome this imbalance, different extensions were suggested which fall under the umbrella term *semi-online optimization*. One possibility is to equip the decision maker with additional information in the form of a map. The Canadian Traveler Problem [PY91] is a corresponding example of the same flavor as the Treasure Hunt problem above. The decision maker is on vacation in the wintry Canadian wilderness and wants to travel on the streets from a point A to a point B . Although the decision maker has a map, the streets might be blocked by a fallen tree. What is the best way to travel from A to B given the uncertainty that a street may be blocked? Here, the map can be used to gain information on which streets may be blocked (only those that are close to a forest) and how to get from A to B .

A visualization of the model for online problems with a map can be found in Section 2.6. The input is given to the decision maker and the adversary, and is itself not influenceable by the adversary, in contrast to the general model. The adversary is only able to control the revelation

of the existing instance, i.e., he is able to control whether a tree lies on the visited road and which exact road is traveled by the decision maker if isomorphic roads exist in correspondence to the map. The relation between the decision maker and the adversary corresponds to players in an asymmetric two-player game, in which the algorithm wants to maximize its performance and the adversary's goal is to minimize it. We can interpret the input as the game board of the game. The question is whether the algorithm has a *winning strategy*, i.e., whether there is a solution of costs that are smaller than some threshold t for all possible strategies of the adversary. As the first complexity result, the Canadian Traveler Problem was shown to be PSPACE-complete [PY91].

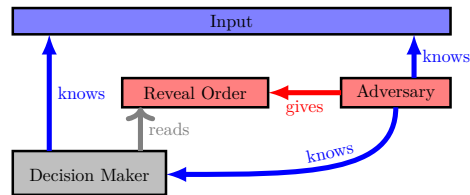


Figure 2.4: The model of worst-case analysis of online computation with a map as additional information.

Part I

Subset Search Problem Framework

Chapter 3

Subset Search Problem Framework

3.1 Introduction

In recent years, there has been enormous interest in the areas of Bilevel Optimization [DZ20, KLLS21], Robust Optimization [BGN09, GH24], Network Interdiction [SPG13], Stackelberg Games [LS17], Attacker-Defender games [HZ23], and many other bilevel problems. These research areas are vital in helping us to understand, which parts of a network are most vulnerable, prevent terrorist attacks, understand economic processes, and to understand and improve the robustness properties of many systems.

The common property of all these areas of research is that they study min-max optimization problems. Equivalently, these problems can be characterized as an abstract two-turn game between two players. The first player (from now on called Alice) starts the game and takes an action with the goal of minimizing some objective. Afterwards, the second player (from now on called Bob) responds to Alice. Typically (but not always), Alice's and Bob's goals are opposite of each other.

Let us take as a prototypical example the maximum clique interdiction problem [PBP14, PPR16, FLMS19, Paj20]. In this problem, we are given a graph $G = (V, E)$ and some budget $k \in \mathbb{N}$. The goal of Bob is to find a clique of largest possible size in the graph. However, before Bob's turn, Alice can delete up to k vertices from the graph in order to impair Bob's objective. Hence the game is a min-max optimization problem described by

$$\min_{\substack{W \subseteq V \\ |W| \leq k}} \max\{|C| : C \text{ is a clique in the graph } G \text{ with } W \cap C = \emptyset\}.$$

We remark that this problem follows a natural pattern, in which researchers often come up with new problems: First, some *nominal problem* is taken (in this case, the maximum clique problem), and afterwards it is *modified* into a more complicated min-max problem by adding an additional component (in this case, the possibility of Alice to interdict). The goal of this part is to consider this very general pattern and shed some light on the understanding of its computational complexity. Roughly speaking, we show that whenever the nominal problem is already an NP-complete problem, then under some mild assumptions the complexity of the min-max variant will be significantly larger than the complexity of the nominal variant. As a first example of this behavior, consider the following: It was proven by Rutenburg [Rut93] that the maximum clique interdiction problem is complete for the complexity class Σ_2^p .

The natural complexity class for bilevel problems. In the year 1976, Stockmeyer [Sto76] introduced the *polynomial hierarchy*, containing the complexity classes Σ_k^p for all $k \in \mathbb{N}$. While

the complexity class $\Sigma_1^P = \text{NP}$ is very well known even to outsiders of theoretical computer science, the complexity class Σ_2^P seems to be less known to non-specialists. However, for min-max optimization problems, it turns out that Σ_2^P is the natural class to describe them. Roughly speaking, the class Σ_2^P contains all the problems of the form: Does there EXIST some object x , such that FOR ALL objects y some easy-to-check property $P(x, y)$ holds? For example, the (decision version of the) maximum clique interdiction problem belongs to Σ_2^P . This is because it has objective value t if and only if there EXISTS some set $W \subseteq V$ of size at most k , such that ALL cliques of size $t + 1$ or more intersect W . Hence the class Σ_2^P naturally corresponds to decision variants of min-max optimization problems. Analogously, there exists the class Σ_3^P , corresponding to min-max-min optimization problems, and so on.

Researchers are interested in the question which problems are complete for the class Σ_2^P , since this has several interesting consequences. Similarly to the widely believed conjecture $\text{P} \neq \text{NP}$, it is also widely believed that $\text{NP} \neq \Sigma_2^P$. If this conjecture is true, it means that a Σ_2^P -complete or Σ_3^P -complete problem can not be described by a mixed integer program of polynomial size (because this would imply a polynomial-time reduction to MIP, which is NP-complete). This means that current existing integer program solvers, which have in some cases been very successful in tackling NP-complete problems, are not very successful at solving the harder case of Σ_2^P -problems. This theoretical difference can also be observed in practice. For example, Woeginger [Woe21] points out as an example a bilinear program, which comes from social choice theory. Solving this program was posed to the Operations Research community as a challenge by Kurz and Napel [KN16]. Despite the problem having very small input, the challenge is still widely open and seems out of reach of current methods. Woeginger argues that the mathematical optimization community in the past has proven greatly successful at tackling NP-complete problems through the means of sophisticated IP-solvers. Today, the community is at a threshold, where new and more powerful tools and techniques need to be developed in order to tackle Σ_2^P -hard problems. Similar observations were the motivation for a recent DAGSTUHL seminar on the topic of optimization at the second level. In its report [BBdHH22], it is noted that “methodologies that have been developed for NP-complete problems over the last 50 years do not directly apply to robust and/or bilevel optimization problems”, and furthermore that “we will need to develop new techniques, new tricks, new insights, new algorithms, and new theorems to get a grip on this area”. The goal of this part is to address this problem by providing a novel and powerful tool, which can be used to shed light on the complexity landscape of bilevel (and more generally, multi-level) optimization.

3.1.1 Literature overview

Despite a tremendous interest in bilevel optimization, and despite the complexity classes Σ_k^P being the natural complexity classes for this type of decision problems, we are aware of only a handful of publications on the matter of Σ_k^P -completeness with respect to these areas. Rutenburg [Rut93] shows Σ_2^P -completeness of the maximum clique interdiction problem. Deineko and Woeginger [DW10] show Σ_2^P -completeness of the min-max regret interval knapsack problem. Caprara, Carvalho, Lodi and Woeginger [CCLW14] show Σ_2^P -completeness of an interdiction variant of knapsack that was originally introduced by DeNegre [Den11]. Fröhlich and Ruzika [FR21] show the Σ_2^P -completeness of two versions of a location-interdiction problem. Nabli, Carvalho and Hosteins [NCH22] study the multilevel critical node problem and prove its Σ_3^P -completeness. Coco, Santos and Noronha [CSN22] show Σ_2^P -completeness of the min-max regret maximum benefit set covering problem. Goerigk, Lendl and Wulf [GLW24] show Σ_3^P -completeness of two-stage adjustable variants of TSP, independent set, and vertex cover. Tomasaz, Carvalho, Cordone and Hosteins [TCCH24] show Σ_2^P -completeness of an interdiction-knapsack problem and Σ_3^P -completeness of

an fortification-interdiction-knapsack problem. The compendium by Umans and Schaefer [SU08] contains many Σ_2^p -complete problems, but few of them are related to bilevel optimization. A seminal paper by Jeroslow [Jer85] shows Σ_k^p -completeness for various classes of general multi-level programs.

The research that is most closely connected to this thesis is the Ph.D. thesis of Johannes [Joh11]. She proves Σ_2^p -hardness for a variety of problems including “adversarial problems”, “partial inverse optimization problems” and several other types of classes. She also discusses a transitive transformation for NP-completeness results that she calls “value preserving” [Joh11, Theorem 2.1.1]. She shows that a wide range of adversarial problems and partial preprocessing problems can be proven to be Σ_2^p -hard by relying on their corresponding NP-completeness proofs, providing the NP-completeness transformation was “value preserving.” In our paper, we establish a similar meta-theorem for other classes of problems, including interdiction, min-max regret, and two-stage adjustable optimization.

Recently, we have learned that Johannes and Orlin [2020] have written an unpublished manuscript that extended the research in Johannes’s thesis. They refer to certain NP-completeness transformations as “ratcheting transformations” because they can be used to prove hardness for extensions of these problems to problems higher in the polynomial time hierarchy. There are additional overlaps of their work with our meta-theorem.

To the best of our knowledge, this completes the list of known Σ_k^p -completeness results for bilevel optimization. We find it remarkable that so few results exist in this area, despite being an active research area for well over two decades. In contrast, NP-completeness proofs are known for a huge number of problems. One possible reason for this, according to Woeginger [Woe21], is that “at the current moment, establishing Σ_2^p -completeness is usually tedious and mostly done via lengthy reductions that go all the way back to 2-Quantified Satisfiability”.

3.1.2 Our contribution

We make a large step towards understanding the complexity of bilevel combinatorial optimization, by introducing a general and powerful meta-theorem to prove Σ_k^p -completeness. Our meta-theorem can be applied to such problems which are already NP-complete and have an additional property (explained below). Using the meta-theorem, we can “upgrade” an existing NP-completeness proof to a Σ_k^p -completeness proof with very little work. In other words, this means that for a lot of NP-complete problems, the Σ_2^p -completeness of its min-max variant follows essentially “for free” from its NP-completeness. An analogous statement holds for min-max-min problems and Σ_3^p -completeness. We apply our meta-theorem to the areas of bilevel optimization, in particular network interdiction, min-max regret robust optimization, two-stage adjustable optimization, and recoverable robust optimization. This way, we obtain over 100 natural Σ_2^p - or Σ_3^p -complete problems relevant to these areas. We remark that earlier papers showed Σ_2^p - or Σ_3^p -completeness only for one problem at a time, usually in a tedious fashion. In contrast, our meta-theorem contains essentially all known Σ_2^p - or Σ_3^p -completeness results in the areas of network interdiction, min-max regret, two-stage adjustable, and recoverable robust optimization as a special case (namely, min-max-regret knapsack [DW10], min-max regret maximum benefit set cover [CSN22], interdiction maximum clique [Rut93], interdiction knapsack [CCLW14, TCCH24], two-stage adjustable TSP, vertex cover, independent set [GLW24], as well as recoverable robust TSP, vertex cover, independent set [GLW24]). While each of these earlier works proves Σ_2^p -hardness of only one specific problem, our meta-theorem proves Σ_2^p -completeness simultaneously for a large class of problems, including all problems mentioned in Chapter 9, as well as possible further problems which are added in the future.

We call this class of problems for which the meta-theorem is applicable the class SSP-NP-

complete (SSP-NPc), for reasons which are explained in Section 3.2. We show that at least the following 25 classical problems are contained in the class SSP-NPc:

Satisfiability, 3Satisfiability, Vertex Cover, Dominating Set, Hitting Set, Set Cover, Feedback Vertex Set, Feedback Arc Set, Uncapacitated Facility Location, p-Center, p-Median, Independent Set, Clique, Subset Sum, Knapsack, Partition, Two Machine Scheduling, Directed/Undirected Hamiltonian Path, Directed/Undirected Hamiltonian Cycle, Traveling Salesman Problem, Two Directed Vertex Disjoint Path, k -Directed Vertex Disjoint Path, Steiner Tree.

A formal description of all these problems is provided in Chapter 9. Furthermore, one can add new problems to the class SSP-NPc with very little work by finding a so-called *SSP reduction* starting from any problem which is already contained in the class. Since we could easily show for many classic problems that they are contained in SSP-NPc, we suspect that many more problems can be added in the future. In fact, we observed the pattern that proving some problem to be contained in SSP-NPc is usually significantly easier than formulating a complex and technically challenging Σ_k^p -completeness proof. This offers future researchers a convenient way to prove Σ_k^p -completeness of relevant problems.

Since the class SSP-NPc contains so many well-known problems, our work shows that, in a certain sense, the “standard” or “normal” behavior of a NP-complete problem is to become Σ_2^p -complete when modified to be a min-max problem. While this behavior is very intuitive, we are the first to be able to prove this intuition to be true for a very broad range of problems in robust and bilevel optimization.

3.1.3 Technical Overview

We give a short overview of the techniques and ideas used to obtain our main theorems. For that purpose it becomes necessary to formally describe to what kind of optimization problems the meta-theorem applies. In this thesis, we consider *linear optimization problems* (LOP). Inspired by problems appearing in the literature, we define an LOP to be a problem expressed as a tuple $(\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$. Here, $\mathcal{I} \subseteq \{0, 1\}^*$ is the set of input instances of the problem encoded as words in binary. Associated to each input instance $I \in \mathcal{I}$, we assume that there is a *universe* $\mathcal{U}(I)$, a *linear cost function* $d^{(I)} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ over the universe, the *feasible sets* $\mathcal{F}(I) \subseteq 2^{\mathcal{U}(I)}$ containing all feasible subsets of the universe, and a *threshold* $t^{(I)}$.

Our ideas are best explained using an example. Consider (the decision version of) the vertex cover problem. For an instance I , we are given an undirected graph $G = (V, E)$, and some threshold $t^{(I)} \in \mathbb{Z}_{\geq 0}$. The question is if there is a vertex cover of size at most $t^{(I)}$. We can rephrase this question as $d^{(I)}(F) \leq t^{(I)}$, where F is some vertex cover and $d^{(I)} = \mathbf{1}$ is the unit cost function. Interpreting the vertex cover problem as an LOP in the above sense means the following: The input instance is given as a tuple $I = (G, t)$ (encoded in binary). The universe associated to some input instance I is given by $\mathcal{U}(I) = V$, and the feasible sets are given by $\mathcal{F}(I) = \{F \subseteq V : F \text{ is a vertex cover}\}$. The cost function and threshold are given by $d^{(I)}$ and $t^{(I)}$.

The decision question associated to the problem is to decide if there is a feasible set $F \in \mathcal{F}(I)$ such that its cost is below the threshold, that is $d^{(I)}(F) \leq t^{(I)}$. (Note that this models a minimization problem, but maximization problems can be modeled as well by using negative coefficients.) In general, we define the *solution set* $\mathcal{S}(I)$ as the set of all solutions of the instance I , that is

$$\mathcal{S}(I) := \{F \in \mathcal{F}(I) : d^{(I)}(F) \leq t^{(I)}\}. \quad (3.1)$$

Roughly stated, our main idea is now to show that many well-known NP-completeness proofs from some problem Π_1 to some other problem Π_2 have a special property, which we call the *SSP property*. On an intuitive level, this property states that the universe of Π_1 can be injectively embedded into the universe of Π_2 in such a way that the following two properties hold: (P1). Every solution of Π_1 corresponds to a partial solutions of Π_2 , and (P2). every solution of Π_2 when restricted to the image of the embedding corresponds to a solution of Π_1 .

We show that a surprisingly large amount of NP-completeness reductions which are known from the literature actually have the SSP property (Chapter 9). Even the historically first NP-completeness reduction, i.e. the reduction used by Cook and Levin to show that SATISFIABILITY is NP-complete has the SSP property (Theorem 3.1). We then proceed to show that every NP-completeness reduction with the SSP property can be upgraded to a Σ_2^P -completeness proof (or Σ_3^P -completeness proof, respectively) between the min-max variants (min-max-min variants, respectively) of the problems Π_1 and Π_2 , hence proving the desired result of Σ_2^P -completeness (Σ_3^P -completeness, respectively).

The next step for us is to consider a slight generalization of the concept of an LOP. This has two reasons: First, it turns out that for most of our arguments, we do not make explicit use of \mathcal{F} , d and t . We only make use of \mathcal{I} , \mathcal{U} , and \mathcal{S} . This means that we can abstract from these unnecessary details to have a cleaner argument. The second reason is that there exist many optimization problems, which can be expressed as an LOP only in an awkward, non-natural way. For example, consider the Hamiltonian cycle problem. Even though it is possible to express the Hamiltonian cycle problem as an LOP, using trivial values for d and t , this definition seems a bit unnatural. For this reason, we consider the concept of so-called *subset search problems (SSP)*. An SSP is a problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$, where \mathcal{I} is the set of input instances, $\mathcal{U}(I)$ is the universe associated to each instance, and $\mathcal{S}(I)$ is the set of solutions associated to each instance. A formal definition is provided in Definition 3.2.

Every LOP can be interpreted as an SSP in a straight-forward way, using eq. (3.1). Hence the vertex cover problem is an example of an SSP problem. Another example of an SSP problem is the problem 3-SATISFIABILITY. The input is some formula φ with clauses c_1, \dots, c_m . The universe is the set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$ of all literals. The solution set of φ is the set of all the subsets of the literals which encode a satisfying solution, i.e.

$$\mathcal{S}(\varphi) = \{L' \subseteq L : |L' \cap \{\ell_i, \bar{\ell}_i\}| = 1 \forall i \in \{1, \dots, n\}, L' \cap c_j \neq \emptyset \forall j \in \{1, \dots, m\}\}.$$

We are now ready to explain our main idea of SSP reductions. Our ideas are best explained with an example. Consider the SSP problem 3SAT with universe $\mathcal{U} = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$ (the literals) and the SSP problem VERTEX COVER with universe $\mathcal{U}' = V$ (the vertices). We recall the classical NP-hardness reduction from 3SAT to VERTEX COVER, depicted in Figure 9.3 from the book of Garey and Johnson [GJ79]. Given a 3SAT instance consisting out of literals $\{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$ and clauses C , the reduction constructs a graph $G = (V, E)$ the following way: The graph contains vertices $W := \{v_{\ell_1}, \dots, v_{\ell_n}\} \cup \{v_{\bar{\ell}_1}, \dots, v_{\bar{\ell}_n}\}$ such that each vertex v_{ℓ_i} is connected to vertex $v_{\bar{\ell}_i}$ with an edge. Furthermore, for each clause, we add a new triangle to the graph, such that the three vertices of the triangle are connected to the corresponding vertices of the literals appearing in the clause. The following is easily verified: Every vertex cover of G has size at least $|L|/2 + 2|C|$ and G has a vertex cover of size $|L|/2 + 2|C|$ if and only if the 3SAT instance is a YES-instance.

The above reduction is of course well-known. However, we want to bring attention to the fact that this reduction has the SSP property. This property is that the reduction maps the set of all solutions of the 3SAT instance to the set of all solutions of the VERTEX COVER instance in a one-to-one fashion. More precisely, consider the set W . Every small vertex cover (of size $|L|/2 + 2|C|$) restricted to the set W directly encodes a possible solution of the 3SAT instance.

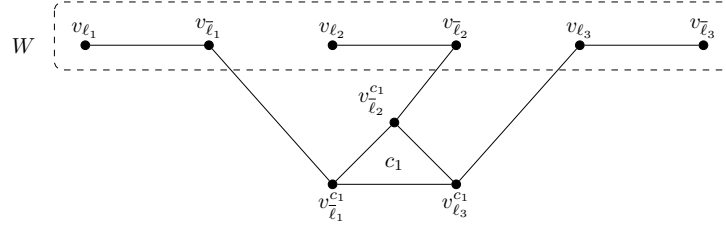


Figure 3.1: Classic reduction of 3SAT to VERTEX COVER for $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$.

Conversely, for every single solution α of the 3SAT instance, we can find a small vertex cover S' (of size $|L|/2 + 2|C|$) such that $S' \cap W$ encodes α .

We describe this one-to-one correspondence more formally. Let $\mathcal{S}(\varphi) \subseteq 2^{\mathcal{U}}$ denote the solutions of 3SAT (i.e. the set of all subsets of the literals which encode a satisfying assignment). Let $k := |L|/2 + 2|C|$ and $\mathcal{S}'(G, k) \subseteq 2^{\mathcal{U}'}$ denote the solutions of VERTEX COVER (i.e. the set of all vertex covers of size at most k). We consider the injective function $f : \mathcal{U} \rightarrow \mathcal{U}'$ with $f(\ell_i) = v_{\ell_i}$ and $f(\bar{\ell}_i) = v_{\bar{\ell}_i}$. This function f can be interpreted as a function which embeds the universe \mathcal{U} into the universe \mathcal{U}' . It describes which literals in \mathcal{U} correspond to which vertices in \mathcal{U}' . Note that the vertex subset $W = f(\mathcal{U})$ is the image of \mathcal{U} . Then the following holds: For every satisfying assignment $S \in \mathcal{S}(\varphi)$, there exists at least one vertex cover $S' \in \mathcal{S}'(G, k)$ such that $S' \cap W = f(S)$. This is property (P1). Conversely, we also have that for every vertex cover $S' \in \mathcal{S}'(G, k)$, the set $f^{-1}(S' \cap W)$ is contained in $\mathcal{S}(\varphi)$. This is property (P2). It can be seen that $(P1) \wedge (P2)$ is equivalent to the set-based equation

$$\{f(S) : S \in \mathcal{S}(\varphi)\} = \{S' \cap f(\mathcal{U}) : S' \in \mathcal{S}'(G, k)\}. \quad (3.2)$$

The above equation defines the SSP property. It turns out to be the key ingredient which is required for our meta-theorem. We call a reduction with the SSP property an SSP reduction (compare Definition 3.4). We write $\Pi_1 \leq_{\text{SSP}} \Pi_2$ for the fact that there exists an SSP reduction from Π_1 to Π_2 . We introduce the class SSP-NPc as an analogon to the class of NP-complete problems, but using polynomial-time SSP reductions instead of normal polynomial-time reductions. Chapter 9 contains a list of problems in SSP-NPc. In order to add a new problem Π to the list, it suffices to prove $\Pi' \leq_{\text{SSP}} \Pi$ for an arbitrary problem $\Pi' \in \text{SSP-NPc}$.

This completes the description of the idea of SSP-reductions. We note that another well-known variants of reductions exists in the literature, so-called *parsimonious* reductions. On the first glance, our reductions seem similar to parsimonious reductions. However, these two concepts are not the same, because parsimonious reductions map solutions to solutions bijectively, while our reductions $f : \mathcal{U} \rightarrow \mathcal{U}'$ map elements to elements injectively.

Finally, we explain how the idea of SSP reductions is used to obtain a meta-theorem. The main idea is that the existence of an SSP reduction tells us that the two involved problems have a very similar solution structure. Therefore, it suffices to prove Σ_2^p -completeness for the min-max variant of only one single problem in SSP-NPc (say SAT, for example), and then invest a little extra work to show that this Σ_2^p -completeness actually carries over to the min-max variant of all other problems in SSP-NPc. For example, in Section 4.2 on interdiction problems, we apply the following proof strategy:

First, we consider the interdiction variant only for the single problem $\Pi = \text{SAT}$. By using traditional techniques, we show that Interdiction-SAT is Σ_2^p -complete (with a reduction from $\exists\forall$ -3DNF-SAT). Next, we consider an arbitrary SSP-NP-complete problem Π' . Since Π' is SSP-NP-complete, there is a reduction $\text{SAT} \leq_{\text{SSP}} \Pi'$. This SSP reduction implies that given a SAT

instance I , we can find a Π' instance I' , such that I can be imagined as a “sub-instance” of I' . Specifically, there is an injective function f mapping the universe of $\mathcal{U}(I)$ into the universe $\mathcal{U}(I')$. Furthermore, the topology of solutions is maintained (by eq. (3.2), or equivalently properties (P1) and (P2)). Hence the SAT instance I can be imagined as a Π' sub-instance of I' . We show that this relation extends in such a way that the corresponding INTERDICTION-SAT instance can be imagined as a sub-instance of INTERDICTION- Π' . We can modify the costs of interdiction such that all newly elements that are part of I' , but not part of I receive infinite costs. On the other hand, all universe elements that are part of I receive the same costs in I' . Therefore all solutions of I' are blocked by some blocker if and only if the sub-instance of SAT is blocked. Since INTERDICTION-SAT Σ_2^P -hard, it follows that INTERDICTION- Π' is Σ_2^P -hard.

The results about min-max regret robust optimization in Section 5.2 and two-stage adjustable robust optimization in Section 6.2 follow essentially the same strategy. We remark that in all three sections, we actually show Σ_2^P -completeness (Σ_3^P -completeness) of a more restricted problem than the original problem for all $\Pi \in \text{SSP-NPc}$. These versions may be of independent interest, since they show that the considered problems are already hard even if the input parameters are more restricted. We call these restricted versions COMBINATORIAL INTERDICTION- Π (Definition 4.2), RESTRICTED INTERVAL MIN-MAX REGRET- Π (Definition 5.2), and COMBINATORIAL TWO-STAGE ADJUSTABLE- Π (Definition 6.2).

3.2 Framework

The goal of this section is to introduce the class of *SSP-NP-complete* problems, i.e. the class of all problems for which our meta-theorem is applicable. As explained in Subsection 3.1.3, we first consider linear optimization problems (LOP problems) and then an abstraction of LOP problems, which we call SSP problems. We then introduce the concept of an SSP reduction, and finally define the class SSP-NPc.

3.2.1 Linear Optimization Problems

It is important to remark that our meta-theorem cannot cover every single discrete optimization problem. This is for two reasons: First, the set of all discrete optimization problems is incredibly diverse. There seems to be no universally agreed upon definition of the term “discrete optimization problem”. Secondly, we need to assume a minimal amount of structure in order to meaningfully describe min-max variants of some problem. For this reason, we make the following assumption: We assume that the problem in question has some *universe* \mathcal{U} . We also assume that there are some feasible solutions associated to the problem, that every feasible solution can be encoded purely as a subset of the universe, and it can be checked efficiently whether some proposed subset is a feasible solution. Finally, we assume that there is some linear cost function on the universe, and the goal of the problem is to find a feasible solution of small cost. Note that all of these assumptions are typical for discrete optimization problems. In order to talk about the computational complexity of an LOP problem, we need to treat it as a decision problem. Therefore, we assume that the input contains some threshold, and the question is whether there is some feasible set whose cost is below the threshold. Formally, this leads to the following definition.

Definition 3.1 (Linear Optimization Problem). *A linear optimization problem (or in short LOP problem) Π is a tuple $(\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$, such that*

- ▶ $\mathcal{I} \subseteq \{0, 1\}^*$ is a language. We call \mathcal{I} the set of instances of Π .
- ▶ To each instance $I \in \mathcal{I}$, there is some

- ▶ set $\mathcal{U}(I)$ which we call the universe associated to the instance I .
- ▶ set $\mathcal{F}(I) \subseteq 2^{\mathcal{U}(I)}$ that we call the feasible solution set associated to the instance I .
- ▶ function $d^{(I)} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ mapping each universe element e to its costs $d^{(I)}(e)$.
- ▶ threshold $t^{(I)} \in \mathbb{Z}$.

For $I \in \mathcal{I}$, we define the solution set $\mathcal{S}(I) := \{S \in \mathcal{F}(I) : d^{(I)}(S) \leq t^{(I)}\}$ as the set of feasible solutions below the cost threshold. The instance I is a YES-instance, if and only if $\mathcal{S}(I) \neq \emptyset$. We assume (for LOP problems in NP) that it can be checked in polynomial time in $|I|$ whether some proposed set $F \subseteq \mathcal{U}(I)$ is feasible.

The following are two examples of LOP problems:

TRAVELING SALESMAN PROBLEM

Instances: Complete graph $G = (V, E)$, weight function $w : E \rightarrow \mathbb{Z}_{\geq 0}$, number $k \in \mathbb{N}$.

Universe: Edge set $E =: \mathcal{U}$.

Feasible solution set: The set of all TSP tours $T \subseteq E$.

Solution set: The set of feasible T with $w(T) \leq k$.

VERTEX COVER

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all vertex covers of G .

Solution set: The set of all vertex covers of G of size at most k .

Recall that a TSP tour is defined as a simple cycle traversing every vertex. Note that for the vertex cover problem, the function $d^{(I)}$ is the unit cost function. The threshold for both of these problems is the number k . The two problems above are minimization problems, but we can model maximization problems in this framework by using negative cost functions. (For example, for the knapsack problem, $d^{(I)}$ is equal to the negative profits.)

3.2.2 Introducing SSPs as an abstraction of LOPs

As explained in Subsection 3.1.3, for most of the arguments in the following paragraphs, we do not really care about the set $\mathcal{F}(I)$ of feasible solutions, the cost function $d^{(I)}$, or the threshold $t^{(I)}$. Rather, these are distracting details which we would like to get rid of. For this reason, we introduce the concept of a *subset search problem* (SSP). An SSP problem is simply a tuple $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$, where \mathcal{I} is the set of instances, $\mathcal{U}(I)$ is the universe, and $\mathcal{S}(I)$ is the solution set associated to each instance. The name “subset search problem” stems from the fact, that we assume that each solution is encoded purely as a subset of the universe, and that the goal of the problem is to search for and find a solution. The SSP concept has the advantage that it captures also such problems, which do not really fit into the LOP scheme. For example, consider the Hamiltonian cycle problem. It is in a certain sense unnatural to describe it using feasible sets, a cost function, and a cost threshold. However, we can define it perfectly well as an SSP problem: The instance is given by $I = (V, E)$ for some graph, the universe is $\mathcal{U}(I) = E$, and the solutions set is $\mathcal{S}(I) = \{T \subseteq E : T \text{ is a Hamiltonian cycle}\}$. Formally, we define an SSP the following way:

Definition 3.2 (Subset Search Problem (SSP)). *A subset search problem (or short SSP problem) Π is a tuple $(\mathcal{I}, \mathcal{U}, \mathcal{S})$, such that*

- ▶ $\mathcal{I} \subseteq \{0, 1\}^*$ is a language. We call \mathcal{I} the set of instances of Π .
- ▶ To each instance $I \in \mathcal{I}$, there is some set $\mathcal{U}(I)$ which we call the universe associated to the instance I .

- To each instance $I \in \mathcal{I}$, there is some (potentially empty) set $\mathcal{S}(I) \subseteq 2^{\mathcal{U}(I)}$ which we call the solution set associated to the instance I .

As a remark, every LOP problem becomes an SSP problem with the definition $\mathcal{S}(I) := \{S \in \mathcal{F}(I) : d^{(I)}(S) \leq t^{(I)}\}$. We call this the *SSP problem derived from an LOP problem*. An example of a natural SSP problem is the satisfiability problem:

SATISFIABILITY

Instances: Literal set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$, clause set $C = \{c_1, \dots, c_m\}$ such that $c_j \subseteq L$ for all $j \in \{1, \dots, m\}$.

Universe: $L =: \mathcal{U}$.

Solution set: The set of all subsets $L' \subseteq \mathcal{U}$ of the literals such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c_j| \geq 1$ for all clauses $c_j \in C$.

Definition 3.3. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem. An instance $I \in \mathcal{I}$ is called *YES-instance*, if $\mathcal{S}(I) \neq \emptyset$. The decision problem associated to the SSP problem Π is the language $\{I \in \mathcal{I} : \mathcal{S}(I) \neq \emptyset\}$ of all YES-instances.

3.2.3 A new type of reduction

In this subsection, we introduce the concept of the *SSP property*. As explained in Subsection 3.1.3, the SSP property states that there is an injective embedding of one SSP problem into another, such that the solution sets of the two SSP problems correspond one-to-one to each other in a strict fashion. An *SSP reduction* is a usual many-one reduction which additionally has the SSP property. By formalizing the intuition gained in Subsection 3.1.3, we obtain the following definition. Note that in this definition the function g corresponds to the standard many-one reduction, while the functions $(f_I)_{I \in \mathcal{I}}$ are the injective embedding functions corresponding to the SSP property (analogous to eq. (3.2)). Note that since $\mathcal{U}(I)$ can be different for every instance I , we have that $(f_I)_{I \in \mathcal{I}}$ is a family of functions, and not a single function.

Definition 3.4 (SSP Reduction). Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ and $\Pi' = (\mathcal{I}', \mathcal{U}', \mathcal{S}')$ be two SSP problems. We say that there is an SSP reduction from Π to Π' , and write $\Pi \leq_{SSP} \Pi'$, if

- There exists a function $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$ computable in polynomial time in the input size $|I|$, such that I is a YES-instance iff $g(I)$ is a YES-instance (i.e. $\mathcal{S}(I) \neq \emptyset$ iff $\mathcal{S}'(g(I)) \neq \emptyset$).
- There exist functions $(f_I)_{I \in \mathcal{I}}$ computable in polynomial time in $|I|$ such that for all instances $I \in \mathcal{I}$, we have that $f_I : \mathcal{U}(I) \rightarrow \mathcal{U}'(g(I))$ is an injective function mapping from the universe of the instance I to the universe of the instance $g(I)$ such that

$$\{f_I(S) : S \in \mathcal{S}(I)\} = \{S' \cap f_I(\mathcal{U}(I)) : S' \in \mathcal{S}'(g(I))\}.$$

An example of an SSP reduction from 3-SATISFIABILITY to VERTEX COVER was shown in Subsection 3.1.3. Many more examples of SSP reductions are shown in Chapter 9. A schematic description how the mapping f_I of an SSP reduction between SSP problems Π and Π' applies for a specific instance I of Π is depicted in Figure 3.2. Next, we show that SSP reductions are transitive, which enables us to easily show reductions between a multitude of problems.

Lemma 3.1. SSP reductions are transitive, i.e. for SSP problems Π_1, Π_2, Π_3 with $\Pi_1 \leq_{SSP} \Pi_2$ and $\Pi_2 \leq_{SSP} \Pi_3$, it holds that $\Pi_1 \leq_{SSP} \Pi_3$.

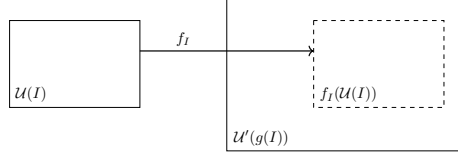


Figure 3.2: The relation between the universes by applying an SSP reduction between the problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ and $\Pi' = (\mathcal{I}', \mathcal{U}', \mathcal{S}')$ for a given instance $I \in \mathcal{I}$. Let $I \in \mathcal{I}$ be that instance of Π , then the SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ maps the universe $\mathcal{U}(I)$ into the universe $\mathcal{U}'(g(I))$ of problem Π' such that $f_I(\mathcal{U}(I)) \subseteq \mathcal{U}'(g(I))$. Note that $g(I)$ is the instance defined by the usual reduction mapping g . The function f_I maintains a one-to-one correspondence between the elements of $\mathcal{U}(I)$ and $f_I(\mathcal{U}(I))$.

Proof. Consider for $i = 1, 2, 3$ the three SSP problems $\Pi_i = (\mathcal{I}_i, \mathcal{U}_i, \mathcal{S}_i)$. There is an SSP reduction (g_1, f_1) from Π_1 to Π_2 and an SSP reduction (g_2, f_2) from Π_2 to Π_3 . We describe an SSP reduction from Π_1 to Π_3 . We require a tuple $(g, (f_I)_{I \in \mathcal{I}})$. For the first function g , we set $g := g_2 \circ g_1$. This suffices since $\mathcal{S}_1(I) \neq \emptyset \Leftrightarrow \mathcal{S}_2(g_1(I)) \neq \emptyset \Leftrightarrow \mathcal{S}_3((g_2 \circ g_1)(I)) \neq \emptyset$ and g is poly-time computable. Let $I_1 := I$ be the initial instance of Π_1 , $I_2 := g_1(I_1)$ be the instance of Π_2 and $I_3 := g_2(I_2)$ be the instance of Π_3 .

For the second function f_I , we define for each instance $I \in \mathcal{I}_1$ the map $f_I := (f_2)_{I_2} \circ (f_1)_I$. Observe that for each instance $I \in \mathcal{I}_1$, the function f_I is injective and maps to $\mathcal{U}_3(I_3)$ and is poly-time computable. It remains to show that f has the desired SSP property. In order to reduce the notation, we omit the subscript in f (i.e. $f = f_2 \circ f_1$). We also write $\mathcal{U}_1, \mathcal{U}_2, \mathcal{U}_3$ instead of $\mathcal{U}_1(I_1), \mathcal{U}_2(I_2), \mathcal{U}_3(I_3)$ and $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ instead of $\mathcal{S}_1(I_1), \mathcal{S}_2(I_2), \mathcal{S}_3(I_3)$. Since $\Pi_1 \leq_{\text{SSP}} \Pi_2$ and $\Pi_2 \leq_{\text{SSP}} \Pi_3$ and since for injective functions it holds that $f(A \cap B) = f(A) \cap f(B)$, we have

$$\begin{aligned}
 \{f(\mathcal{S}_1) : \mathcal{S}_1 \in \mathcal{S}_1\} &= \{f_2(f_1(\mathcal{S}_1)) : \mathcal{S}_1 \in \mathcal{S}_1\} \\
 &= \{f_2(Y) : Y \in \{f_1(\mathcal{S}_1) : \mathcal{S}_1 \in \mathcal{S}_1\}\} \\
 &= \{f_2(Y) : Y \in \{\mathcal{S}_2 \cap f_1(\mathcal{U}_1) : \mathcal{S}_2 \in \mathcal{S}_2\}\} \\
 &= \{f_2(\mathcal{S}_2) \cap f_2(f_1(\mathcal{U}_1)) : \mathcal{S}_2 \in \mathcal{S}_2\} \\
 &= \{\mathcal{S}_3 \cap f_2(\mathcal{U}_2) \cap f_2(f_1(\mathcal{U}_1)) : \mathcal{S}_3 \in \mathcal{S}_3\} \\
 &= \{\mathcal{S}_3 \cap f(\mathcal{U}_1) : \mathcal{S}_3 \in \mathcal{S}_3\}.
 \end{aligned}$$

□

Further, we define the class **SSP-NP**, which is the analogue of **NP** restricted to SSP problems.

Definition 3.5 (SSP-NP). *The class SSP-NP consists out of all the SSP problems which are polynomial-time verifiable. Formally, an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ belongs to SSP-NP, if $|\mathcal{U}(I)| = \text{poly}(|I|)$ and if there is an algorithm receiving tuples of an instance $I \in \mathcal{I}$ and a subset $S \subseteq \mathcal{U}(I)$ as input and decides in time polynomial in $|I|$, whether $S \in \mathcal{S}(I)$.*

For the remainder of the part, in a slight abuse of notation, let us say that $\text{SSP-NP} \subseteq \text{NP}$. Note that this is not formally completely correct, since the class **NP** is a set of languages, while the class **SSP-NP** is a set of SSP problems. However, we can say that some SSP problem Π is in **NP**, if the corresponding decision problem $\{I \in \mathcal{I} \mid \mathcal{S}(I) \neq \emptyset\}$ is in **NP**.

Because of the analogous definition of the class **SSP-NP** to **NP** and the natural SSP adaptation of **SATISFIABILITY**, we are able to adapt the theorem of Cook and Levin [Coo71, Tra84] to the class **SSP-NP** and show that **SATISFIABILITY** is the canonical **SSP-NP**-complete problem.

Theorem 3.1 (Cook-Levin Theorem Adapted to SSPs). *SATISFIABILITY is SSP-NP-complete with respect to polynomial-time SSP reductions, i.e. for every SSP problem Π contained in SSP-NP, we have $\Pi \leq_{SSP}$ SATISFIABILITY.*

Proof. We consider the original proof by Cook and show that it is actually a polynomial-time SSP reduction. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an arbitrary problem in SSP-NP with universe $\mathcal{U}(I)$ and solution set $\mathcal{S}(I)$ associated to each instance $I \in \mathcal{I}$ of Π . We have to show that $\Pi \leq_{SSP}$ SATISFIABILITY. Recall that SATISFIABILITY = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$, where \mathcal{I}' is the set of SAT-instances, and for each formula $\varphi \in \mathcal{I}'$, the set $\mathcal{U}'(\varphi)$ is its literal set, and $\mathcal{S}(\varphi)$ is the set of literal sets corresponding to satisfying assignments.

Since Π is in SSP-NP, there exists a deterministic Turing machine M , such that given as input some tuple (I, S) with $I \in \mathcal{I}$ and $S \subseteq \mathcal{U}(I)$ (encoded in binary), the Turing machine M decides in polynomially many steps (say at most $|I|^k$ for some k), whether $S \in \mathcal{S}(I)$. Here we use the equivalent definition of the class NP in terms of verifiers and in terms of nondeterministic Turing machines [AB09]. Now, the proof of Cook implies that there exists a CNF-formula $\varphi(Y, Z)$ with the following properties:

- ▶ The formula has size polynomial in $|I|$ and can be constructed in polynomial time from I .
- ▶ The variables are split into two parts Y, Z . Here, the variables Z encode in binary the input $S \subseteq \mathcal{U}(I)$ of the Turing machine M . The number of these variables is $|Z| = |\mathcal{U}(I)|$. The set Y contains all other variables.
- ▶ The partial formula “ $\varphi(Y, S)$ ” is satisfiable if and only if M accepts (I, S) . More formally, for all assignments $\alpha : Z \rightarrow \{0, 1\}$, we let S_α be the corresponding subset of $\mathcal{U}(I)$ (defined by letting $\mathcal{U}(I) = \{u_1, \dots, u_m\}$ and $Z = \{z_1, \dots, z_m\}$ and considering the binary encoding $u_i \in S_\alpha$ iff $\alpha(z_i) = 1$ for $i = 1, \dots, m$). Furthermore, we let $\varphi(Y, \alpha)$ be the formula where the Z -variables are assigned by α , and the Y -variables are still free. Then we have for all $\alpha : Z \rightarrow \{0, 1\}$:

$$\varphi(Y, \alpha) \text{ is satisfiable} \Leftrightarrow M \text{ accepts } (I, S_\alpha) \text{ after at most } |I|^k \text{ steps.}$$

We now claim that this reduction by Cook immediately yields a polynomial-time SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$. Formally, we let $g(I) := \varphi(Y, Z)$. Note that by the properties of Cook’s reduction, I is a YES-instance of Π if and only if $\exists Y, Z \varphi(Y, Z)$ is satisfiable. Hence this is a correct many-to-one reduction. For the SSP property, we define $f_I(u_i) := z_i$ for all $i \in \{1, \dots, |\mathcal{U}(I)|\}$. Informally speaking, the universe element u_i is mapped to the positive literal $z_i \in \mathcal{U}(\varphi)$ which encodes in binary in the input to M , whether the element u_i is included in S . It now follows from the above equivalence that this is an SSP reduction: If $S \in \mathcal{S}(I)$, then M accepts (I, S) after $|I|^k$ steps, and for the corresponding assignment $f_I(S)$ it holds that it can be completed to a satisfying assignment of φ . On the other hand, every satisfying assignment $S' \in \mathcal{S}(\varphi)$ restricted to the positive literal set $Z = f_I(\mathcal{U}(I))$ encodes a set $S = f_I^{-1}(S' \cap Z)$ such that $S \in \mathcal{S}(I)$. This proves the SSP property and hence $\Pi \leq_{SSP}$ SATISFIABILITY. \square

With Theorem 3.1 in mind, we define the class of SSP-NP-complete problems (SSP-NPc) as the set of all SSP-NP problems that are complete for the class SSP-NP with respect to SSP reductions.

Definition 3.6. *The class of SSP-NP-complete problems is called SSP-NPc and consists of all $\Pi \in$ SSP-NP such that $SATISFIABILITY \leq_{SSP} \Pi$.*

3.3 SSP-NP-complete Problems

In this section, we give a short overview over the problem and reduction landscape of the complexity class SSP-NP. All formal problem definitions and reductions can be found in Chapter 9. Here, we only state the following theorem and present the corresponding reduction relation between the problems in Figure 3.3.

Theorem 3.2. *The following problems are SSP-NP-complete: SATISFIABILITY, 3SATISFIABILITY, VERTEX COVER, DOMINATING SET, SET COVER, HITTING SET, FEEDBACK VERTEX SET, FEEDBACK ARC SET, UNCAPACITATED FACILITY LOCATION, P-CENTER, P-MEDIAN, INDEPENDENT SET, CLIQUE, SUBSET SUM, KNAPSACK, PARTITION, SCHEDULING, DIRECTED HAMILTONIAN PATH, DIRECTED HAMILTONIAN CYCLE, UNDIRECTED HAMILTONIAN CYCLE, TRAVELING SALESMAN PROBLEM, TWO DIRECTED VERTEX DISJOINT PATH, k -VERTEX DIRECTED DISJOINT PATH, STEINER TREE*

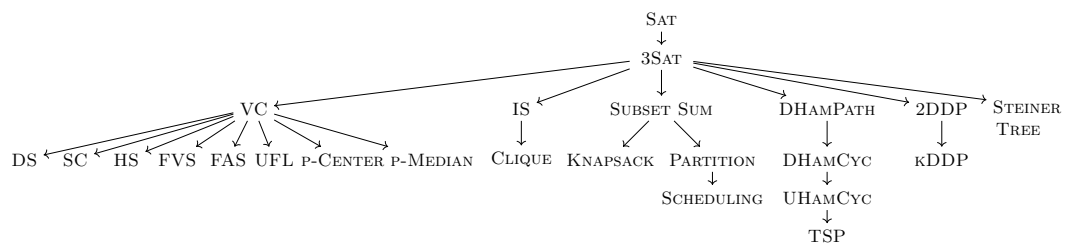


Figure 3.3: The tree of SSP reductions for all considered problems.

Chapter 4

Minimum Cost Interdiction

4.1 Introduction

This chapter is concerned with the *minimum cost interdiction problem*, by which we understand the following task: Given some base problem (the so-called *nominal problem*) we wish to find a subset of elements of small costs such that this subset has a non-empty intersection with every optimal solution of the base problem. The concept of interdiction is so natural that it has reappeared under many different names in different research communities. Depending on the context, the interdiction problem (or slight variants of it) has been called the *most vital node/most vital edge* problem, the *blocker* problem, and *node deletion/edge deletion* problem. As an example for the type of problems that this chapter is concerned with, consider the following problem:

MIN COST CLIQUE INTERDICTION

Input: Graph $G = (V, E)$, cost function $c : E \rightarrow \mathbb{Z}$

Task: Find a minimum-cost subset $V' \subseteq V$ such that every maximum clique shares at least one vertex with V' .

In particular, if in the above example the set V' is deleted from the graph, the maximum clique size decreases. Hence the interdiction problem can be interpreted as the minimal effort required to destroy all optimal solutions. Clearly, analogous problems can be defined and analyzed for a wealth of different nominal problems. Indeed, this has been done extensively by past researchers. The following is a non-exhaustive list: Interdiction-like problems have been considered already since the '90s for a large amount of problems, among others for shortest path [BNKS98, KBB⁺08, MMG89], matching [Zen10], minimum spanning tree [LC93], or maximum flow [Woo93]. Note that in all these cases the nominal problem can be solved in polynomial time. Interdiction for nominal problems that are NP-complete has also been extensively considered, for example for vertex covers [BTT10, BTT11], independent sets [BBPR15, BTT10, BTT11, HLW23, LR24, PPR17], colorings [BBPR15, PPR16, PPR17], cliques [FLMS19, Paj20, PBP14, PPR16], knapsack [CCLW13, WF24], dominating sets [GLR21, PWBP15], facility location [FR21], 1- and p -center [BTV10, BTV13], and 1- and p -median [BTV10, BTV13]. A general survey is provided by Smith, Prince, and Geunes [SPG13].

This large interest is due to the fact that interdiction problems are well-motivated from many different directions. In the area of robust optimization, interdiction is studied because it concerns robust network design, defense against (terrorist) attacks, and sensitivity analysis [TCCH24]. In particular, we want to find the most vital nodes/edges of a given network in order to identify its

most vulnerable points and understand where small changes have the largest impact. Interdiction in these contexts is often interpreted as a min-max optimization problem or alternatively as a game between a network interdictor (attacker) and a network owner (defender) with competing goals. In the area of bilevel optimization, interdiction-like problems arise naturally from the dynamic between two independent hierarchical agents [CCLW13]. In the area of pure graph theory, interdiction problems are usually called vertex and edge blocker problems. They relate to the important concepts of maximum induced subgraphs, critical vertices and edges, cores, and transversals (with respect to some fixed property) [PPR17]. In the area of (parameterized) complexity, interdiction-like problems are usually called vertex deletion problems. They arise from the desire to delete a constant number of vertices until the resulting graph has some desirable property, for example so that it can be handled by an efficient algorithm. For instance, Lewis and Yannakakis showed that the vertex deletion problem for hereditary graph properties is NP-complete [LY80] and Bannach, Chudigiewitsch and Tantau analyzed the parameterized complexity for properties definable by first-order formulas [BCT24].

Our Results. In Section 4.2, we are concerned with the *minimum cost interdiction problem*. In the minimum cost interdiction problem, one is given a nominal problem Π and additionally a cost function. The question is for Alice to find a *blocker* of minimum cost. A blocker is a set which intersects every solution of the nominal problem. (For example, every Hamiltonian cycle, every minimum vertex cover, every maximum clique, etc.) Our main result is that for every problem $\Pi \in \text{SSP-NPc}$, the corresponding minimum cost interdiction problem is Σ_2^p -complete. We remark that Σ_2^p -completeness was already known in the case of clique/independent set, and knapsack [CCLW13, Rut93, TCCH24]. Hence our work is a generalization of these results. Concretely, we define a minimum cost interdiction problem for the following nominal problems and prove their Σ_2^p -completeness:

Sat, 3Sat, vertex cover, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median, independent set, clique, subset sum, knapsack, partition, scheduling, Hamiltonian path/cycle (directed/undirected), TSP, k -directed disjoint path ($k \geq 2$), and Steiner tree.

We remark that this chapter is the first of two chapters on interdiction. In this Chapter 4, we are only concerned with the more specialized version of minimum cost interdiction. In Chapter 8, we generalize the results to minimum cardinality interdiction.

Related Work. Usually in the literature, the complexity of interdiction problems is not discussed beyond NP-hardness in the context of the polynomial hierarchy. However, there are the following exceptions: Rutenburg [Rut93] proves Σ_2^p -completeness for clique interdiction. Caprara, Carvalho, Lodi & Woeginger [CCLW13] consider different bilevel knapsack formulations and prove Σ_2^p -completeness of the DeNegre [Den11] knapsack variant, which can be interpreted as an interdiction knapsack variant. Tomasaz, Carvalho, Cordone & Hosteins [TCCH24] consider interdiction-fortification games and prove Σ_2^p -completeness of another knapsack interdiction variant. Fröhlich and Ruzika prove Σ_2^p -completeness of a facility location interdiction problem on graphs (in contrast to our work, the interdictor attacks edges instead of vertices) [FR21, Section 4]. Our work extends these results to more problem classes. Finally, in a seminal paper, Lewis & Yannakakis prove the very general result that the most vital vertex problem is NP-hard for every nontrivial hereditary graph property [LY80]. Our work adds to these results by showing that in many cases, interdiction is even harder than NP-hard.

4.2 Interdiction Problems

We consider the closely related topics of *minimum cost blocker problems*, *most vital vertex/edge problems* and *interdiction problems*. All of these problems are slight variants of each other. Formally, we consider the following problem.

Definition 4.1 (Interdiction Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The interdiction problem associated to Π is denoted by INTERDICTION- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with a cost function $c : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and a threshold $t \in \mathbb{Z}$. The question is whether*

$$\exists B \subseteq \mathcal{U}(I) \text{ with } c(B) \leq t : \forall S \in \mathcal{S}(I) : B \cap S \neq \emptyset.$$

The main result of this chapter is that INTERDICTION- Π is Σ_2^p -complete for all SSP-NP-complete problems Π (Theorem 4.2). We also show in Section 4.3 the Σ_2^p -completeness of a more restricted version of interdiction, which could be of independent interest.

We make a few remarks regarding Definition 4.1: First, note that the interdiction variant of all LOP problems can be defined over the SSP problem derived from it as described in Section 3.2. The set $\mathcal{S}(I)$ then contains all the sets the attacker wants to block. For example in the vertex cover problem, $\mathcal{S}(I)$ contains all vertex covers of instance I of size smaller or equal to t . (W.l.o.g. due to the properties of the reductions studied in this part, we can for all problems from Chapter 9 assume that t is chosen to be the optimal threshold.) Second, note that in our problem INTERDICTION- Π , the underlying instance is not changed. In particular, we do not delete elements of the universe (e.g. vertices of the graph). For some problems, there might be a subtle difference between deleting elements and forbidding elements to be in the solution. (The vertex cover problem is one such example: It makes a big difference of deleting a vertex v and its incident edges, or forbidding that v is contained in the solution, but still having the requirement that all edges incident to v get covered by the vertex cover. In the first case, the vertex cover interdiction problem stays in NP, hence we can not hope to obtain a general Σ_2^p -completeness result. In this part we only consider the second case.)

As the third remark, we note that in the literature, usually the minimum cost blocker problem and the most vital nodes/edges problem are slightly differently defined: The minimum cost blocker problem asks for a minimum cost blocker which decreases the objective value by a set amount. On the other hand, the most vital nodes asks for the maximum value by which the objective can be decreased, when given a certain cost budget for the blocker. The interdiction problem, which we formulated here as a decision problem enables us to capture the Σ_2^p -completeness of both these variants. It follows by standard arguments that both of the above problems become Σ_2^p -complete in our setting.

The last remark is that INTERDICTION- Π can be understood as a game between Alice (\exists -player, trying to find a blocker) and Bob (\forall -player, trying to find a solution). Note that this could be considered different from other typical robust optimization problems, where the \exists -player tries to find a solution. In the remainder of this chapter, we locate the complexity of INTERDICTION- Π exactly. The easy part is to show containment in Σ_2^p .

Lemma 4.1. *If $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ is a problem in SSP-NP, then INTERDICTION- Π is in Σ_2^p .*

Proof. We provide a polynomial time algorithm V such that for $m_1, m_2 = |I|^{O(1)}$:

$$I \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} : V(I, y_1, y_2) = 1.$$

With the \exists -quantified y_1 , we encode the blocker $B \subseteq \mathcal{U}(I)$. The encoding size of y_1 is polynomially bounded in the input size of Π because $|\mathcal{U}(I)| = |I|^{O(1)}$. Next, we encode the

solution $S \in \mathcal{S}(I)$ to the nominal problem Π using the \forall -quantified y_2 within polynomial space. This is doable because the problem Π is in NP. At last, the verifier V has to verify the correctness of the given tuple (B, S) provided by the \exists -quantified y_1 and \forall -quantified y_2 . Checking whether $c(B) \leq t$ and $B \cap S \neq \emptyset$ is trivial and checking whether $S \in \mathcal{S}(I)$ is clearly in polynomial time because Π is in SSP-NP. It follows that INTERDICTION- Π is in Σ_2^p . \square

4.3 Combinatorial Interdiction Problems

In the literature, often the cost version from above is analyzed and used to model real-world problems. However, for us it proves to be helpful to introduce a more restricted variant of INTERDICTION- Π , which we call the *combinatorial version* of INTERDICTION- Π . This combinatorial version is slightly more specific than the cost version and allows for a more precise reduction in the SSP framework. We show the Σ_2^p -hardness of the combinatorial version of INTERDICTION- Π for all SSP-NP-complete problems. In the end, we adapt this result to the cost version. This shows the Σ_2^p -hardness of all interdiction problems, for which the nominal problem is SSP-NP-complete.

Definition 4.2 (Combinatorial Interdiction Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The combinatorial interdiction problem associated to Π is denoted by COMB. INTERDICTION- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with a set of blockable elements $B \subseteq \mathcal{U}(I)$ and a threshold $t \in \mathbb{Z}$. The question is whether*

$$\exists B' \subseteq B \text{ with } |B'| \leq t : \forall S \in \mathcal{S}(I) : B' \cap S \neq \emptyset.$$

The difference between the combinatorial version and the cost version is that we ignore the costs of the elements and introduce a set of possibly blockable elements. We use the canonical SATISFIABILITY problem as the starting point for our meta reduction. Therefore, we apply Definition 4.2 to $\Pi = \text{SATISFIABILITY}$ yielding the following:

Definition 4.3 (COMBINATORIAL INTERDICTION-SATISFIABILITY). *We denote the combinatorial interdiction version of SAT by COMB. INTERDICTION-SAT. The input is a CNF with clauses C and literals L , a set blockable literals $B \subseteq L$ and a threshold t . The question is whether there is a set $B' \subseteq B$ with $|B'| \leq t$ such that for all $S \in \mathcal{S}(I)$, we have $B' \cap S \neq \emptyset$. (In other words, there is no satisfying assignment whose literals are completely disjoint from B' .)*

4.4 A Meta-Reduction for Combinatorial Interdiction Problems

For the beginning of our meta-reduction, we prove that the canonical SATISFIABILITY problem COMBINATORIAL INTERDICTION-SATISFIABILITY is Σ_2^p -complete.

Lemma 4.2. *COMBINATORIAL INTERDICTION-SATISFIABILITY is Σ_2^p -complete.*

Proof. Analogously to Lemma 4.1, COMB. INTERDICTION-SAT is in Σ_2^p . As the basis of our hardness proof, we use the problem $\exists\forall\text{DNF-SAT}$, which is Σ_2^p -hard as shown by Stockmeyer [Sto76]. In this problem, we are given a SAT formula $\varphi(X, Y)$ in disjunctive normal form (DNF), such that its variables are partitioned into two parts X, Y . The question is whether there is a variable assignment for X such that for all variable assignments for Y we have $\varphi(X, Y) = 1$. We reduce $\exists\forall\text{DNF-SAT}$ to COMB. INTERDICTION-SAT. Let $\exists X \forall Y \varphi(X, Y)$ be the $\exists\forall\text{DNF-SAT}$ instance. We transform this instance into an equivalent instance (ψ, B, t) of COMB. INTERDICTION-SAT.

More precisely, this means that $\exists X \forall Y \varphi(X, Y)$ if and only if there is a set $B' \subseteq B$ with $|B'| \leq t$ such that there is no solution $S \in \mathcal{S}(\exists X' \psi(X'))$ with $B \cap S = \emptyset$.

We use an idea of Goerigk, Lendl and Wulf [GLW24, Theorem 1]. We quickly sketch the main idea: Let $n := |X|$, i.e. $X = \{x_1, \dots, x_n\}$. COMB. INTERDICTION-SAT can be understood as a game between Alice and Bob, where Alice selects a blocker $B' \subseteq B$ and Bob tries to select a satisfying assignment avoiding B' . How can we model the formula $\exists X \forall Y \varphi(X, Y)$ with this game? The idea is that Alice's choice of a blocker B' should correspond to the $\exists X$ stage and Bob's choice of assignment should correspond to the $\forall Y$ stage. How can we encode an assignment of the X -variables in terms of a blocker B' ? The idea is to introduce new variables $X^t = \{x_1^t, \dots, x_n^t\}$ and $X^f = \{x_1^f, \dots, x_n^f\}$. We say that Alice plays honestly, if $|B' \cap \{x_i^t, x_i^f\}| = 1$ for all $i = 1, \dots, n$. In the other case, i.e. B' contains both x_i^t, x_i^f for some i , we say that Alice cheats. We will add some "cheat-detection" gadgets to the formula, which make sure that if Alice cheats, then Bob can trivially win the game. We make sure that the cheat-detection gadget can be used if and only if Alice cheats. If Alice plays honestly, note that $x_i^t \in B'$ enables Bob to choose x_i^f as part of his solution. This corresponds to the assignment $\alpha(x) = 0$. On the other hand, $x_i^f \in B'$ corresponds to $\alpha(x_i) = 1$. We are now ready to give the formal reduction.

Definition of the instance Given an instance φ of $\exists \forall$ DNF-SAT, the instance (ψ, B, t) of combinatorial interdiction SAT is defined as follows: We start by considering the auxiliary formula $\varphi'(X, Y) := \neg \varphi(X, Y)$. Note that φ' is in CNF by De Morgan's law. We furthermore have

$$\exists X \forall Y \varphi(X, Y) \leftrightarrow \exists X \neg \exists Y \varphi'(X, Y).$$

A new formula φ'' is created from φ' in terms of a substitution process: We introduce $2n$ new variables $X^t = \{x_1^t, \dots, x_n^t\}$ and $X^f = \{x_1^f, \dots, x_n^f\}$. For all $i = 1, \dots, n$, we substitute each occurrence of some literal x_i by the positive literal x_i^t . We substitute each occurrence of some literal \bar{x}_i by the positive literal x_i^f . All other literals are kept the same. A new formula φ''' is created from φ'' by introducing a new variable s and appending the positive literal s to every clause, that is

$$\varphi''' \equiv \varphi'' \vee s.$$

Finally, we introduce n new variables $\{s_1, \dots, s_n\}$. We let $Z = \{s\} \cup \{s_1, \dots, s_n\}$ and define the formula ψ by

$$\psi(X^t, X^f, Y, Z) = \varphi'''(X^t, X^f, Y, s) \wedge \left(\bigwedge_{i=1}^n (x_i^t \vee \bar{s}_i) \wedge (x_i^f \vee \bar{s}_i) \right) \wedge (\bar{s} \vee s_1 \vee s_2 \vee \dots \vee s_n).$$

We remark that the newly added elements between φ'' and ψ form the cheat-detection gadget. Finally, we define the set of blockable literals by $B := X^t \cup X^f$ and the number of blockable literals by $t := n$. This completes the description of the instance (ψ, B, t) .

Correctness Note that Alice can only block the positive literals of $X^t \cup X^f$ because by definition of the comb. interdiction problem we have $B' \subseteq B = X^t \cup X^f$. Furthermore, we claim that in an optimal game, Alice has to play honestly. To prove this, consider the case where both literals x_i^t and x_i^f are blocked or less than t literals in total are blocked by Alice. In both cases there is a $j \neq i$ such that Alice blocks neither x_j^t nor x_j^f (due to $|B'| \leq t$ and the pigeonhole principle). Hence Bob is able to take both literals x_j^t and x_j^f into his solution. This enables Bob to also take both s_j and s into the solution without violating the constraints of ψ . For all other $p \neq j$, Bob can take the literals \bar{s}_p into his solution. In this case ψ is trivially satisfied. We conclude that Alice has to play honestly, i.e. $|B' \cap \{x_i^t, x_i^f\}| = 1$ for all $i = 1, \dots, n$.

Hence for a fixed honest choice of Alice, we obtain a fixed chosen assignment $\alpha(X) \rightarrow \{0, 1\}^{|X|}$ which Bob is forced to take. (More precisely, Bob is forced to take $\bar{s}, \bar{s}_1, \dots, \bar{s}_n$ into his solution. Then the cheat-detection clauses are trivially verified. The remaining formula does not contain any negative literal \bar{x}_i^f, \bar{x}_i^t , so we can w.l.o.g. assume that under optimal play Bob takes the one from the two positive literals x_i^t, x_i^f that is not blocked by Alice.) Following the above restriction, Alice's goal is described by the formula $\neg\exists Y, Z \psi(X^t, X^f, Y, Z)$. This in the end is equivalent to the formula $\neg\exists Y \varphi'(\alpha, Y)$, since for honest behavior of Alice, the only way to satisfy the formula is to set s, s_1, \dots, s_n to false. We conclude that if φ is a YES-instance of $\exists\forall$ DNF-SAT, then Alice can play honestly and win the game, hence we have a YES-instance of COMB. INTERDICTION-SAT.

On the other hand, assume that the described tuple (ψ, B, t) is a YES-instance of COMB. INTERDICTION-SAT. By the previous argument, since Alice was able to win, she must have played honestly. Then, only one of x_i^t and x_i^f can be in the blocker B' for all $i \in \{1, \dots, |X|\}$. Thus, there is a blocker $B' \subseteq B$ fixing the assignment on $X^t \cup X^f$ such that there is no solution to $\exists Y, Z \psi(X^t, X^f, Y, Z)$. This is equivalent to fixing the assignment on $X^t \cup X^f$, such that $\neg\exists Y, Z \psi(X^t, X^f, Y, Z)$. By transforming $\psi(X^t, X^f, Y, Z)$ back to $\neg\varphi(X, Y)$, we get that there is an assignment to X such that $\forall Y \varphi(X, Y)$, which is equivalent to $\exists X \forall Y \varphi(X, Y)$, which is a YES-instance for $\exists\forall$ DNF-SAT.

Polynomial Time All transformations are doable in polynomial time because only a polynomial number of additional variables as well as clauses are added to the formula. □

With the Σ_2^p -hardness of COMB. INTERDICTION-SAT established, we are able to provide a meta-reduction to all COMB. INTERDICTION-II, if there is an SSP reduction between the nominal SATISFIABILITY and the nominal Π . In other words, we prove the Σ_2^p -hardness of COMB. INTERDICTION-II.

Theorem 4.1. *For all SSP-NP-complete problems Π , the combinatorial interdiction variant COMB. INTERDICTION-II is Σ_2^p -complete.*

Proof. Analogously to Lemma 4.1, COMB. INTERDICTION-II is in Σ_2^p . For the hardness, we use the observation that if a problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ is SSP-NP-complete, there is an SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ from SAT to Π . We extend the SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ to a polynomial-time reduction g' from COMB. INTERDICTION-SAT to COMB. INTERDICTION-II as depicted in Figure 4.1. The main idea is that the underlying reduction from SAT to Π remains the same function g and the additional set of blockable elements is redefined such that the blocking sets (which are the solutions to the problems) have a one-to-one correspondence.

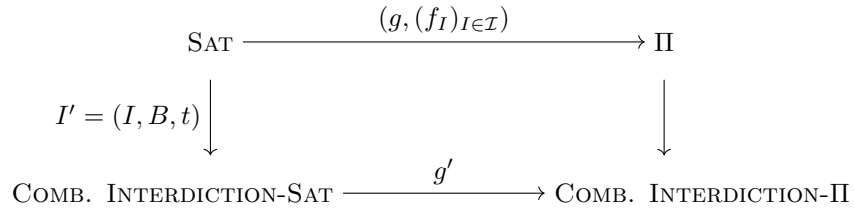


Figure 4.1: The fact that SAT is SSP reducible to Π induces a reduction from COMB. INTERDICTION-SAT to COMB. INTERDICTION-II.

Let $I' = (I, B, t)$ be the instance of COMB. INTERDICTION-SAT, where $I \in \mathcal{I}$ is the corresponding SAT instance, B is the set of blockable elements and t is the threshold. Then, the reduction g' is defined by $g'(I') = (g(I), B_{\text{new}}, t)$, where B_{new} is the new blockable set in COMB. INTERDICTION-II defined as

$$B_{\text{new}} = f_I(B) \subseteq \mathcal{U}(g(I)).$$

I.e. the injectively mapped universe elements from $\mathcal{U}(I)$ remain blockable in COMB. INTERDICTION-II if and only if they are blockable in COMB. INTERDICTION-SAT. Furthermore, observe that by this definition of B_{new} , we have that all newly introduced universe elements, i.e. all elements in $\mathcal{U}(g(I)) \setminus f_I(\mathcal{U}(I))$ are not blockable. Now, recall that the SSP property for $(g, (f_I)_{I \in \mathcal{I}})$ states that the solutions of SAT and Π correspond one-to-one to each other on the set $f_I(\mathcal{U}(I))$. Namely, if $\mathcal{S}(I)$ denotes the solutions of the SAT instance and $\mathcal{S}(g(I))$ denotes the solutions of the corresponding Π instance, then

$$\{f_I(S) : S \in \mathcal{S}(I)\} = \{S' \cap f_I(\mathcal{U}(I)) : S' \in \mathcal{S}(g(I))\}.$$

As a consequence of this, and the fact that $B_{\text{new}} \subseteq f_I(\mathcal{U}(I))$, we have that each blocking set $B \subseteq \mathcal{U}(I)$ in COMB. INTERDICTION-SAT is in a direct one-to-one correspondence to some blocking set in COMB. INTERDICTION-II. In particular, there exists a blocker for the instance (I, B, t) if and only if there exists a blocker for the instance $(g(I), B_{\text{new}}, t)$. Finally, note that the whole reduction g' can be computed in polynomial time, in particular since g and f_I (and therefore B_{new}) can be computed in polynomial time. In summary, COMB. INTERDICTION-SAT reduces to COMB. INTERDICTION-II and COMB. INTERDICTION-II is Σ_2^p -hard. \square

4.5 Adapting the Meta-Reduction to the Cost Version

While the previous subsection showed Σ_2^p -completeness of the combinatorial interdiction problem COMB. INTERDICTION-II, the goal of this subsection is to show the same for INTERDICTION-II, i.e. the version with element costs. This is done via an easy reduction, which re-adapts the combinatorial interdiction problem to the cost version. Note that the following theorem also holds for all LOP problems. Specifically, if one is given an LOP problem, one can consider the SSP problem derived from it as described in Section 3.2.

Theorem 4.2. *For all SSP-NP-complete problems Π , the interdiction variant INTERDICTION-II is Σ_2^p -complete.*

Proof. Due to Lemma 4.1, INTERDICTION-II is in Σ_2^p . We further reduce COMB. INTERDICTION-II to INTERDICTION-II. Assume an instance (I, B, t) of COMB. INTERDICTION-II is given. We use the cost function $c : \mathcal{U}(I) \rightarrow \mathbb{Z}$ in INTERDICTION-II to distinguish the elements in the blockable set B from those that are not blockable. For this, we set $c(b) = 1$ for all $b \in B$ (blockable) and $c(u) = t + 1$ for all $u \in \mathcal{U}(I) \setminus B$ (not blockable). It is clear that every blocker B' with $c(B') \leq t$ uses no elements from $\mathcal{U}(I) \setminus B$ and at most t elements from B . The reduction is obviously polynomial-time computable. Consequently, the reduction is correct and INTERDICTION-II is Σ_2^p -complete as well. \square

4.6 The Meta-Reduction is also an SSP reduction

In Section 4.3, we showed that COMB. INTERDICTION-SAT reduces to COMB. INTERDICTION-II. The goal of this subsection is to show the slightly stronger statement that this reduction is again

an SSP reduction itself. This fact can be succinctly stated as “SATISFIABILITY \leq_{SSP} Π implies COMB. INTERDICTION-SAT \leq_{SSP} COMB. INTERDICTION-II”. We believe that this is an elegant result which deserves to stand on its own right. In order state this result it becomes necessary to explain in which way COMB. INTERDICTION-SAT and COMB. INTERDICTION-II are interpreted as SSP problems. This is done the following way.

Observation 4.1. *The combinatorial interdiction variant COMB. INTERDICTION-II of an SSP problem Π is an SSP problem.*

Proof. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem and we denote COMB. INTERDICTION-II = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$. Note that by definition, we have $\mathcal{U} = \mathcal{U}'$. Now, let $I \in \mathcal{I}$ be an instance of Π . Then, we can define the corresponding instances of COMB. INTERDICTION-II by setting

$$\mathcal{I}' = \{(I, B, t) \mid I \in \mathcal{I}, B \subseteq \mathcal{U}(I), t \in \mathbb{N}\}.$$

Furthermore, we set the solutions for the combinatorial variant to

$$\mathcal{S}'(I') = \{B' \subseteq B \mid |B'| \leq t \text{ and } \forall S \in \mathcal{S}(I) : B' \cap S \neq \emptyset\}$$

Thus, COMB. INTERDICTION-II = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$ is an SSP problem. \square

According to Observation 4.1, the equivalent definition of COMB. INTERDICTION-SATISFIABILITY as SSP problem is the following.

Definition 4.4 (COMBINATORIAL INTERDICTION-SATISFIABILITY as SSP Problem). *The interdiction version of SAT is a tuple $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$ with input $I = (L, C, B, t) \in \mathcal{I}'$ of literals and clauses, universe $\mathcal{U}'(I) = L$, solution set \mathcal{S} , blockable set B and threshold t . The solution set is given by $\mathcal{S}'(I) = \{B' \subseteq B : |B'| \leq t \text{ and for all } S \in \mathcal{S}(I) \text{ we have } B' \cap S \neq \emptyset\}$.*

Now, we prove that the meta-reduction from Theorem 4.1 is also an SSP reduction by using the reduction from Theorem 4.1 and defining the corresponding function $(f'_{I'})_{I' \in \mathcal{I}'}$ to complete the SSP reduction.

Corollary 4.1. *For all SSP-NP-complete problems Π , COMB. INTERDICTION-SAT \leq_{SSP} COMB. INTERDICTION-II.*

Proof. As implied by Observation 4.1, COMB. INTERDICTION-SAT and COMB. INTERDICTION-II are also an SSP problems. We extend the SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ to an SSP reduction $(g', (f'_{I'})_{I' \in \mathcal{I}'})$ from COMB. INTERDICTION-SAT to COMB. INTERDICTION-II as depicted in Figure 4.2, where g' is the reduction from the proof of Theorem 4.1.

$$\begin{array}{ccc} \text{SAT} & \xrightarrow{(g, (f_I)_{I \in \mathcal{I}})} & \Pi \\ \downarrow I' = (I, B, t) & & \downarrow \\ \text{COMB. INTERDICTION-SAT} & \xrightarrow{(g', (f'_{I'})_{I' \in \mathcal{I}'})} & \text{COMB. INTERDICTION-II} \end{array}$$

Figure 4.2: SATISFIABILITY \leq_{SSP} Π implies COMB. INTERDICTION-SAT \leq_{SSP} COMB. INTERDICTION-II.

We recall the reduction from the proof of Theorem 4.1. As we have already proven the correctness of g' , we focus solely on the function $(f'_{I'})_{I' \in \mathcal{I}'}$. We define $f_I = f'_{I'}$ such that B_{new} is the new blockable set in $\text{COMB. INTERDICTION-II}$ defined as

$$B_{\text{new}} = f'_{I'}(B) = \{u \in \mathcal{U}(g(I)) \mid u \in f'_{I'}(B)\} = \{u \in \mathcal{U}(g(I)) \mid u \in f_I(B)\}.$$

Because the universe for SAT and $\text{COMB. INTERDICTION-SAT}$ is the same, $f'_{I'}$ is well-defined. Consequently, we have $f_I(B) = f'_{I'}(B) \subseteq \mathcal{U}(g(I))$. As was stated in the proof of Theorem 4.1, the blockers correspond one-to-one to each other on the set $f_I(B)$. Hence, this reduction is indeed solution preserving. \square

Chapter 5

Min-Max Regret Optimization

5.1 Introduction

In the area of *min-max regret robust optimization*, one is faced with an uncertain cost scenario. The decision maker seeks to minimize the regret that he is experiencing after the actual costs have been realized. In other words, the decision maker wants to minimize the maximum deviation between incurred cost and optimal cost. This problem is motivated by risk aversion strategies and human behavior to evade negative emotions such as regret. Accordingly, critical events of high costs can sufficiently be modeled under this regime since a decision maker is explicitly encouraged to minimize the impact of such events. An example of such events is contamination of water supplies, which poses an inherent danger to the people of a municipality. One question in this regard is how to place a given number of contamination sensors to install a robust warning system for the water supply of a municipality, such that the impact of critical events is minimized [WHM06, CGH⁺06]. Furthermore, in decision theory and computational social choice, this concept is of interest in analyzing voting behavior. A public decision maker (e.g., a politician or a manager of a public stock company) can be evaluated retrospectively. These decision makers have to justify their actions to the public, and these actions are typically retrospectively evaluated by the public when the uncertain costs have been realized, and thus, the actual scenario is clear. Accordingly, the decision maker may choose a strategy that causes minimum regret based on the information that is available at the point in time at which the decision has to be made. Vice versa, the voters may vote based on a minimum regret strategy given the possible candidates, too [MG75, BYFL95, DL15].

The min-max regret criterion is popular in robust optimization and has been considered for many standard optimization problems. Among the problems, where the nominal problem is contained in \mathcal{P} and is analyzed from a complexity viewpoint, are, for example shortest path [AL04], minimum spanning tree [AL04], min cut [ABV08], min s - t cut [ABV08], 1-center and 1-median with uncertainty in the node or edge weights [AB00, Ave03], and scheduling variants [LA06, Con14]. Among these, some interval min-max regret problems are shown to be in \mathcal{P} , such as min cut, 1-center, and 1-median. The rest of the problems are shown to be NP-hard. In the realm of interval min-max regret problems, for which the nominal problem is NP-complete, there are publications considering the complexity of the knapsack problem [DW10], the set cover problem [CSN22], and the TSP and Steiner tree problem [GMP23]. Further but more general publications on this topic are [KY13, KZ16]. A general survey is provided by Aissi, Bazgan, and Vanderpooten [ABV09].

Our Results. In this chapter, we are concerned with the *min-max regret* robust optimization

problem with interval uncertainty. Faced with an uncertain cost function, the goal is to minimize the maximum deviation between incurred cost and optimal cost. The uncertain cost is modeled by assuming that each cost coefficient stems from a pre-specified interval. Our main result in this chapter is that for every problem $\Pi \in \text{SSP-NPc}$, the corresponding min-max regret robust optimization problem with interval uncertainty is Σ_2^p -complete. Concretely, we define a min-max regret problem with interval uncertainty by embedding it into the SSP framework. Based on this, we provide a general reduction for these min-max regret problems for the following nominal problems:

Sat, 3Sat, vertex cover, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median, independent set, clique, subset sum, knapsack, partition, scheduling, Hamiltonian path/cycle (directed/undirected), TSP, k -directed disjoint path ($k \geq 2$), and Steiner tree.

Related Work. Usually in the literature, the complexity of min-max regret problems is not discussed beyond NP-hardness in the context of the polynomial hierarchy. However, there are two exceptions, which we generalize by introducing the SSP framework and applying it to min-max regret optimization. The first paper by Deineko and Woeginger [DW10] shows the Σ_2^p -completeness of min-max regret knapsack. The second paper by Coco, Santos, and Noronha [CSN22] establishes Σ_2^p -completeness of min-max regret maximum benefit set cover.

5.2 Min-Max Regret Problems with Interval Uncertainty

We consider min-max regret robust optimization problems with interval uncertainty. Our main result in this chapter is that for every single LOP problem with the property that the SSP problem derived from it is in SSP-NPc (compare Section 3.2) by a so-called *tight* reduction – we define this term in Section 5.5 –, the corresponding min-max regret problem is Σ_2^p -complete. This means that the problem is described by $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$, where for every instance $I \in \mathcal{I}$, $\mathcal{U}(I)$ denotes its universe, $\mathcal{F}(I)$ denotes its feasible solutions, $d^{(I)} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ denotes its cost function, and $t^{(I)}$ denotes its threshold. Like in Definition 3.1, we define $\mathcal{S}(I) = \{S \in \mathcal{F}(I) : d^{(I)}(S) \leq t^{(I)}\}$. As an example, if $\Pi = \text{VERTEX COVER}$, then $\mathcal{F}(I)$ contains all vertex covers, but $\mathcal{S}(I)$ only contains those vertex covers of size at most the threshold $t^{(I)}$.

In order to define the min-max regret version of Π , we use the following definitions, which are standard in the area of min-max regret robust optimization [DW10]: For some cost function $c : \mathcal{U}(I) \rightarrow \mathbb{Z}$, we define $S_c^* \in \arg \min_{S' \in \mathcal{F}(I)} c(S')$ to be an optimal feasible solution of Π with respect to cost function c . The *regret* of some feasible solution $S \in \mathcal{F}(I)$ is defined as $\text{reg}(S, c) := c(S) - c(S_c^*)$. Given numbers $\underline{c}_e, \bar{c}_e \in \mathbb{Z}$ for every universe element $e \in \mathcal{U}(I)$ such that $\underline{c}_e \leq \bar{c}_e$, the *interval uncertainty set* defined by the coefficient sequence $(\underline{c}_e, \bar{c}_e)_{e \in \mathcal{U}(I)}$ is defined as

$$C := \{c \mid c : \mathcal{U}(I) \rightarrow \mathbb{R} \text{ is a function s.t. } c(e) \in [\underline{c}_e, \bar{c}_e] \forall e \in \mathcal{U}(I)\}.$$

(We remark that we are using the letter C for the uncertainty set instead of the more standard letter \mathcal{U} , since we are using \mathcal{U} already to denote the universe of the nominal problem.)

Definition 5.1 (Min-Max Regret Problem with Interval Uncertainty). *Let an LOP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be given. The min-max regret problem with interval uncertainty associated to Π is denoted by INTERVAL MIN-MAX REGRET- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with integers $\underline{c}_e \leq \bar{c}_e$ for all $e \in \mathcal{U}(I)$ and a threshold $t_R \in \mathbb{Z}$. The question is whether*

$$\min_{S \in \mathcal{F}(I)} \max_{c \in C} \text{reg}(S, c) \leq t_R.$$

We remark that this definition only applies to LOP problems, i.e. not to every problem in Chapter 9. This is simply due to the fact that it makes use of the concept of the feasible solutions $\mathcal{F}(I)$, and not every SSP problem has a set $\mathcal{F}(I)$. However, in the next subsection, we introduce a slight variant of the min-max regret problem which is defined for every SSP problem and also show Σ_2^p -completeness for that variant.

Before we start, we define the *maximum regret* $r_{\max}(S) := \max_{c \in C} \text{reg}(S, c)$. For a fixed feasible set $S \in \mathcal{F}(I)$, it is a well-known fact [DW10] that the maximum regret is caused by the so-called *canonical cost scenario* $c_S : \mathcal{U}(I) \rightarrow \mathbb{Z}$, defined by

$$c_S(e) := \begin{cases} \bar{c}_e & \text{if } e \in S \\ \underline{c}_e & \text{if } e \notin S. \end{cases}$$

This implies that

$$r_{\max}(S) = \text{reg}(S, c_S) = c_S(S) - c_S(S_{c_S}^*) = c_S(S) - \min_{S' \in \mathcal{F}(I)} c_S(S').$$

With this observation on the canonical cost scenario, we can view interval min-max regret problems as a two-player game of an \exists -player that wants to find a feasible solution against an adversary (the \forall -player). The \exists -player is in control of the *min* and thus is able to choose the solution S . On the other hand, the adversary is in control of the *max* and is thus able to choose the cost function c from the set C as well as the solution S' . Consequently, we can reformulate the question to

$$\exists S \in \mathcal{F}(I) : \forall S' \in \mathcal{F}(I) : c_S(S) - c_S(S') \leq t_R.$$

With this perspective, it is easy to show the containment in the class Σ_2^p .

Lemma 5.1. *If $\Pi = (\mathcal{I}, \mathcal{F}, \mathcal{U}, d, t)$ is an LOP problem such that the derived SSP problem is in SSP-NP, then INTERVAL MIN-MAX REGRET- Π is contained in the class Σ_2^p .*

Proof. We provide a polynomial time algorithm V such that for $m_1, m_2 \leq \text{poly}(|I|)$:

$$I \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} : V(I, y_1, y_2) = 1.$$

Observe that for INTERVAL MIN-MAX REGRET- Π we can characterize the YES-instances as follows:

$$I \in L \Leftrightarrow \exists S \in \mathcal{F}(I) \forall S' \in \mathcal{F}(I) : c_S(S) - c_S(S') \leq t_R.$$

Therefore with the \exists -quantified y_1 , we encode the solution $S \subseteq \mathcal{U}(I)$ with $S \in \mathcal{F}(I)$. Because $|\mathcal{U}(I)| \leq \text{poly}(|I|)$, the encoding size of y_1 is polynomially bounded in the input size of Π . Next, we encode all solutions $S' \in \mathcal{F}(I)$ to the nominal problem Π using the \forall -quantified y_2 within polynomial space as well. At last, the verifier V has to verify the correctness of the given solution provided by the \exists -quantified y_1 and \forall -quantified y_2 . Since we have assumed that we can efficiently check (for Π in NP) whether proposed solutions $F \subseteq \mathcal{U}(I)$ are indeed feasible solutions, it can be checked in polynomial time whether S and $S' \in \mathcal{F}(I)$. Furthermore, the canonical cost scenario c_S can be efficiently computed from S . Consequently, it can be checked in polynomial time whether $c_S(S) - c_S(S') \leq t_R$. It follows that INTERDICTION- Π is in Σ_2^p . \square

5.3 Simplifying the Structure: Restricted Interval Min-Max Regret Problems

In order to show the meta-theorem, we define a restricted variant of interval min-max regret problems. In contrast to the non-restricted variant, which can only be defined for LOP problems

(since the definition relies on the concept of $\mathcal{F}(I)$), the restricted version can be defined more generally for every SSP. We call these problems RESTRICTED INTERVAL MIN-MAX REGRET-II for a nominal SSP problem Π .

Definition 5.2 (Restricted Interval Min-Max Regret Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The restricted regret problem associated to Π is denoted by RESTRICTED INTERVAL MIN-MAX REGRET-II and defined as follows: The input is an instance $I \in \mathcal{I}$ together with integers $\underline{h}_e, \bar{h}_e$ for all $e \in \mathcal{U}(I)$, such that $\underline{h}_e \leq \bar{h}_e$ and both $\underline{h}_e, \bar{h}_e \in \{0, 1\}$, and a threshold $q \in \mathbb{Z}$. The question is whether*

$$\min_{S \in \mathcal{S}(I)} \max_{h \in H} \left(h(S) - \min_{S' \in \mathcal{S}(I)} h(S') \right) \leq q.$$

Here H denotes the interval uncertainty set defined by the coefficients $\underline{h}_e, \bar{h}_e$. We assume that the input is a YES-instance ($\mathcal{S}(I) \neq \emptyset$), otherwise the (restricted) min-max regret is undefined.

Observe that the restricted interval min-max regret is defined over the solutions $\mathcal{S}(I)$ instead of the feasible solutions $\mathcal{F}(I)$ as for the standard interval min-max regret problems based on an LOP problem. Furthermore, the uncertainty set is restricted by $\underline{h}_e, \bar{h}_e$ to be from the set $\{0, 1\}$. (We remark that while it is not standard to calculate the regret relative to $\mathcal{S}(I)$ instead of $\mathcal{F}(I)$, this has been considered recently by Coco, Santos and Noronha for the case of min-max regret for the set cover problem [CSN22].)

We begin with the canonical SAT variant RESTRICTED INTERVAL MIN-MAX REGRET-SAT and show that it is Σ_2^P -complete. Then, using this SAT problem as a basis for a reduction, we explain how this can be used to show that RESTRICTED INTERVAL MIN-MAX REGRET-II is Σ_2^P -hard. In a similar spirit to previous chapters of this part, we consider L as the universe of the problem and we define the set of solutions

$$\mathcal{S}(\varphi) := \{L' \subseteq L : |L' \cap \{x_i, \bar{x}_i\}| = 1 \ \forall i = 1, \dots, n; |L' \cap c_j| \geq 1 \ \forall j = 1, \dots, m\}$$

as the subset of all literals encoding a satisfying assignment. Applying Definition 5.2 to the satisfiability problem yields the following.

Definition 5.3 (RESTRICTED INTERVAL MIN-MAX REGRET-SAT). *We denote the restricted interval min-max regret version of SAT by RESTRICTED INTERVAL MIN-MAX REGRET-SAT. The input is a CNF with clauses C and literals L , together with a threshold parameter $q \in \mathbb{Z}$ and integers $\underline{h}_\ell, \bar{h}_\ell$ for all $\ell \in L$, such that $\underline{h}_\ell \leq \bar{h}_\ell$ and both $\underline{h}_\ell, \bar{h}_\ell \in \{0, 1\}$ for all $\ell \in L$. Let H be the interval uncertainty set defined by the $\underline{h}_\ell, \bar{h}_\ell$. The question is whether*

$$\min_{S \in \mathcal{S}(\varphi)} \max_{h \in H} \left(h(S) - \min_{S' \in \mathcal{S}(\varphi)} h(S') \right) \leq q.$$

We assume that the input formula φ is satisfiable ($\mathcal{S}(\varphi) \neq \emptyset$), otherwise the (restricted) min-max regret is undefined.

As next step, we show the Σ_2^P -completeness of RESTRICTED INTERVAL MIN-MAX REGRET-SAT.

Lemma 5.2. *RESTRICTED INTERVAL MIN-MAX REGRET-SAT is Σ_2^P -complete.*

Proof. The containment in the class Σ_2^P is analogous to Lemma 5.1. It remains to show hardness. For the hardness, we reduce from $\exists \forall$ DNF-SAT, which is Σ_2^P -hard as shown by Stockmeyer [Sto76].

Definition of the instance Assume we are given an instance $\exists X \forall Y \psi(X, Y)$ of $\exists \forall$ DNF-SAT, where ψ is in DNF. We show how to construct an equivalent instance of RESTRICTED INTERVAL MIN-MAX REGRET-SAT by describing $\varphi, \underline{h}_\ell, \bar{h}_\ell, t$ as in Definition 5.3 (where φ is a CNF-formula and ℓ runs over the literals of φ).

We start by letting $\psi' := \neg\psi$. Note that because ψ is in DNF, ψ' is in CNF by applying De Morgan's rule. The variables and literals are the same between ψ and ψ' . Observe that

$$\exists X \forall Y \psi(X, Y) \Leftrightarrow \exists X \neg \exists Y \psi'(X, Y).$$

We define a new CNF formula φ , by introducing a new variable z and appending it to every clause of ψ' , that is $\varphi \equiv \psi' \vee z$. Note that φ is satisfiable ($\mathcal{S}(\varphi) \neq \emptyset$), since any assignment α with $\alpha(z) = 1$ is satisfying. Let $L_X := X \cup \bar{X}$ be the literal set belonging to the variables X and $L_Y := Y \cup \bar{Y}$ be the literal set belonging to the variables Y . The formula φ has literal set $L := L_X \cup L_Y \cup \{z, \bar{z}\}$. We define the uncertainty set H , by making a case distinction. For a literal $\ell \in L_X \cup \{z, \bar{z}\}$, we let $\underline{h}_\ell = 0$ and $\bar{h}_\ell = 1$. For a literal $\ell \in L_Y$, we let $\underline{h}_\ell = \bar{h}_\ell = 0$. Finally we let $q := |X| = |L_X|/2$. This completes the description of the RESTRICTED INTERVAL MIN-MAX REGRET-SAT instance (φ, H, q) .

Correctness We claim that the regret for (φ, H, q) is bounded from above by q if and only if $\exists X \neg \exists Y \psi'(X, Y)$ is satisfiable. The main idea behind this claim is that the min-max regret problem can be understood as a game between Alice and Bob: Alice selects an initial solution $S \in \mathcal{S}(\varphi)$, Bob selects the cost function $h \in H$ together with some other solution $S' \in \mathcal{S}(\varphi)$. Bob's goal is to cause a large regret for Alice (at least $q + 1$). By construction, such a large regret is only possible, if S' and S take exactly opposite literals on the set $L_X \cup \{z, \bar{z}\}$. But this means Alice can "block" Bob, if she chooses her assignment on X in a smart way.

We are now ready to prove the claim. First, assume that the formula $\exists \alpha_1 \neg \exists \alpha_2 \psi'(\alpha_1, \alpha_2)$ is true, where $\alpha_1 : X \rightarrow \{0, 1\}$ and $\alpha_2 : Y \rightarrow \{0, 1\}$ are truth assignments. We let A_1 be the literals corresponding to α_1 (i.e. if $\alpha_1(x) = 1$ then $x \in A_1$, else $\bar{x} \in A_1$). Then Alice chooses a solution S the following way: From $\{z, \bar{z}\}$, she chooses the literal z . From L_X , she chooses exactly the literals \bar{A}_1 opposite to A_1 . From L_Y , she chooses an arbitrary literal for each variable $y \in Y$. Observe that the subset $S \subseteq L$ described this way encodes a satisfying assignment of φ , since $z \in S$. Furthermore, observe that the canonical cost scenario causing the maximum regret to Alice is given by

$$h_S(\ell) = \begin{cases} 1 & \text{if } \ell \in (L_X \cup \{z, \bar{z}\}) \cap S \\ 0 & \text{if } \ell \in (L_X \cup \{z, \bar{z}\}) \setminus S \\ 0 & \text{if } \ell \in L_Y. \end{cases}$$

The properties of h_S imply the following: The only way for Bob to cause a regret of $q + 1 = |X| + 1$ is if $S' \supseteq A_1 \cup \{\bar{z}\}$ (i.e. if he makes the exact opposite choice of Alice on the set $L_X \cup \{z, \bar{z}\}$). But this is impossible: since $\neg \exists \alpha_2 \psi'(\alpha_1, \alpha_2)$, Bob can never complete such a choice to a satisfying assignment $S' \in \mathcal{S}(\varphi)$. This means that the maximum regret is at most q .

On the other hand, assume that the regret is at most q . In this case, Alice must have chosen z , (that is, $z \in S$), because otherwise

$$\min_{S \in \mathcal{S}(\varphi)} \max_{h \in H} \left(h(S) - \min_{S' \in \mathcal{S}(\varphi)} h(S') \right) = q + 1.$$

This is because if Alice did not choose z , then Bob can choose z and exactly the opposite of Alice's choices on the set L_X and an arbitrary assignment of Y . (This choice by Bob satisfies the formula, since every assignment with $z = 1$ trivially satisfies the formula.) Furthermore, it causes a regret of $q + 1$ since on the set $L_X \cup \{z, \bar{z}\}$ it is exactly the opposite of Alice's choice. Hence we conclude that Alice must have chosen z . But then, by a similar argument to the above, we have that $\exists\alpha_1 \neg \exists\alpha_2 \psi'(\alpha_1, \alpha_2)$.

We have successfully proven the claim. In conclusion, we have that the regret is bounded by q if and only if $\exists X \forall Y \psi(X, Y)$ is satisfiable.

Polynomial Time Moreover, the reduction is polynomial-time computable because all transformations of the formula ψ as well as the computation of the numbers $\underline{h}_\ell, \bar{h}_\ell$ are polynomial-time computable.

In conclusion, we have shown that RESTRICTED INTERVAL MIN-MAX REGRET-SAT is Σ_2^P -complete. \square

5.4 A Meta-Reduction from Restricted Interval Min-Max Regret Problems

Having established the hardness of our base problem RESTRICTED INTERVAL MIN-MAX REGRET-SAT, we are now ready to prove our meta-theorem, i.e. we prove that for every SSP problem Π , which is also SSP-NP-complete, its corresponding restricted interval min-max regret version is Σ_2^P -complete.

Theorem 5.1. *For all SSP-NP-complete problems Π , the restricted interval min-max regret variant RESTRICTED INTERVAL MIN-MAX REGRET- Π is Σ_2^P -complete.*

Proof. The containment in the class Σ_2^P is analogous to Lemma 5.1. Let an abstract, SSP-NP-complete problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. SSP-NP-completeness implies that SATISFIABILITY $\leq_{\text{SSP}} \Pi$ (where SATISFIABILITY is defined like in Section 3.2). By the definition of an SSP reduction, there exists a poly-time computable function g which maps CNF-formulas φ to equivalent instances $g(\varphi) \in \mathcal{I}$ of Π . Let $I := g(\varphi)$. Furthermore there exist poly-time computable functions $f_\varphi : \mathcal{U}(\varphi) \rightarrow \mathcal{U}(I)$ with the SSP property. Here, $\mathcal{U}(\varphi)$ is the literal set of φ , and $\mathcal{U}(I)$ is the universe of the instance $I = g(\varphi)$ of Π . In order to reduce the notation, we write f instead of f_φ . The SSP property states that

$$\{f(S) : S \in \mathcal{S}(\varphi)\} = \{S' \cap f(\mathcal{U}(\varphi)) : S' \in \mathcal{S}(I)\}$$

where $\mathcal{S}(\varphi)$ is the set of all subsets of literals which satisfy φ .

The above is a reduction from SATISFIABILITY to Π . We now show that we can “upgrade” the reduction to obtain a reduction from RESTRICTED INTERVAL MIN-MAX REGRET-SAT to RESTRICTED INTERVAL MIN-MAX REGRET- Π . This upgraded reduction is described the following way:

Assume we are given an instance $(\varphi, (\underline{h}_\ell, \bar{h}_\ell)_{\ell \in L}, q)$ of RESTRICTED INTERVAL MIN-MAX REGRET-SAT with the properties as described in Definition 5.3. We define a RESTRICTED INTERVAL MIN-MAX REGRET- Π instance $(I, t_R, (\underline{h}_e, \bar{h}_e)_{e \in \mathcal{U}(I)})$ in the following way:

- ▶ The instance is given by $I = g(\varphi)$. (Note this can be computed in polynomial time).
- ▶ Let $n := |\mathcal{U}(I)|$. The numbers $\underline{h}_e, \bar{h}_e$ for $e \in \mathcal{U}(I)$ are defined by a case distinction. If $e \in f(\mathcal{U}(\varphi))$, then we let $\ell := f^{-1}(e)$ and

$$\underline{h}_e := \underline{h}_\ell, \quad \text{and} \quad \bar{h}_e := \bar{h}_\ell.$$

In the other case, $e \notin f(\mathcal{U}(\varphi))$, we let $\underline{h}_e = \bar{h}_e := 0$. We define the resulting interval uncertainty set by \tilde{H} .

- The threshold t_R is given by $t_R := q$.

This completes our description of the instance of RESTRICTED INTERVAL MIN-MAX REGRET-II. Note that φ is a satisfiable formula by Definition 5.3, hence $g(\varphi) = I \in \mathcal{I}$ is a YES-instance of Π and $\mathcal{S}(I) \neq \emptyset$. Furthermore, the transformation is computable in polynomial time because g and $f(\mathcal{U})$ are polynomial-time computable.

At last, we prove that for the two instances of RESTRICTED INTERVAL MIN-MAX REGRET SAT and RESTRICTED INTERVAL MIN-MAX REGRET-II as defined above, it holds that their min-max regrets are equal. Let $W := f(\mathcal{U}(\varphi))$ and consider the following chain of equalities:

$$\begin{aligned}
& \text{min-max-regret}(I, (\underline{h}_e, \bar{h}_e)_{e \in \mathcal{U}(I)}) \\
&= \min_{S \in \mathcal{S}(I)} \max_{\tilde{h} \in \tilde{H}} \left(\tilde{h}(S) - \min_{S' \in \mathcal{S}(I)} \tilde{h}(S') \right) \\
&= \min_{S \in \mathcal{S}(I)} \max_{\tilde{h} \in \tilde{H}} \left(\tilde{h}(S \cap W) - \min_{S' \in \mathcal{S}(I)} \tilde{h}(S' \cap W) \right) \\
&= \min_{S \in \mathcal{S}(I)} \max_{h \in H} \left(h(f^{-1}(S \cap W)) - \min_{S' \in \mathcal{S}(I)} h(f^{-1}(S' \cap W)) \right) \\
&= \min_{T \in \mathcal{S}(\varphi)} \max_{h \in H} \left(h(T) - \min_{T' \in \mathcal{S}(\varphi)} h(T') \right) \\
&= \text{min-max-regret}(\varphi, (\underline{h}_\ell, \bar{h}_\ell)_{\ell \in L})
\end{aligned}$$

The first equality is by definition. The second equality follows from the fact that \tilde{h} is only non-zero on the set W . The third equality follows since the uncertainty set \tilde{H} behaves on W analogous to H on $f^{-1}(W) = L$. The fourth equality is due to the SSP property. To see this, apply f^{-1} to both sides of the equation $\{f(S) : S \in \mathcal{S}(\varphi)\} = \{S' \cap f(\mathcal{U}(\varphi)) : S' \in \mathcal{S}(I)\}$. The last equality is again by definition.

We remark that the crucial step in the proof of the above lemma is the usage of the SSP property. In summary, we have described a polynomial time reduction from the Σ_2^P -complete problem RESTRICTED INTERVAL MIN-MAX REGRET SAT to RESTRICTED INTERVAL MIN-MAX REGRET-II, such that the objective value of both problems stays exactly the same. This proves that RESTRICTED INTERVAL MIN-MAX REGRET-II is Σ_2^P -hard. Together with the containment, as argued above, the problem is Σ_2^P -complete. \square

5.5 Adapting the Meta-Reduction to the Interval Min-Max Regret Version

For the completion of our main result, we have to show that the result on the restricted interval min-max regret problem is also applicable to the non-restricted interval min-max regret problem. We do this by re-adapting the cost function of the restricted version back to the standard version based on the LOP problem. Furthermore, we have to prove that the regret is not affected by this adaptation. The underlying instance, however, stays the same within the reduction.

In order not to affect the regret, we use an additional property on the SSP reduction from the SSP problem Π_1 to the (SSP version derived from) LOP problem Π_2 . Specifically, the reduction needs to include only optimal solutions in the solution set. Then, we can split the costs of each element into a dominant part, the original costs d from the LOP problem, and a subordinate part,

the costs based on the restricted interval min-max regret problem h . Thus, all possible solutions that are considered for the regret are minimizers for d , i.e., optimal solutions for problem Π_2 . Accordingly, we are able to subtract the costs for d and obtain the regret solely based on the costs h . We call these reductions *tight*.

Definition 5.4. Let Π_1 be an SSP problem and $\Pi_2 = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be an LOP problem. Consider an SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ from Π_1 to (the SSP problem derived from) Π_2 . The reduction is called *tight* if for all yes-instances I_1 of Π_1 , the corresponding instance $I_2 = g(I_1)$ of Π_2 with the associated parameter $t := t^{(I_2)}$ and associated cost function $d := d^{(I_2)}$, the following holds:

$$\{F \in \mathcal{F}(I_2) : d(F) \leq t\} \neq \emptyset \text{ and } \{F \in \mathcal{F}(I_2) : d(F) \leq t - 1\} = \emptyset$$

All SSP reductions (to SSP problems derived from LOP problems) that can be found in Chapter 9 fulfill this definition and are thus tight.

Theorem 5.2. For all LOP problems Π with the property that the SSP problem derived from them is contained in SSP-NPc by a tight SSP reduction, the interval min-max regret variant INTERVAL MIN-MAX REGRET-II is Σ_2^p -complete.

Proof. The containment in the class Σ_2^p follows from Lemma 5.1. Let an abstract, NP-complete LOP problem $\Pi = (\mathcal{I}, \mathcal{F}, \mathcal{U}, d, t)$ be given. The corresponding SSP problem is defined by $\Pi' = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ with

$$\mathcal{S} = \{f \in \mathcal{F}(I) \mid d^{(I)}(f) \leq t^{(I)}\}.$$

Assume we are given an instance $(I, q, (\underline{h}_e, \bar{h}_e)_{e \in \mathcal{U}(I)})$ of RESTRICTED INTERVAL MIN-MAX REGRET-II' with the properties as described in Definition 5.2. We transform this instance to an instance $(I, t_R, (\underline{c}_e, \bar{c}_e)_{e \in \mathcal{U}(I)})$ of INTERVAL MIN-MAX REGRET-II in the following way:

- ▶ The underlying instance I , as well as the corresponding universe $\mathcal{U}(I)$ remain the same.
- ▶ Let $n := |\mathcal{U}(I)|$. We define the numbers $\underline{c}_e, \bar{c}_e$ for all $e \in \mathcal{U}(I)$ by

$$\underline{c}_e := 2(n+1)d^{(I)}(e) + \underline{h}_e, \quad \text{and} \quad \bar{c}_e := 2(n+1)d^{(I)}(e) + \bar{h}_e.$$

We define the resulting interval uncertainty set by C .

- ▶ The threshold t_R is given by $t_R := q$.

This completes our description of the instance of INTERVAL MIN-MAX REGRET-II. (Note that t_R and $t^{(I)}$ are different thresholds). The transformation is computable in polynomial time, because the underlying instance is unchanged and only numbers encoded in polynomial size (Π is in NP) are added to \underline{h}_e and \bar{h}_e , resulting in \underline{c}_e and \bar{c}_e .

Again, we prove that for the two instances of RESTRICTED INTERVAL MIN-MAX REGRET-II and INTERVAL MIN-MAX REGRET-II as defined above, it holds that their min-max regrets are equal. For this, consider the following chain of equalities:

$$\begin{aligned} \text{min-max-regret}(I, (\underline{c}_e, \bar{c}_e)_e) &= \min_{S \in \mathcal{F}(I)} \max_{c \in C} \left(c(S) - \min_{S' \in \mathcal{F}(I)} c(S') \right) \\ &= \min_{S \in \mathcal{S}(I')} \max_{c \in C} \left(c(S) - \min_{S' \in \mathcal{S}(I')} c(S') \right) \\ &= \min_{S \in \mathcal{S}(I')} \max_{h \in H} \left(h(S) - \min_{S' \in \mathcal{S}(I')} h(S') \right) \\ &= \text{min-max-regret}(I', (\underline{h}_e, \bar{h}_e)_e) \end{aligned}$$

The first equality is by definition. For the second equality, note that no matter which $c \in C$ is chosen, we always have $c(S') \in [2(n+1)d^{(I)}(S'), 2(n+1)d^{(I)}(S') + n]$ by definition of the coefficients $\underline{c}_e, \bar{c}_e$. Hence any S' minimizing $c(S')$ also minimizes $d^{(I)}(S')$. Hence, by the definition of $\mathcal{S}(I)$ and by the fact that $\mathcal{S}(I) \neq \emptyset$, we have that $S' \in \mathcal{S}(I)$. By a similar argument, if S optimizes the first minimum, but $S \notin \mathcal{S}(I)$, then $c(S) - \min_{S' \in \mathcal{F}(I)} c(S') \geq 2(n+1) - n = n+1$. But this is a contradiction, since this last expression is always at most n if both $S, S' \in \mathcal{S}(I)$ and the SSP reduction is tight.

The third equality follows from the fact that the function c can be decomposed into the part that is contributed by $d^{(I)}$ and the part contributed by h . Since both S, S' are minimizers of $d^{(I)}$, we have that $d^{(I)}(S) = d^{(I)}(S')$ and this part cancels out. The last equality is again by definition. \square

5.6 The Meta-Reduction is an SSP reduction

In Section 5.4, we showed that RESTRICTED INTERVAL MIN-MAX REGRET SAT reduces to RESTRICTED INTERVAL MIN-MAX REGRET-II. The goal of this subsection (analogous to Section 4.6) is to show the slightly stronger statement that this reduction is again an SSP reduction itself. In order to state this result, it becomes necessary to explain in which way RESTRICTED INTERVAL MIN-MAX REGRET-II is interpreted as SSP problem. This is done the following way.

Observation 5.1. *The restricted interval min-max regret variant RESTRICTED INTERVAL MIN-MAX REGRET-II of an SSP problem Π is an SSP problem.*

Proof. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem. We denote RESTRICTED INTERVAL MIN-MAX REGRET-II =: $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$. First, we set $\mathcal{U} = \mathcal{U}'$. Second, we can define the instance set \mathcal{I}' of the restricted min-max regret variant of Π by setting

$$\mathcal{I}' = \{(I, H, q) \mid I \in \mathcal{I}, H \text{ is an interval uncertainty set}, q \in \mathbb{Z}\}.$$

Furthermore, for an instance $I' = (I, H, q) \in \mathcal{I}'$, we define its solution set as

$$\mathcal{S}'(I') = \left\{ S \in \mathcal{S}(I) \mid \max_{c \in H} \left(c(S) - \min_{S' \in \mathcal{S}(I)} c(S') \right) \leq q \right\}.$$

Thus, RESTRICTED INTERVAL MIN-MAX REGRET-II = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$ is an SSP problem. \square

It remains to prove that the meta-reduction from Theorem 5.1, which we denote by g' , is also an SSP reduction by defining the corresponding function $((f'_{I'})_{I' \in \mathcal{I}'})$ to complete the SSP reduction.

Corollary 5.1. *For all SSP-NP-complete problems Π : RESTRICTED INTERVAL MIN-MAX REGRET-SAT \leq_{SSP} RESTRICTED INTERVAL MIN-MAX REGRET-II.*

Proof. The underlying instance of the reduction $(g, (f_I)_{I \in \mathcal{I}})$ from SAT to Π remains the same in the reduction of Theorem 5.1. Note that this reduction transforms instances I of SAT into instances $g(I)$ of Π . We now want to show that the reduction $(g', (f'_{I'})_{I' \in \mathcal{I}'})$ with some additional function $((f'_{I'})_{I' \in \mathcal{I}'})$ from RESTRICTED INTERVAL MIN-MAX REGRET-SAT to RESTRICTED INTERVAL MIN-MAX REGRET-II indeed is an SSP reduction. For this, we set $f'_{I'} := f_I$. Recall that if $e \notin f(\mathcal{U}(\varphi))$, the costs defined by the reduction are $\underline{h}_e = \bar{h}_e = 0$. Thus, no restriction is imposed by these elements on any solution. Now, we consider the elements that are part of

$f_I(\mathcal{U}(\varphi)) = f_{I'}(\mathcal{U}(\varphi))$. (Note that the instance remains the same.) If $e \in f(\mathcal{U}(\varphi))$, the costs are set to $\underline{h}_e = \underline{h}_\ell$ and $\bar{h}_e = \bar{h}_\ell$. The original reduction $\text{SAT} \leq_{\text{SSP}} \Pi$ has the solution preserving property. Hence the solution sets $\mathcal{S}(I)$ and $\mathcal{S}(g(I))$ correspond to each other. Furthermore, we have that $\mathcal{S}'(I')$ are exactly those elements of $\mathcal{S}(I)$ with regret at most q , and $\mathcal{S}'(g'(I'))$ are exactly those elements of $\mathcal{S}(g(I))$ with regret at most q . At last, the regret of a solution stays the same when transferring it from SAT to Π . Consequently, we conclude that the reduction $(g', (f_{I'})_{I' \in \mathcal{I}'})$ is an SSP reduction. \square

Chapter 6

Two-Stage Adjustable Robustness

6.1 Introduction

In the area of *two-stage adjustable robust optimization*, one is faced with an uncertain cost scenario. The decision-maker has to make a two-step decision, where in the first step a partial decision has to be made without full knowledge of the scenario, also called here-and-now decisions, and in a second step a partial corrective decision can be made with full knowledge of the scenario, also called wait-and-see decisions.

The applications of this concept are manifold. Whenever there are multiple stages, which can be divided into these here-and-now and wait-and-see decisions, we can apply adjustable robustness concepts to include uncertainty that occurs in between the stages. The added dynamic can also avoid the conservatism that traditional one-stage robust approaches induce. In one area, this concept is prevalent: planning and operating energy networks. Especially in the context of renewable energies, the increased volatility in the delivery of energy demands requires robust approaches [BLS⁺12, DLY⁺15, SSM⁺21]. Similar ideas are also used in various scheduling problems to model process optimization [LG16] and inter-city bus markets [YT09].

Furthermore, two-stage adjustable optimization has been considered for a large number of standard optimization problems, for example, in [KZ15, KZ16, KZ17, CGKZ18, GLW22b]. A general survey is provided by Yanikoğlu, Gorissen, and den Hertog [YGdH19].

Our Results. In this chapter, we are concerned with *two-stage adjustable robust optimization* with discrete budgeted uncertainty. Roughly speaking, the class of two-stage problems models problems that are divisible into two stages: The decision on the first stage has to be made without full information on the real instance (here-and-now), and the decision in the second stage is made after the uncertainty is revealed (wait-and-see). Two-stage adjustable problems can be formulated via min-max-min expressions. Therefore, the natural class for them is the class Σ_3^p . Consequently, in Section 6.2, we demonstrate how our main idea can also be used to show completeness for the third stage of the polynomial hierarchy. Our main result in this chapter is that for every problem $\Pi \in \text{SSP-NPc}$, the corresponding two-stage adjustable problem with discrete budgeted uncertainty is Σ_3^p -complete. Concretely, we define a two-stage adjustable problem with discrete budgeted uncertainty by embedding it into the SSP framework. Based on this, we provide a general reduction for these two-stage adjustable problems for the following nominal problems:

Sat, 3Sat, vertex cover, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median, independent

set, clique, subset sum, knapsack, partition, scheduling, Hamiltonian path/cycle (directed/undirected), TSP, k -directed disjoint path ($k \geq 2$), and Steiner tree.

These results also answer a question asked by Goerigk, Lendl, and Wulf [GLW24].

Related Work. Usually in the literature, the complexity of two-stage adjustable robust problems is not discussed beyond NP-hardness in the context of the polynomial hierarchy. The paper of Goerigk, Lendl, and Wulf [GLW24] is an exception: The authors prove the Σ_3^p -completeness of TSP, vertex cover, and independent set. We generalize these results by applying the SSP framework to two-stage adjustable optimization.

6.2 Two-Stage Adjustable Problems

We consider two-stage adjustable robust optimization problems with discrete budgeted uncertainty. We show that for every single LOP problem, whose derived SSP problem is SSP-NP-complete, that the corresponding two-stage adjustable problem is Σ_3^p -complete. We choose to make the following definition of a two-stage adjustable problem, where in the first stage an arbitrary set S_1 can be selected, but needs to be completed to a feasible set in the second stage (i.e. $S_1 \cup S_2 \in \mathcal{F}(I)$). We remark that the formal definition mentions the cost function d and the cost threshold t of the LOP. While these are not necessary for the definition (the function d and the threshold t get substituted with new ones), they are necessary for the proofs.

Definition 6.1 (Two-Stage Adjustable Problem). *Let an LOP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be given. The two-stage adjustable problem associated to Π is denoted by TWO-STAGE ADJUSTABLE- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with three cost functions: the first stage cost function $c_1 : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and the second stage cost functions $c_2 : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and $\bar{c}_2 : \mathcal{U}(I) \rightarrow \mathbb{Z}$, a threshold $t_{TS} \in \mathbb{Z}$ and an uncertainty parameter $\Gamma \in \mathbb{Z}$. Then, the uncertainty set is defined by all cost functions such that at most Γ elements $u \in \mathcal{U}(I)$ have costs of $\bar{c}_2(u)$, while all other have $c_2(u)$:*

$$C_\Gamma := \{c_2 \mid \forall u \in \mathcal{U}(I) : c_2(u) = c_2(u) + \delta_u(\bar{c}_2(u) - c_2(u)), \delta_u \in \{0, 1\}, \sum_{u \in \mathcal{U}(I)} \delta_u \leq \Gamma\}.$$

The question is whether

$$\min_{S_1 \subseteq \mathcal{U}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \subseteq \mathcal{U}(I) \setminus S_1 \\ S_1 \cup S_2 \in \mathcal{F}(I)}} c_1(S_1) + c_2(S_2) \leq t_{TS}.$$

We remark that this definition only applies to LOP problems, i.e. not to every problem in Chapter 9. This is simply due to the fact that it makes use of the concept of the feasible solutions $\mathcal{F}(I)$, and not every SSP problem has a set $\mathcal{F}(I)$. However, in the next subsection, we introduce a slight variant of the two-stage problem which is defined for every SSP problem and also show Σ_3^p -completeness for that variant.

Two-stage adjustable problems can be also viewed as a two-player game of an \exists -player that wants to find a feasible solution against an adversary (the \forall -player). The \exists -player is in control of both \min and thus is able to choose the partial solutions S_1 and S_2 . On the other hand, the adversary is in control of the \max and is thus able to choose the cost function c_2 from the set C_Γ . Consequently, we can reformulate the question to

$$\exists S_1 \subseteq \mathcal{U}(I) : \forall c_2 \in C_\Gamma : \exists S_2 \subseteq \mathcal{U}(I) \setminus S_1 : S_1 \cup S_2 \in \mathcal{F}(I) \text{ and } c_1(S_1) + c_2(S_2) \leq t_{TS}.$$

Again, we aim to locate the complexity of two-stage adjustable problems based on LOP problems exactly. By the game theoretical perspective, it is intuitive to see the containment in Σ_3^p for all problems that are polynomial-time checkable. Thus, we derive the following lemma and prove this intuition to be true.

Lemma 6.1. *If $\Pi = (\mathcal{I}, \mathcal{F}, \mathcal{U}, d, t)$ is an LOP problem such that the derived SSP problem is in SSP-NP, then TWO-STAGE- Π is in Σ_3^p .*

Proof. We provide a polynomial time algorithm that verifies a specific solution y_1, y_2, y_3 of polynomial size for instance I such that

$$I \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} \exists y_3 \in \{0, 1\}^{m_3} : V(I, y_1, y_2, y_3) = 1.$$

With the \exists -quantified y_1 , we encode the first partial solution to $S_1 \subseteq \mathcal{U}(I)$. Because $|\mathcal{U}(I)| \leq \text{poly}(|I|)$, the encoding size of y_1 is polynomially bounded in the input size of Π . Next, we encode all cost functions $c_2 \in C_\Gamma$ over the \forall -quantified y_2 . For this, we encode which of the at most Γ elements are chosen to have costs \bar{c}_2 . This is at most polynomial in the input size. At last, we encode the second partial solution $S_2 \subseteq \mathcal{U}(I)$ with the help of the \exists -quantified y_3 . Again, this is at most polynomial in the input size as before.

Now, V has to check whether $S_2 \subseteq \mathcal{U}(I) \setminus S_1$, $S_1 \cup S_2 \in \mathcal{F}(I)$ and $c_1(S_1) + c_2(S_2) \leq t_{TS}$. All of these checks can be done in polynomial time, where $S_1 \cup S_2 \in \mathcal{F}(I)$ and $c_1(S_1) + c_2(S_2) \leq t_{TS}$ can be checked in polynomial time because $\Pi \in \text{NP}$ and we have assumed that feasible solutions can be checked in polynomial time. It follows that TWO-STAGE ADJUSTABLE- Π is in Σ_3^p . \square

6.3 Combinatorial Two-Stage Adjustable Problems

To show the hardness of two-stage adjustable problems, we introduce a new problem which we call the *combinatorial version* of two-stage adjustable problems. In this problem, the cost function is substituted by a set of blockable elements. This combinatorial version is slightly more specific than the cost version and allows for a more precise reduction in the SSP framework. The hardness of the combinatorial version also implies the hardness of the cost version as we show later. For this, we redefine the cost functions into sets of elements: The set of first stage elements $U_1 \subseteq \mathcal{U}(I)$ and the set of second stage elements $U_2 = \mathcal{U}(I) \setminus U_1$. The set of blockable elements $B \subseteq U_2$ is a subset of the second stage elements.

We define the combinatorial version on basis of an SSP problem, which is no restriction because LOP problems are special SSP problems, which contain more structure. On the contrary, SSP reductions allow for a more succinct presentation of the hardness proof and its preliminaries.

Definition 6.2 (Combinatorial Two-Stage Adjustable Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The combinatorial two-stage adjustable problem associated to Π is denoted by COMB. TWO-STAGE ADJUSTABLE- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with three sets $U_1 \subseteq \mathcal{U}(I)$, $U_2 = \mathcal{U}(I) \setminus U_1$ and $B \subseteq U_2$, and an uncertainty parameter $\Gamma \in \mathbb{N}_0$. The question is whether*

$$\exists S_1 \subseteq U_1 : \forall B' \subseteq B \text{ with } |B'| \leq \Gamma : \exists S_2 \subseteq U_2 : S_1 \cup S_2 \in \mathcal{S}(I) \text{ and } S_2 \cap B' = \emptyset.$$

As the basis for our meta-reduction, we use the canonical SATISFIABILITY problem. Definition 6.2 applied to $\Pi = \text{SATISFIABILITY}$ yields the following:

Definition 6.3 (COMBINATORIAL TWO-STAGE ADJUSTABLE-SATISFIABILITY). *The problem COMB. TWO-STAGE ADJUSTABLE-SAT is defined as follows. The input is a CNF with clauses*

C and literal set L , sets $U_1 \subseteq L$, $U_2 = L \setminus U_1$ and $B \subseteq U_2$, and an uncertainty parameter $\Gamma \in \mathbb{N}_0$. Then the question is whether there is $S_1 \subseteq U_1$ such that for all $B' \subseteq B$ with $|B'| \leq \Gamma$, there is $S_2 \subseteq U_2$, such that $|(S_1 \cup S_2) \cap c_j| \geq 1$ for all $c_j \in C$, $|(S_1 \cup S_2) \cap \{x_i, \bar{x}_i\}| = 1$ for all pairs $x_i, \bar{x}_i \in L$, and $S_2 \cap B' = \emptyset$.

6.4 A Meta-Reduction for Combinatorial Two-Stage Adjustable Problems

Now, we show the Σ_3^p -hardness of the canonical SATISFIABILITY problem, COMB. TWO-STAGE ADJUSTABLE-SATISFIABILITY.

Lemma 6.2. *COMB. TWO-STAGE ADJUSTABLE-SATISFIABILITY is Σ_3^p -hard.*

Proof. The containment in the class Σ_3^p is analogous to Lemma 6.1. For the hardness, we reduce $\exists\forall\exists\text{CNF-SAT}$, which is Σ_3^p -hard as shown by Stockmeyer [Sto76], to COMB. TWO-STAGE ADJUSTABLE-SAT. Let $\exists X\forall Y\exists Z\varphi(X, Y, Z)$ be the $\exists\forall\exists\text{CNF-SAT}$ instance. We transform this into an instance $(\exists X'\psi(X'), U_1, U_2, B, \Gamma)$ of the combinatorial two-stage adjustable satisfiability problem, with the set of first stage elements U_1 , set of second stage elements U_2 , set of blockable elements B , and uncertainty parameter Γ .

Again, we use the idea of Goerigk, Lendl and Wulf [GLW24, Theorem 1] for their reduction for ROBUST ADJUSTABLE SAT to model the \forall -quantifier with Γ -uncertainty. The proof is very similar to the proof of Lemma 4.2. The difference this time is that the two-stage problem can be understood as a game between Alice and Bob, where Alice selects the literals from U_1 , Bob selects some blocker $B' \subseteq B$ with $|B'| \leq \Gamma$, and Alice responds by selecting literals from $U_2 \setminus B'$. Another difference is that we start with a CNF formula. Analogously to Lemma 4.2, for $n := |Y|$, we introduce new variables $Y^t = \{y_1^t, \dots, y_n^t\}$ and $Y^f = \{y_1^f, \dots, y_n^f\}$. We say that Bob plays honestly, if he chooses a blocker B' with $|B' \cap \{y_i^t, y_i^f\}| = 1$ for all $i = 1, \dots, n$. Otherwise we say that Bob cheats.

Description of the instance Let an instance $\exists X\forall Y\exists Z\varphi(X, Y, Z)$ of $\exists\forall\exists\text{CNF-SAT}$ be given. We define an instance of the two-stage adjustable satisfiability problem the following way: We introduce new variables Y^t, Y^f as explained above. A new formula φ' is created from φ in terms of a literal substitution process. For each $i \in \{1, \dots, n\}$, each occurrence of some literal y_i is substituted with the positive literal y_i^t . Each occurrence of some literal \bar{y}_i is substituted with the positive literal y_i^f . We introduce a new variable s and obtain a new formula φ'' by adding s to every clause of φ' , that is $\varphi'' \equiv \varphi' \vee s$. We introduce new variables s_1, \dots, s_n . we let $W := \{s\} \cup \{s_1, \dots, s_n\}$. The formula ψ is then defined by

$$\psi(X, Y^t, Y^f, Z, W) = \varphi''(X, Y^t, Y^f, Z, s) \wedge \left(\bigwedge_{i=1}^n (y_i^t \vee \bar{s}_i) \wedge (y_i^f \vee \bar{s}_i) \right) \wedge (\bar{s} \vee s_1 \vee s_2 \vee \dots \vee s_n).$$

Finally, we define L to be the literal set of ψ , i.e. the set of positive and negative literals corresponding to the variables X, Y^t, Y^f, Z, W . We let $U_1 := X \cup \bar{X}$, and $U_2 := L \setminus U_1$, and $B := Y^t \cup Y^f$, and $\Gamma = n$. This completes the description of the two-stage SAT instance.

Correctness Since the proof is very similar to Lemma 4.2, we only sketch the high-level argument.

Observe that in an optimal game, Bob can not cheat. On the other hand, if Bob plays honestly, Alice can not take any of the positive literals s, s_1, \dots, s_n into her second-stage solution S_2 . In other words, the cheat-detection gadget can be used if and only if Bob cheats.

First, assume that $\varphi(X, Y, Z)$ is a YES-instance of $\exists\forall\exists$ CNF-SAT. That is, the formula $\exists\alpha_X\forall\alpha_Y\exists\alpha_Z\varphi(\alpha_X, \alpha_Y, \alpha_Z)$ is true with the corresponding assignments $\alpha_X : X \rightarrow \{0, 1\}$, $\alpha_Y : Y \rightarrow \{0, 1\}$ and $\alpha_Z : Z \rightarrow \{0, 1\}$. We claim that Alice has a winning strategy for the two-stage SAT game. In fact, her winning strategy is as follows: In the first step, on the set U_1 , she chooses literals which correspond to α_X . Now there are two cases: If Bob cheats, Alice can win because of the cheat-detection gadget. If Bob plays honestly, Alice can win because $\forall\alpha_Y\exists\alpha_Z\varphi(\alpha_X, \alpha_Y, \alpha_Z)$.

Second, assume that $\varphi(X, Y, Z)$ is a NO-instance of $\exists\forall\exists$ CNF-SAT. Consider the first-stage choice $S_1 \subseteq U_1$. Note that Alice is forced to take a set $S_1 \subseteq U_1$ such that $|S_1 \cap \{x_i, \bar{x}_i\}| = 1$ for all i , because otherwise it is impossible that $S_1 \cup S_2 \in \mathcal{S}(\psi)$. Hence the choice S_1 of Alice corresponds to some assignment $\alpha_X : X \rightarrow \{0, 1\}$. Then, because φ is a NO-Instance, this means that with respect to this assignment α_X , we have $\exists\alpha_Y\forall\alpha_Z\neg\varphi(\alpha_X, \alpha_Y, \alpha_Z)$. This means that Bob can play honestly according to α_Y and have a winning strategy. In total, we have shown that $\exists\forall\exists$ CNF-SAT reduces to the COMB. TWO-STAGE ADJUSTABLE-SAT.

Polynomial Time The transformation is doable in polynomial time because for each variable a constant number of variables and clauses is introduced. □

Now, we are able to develop the meta-reduction for two-stage problems based on SSP-NP-complete problems Π . Precisely, we show a meta-reduction from COMBINATORIAL TWO-STAGE ADJUSTABLE-SATISFIABILITY to COMBINATORIAL TWO-STAGE ADJUSTABLE- Π .

Theorem 6.1. *For all SSP-NP-complete problems Π , the combinatorial two-stage adjustable variant COMB. TWO-STAGE ADJUSTABLE- Π is Σ_3^p -complete.*

Proof. The containment in the class Σ_3^p is analogous to Lemma 6.1. For the hardness, we again use that there is an SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ from SAT to Π because Π is an SSP-NP-complete problem. We design a reduction g' from COMB. TWO-STAGE ADJUSTABLE-SAT to COMB. TWO-STAGE ADJUSTABLE- Π by extending the existing reduction $(g, (f_I)_{I \in \mathcal{I}})$ as depicted in Figure 6.1. As in the reduction for interdiction problems, the underlying reduction from SAT to Π remains and we construct the corresponding combinatorial two-stage instance.

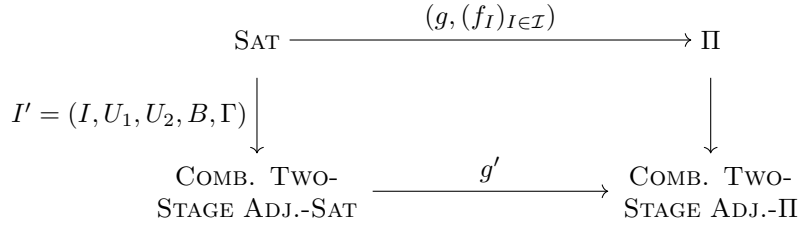


Figure 6.1: The fact that SAT is SSP reducible to Π induces a reduction from COMB. TWO-STAGE ADJUSTABLE-SAT to COMB. TWO-STAGE ADJUSTABLE- Π .

Let $I' = (I, U_1, U_2, B, \Gamma)$ be the instance of COMB. TWO-STAGE ADJUSTABLE-SAT with the corresponding SAT instance $I \in \mathcal{I}$. We denote the set of first stage elements by U_1 , the set of second stage elements by U_2 and the set of blockable elements by B . Furthermore, we denote the uncertainty parameter by Γ . The reduction g' is defined by $g'(I') = (g(I), U'_1, U'_2, B_{\text{new}}, \Gamma)$. The new sets are defined by

$$U'_1 = f_I(U_1), \quad U'_2 = \mathcal{U}(g(I)) \setminus U'_1, \quad B_{\text{new}} = f_I(B).$$

The uncertainty parameter Γ remains the same. This completes the description of the reduction. We claim that we have a direct one-to-one correspondence between the solutions of **COMB. TWO-STAGE ADJUSTABLE-SAT** and **COMB. TWO-STAGE ADJUSTABLE-II**. Indeed, this follows from the fact $B_{\text{new}} \subseteq f_I(\mathcal{U}(I))$ together with the SSP property of the SSP-reduction $\text{SAT} \leq \Pi$, which states that the solutions of **SAT** and Π correspond one-to-one to each other on the set $f_I(\mathcal{U}(I))$. Furthermore, the transformation is computable in polynomial time because g and f are polynomial-time computable. It follows that **COMB. TWO-STAGE ADJUSTABLE-SAT** \leq **COMB. TWO-STAGE ADJUSTABLE-II**. In particular, this implies that **COMB. TWO-STAGE ADJUSTABLE-II** is Σ_3^p -complete. \square

6.5 Adapting the Meta-Reduction to the Cost Version

As the last step of the meta-reduction, we have to show that the result on the combinatorial version of two-stage adjustable problems is also applicable to the cost version. This is indeed the case and we show this by reintroducing the cost function d and threshold t of the original nominal LOP problem.

Theorem 6.2. *For all LOP problems Π with the property that the SSP problem derived from them is contained in **SSP-NPc**, the two-stage adjustable variant **TWO-STAGE ADJUSTABLE-II** is Σ_3^p -complete.*

Proof. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be the LOP problem of consideration and **TWO-STAGE ADJUSTABLE-II** be the corresponding two-stage adjustable version. The containment of **TWO-STAGE ADJUSTABLE-II** in the complexity class Σ_3^p follows from Lemma 6.1.

Let $\Pi' = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be the SSP problem derived from the LOP problem Π . Recall that by the definition of the derived SSP problem, we have that Π and Π' share the same input instances and universe, and for $I \in \mathcal{I}$, we have $\mathcal{S}(I) = \{F \in \mathcal{F}(I) : d^{(I)}(F) \leq t^{(I)}\}$.

Let **COMB. TWO-STAGE ADJUSTABLE-II'** be the corresponding combinatorial two-stage adjustable version. By assumption on Π and by the previous section, **COMB. TWO-STAGE ADJUSTABLE-II'** is Σ_3^p -complete. The goal is now to reduce **COMB. TWO-STAGE ADJUSTABLE-II'** to **TWO-STAGE ADJUSTABLE-II**. This reduction is defined the following way: Let an instance (I, U_1, U_2, B, Γ) of **COMB. TWO-STAGE ADJUSTABLE-II'** be given. Recall that associated to the underlying instance I of Π there is the cost function $d^{(I)}$ and threshold $t^{(I)}$. We let the underlying instance I of Π remain the same and define an instance of **TWO-STAGE ADJUSTABLE-II** via the following tuple $(c_1, c_2, \bar{c}_2, \Gamma', t_{TS})$:

$$c_1(u) = \begin{cases} d^{(I)}(u), & u \in U_1 \\ t^{(I)} + 1, & u \in U_2 \end{cases}$$

$$c_2(u) = \begin{cases} t^{(I)} + 1, & u \in U_1 \\ d^{(I)}(u), & u \in U_2 \end{cases} \quad \bar{c}_2(u) = \begin{cases} t^{(I)} + 1, & u \in U_1 \\ t^{(I)} + 1, & u \in B \\ d^{(I)}(u), & u \in U_2 \setminus B \end{cases}$$

At last, we set $t_{TS} = t^{(I)}$ and $\Gamma' = \Gamma$ and let C_Γ be the discrete budgeted uncertainty set defined by \underline{c}, \bar{c} . This completes the description of the **TWO-STAGE ADJUSTABLE-II** instance. Recall that this instance by definition is a **YES**-instance if and only if the following inequality is true:

$$\min_{S_1 \subseteq \mathcal{U}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \subseteq \mathcal{U}(I) \setminus S_1 \\ S_1 \cup S_2 \in \mathcal{F}(I)}} c_1(S_1) + c_2(S_2) \leq t_{TS} \quad (6.1)$$

To prove the correctness of the reduction, observe that whenever an element has cost of $t^{(I)} + 1$, the element cannot be in a solution because the threshold t_{TS} is set to $t^{(I)}$.

Now, assume that inequality (6.1) is true. Then the solution S_1 in the first stage has to meet the condition $S_1 \subseteq U_1$. For a similar reason, the second stage solution S_2 has to meet the condition $S_2 \subseteq \mathcal{U}(I) \setminus (U_1 \cup B') = U_2 \setminus B'$, where $B' \subseteq B$ denotes the set of elements whose costs were increased in the second stage. Note that $|B'| \leq \Gamma$. These two facts together imply $c_1(S_1) + c_2(S_2) = d^{(I)}(S_1 \cup S_2)$. This in turn implies that $S_1 \cup S_2$ is not only contained in $\mathcal{F}(I)$ as described in inequality (6.1), but we even have the stronger condition $S_1 \cup S_2 \in \mathcal{S}(I)$. All the arguments above together show that (I, U_1, U_2, B, Γ) is a YES-instance of COMB. TWO-STAGE ADJUSTABLE-II.

For the other direction, assume that (I, U_1, U_2, B, Γ) is a YES-instance of COMB. TWO-STAGE ADJUSTABLE-II'. We can argue in a very similar way to the above that the inequality (6.1) is true. At last, this transformation is computable in polynomial time, because the underlying instance is unchanged and only numbers encoded in polynomial size (Π is in NP) are added. In total, we get that the two instances are equivalent, thus the reduction is correct and TWO-STAGE ADJUSTABLE-II is Σ_3^p -complete. \square

6.6 The Meta-Reduction is an SSP reduction

The meta-reduction from Theorem 6.1 again relies heavily on the one-to-one correspondence between the set of first stage elements and second stage elements as well as the set of blockable elements of the problems COMB. TWO-STAGE ADJUSTABLE-SAT and COMB. TWO-STAGE ADJUSTABLE-II. Because all of the sets from above are subsets of the universe, we are able to prove that the meta-reduction is also an SSP reduction. For this, we first show that the combinatorial two-stage adjustable variant of an SSP problem again is an SSP problem.

Observation 6.1. *The combinatorial two-stage variant COMB. TWO-STAGE ADJUSTABLE-II of an SSP problem Π is an SSP problem.*

Proof. Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem and denote COMB. TWO-STAGE ADJUSTABLE-II = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$. Then, set $\mathcal{U} = \mathcal{U}'$. Now, let $I \in \mathcal{I}$ be an instance of Π . Then, we can define the corresponding instances of COMB. TWO-STAGE ADJUSTABLE-II by setting $I' = \{(I, U_1, U_2, B, \Gamma) \mid I \in \mathcal{I}, U_1 \subseteq \mathcal{U}, U_2 = \mathcal{U} \setminus U_1, B \subseteq U_2, \Gamma \in \mathbb{Z}\}$. We define the solutions $\mathcal{S}'(I')$ to be all sets $S_1 \subseteq U_1 \subseteq \mathcal{U}'(I')$ such that for all $B' \subseteq B$ with $|B'| \leq \Gamma$ there is $S_2 \subseteq U_2$ such that $S_2 \cap B' = \emptyset$ and $S_1 \cup S_2 \in \mathcal{S}(I)$. Thus, COMB. TWO-STAGE ADJUSTABLE-II = $(\mathcal{I}', \mathcal{U}', \mathcal{S}')$ is an SSP problem. \square

With this additional observation, we are able to elegantly prove the meta-reduction from any SSP-NP-complete problem Π by extending the existing SSP reduction $\text{SAT} \leq_{\text{SSP}} \Pi$ to be an SSP reduction $\text{COMB. TWO-STAGE ADJUSTABLE-SAT} \leq_{\text{SSP}} \text{COMB. TWO-STAGE ADJUSTABLE-II}$.

Corollary 6.1. *For all SSP-NP-complete problems Π , $\text{COMB. TWO-STAGE ADJUSTABLE-SAT} \leq_{\text{SSP}} \text{COMB. TWO-STAGE ADJUSTABLE-II}$.*

Proof. By Observation 6.1, COMB. TWO-STAGE ADJUSTABLE-SAT and COMB. TWO-STAGE ADJUSTABLE-II are SSP problems and we design an SSP reduction $(g', (f_{I'})_{I' \in \mathcal{I}'})$ from COMB. TWO-STAGE ADJUSTABLE-SAT to COMB. TWO-STAGE ADJUSTABLE-II by extending the existing reduction $(g, (f_I)_{I \in \mathcal{I}})$ as depicted in Figure 6.2, where g' is the reduction from the proof of Theorem 6.1.

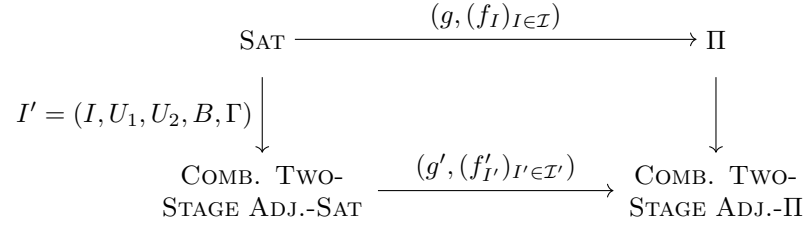


Figure 6.2: The fact that SAT is SSP reducible to Π induces an SSP reduction from COMB. TWO-STAGE ADJUSTABLE-SAT to COMB. TWO-STAGE ADJUSTABLE- Π .

We recall the reduction from the proof of Theorem 6.1. As we have already proven the correctness of g' , we focus solely on the function $(f'_{I'})_{I' \in \mathcal{I}'}$. We define $f_I = f'_{I'}$ such that the following holds:

$$U'_1 = f'_{I'}(U_1), \quad U'_2 = \mathcal{U}(g(I)) \setminus U'_1, \quad B' = f'_{I'}(B).$$

The function $f'_{I'}$ is well-defined because the universe \mathcal{U} of the SAT instance I and the COMB. TWO-STAGE ADJUSTABLE-SAT instance I' are the same. Therefore, $(g', (f'_{I'})_{I' \in \mathcal{I}'})$ is an SSP reduction. \square

Chapter 7

Recoverable Robustness

7.1 Introduction

Real-world decision makers are faced with a large degree of uncertainty when making important decisions in economics, planning, and operations research. The successful area of *robust optimization* [BGN09, GH24, KY13] has been developed as one possible way to deal with these uncertainties. However, sometimes the classical robust optimization approach has turned out to be too conservative. For this reason, *recoverable robust optimization* was introduced by Liebchen, Lübbecke, Möhring and Stiller [LLMS09]. Initially motivated by train scheduling problems, it has since then found wide-spread application in practice and in the analysis of standard optimization problems. For instance, it has been successfully applied to combinatorial problems such as s-t-path [Büs11, JKZ24a, JKZ24b], matching [DMP⁺15], scheduling [BG22], spanning tree [HKZ17a, HKZ17b], knapsack [BGKK19, BKK11a, BKK11b, LLW21], and TSP [CG16, GLW22a]. Recoverable robust optimization follows a two-step robust optimization approach. In contrast to classic robust optimization, where a first-stage solution cannot be changed, in recoverable robust optimization the decision maker is allowed to incorporate a limited recovery action after the underlying uncertainty is revealed. Mathematically, this is described by the expression

$$\min_{S_1} \max_{c_2 \in C} \min_{S_2, \text{dist}(S_1, S_2) \leq \kappa} c_1(S_1) + c_2(S_2).$$

Here, S_1 denotes the first-stage solution, S_2 denotes the second-stage solution, c_1 denotes the first-stage cost function, C denotes the set of uncertain scenarios, and $dist$ is some distance measure between the solutions. A more formal definition will be given later.

In this chapter, we make the standard assumption that the underlying uncertainty for the robust problem is given as a discrete budgeted uncertainty set, also denoted as discrete Γ -uncertainty [BS03]. The main goal of this chapter is to extend the SSP framework also to the setting of recoverable optimization. It turns out that compared to the standard SSP framework more complicated assumptions are needed. We introduce a set of sufficient conditions for some nominal problem, which imply that the recoverable robust version of that problem is Σ_3^p -complete.

Our Results. We consider recoverable robust optimization for the following nominal problems:

Sat, 3Sat, vertex cover, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median, independent set, clique, subset sum, knapsack, partition, scheduling, Hamiltonian path/cycle (directed/undirected), TSP, k -directed disjoint path ($k \geq 2$), and Steiner tree.

In addition we consider the three most popular distance measures *dist* used in recoverable robust optimization: The κ -addition distance, the κ -deletion distance, and the Hamming distance.

We show that for every combination of the above problems with any of the three distance measures, the recoverable robust problem (with discrete budgeted uncertainty) is Σ_3^p -complete. More generally, we identify an abstract property of all our studied problems, and prove that this abstract property already implies that the recoverable robust problem becomes Σ_3^p -complete. This answers a question asked by Goerigk, Lendl and Wulf [GLW24]. We remark that Σ_3^p -completeness was already known in the case of clique/independent set, TSP or shortest path combined with the Hamming distance [GLW24]. It was also already known for shortest path in combination with all three distance measures [JKZ24a]. Hence our work is an extension of these results.

Related Work. Recoverable robustness concepts were analyzed for a variety of different standard optimization problems. Büsing [Büs11] analyzed the recoverable shortest path problem with discrete budgeted uncertainty, in which adding at most κ elements to the second stage solution are allowed. This analysis was lately continued by Jackiewicz, Kasperski, and Zieliński [JKZ24b] for several different graph classes and for interval budgeted uncertainty. Furthermore, recoverable robust knapsack was analyzed by Büsing, Koster, and Kutschka [BKK11b] for discrete scenarios and in which adding at most κ elements and deleting at most ℓ elements are allowed. Büsing, Koster, and Kutschka [BKK11a] additionally analyzed the Γ -scenario case (discrete budgeted uncertainty) while allowing at most ℓ elements to be deleted. This work was also continued by Büsing, Goderbauer, Koster, and Kutschka [BGKK19]. Further classical optimization problems that were studied in the recoverable robustness context are matching by Dourado, Meierling, Penso, Rautenbach, Protti, and de Almeida [DMP⁺15] and spanning tree under interval cost uncertainty by Hradovich, Kasperski, and Zieliński [HKZ17a, HKZ17b]. Additionally, Lendl, Peis, and Timmermans [LPT22] examined matroidal problems. One variant of independent set for recoverable robustness with a commitment property was analyzed by Hommelsheim, Megow, Muluk, and Peis [HMMP23]. Beside these problems, recoverable robust selection was studied by Kasperski and Zieliński [KZ17], Chassein, Goerigk, Kasperski, and Zieliński [CGKZ18], and Goerigk, Lendl, and Wulf [GLW22a]. Moreover, Lachmann, Lendl, and Woeginger [LLW21] developed a linear time algorithm for the recoverable Γ -robust knapsack problem. The recoverable robust assignment problem was also investigated by Fischer, Hartmann, Lendl, and Woeginger [FHLW21].

Closely related to our work are the complexity studies by Goerigk, Lendl, and Wulf [GLW24] who analyze the problems independent set, traveling salesman and vertex cover and obtain Σ_3^p -completeness for the three problems. Additionally, Jackiewicz, Kasperski, and Zieliński [JKZ24a] show that the shortest path problem with discrete budgeted interval uncertainty is Σ_3^p -complete.

7.2 Recoverable Robust Problems

We consider recoverable robust optimization problems. We show that the recoverable robust optimization problem is Σ_3^p -complete for the following nominal problems: satisfiability, 3-satisfiability, vertex cover, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p-center, p-median, independent set, clique, subset sum, knapsack, partition, scheduling, (un)directed Hamiltonian path, (un)directed Hamiltonian cycle, traveling salesman, two directed disjoint path, k directed disjoint path, and Steiner tree.

Recoverable robust optimization problems are defined as follows: We are given some instance I of a linear optimization problem (like the shortest path problem, the traveling salesman problem, etc.), and are faced with an uncertain future. The goal is to find a feasible solution S_1

for the first stage (called here-and-now decision) such that after the reveal of the uncertainty we can find a feasible solution S_2 in the second stage (called wait-and-see decision) such that S_1 and S_2 are not far away from each other according to some distance measure. Formally we require $\text{dist}(S_1, S_2) \leq \kappa$, where $\text{dist}(S_1, S_2)$ is some abstract distance function (for example the Hamming distance $|S_1 \Delta S_2|$). The cost of the solution is given by $c_1(S_1) + c_2(S_2)$. Here, c_1 is a fixed cost function not affected by uncertainty, called the setup costs, and c_2 is affected by uncertainty. More specifically, we assume that $c_2 \in C_\Gamma$, that is, c_2 is affected by discrete budgeted uncertainty C_Γ . Precisely, given $\Gamma \in \mathbb{N}$ and upper and lower bounds $\underline{c}(u) \leq \bar{c}(u)$ for all elements in the universe, the set C_Γ contains all cost functions such that at most Γ elements $u \in \mathcal{U}(I)$ have costs of $\bar{c}(u)$, while all other have $\underline{c}(u)$:

$$C_\Gamma := \{c_2 \mid \forall u \in \mathcal{U}(I) : c_2(u) = \underline{c}(u) + \delta_u(\bar{c}(u) - \underline{c}(u)), \delta_u \in \{0, 1\}, \sum_{u \in \mathcal{U}(I)} \delta_u \leq \Gamma\}$$

This leads to the following abstract definition of a recoverable robust problem. We remark that the formal definition in some sense “disregards” the cost functions d and the cost threshold t of the original LOP problem. This is intentional because in the new instance of the recoverable robust problem, it becomes necessary to substitute the old function d and the old threshold t by new ones. However, these concepts are necessary to correctly understand the proofs.

Definition 7.1 (Recoverable Robust Problem). *Let an LOP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ and a distance measure $\text{dist} : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \rightarrow \mathbb{R}_{\geq 0}$ be given. The recoverable robust problem associated to Π is denoted by $RR\text{-}\Pi$ and defined as follows: The input is an instance $I \in \mathcal{I}$ together with three cost functions $c_1 : \mathcal{U}(I) \rightarrow \mathbb{Z}$, $\underline{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and $\bar{c} : \mathcal{U}(I) \rightarrow \mathbb{Z}$ and a cost threshold $t_{RR} \in \mathbb{Z}$, an uncertainty parameter $\Gamma \in \mathbb{N}_0$ and a recoverability parameter $\kappa \in \mathbb{N}_0$. The question is whether*

$$\min_{S_1 \in \mathcal{F}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \in \mathcal{F}(I) \\ \text{dist}(S_1, S_2) \leq \kappa}} c_1(S_1) + c_2(S_2) \leq t_{RR}. \quad (7.1)$$

Example. Let $\Pi = \text{TSP}$. TSP is encoded as LOP problem the following way: An instance is given by $I = (G, d, t)$, where $G = (V, E)$ is a complete undirected graph, $d : E \rightarrow \mathbb{N}_0$ are the edge costs and t is the cost threshold. The decision problem of TSP asks if there is a tour $T \subseteq E$ with cost $d(T) \leq t$ visiting all vertices from V exactly once. The universe is $\mathcal{U}(I) = E$. The set $\mathcal{F}(I) \subseteq 2^E$ is the set of all feasible tours (including those of cost greater than t). The set $\mathcal{S}(I)$ is the set of all tours of cost at most t . To turn the TSP into a recoverable robust problem, we “forget” about the cost function d and the threshold t . Given $c_1, \underline{c}, \bar{c}, \Gamma, \kappa, t_{RR}$, the decision problem associated to the recoverable robust TSP is to decide whether Equation (7.1) holds.

We remark that this definition does not include all SSP problems but only LOP problems. This is for the reason that recoverable robust optimization is usually considered only for linear optimization problems, which distinguish between feasible solutions $\mathcal{F}(I)$ and optimal solutions $\mathcal{S}(I)$. In contrast, recoverable robust optimization is usually not considered for pure feasibility problems, like SAT, which are modeled as SSP problems in our framework. However, it is still possible to define a variant of recoverable robust optimization that is applicable to *all* SSP problems. Indeed, it turns out that such a definition becomes helpful for our proof. Hence, in Subsection 7.2.3, we introduce a corresponding definition (and also show Σ_3^P -completeness of several problems with this new definition).

7.2.1 Distance Measures

Recoverable robust problems require a distance measure to model the constraint that the solutions remain close to each other. However, there are numerous problems having different

structures such that it is not possible to define distance measures for all possible types of recoverable robust problems. Because we restrict ourselves to a certain kind of problems, namely SSP problems, we consider distance measures defined over sets. This allows us to determine a specific enough definition of distance measure to be meaningful. Furthermore, our definition needs to be general enough to include the distance measures used in the literature. Among those are the following:

- ▶ κ -addition or simply κ -distance measure ([Büs12, HKZ17a]): $\text{dist}(A_1, A_2) = |A_2 \setminus A_1|$
- ▶ κ -deletion distance ([BKK11a]): $\text{dist}(A_1, A_2) = |A_1 \setminus A_2|$
- ▶ Hamming distance measure ([DMP⁺15, GLW24, Grü24]): $\text{dist}(A_1, A_2) = |A_1 \Delta A_2|$

Definition 7.2 (Distance Measure). *Let U be a set and $A_1, A_2 \subseteq U$ subsets. A distance measure on set U is a map $\text{dist}_U : 2^U \times 2^U \rightarrow \mathbb{R}_{\geq 0}$ that adheres to the following properties*

- ▶ computable in polynomial time
- ▶ invariant on injective mappings $f : U \rightarrow U'$, i.e. $\text{dist}_U(A_1, A_2) = \text{dist}_{U'}(f(A_1), f(A_2))$,
- ▶ invariant on union, i.e. for $x \in U \setminus (A_1 \cup A_2)$: $\text{dist}_U(A_1, A_2) = \text{dist}_U(A_1 \cup \{x\}, A_2 \cup \{x\})$.
- ▶ $\text{dist}_U(A_1, A_1) = 0$

If U is clear from the context, we omit the subscript.

One can easily verify that all of the distance measures from above fulfill these criteria.

7.2.2 Containment in Σ_3^p

Recoverable robust problems can also be understood as a three-stage two-player game of an \exists -player playing against an adversary (the \forall -player). The \exists -player controls both min operators and is able to choose the solutions S_1 and S_2 . On the other hand, the adversary controls the max operator and is able to choose the uncertainty scenario, i.e. the cost function c_2 from the set C_Γ . Thus, it is possible to reformulate the question to

$$\exists S_1 \subseteq \mathcal{U}(I) : \forall c_2 \in C_\Gamma : \exists S_2 \subseteq \mathcal{U}(I) : S_1, S_2 \in \mathcal{F}(I), c_1(S_1) + c_2(S_2) \leq t \text{ and } \text{dist}(S_1, S_2) \leq \kappa.$$

With the game-theoretical perspective, it is intuitive to see the containment in Σ_3^p for all problems that have a polynomial-time verifier.

Theorem 7.1. *If $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ is an LOP problem in NP, then RR-II is in Σ_3^p .*

Proof. We provide a polynomial-time algorithm V that verifies a specific solution for the three quantifiers y_1, y_2, y_3 of polynomial size for instance I such that

$$I \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} y_3 \in \{0, 1\}^{m_3} : V(I, y_1, y_2, y_3) = 1.$$

With the first \exists -quantified y_1 , we encode the first solution $S_1 \subseteq \mathcal{U}(I)$. Because the universe is part of the input of the problem, this is at most linear in the input. Next, we encode all cost functions $c_2 \in C_\Gamma$ with the \forall -quantified y_2 . For this, we encode which of the at most Γ elements are chosen to have costs \bar{c} . This is at most linear in the input size. At last, we encode the second solution $S_2 \subseteq \mathcal{U}(I)$ with the help of the second \exists -quantified y_3 . Again, this is at most linear in the input size as before. Now, the verifier has to check whether $\text{dist}(S_1, S_2) \leq \kappa$, $S_1, S_2 \in \mathcal{F}(I)$ and $c_1(S_1) + c_2(S_2) \leq t$. All of these checks can be done in polynomial time, where $S_1, S_2 \in \mathcal{F}(I)$ and $c_1(S_1) + c_2(S_2) \leq t$ can be checked in polynomial time because $\Pi \in \text{NP}$. Furthermore, $\text{dist}(S_1, S_2) \leq \kappa$ can be checked by our assumption that the abstract distance function is computable in polynomial time. It follows that RR-II is in Σ_3^p . \square

7.2.3 Combinatorial Recoverable Robust Problems

To show the hardness of recoverable robust problems, we introduce a new problem which we call the *combinatorial version* of recoverable robust problems. There are two differences to Definition 7.1: In this new problem, the cost function is substituted by a set B of so-called *blockable elements* (this can be interpreted as the case where cost coefficients \underline{c}, \bar{c} are restricted to come from $\{0, \infty\}$). Furthermore, in this new definition we substitute $\mathcal{F}(I)$ by $\mathcal{S}(I)$. As we show later, the hardness of the new combinatorial version also implies the hardness of the cost version. Finally, as explained above, Definition 7.3 applies to all SSP problems.

Definition 7.3 (Combinatorial Recoverable Robust Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The combinatorial recoverable robust problem associated to Π is denoted by $\text{COMB. RR-}\Pi$ and defined as follows: The input is an instance $I \in \mathcal{I}$ together with a set $B \subseteq \mathcal{U}(I)$, an uncertainty parameter $\Gamma \in \mathbb{N}_0$, and a recoverability parameter $\kappa \in \mathbb{N}_0$. The question is whether*

$$\begin{aligned} \exists S_1 \subseteq \mathcal{U}(I) : \forall B' \subseteq B \text{ with } |B'| \leq \Gamma : \exists S_2 \subseteq \mathcal{U}(I) : \\ S_1, S_2 \in \mathcal{S}(I), S_2 \cap B' = \emptyset \text{ and } \text{dist}(S_1, S_2) \leq \kappa. \end{aligned}$$

7.3 An SSP Framework for Recoverable Robust Problems

In this section, we prove our main result, i.e. we introduce and prove a sufficient condition for nominal problems such that the corresponding recoverable robust problem is Σ_3^p -hard. We first explain the rough idea.

We want to show that recoverable problems are Σ_3^p -hard, so we have to choose some Σ_3^p -hard problem from which we start our reduction. The canonical Σ_3^p -complete problem is $\exists\forall\exists$ -SATISFIABILITY [Sto76]. Intuitively, this satisfiability problem can be understood as a game between Alice and Bob. Alice first chooses a variable assignment on the variables of X , then Bob chooses an assignment of the variables Y , and then again Alice selects an assignment on the variables Z , where Alice wishes to satisfy formula $\varphi(X, Y, Z)$, and Bob wishes the opposite. On the other hand, in our target problem $\text{RR-}\Pi$, any solution consists of two solutions of the nominal problem Π , the first stage solution S_1 and the second stage solution S_2 . The main challenge of our reduction is, that we have to model three variable sets X, Y, Z of the formula $\exists X \forall Y \exists Z \varphi(X, Y, Z)$ in the order of quantification into the problem $\text{RR-}\Pi$. Accordingly, both solutions S_1, S_2 need to include an assignment to all variables from X, Y, Z that satisfy $\varphi(X, Y, Z)$ in agreement with the nominal SAT problem. However, note that there is a difference in the solution structure of both problems, which we need to address. If we model Alice's decision on the X -variables in $\exists\forall\exists$ -SAT in first stage solution S_1 of $\text{RR-}\Pi$, we need to make sure that Alice is not able to reassign the the X -variables in the second stage solution S_2 of $\text{RR-}\Pi$. Otherwise, Alice does not adhere to the order of quantification.

This problem can be circumvented by adding an additional property to the SSP reduction. We demand that for a given set of literals L_b the distance of two solutions S_1 and S_2 is small if and only if the partial solution of S_1 and S_2 restricted to L_b is the same. One can interpret this construction as a gadget that *blows up* the literals of L_b in comparison to all the other literals. Accordingly, we call the literals of L_b *blow-up literals* and the corresponding reduction *blow-up SSP reduction*. Then, we can set $L_b = L_X$, where L_X is the set of literals corresponding to variables of X . Therefore, Alice is not able to change the assignment of the variable set X from the first stage solution S_1 to the second stage solution S_2 and thus has to adhere to the order of quantification. Then, it is also possible to reuse the injective correspondence function f of the SSP reduction to set the blockable elements B and to show the correctness of the reduction by

using the elementary correspondence between the SAT solution and the solution of Π . Formally, we define *blow-up SSP reductions* as follows.

Definition 7.4 (Blow-Up SSP Reduction). *Let dist be a distance measure. Let $3\text{SAT} = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ with $\mathcal{U} = L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$ and $\Pi' = (\mathcal{I}', \mathcal{U}', \mathcal{S}')$ be two SSP problems. Then, 3SAT is SSP blow-up reducible to Π' with respect to $\text{dist}(\cdot, \cdot)$ if for all sets $L_b \subseteq L$ fulfilling $\ell_i \in L_b \leftrightarrow \bar{\ell}_i \in L_b$ there exists an SSP reduction (g, f_I) with the following property: There is a polynomial time computable blow-up factor $\beta_I \in \mathbb{N}$ corresponding to each instance I such that for all solutions $S_1, S_2 \in \mathcal{S}'(g(I))$ of $g(I)$:*

$$f(L_b) \cap S_1 = f(L_b) \cap S_2 \Leftrightarrow \text{dist}_{\mathcal{U}'(g(I))}(S_1, S_2) \leq \beta_I.$$

A concrete example of a blow-up reduction is given in Subsection 7.3.1.

Remark 1. Note that we define blow-up reductions to start at 3SAT . Because 3SAT is one of the 'first' NP-complete problems, many reductions start at 3SAT or have a short reduction chain from 3SAT . Thus, it is convenient to reuse these reductions or respectively reduction chains to construct blow-up reductions. As we will show later in Subsection 7.3.1 and Section 7.4 this is not really a restriction.

Remark 2. Blow-up SSP reductions are not transitive. (This is because there is no restriction on newly introduced elements in Π' and how they behave corresponding to solutions in Π' . Therefore, the distance between solutions in Π' cannot be related to the distance between the corresponding solutions in Π .) We will tackle this problem in Section 7.4, in which we will show that we only need to find a blow-up reduction once at the beginning of a reduction chain and we can use so-called blow-up preserving reductions to append further problems to the reduction chain starting at 3SAT . This blow-up preserving reduction is typically much easier to find than a new blow-up reduction.

With everything in place, we show that blow-up SSP reductions enable us to show the Σ_3^p -hardness of recoverable robust problems as long as the nominal problem Π is SSP-NP-hard. In particular, our main theorem now states, that our newly introduced blow-up reductions indeed are a sufficient criterion for Σ_3^p -hardness.

Theorem 7.2. *For all SSP-NP-hard problems Π that are blow-up SSP reducible from 3SAT , the combinatorial recoverable robust variant $\text{COMB. RR-}\Pi$ is Σ_3^p -hard.*

Proof. For the proof of the main theorem, we require some definitions. Let $X = \{x_1, \dots, x_n\}$ be a set of binary variables and $L := \{x_1, \dots, x_n\} \cup \{\bar{x}_1, \dots, \bar{x}_n\} = X \cup \bar{X}$ be the set of corresponding literals. An *assignment* is a subset $A \subseteq X \cup \bar{X}$ such that $|A \cap \{x_i, \bar{x}_i\}| = 1$ for all $i = 1, \dots, n$. We say that the assignment A assigns the value 1 to variable x_i , if $x_i \in A$, and A assigns 0 to x_i , if $\bar{x}_i \in A$. We remark that this notation for an assignment is non-standard, but it turns out to be convenient in the context of our framework. A SAT-formula φ is in 3CNF, if it is a conjunction of clauses, and every clause has exactly three literals. We denote by $\varphi(A) \in \{0, 1\}$ the evaluation of the formula φ under the assignment A . In the following, we often consider the case, where the set of variables is partitioned into three disjoint sets X, Y, Z , and denote this case by writing $\varphi(X, Y, Z)$.

For the Σ_3^p -hardness proof, we require a known Σ_3^p -complete problem to reduce from. It turns out that instead of reducing from the classic problem $\exists\forall\exists$ -SATISFIABILITY, it is more convenient to base our proof on ROBUST ADJUSTABLE SAT with budgeted uncertainty (in short, R-ADJ-SAT), introduced by Goerigk, Lendl and Wulf [GLW24], which we reformulate to adhere to the

notation of this part:

R-ADJ-SAT

Instance: A SAT-formula $\varphi(X, Y, Z)$ in 3CNF. A partition of the set of variables into three disjoint parts $X \cup Y \cup Z$ with $|X| = |Y| = |Z|$. An integer $\Gamma \geq 0$.

Question: Is there an assignment $A_X \subseteq X \cup \bar{X}$ such that for all subsets $Y' \subseteq Y$ of size $|Y'| \leq \Gamma$, there exist assignments $A_Y \subseteq Y \cup \bar{Y}$ and $A_Z \subseteq Z \cup \bar{Z}$ such that $A_Y \cap Y' = \emptyset$, i.e. set all variables in Y' to 0, and $\varphi(A_X, A_Y, A_Z) = 1$?

The problem R-ADJ-SAT is best understood as a game between Alice and Bob: Alice wants to satisfy the formula, Bob wants to achieve the opposite. Alice initially selects an assignment A_X of the variables in X . Afterwards, Bob is allowed to select a blocker $Y' \subseteq Y$ of size at most Γ and force all these variables to be “0”. Finally, Alice is allowed to assign values to all variables in $Y \cup Z$ that have not yet been assigned. In [GLW24] it is shown that R-ADJ-SAT is Σ_3^P -complete. In order to provide additional insight for the reader, we quickly sketch the main argument in [GLW24]: The idea is to reduce from the classic problem $\exists\forall\exists$ -SATISFIABILITY. Let $\exists A_X \forall A_Y \exists A_Z \varphi(A_X, A_Y, A_Z)$ be an instance of this problem. How do we transform it into a R-ADJ-SAT problem? It is intuitive, that in such a reduction, Alice’s rule of choosing assignments A_X and A_Z should stay roughly the same. However, how do we express the choice of A_Y in terms of a blocker Y' that is forced to 0? The idea is to split the variable $y_i \in Y$ into two new variables y_i^t, y_i^f for every $i = 1, \dots, n$. We say that Bob plays honestly if for all $i = 1, \dots, n$ we have $|\{y_i^t, y_i^f\} \cap Y'| = 1$. Such an honest choice of Y' naturally encodes an assignment A_Y . It turns out that, using the pigeonhole principle, and the budget constraint $|Y'| \leq \Gamma$, one can enrich the formula φ with a “cheat-detection-gadget”. This gadget can be used by Alice to win the game trivially if and only if Bob cheats. Hence the modified game of R-ADJ-SAT is equivalent to the original instance of $\exists\forall\exists$ -SATISFIABILITY.

We prove Σ_3^P -hardness by providing a reduction from R-ADJ-SAT to COMB. RR-II. This reduction will crucially rely on the blow-up SSP reduction from 3SAT to Π , which we assumed to exist. More formally, let $\Pi = (\mathcal{I}_\Pi, \mathcal{U}_\Pi, \mathcal{S}_\Pi)$ be an SSP problem, such that there is a blow-up reduction $(g, (f_I)_{I \in \mathcal{I}_{\text{SAT}}}, (\beta_I)_{I \in \mathcal{I}_{\text{SAT}}})$ from 3SAT to Π . Recall that by the definition of a blow-up reduction this means the following: Let the term \mathcal{I}_{SAT} denote the set of all possible 3SAT instances, i.e. the set of all 3CNF formulas. For each fixed 3SAT instance $I_{\text{SAT}} \in \mathcal{I}_{\text{SAT}}$, its universe $\mathcal{U}(I_{\text{SAT}})$ is the set of positive and negative literals of variables appearing in that formula. By assumption, for every fixed 3SAT instance $I_{\text{SAT}} \in \mathcal{I}_{\text{SAT}}$ and every set $L_b \subseteq \mathcal{U}(I_{\text{SAT}})$ (with $\ell \in L_b$ iff $\bar{\ell} \in L_b$) there exists an equivalent instance $I_\Pi = g(I_{\text{SAT}})$ of Π . Furthermore, associated to each fixed 3SAT instance $I_{\text{SAT}} \in \mathcal{I}_{\text{SAT}}$ we have a function $f_{I_{\text{SAT}}}$, which describes in which way the universe of I_{SAT} can be injectively embedded into the universe of the equivalent instance I_Π such that the SSP property is true. Finally, there is a tight correspondence between solutions of I_Π having distance at most $\beta_{I_{\text{SAT}}}$, and (the pre-image of) the solutions agreeing on the set L_b . For brevity of notation, in the following we often write $\mathcal{U}(I_\Pi)$ instead of $\mathcal{U}_\Pi(I_\Pi)$ and $\mathcal{S}(I_\Pi)$ instead of $\mathcal{S}_\Pi(I_\Pi)$ to denote the solution set/universe associated to instance I_Π , if the correct subscript is clear from the context.

We now turn our attention to the Σ_3^P -hardness proof. As explained above, the problem R-ADJ-SAT is Σ_3^P -complete. Let an R-ADJ-SAT instance I_{RAS} be given. Our goal is to transform this instance into a new instance $I_{\text{RR}\Pi}$ of COMB. RR-II such that

$$I_{\text{RAS}} \text{ is a YES-instance of R-ADJ-SAT} \Leftrightarrow I_{\text{RR}\Pi} \text{ is a YES-instance of Comb. RR-II.}$$

Clearly if we can achieve this goal we are done. By definition of the problem R-ADJ-SAT, the instance I_{RAS} consists out of the following parts:

$$I_{\text{RAS}} = (\varphi(X, Y, Z), \Gamma),$$

where $\varphi \in \mathcal{I}_{\text{SAT}}$ is a 3CNF-formula, whose variables are partitioned into three parts X, Y, Z of equal size $n := |X| = |Y| = |Z|$, and $\Gamma \in \mathbb{N}_0$. Let the corresponding literal sets X, Y, Z be denoted by $L_X := X \cup \bar{X}$, $L_Y := Y \cup \bar{Y}$, and $L_Z := Z \cup \bar{Z}$. Note that the universe associated to the 3SAT-instance φ is $L_X \cup L_Y \cup L_Z$. Let us denote this fact by writing $\mathcal{U}(\varphi) = L_X \cup L_Y \cup L_Z$.

Given the instance I_{RAS} as input, we describe in the following how to construct the instance I_{RRII} of COMB. RR-II, which consists out of the following parts:

$$I_{\text{RRII}} = (I_{\Pi}, B, \Gamma', \kappa).$$

Here, I_{Π} is an instance of the nominal problem Π , $B \subseteq \mathcal{U}(I_{\Pi})$ is the set of blockable elements, $\Gamma' \in \mathbb{N}_0$ is the uncertainty budget, and $\kappa \in \mathbb{N}_0$ is the recoverability parameter.

Before we give a formal description of I_{Π}, B, Γ' , and κ , we explain the rough idea: In particular, what is the right choice for the instance I_{Π} ? The answer is provided by the blow-up reduction. Note that this reduction can take in any 3CNF formula and a subset L_b of the literals (with the property that $\ell \in L_b$ iff $\bar{\ell} \in L_b$ for all literals ℓ) and produce an instance I of Π . Note that the set L_b gets “blown up” by the reduction. We claim that $L_b = L_X$ is a good choice. Indeed, consider the following formal definition of I_{RRII} .

Description of the Instance I_{RRII} Let $(g, (f_{\varphi})_{\varphi \in \mathcal{I}_{\text{SAT}}}, (\beta_{\varphi})_{\varphi \in \mathcal{I}_{\text{SAT}}})$ be the blow-up reduction from 3SAT to Π . We let $I_{\Pi} = g(\varphi)$ be the instance produced by the blow-up reduction applied to the formula φ and the literal set $L_b := L_X$. We remark that the following holds by the definition of the blow-up reduction: A blow up-reduction in particular is also an SSP reduction. Hence the function $f_{\varphi} : \mathcal{U}(\varphi) \rightarrow \mathcal{U}(I_{\Pi})$ maps the literals of φ injectively to elements of the new universe $\mathcal{U}(I_{\Pi})$. We can hence define the set of blockable elements

$$B := f_{\varphi}(Y),$$

where $Y \subseteq \mathcal{U}(\varphi)$ describes the positive literals in L_Y (recall that $L_Y = Y \cup \bar{Y}$).

Furthermore, note that by definition of a blow-up reduction, there exists some $\beta_{\varphi} \in \mathbb{N}$, computable in poly-time, with the property that for all $S_1, S_2 \in \mathcal{S}(I_{\Pi})$:

$$\text{dist}(S_1, S_2) \leq \beta_{\varphi} \Leftrightarrow S_1 \cap f_{\varphi}(L_X) = S_2 \cap f_{\varphi}(L_X).$$

In other words, the new instance I_{Π} has the property that two solutions of it have small distance if and only if they agree on $f_{\varphi}(L_X)$, i.e. they agree on those universe elements of $\mathcal{U}(I_{\Pi})$ that correspond to L_X . We finally define

$$\kappa := \beta_{\varphi} \text{ and } \Gamma' := \Gamma.$$

This completes the description of the instance $I_{\text{RRII}} = (I, B, \Gamma', \kappa)$.

Correctness We start the correctness proof by showing

$$I_{\text{RAS}} \text{ is a YES-instance} \Rightarrow I_{\text{RRII}} = (I_{\Pi}, B, \Gamma, \kappa) \text{ is a YES-instance.}$$

In this case, let $A_X \subseteq X \cup \bar{X}$ be an assignment of the variables X with the property that

$$\forall Y' \subseteq Y, |Y'| \leq \Gamma : \exists A_Y, A_Z : A_Y \cap Y' = \emptyset \text{ and } \varphi(A_X, A_Y, A_Z) = 1.$$

Such an A_X exists by assumption that I_{RAS} is a YES-instance. Now, let $Y' \subseteq Y$ be an arbitrary subset of Y with $|Y'| \leq \Gamma$. Then, it is possible to choose assignments A_Y, A_Z of Y, Z such that $A_Y \cap Y' = \emptyset$ and $\varphi(A_X, A_Y, A_Z) = 1$. We consider such an assignment

$A_1 := A_X \cup A_Y \cup A_Z$. Note that if we interpret φ as 3SAT instance, $A_1 \subseteq \mathcal{U}(\varphi)$ and even $A_1 \in \mathcal{S}(\varphi)$, due to $\varphi(A_1) = 1$. Since the blow-up reduction is in particular an SSP reduction, we can make use of the central property of SSP reductions. The property implies that the 3SAT solution A_1 can be “lifted” to a solution of Π . More precisely, there exists a solution $S_1 \in \mathcal{S}(I_\Pi)$, such that

$$S_1 \cap f_\varphi(\mathcal{U}(\varphi)) = f_\varphi(A_1).$$

(Intuitively, the solution $S_1 \in \mathcal{S}(I_\Pi)$ of the nominal problem instance I_Π corresponds to the solution $A_1 \in \mathcal{S}(\varphi)$ of the 3SAT instance φ when restricted to the injective embedded “sub-instance” of 3SAT inside Π .)

We claim that S_1 is a solution for $I_{\text{RR}\Pi}$. To prove this claim we have to show that for all blockers $B' \subseteq B$ with $|B'| \leq \Gamma' = \Gamma$, there exists a solution $S_2 \in \mathcal{S}(I_\Pi)$ with $S_2 \cap B' = \emptyset$ and $\text{dist}(S_1, S_2) \leq \kappa$. Indeed, such a solution S_2 exists for all choices of blockers B' . This can be seen by applying the following construction: Repeat exactly the same construction as for S_1 , except that the set Y' is not chosen arbitrarily. Instead, choose Y' by letting $Y' := f_\varphi^{-1}(B')$. Note that by the definition of B , the set Y' is well-defined, i.e. $Y' \subseteq Y$, $|Y'| \leq \Gamma$, and $f_\varphi(Y') = B'$. Repeating the same construction, assignment A_X stays the same, but we obtain different assignments A'_Y, A'_Z with $A'_Y \cap Y' = \emptyset$ and $\varphi(A_X, A'_Y, A'_Z) = 1$. We let $A_2 := A_X \cup A'_Y \cup A'_Z$. Again, by the SSP property there exists a solution $S_2 \in \mathcal{S}(I_\Pi)$ such that $S_2 \cap f_\varphi(\mathcal{U}(\varphi)) = f_\varphi(A_2)$. Therefore we have

$$\emptyset = A_2 \cap Y' = f_\varphi(A_2) \cap f_\varphi(Y') = (S_2 \cap f_\varphi(\mathcal{U}(\varphi))) \cap B' = S_2 \cap B'.$$

Note that A_X is the same in both constructions. Therefore $S_1 \cap f_\varphi(L_X) = S_2 \cap f_\varphi(L_X)$, and hence by the definition of a blow-up reduction, we have $\text{dist}(S_1, S_2) \leq \beta_\varphi = \kappa$. In total, we have shown that S_1 is a solution of Π such that for every blocker $B' \subseteq B$, with $|B'| \leq \Gamma$, there exists a good solution S_2 . This shows that $I_{\text{RR}\Pi}$ is YES-instance.

It remains to consider the reverse direction:

$$I_{\text{RR}\Pi} = (I, B, \Gamma', \kappa) \text{ is a YES-instance} \Rightarrow I_{\text{RAS}} \text{ is a YES-instance}$$

The strategy is very similar, with the difference that we use the SSP property and the blow-up property in the reverse direction. Consider some solution $S_1 \in \mathcal{S}(I_\Pi)$ which satisfies

$$\forall B' \subseteq B, |B'| \leq \Gamma' : \exists S_2 \in \mathcal{S}(I_\Pi), S_2 \cap B' = \emptyset : \text{dist}(S_1, S_2) \leq \kappa.$$

We have to show that there exists an assignment $A_X \subseteq X \cup \bar{X}$ such that for all subsets $Y' \subseteq Y$ there are assignments A_Y, A_Z with $A_Y \cap Y' = \emptyset$ and $\varphi(A_X, A_Y, A_Z) = 1$. Indeed, to define A_X , consider solution S_1 . By the SSP property, since $S_1 \in \mathcal{S}(I_\Pi)$, the set $f_\varphi^{-1}(S_1) \subseteq \mathcal{U}(\varphi)$ is a set of literals which satisfies φ . We let A_X be the restriction of that satisfying assignment to the variables in X . More formally, we let $A_1 := f_\varphi^{-1}(S_1)$.

We claim that this assignment A_X proves that I_{RAS} is a YES-instance. Indeed, let some set $Y' \subseteq Y$, with $|Y'| \leq \Gamma$, be given. We define the blocker to be $B' := f_\varphi(Y')$. Observe that $B' \subseteq f_\varphi(Y) = B$, and $|B'| \leq \Gamma$. Therefore there exists $S_2 \in \mathcal{S}(I)$ such that $S_2 \cap B' = \emptyset$ and $\text{dist}(S_1, S_2) \leq \kappa$. We can again interpret the solution S_2 using the SSP property as an assignment $A_2 := f_\varphi^{-1}(S_2)$. By the SSP property, this assignment satisfies φ , that is $A_2 \in \mathcal{S}(\varphi)$. Since $\text{dist}(S_1, S_2) \leq \kappa = \beta_\varphi$, we have $S_1 \cap f_\varphi(Y) = S_2 \cap f_\varphi(Y)$ by the property of a blow-up reduction. This in turn implies $A_2 \cap (X \cup \bar{X}) = A_X$. Finally, we have $\emptyset = S_2 \cap B' = f_\varphi^{-1}(S_2) \cap f_\varphi^{-1}(B') = A_2 \cap Y'$. This shows that I_{RAS} is a YES-instance, and concludes the proof.

Polynomial Time The instance $I_{\text{RR-II}} = (I_{\text{II}}, B, \Gamma', \kappa)$ can indeed be constructed in polynomial time. The nominal instance I_{II} can be computed polynomially, since the map g in the SSP reduction can be computed polynomially. The set B can be computed polynomially, since the map f_φ in the SSP reduction can be computed polynomially. The number $\kappa = \beta_\varphi$ can be computed polynomially by definition of a blow-up SSP reduction. The number Γ remains the same. □

We have shown that the combinatorial version is indeed Σ_3^p -hard as long as the nominal SSP problem II is SSP-NP-hard with a blow-up SSP reduction from SATISIFIABILITY. This however does not directly imply that the linear optimization version RR-II is also Σ_3^p -hard. We remedy this problem with the following short reduction that simulates the set of blockable elements in COMB. RR-II with the cost functions in RR-II.

Theorem 7.3. *For all LOP problems $\text{II} \in \text{NP}$ with the property that the SSP problem derived from it is blow-up SSP reducible from 3SAT, the recoverable robust version RR-II is Σ_3^p -complete.*

Proof. Let $\text{II} = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be the LOP problem and RR-II be the corresponding recoverable robust version. The containment of RR-II in Σ_3^p follows from Theorem 7.1 by an analogous argument.

For the hardness, let $\text{II}' = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be the derived SSP problem from the LOP problem II . Remember, by definition of derived SSP problems, the set of solutions is defined by $\mathcal{S}(I) = \{F \in \mathcal{F}(I) : d^{(I)}(F) \leq t^{(I)}\}$. Now, let COMB. RR-II' be the corresponding combinatorial recoverable robust version. By assumption that the SSP problem II' is SSP blow-up reducible from SATISIFIABILITY and Theorem 7.2, COMB. RR-II' is Σ_3^p -hard.

We want to reduce COMB. RR-II' to RR-II. For this consider a COMB. RR-II'-instance $(I', U', t', B', \Gamma', \kappa')$, which we transform to the RR-II-instance $(I, U, c_1, \underline{c}, \bar{c}, t, \Gamma, \kappa)$. The instance and the universe stay the same, i.e. $I = I'$ and $U = U'$. We define the first stage cost function to be $c_1 = d^{(I)}$. We further have to model the blocker with the cost functions \underline{c} and \bar{c} . For this, we define the cost functions such that the blockable elements can be correctly blocked by choosing the cost function \underline{c} and \bar{c} correspondingly:

$$\begin{aligned} \underline{c}(u) &= d^{(I)}(u), \quad u \in \mathcal{U}(I) \\ \bar{c}(u) &= \begin{cases} d^{(I)}(u), & u \in \mathcal{U}(I) \setminus B \\ 2t^{(I)} + 1, & u \in B. \end{cases} \end{aligned}$$

At last, we set $\Gamma' = \Gamma$, $\kappa' = \kappa$ and $t' = 2t^{(I)}$. This completes the description of the instance of RR-II.

To prove the correctness of this reduction, we have to show that the following inequality holds if and only if $(I', U', t', B', \Gamma', \kappa')$ a YES-instance:

$$\min_{S_1 \in \mathcal{F}(I)} \max_{c_2 \in C_\Gamma} \min_{\substack{S_2 \in \mathcal{F}(I) \\ \text{dist}(S_1, S_2) \leq \kappa}} c_1(S_1) + c_2(S_2) \leq t'. \quad (7.2)$$

For the first direction, assume that $(I', U', B', \Gamma', \kappa')$ is a YES-instance. Accordingly, it holds that $|B'| \leq \Gamma'$. Furthermore, there are solutions $S_1, S_2 \subseteq U$ with $S_1, S_2 \in \mathcal{S}(I)$ such that $\text{dist}(S_1, S_2) \leq \kappa$ and $S_2 \cap B = \emptyset$. Since $\mathcal{S}(I) = \{F \in \mathcal{F}(I) : d^{(I)}(F) \leq t^{(I)}\}$, we have that $S_1, S_2 \in \mathcal{F}(I)$ and $d^{(I)}(S_1) + d^{(I)}(S_2) \leq 2t^{(I)}$. At last, observe that the blocked elements $u \in B$ are assigned a cost of $2t^{(I)} + 1$. Thus, these cannot be part of a second stage solution S_2 in RR-II

due to $t' = 2t^{(I)}$. Overall, all conditions on Equation (7.2) are fulfilled and $c_1(S_1) + c_2(S_2) \leq t'$ holds.

For the second direction, assume that Equation (7.2) holds. That is, there is an $S_1 \subseteq \mathcal{U}(I)$ and an $S_2 \subseteq \mathcal{U}(I)$ with $S_1, S_2 \in \mathcal{F}(I)$ such that $\text{dist}(S_1, S_2) \leq \kappa$. Furthermore, the set of cost functions C_Γ , defined by all cost functions c_2 such that $c_2(u) = \underline{c}(u) + \delta_u(\bar{c}(u) - \underline{c}(u))$, $\delta_u \in \{0, 1\}$, and $\sum_{u \in \mathcal{U}(I)} \delta_u \leq \Gamma$, guarantees that at most Γ elements have costs of $2t^{(I)} + 1$ while all others have $d^{(I)}(u)$. Since $t' = 2t^{(I)}$, C_Γ simulates all possible blockers B in the combinatorial version and it follows $S_2 \cap B = \emptyset$. Moreover, we have that $\mathcal{S}(I) = \{F \in \mathcal{F}(I) : d^{(I)}(F) \leq t^{(I)}\}$, thus S_1 and S_2 are not only feasible solutions but it also holds that $S_1, S_2 \in \mathcal{S}(I)$. All conditions of the solution pair (S_1, S_2) are fulfilled to be a valid solution and we have a YES-instance for the combinatorial version. Consequently, the reduction is correct.

As last step, we have to prove that the reduction is polynomial time computable. This is indeed the case because the instance remains the same as well as the parameters Γ and κ . All additional numbers to define the cost function \bar{c} are $2t^{(I)} + 1$, which is at most linear in the input because $\Pi \in \text{NP}$. This also holds for the new threshold $t = 2t^{(I)}$.

This concludes the proof and we have shown that RR- Π is indeed Σ_3^P -complete. \square

7.3.1 Blow-Up SSP Reductions for Various Problems

With the theorems proven, we want to apply this framework to several well-known problems, in particular to 3-satisfiability, vertex cover, independent set, subset sum, directed Hamiltonian path, two directed disjoint path and Steiner tree. In order to just convey the intuition how a blow-up SSP reduction works, we only present the reduction from satisfiability to vertex cover and defer the other reductions to Chapter 9. Furthermore, we do not prove the correctness of the actual reduction as this is already presented in the original work, but we prove the correctness of the SSP and blow-up property. In Figure 7.1, the tree of all presented reductions can be found beginning at SATISFIABILITY.

For a blow-up SSP reduction, we need the following mappings: The polynomial-time many-one reduction $g : \{0, 1\}^* \rightarrow \{0, 1\}^*$, the injective SSP mapping $(f_I)_{I \in \mathcal{I}} : \mathcal{U}(I) \rightarrow \mathcal{U}'(g(I))$ and the polynomial-time computable blow-up factor $\beta_I \in \mathbb{N}$ such that $f(L_b) \cap S_1 = f(L_b) \cap S_2 \Leftrightarrow |S_1 \Delta S_2| \leq \beta_I$ for all solutions $S_1, S_2 \in \mathcal{S}'(g(I))$ of $g(I)$.

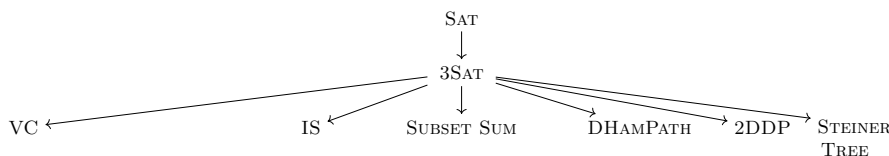


Figure 7.1: The tree of SSP blow-up reductions for all considered problems.

We start by defining the 3satisfiability problem in the SSP framework as presented in Chap-

ter 3. Then, we define the vertex cover problem in the SSP framework.

3SATISFIABILITY

Instances: Literal Set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$, Clauses $C \subseteq L^3$.

Universe: $L =: \mathcal{U}$.

Solution set: The set of all sets $L' \subseteq \mathcal{U}$ such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c| \geq 1$ for all $c \in C$.

VERTEX COVER

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Solution set: The set of all vertex covers of size at most k .

Now, we can use a modification of the reduction by Garey and Johnson [GJ79] from SAT to VERTEX COVER as blow-up SSP reduction. We first describe the original reduction and then argue why it is SSP. At last, we show how to modify the reduction to a blow-up SSP reduction with a blow-up factor β_I .

Let (L, C) be the 3SAT instance of literals L and clauses C . The corresponding VERTEX COVER instance (G, k) with graph $G = (V, E)$ and integer k is then constructed as follows. The reduction maps each literal ℓ to a vertex v_ℓ and we denote the set of all these vertices v_ℓ by W . Then the vertices v_ℓ and $v_{\bar{\ell}}$ of a literal ℓ and its negation $\bar{\ell}$ are connected by the edge $\{v_\ell, v_{\bar{\ell}}\}$. Next, for all clauses $c \in C$ there is a 3-clique. That is, for every literal ℓ in the clause c , there is a vertex v_ℓ^c , which is connected to all other vertices $v_{\ell'}^c$ with $\ell' \in c$ and $\ell' \neq \ell$. At last, for all literal vertices v_ℓ we add an edge $\{v_\ell, v_\ell^c\}$ to all clause vertices related to ℓ . The threshold of the vertex cover instance is then set to $k = |L|/2 + 2|C|$. An example of the reduction can be found in Figure 7.2.

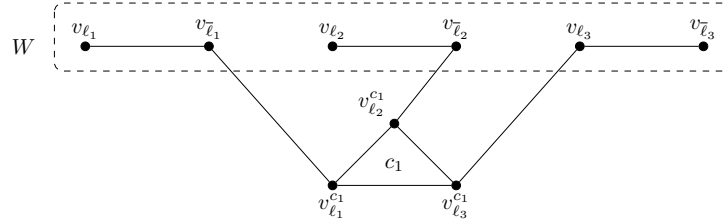


Figure 7.2: Classic reduction of 3SAT to VERTEX COVER for $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$.

Claim 7.1. *The reduction from above from 3SAT to VERTEX COVER is an SSP reduction.*

Proof. The reduction is an SSP reduction because each literal ℓ corresponds to exactly one vertex v_ℓ and the literal ℓ is in the satisfiability solution if and only if the corresponding vertex v_ℓ is in the vertex cover solution. Thus, we are able to prove the following equality, which confirm that the reduction is SSP.

$$\begin{aligned}
 \{f(S) : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{3\text{SAT}}\} &= \{\{f(\ell) : \ell \in S\} : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{3\text{SAT}}\} \\
 &= \{\{v_\ell : \ell \in S\} : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{3\text{SAT}}\} \\
 &= \{\{v_\ell : v_\ell \in S' \cap f(L)\} : S' \cap f(L) \subseteq W \text{ s.t. } S' \in \mathcal{S}_{\text{VC}}\} \\
 &= \{S' \cap f(L) : S' \in \mathcal{S}_{\text{VC}}\}.
 \end{aligned}$$

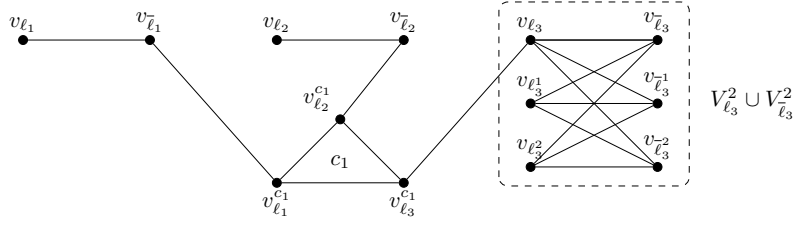


Figure 7.3: The graph $G'(L_b, \beta_I)$ with $L_b = \{\ell_3, \bar{\ell}_3\}$ and $\beta_I = 2$ of the instance $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$. The blow-up gadget of the literals ℓ_3 and $\bar{\ell}_3$ consisting of vertex sets $V_{\ell_3}^2 \cup V_{\bar{\ell}_3}^2$ is outlined with dashed lines.

The correctness proof of the reduction is presented in [GJ79]. ◀

We have shown that the original reduction by Garey and Johnson is an SSP reduction. For a modification to a blow-up SSP reduction, we introduce a blow-up gadget that we attach to each pair of blow-up literals $\ell, \bar{\ell} \in L_b$. Introducing a blow-up gadget that is attached to the blow-up literals is a standard technique to construct blow-up SSP reductions and we use it for all of the following blow-up SSP reductions. The idea is to build up equivalence classes Q_ℓ and $Q_{\bar{\ell}}$ of universe elements for both of the blow-up literals ℓ and $\bar{\ell}$ to which the gadget is connected to. Such a class Q_ℓ has the property that ℓ is in the solution if and only if all elements of Q_ℓ are in the solution. In other words, if one wants to switch from ℓ to $\bar{\ell}$, then additionally all elements from the equivalence class Q_ℓ have to be switched to $Q_{\bar{\ell}}$. Furthermore, a blow-up gadget is variable in size without influencing the rest of the construction, i.e. for each instance we are able to find a large enough β_I that locally guarantees that ℓ is not switchable to $\bar{\ell}$ while retaining a small distance between both solutions.

Given the distance measure, β_I for every possible 3SAT instance I and blow-up literals L_b , we now describe the blow-up gadget. The blow-up gadget for $\ell, \bar{\ell} \in L_b$ is a duplication of the literal vertices v_ℓ and $v_{\bar{\ell}}$ forming a complete bipartite graph K_{β_I+1, β_I+1} between the vertex sets $V_\ell^{\beta_I} = \{v_\ell, v_{\ell^1}, \dots, v_{\ell^{\beta_I}}\}$ and $V_{\bar{\ell}}^{\beta_I} = \{v_{\bar{\ell}}, v_{\bar{\ell}^1}, \dots, v_{\bar{\ell}^{\beta_I}}\}$ as depicted in Figure 7.3. Then, we have the equivalence classes $Q_\ell = \{v_\ell, v_{\ell^1}, \dots, v_{\ell^{\beta_I}}\}$ and $Q_{\bar{\ell}} = \{v_{\bar{\ell}}, v_{\bar{\ell}^1}, \dots, v_{\bar{\ell}^{\beta_I}}\}$. Now, consider the graph G as described by Garey and Johnson and modify it the following way: For each pair $\ell, \bar{\ell} \in L_b$, we identify the two vertices $v_\ell, v_{\bar{\ell}}$ in G and in the blow-up gadget for $\ell, \bar{\ell} \in L_b$ and merge them together. This results in the graph

$$G'(L_b, \beta_I) = (V(G) \cup \bigcup_{\ell \in L_b} V_\ell^{\beta_I}, E(G) \cup \bigcup_{\ell \in L_b} \{\{a, b\} \mid a \in V_\ell^{\beta_I}, b \in V_{\bar{\ell}}^{\beta_I}\})$$

The threshold for the vertex cover is then increased such that $\beta_I + 1$ vertices (instead of only one) for every blow-up literal pair can be taken into the solution, i.e.

$$k'(\beta_I) = (\beta_I + 1) \cdot |L_b|/2 + |L \setminus L_b|/2 + 2|C|.$$

Claim 7.2. For the three choices of dist from Subsection 7.2.1 and $\beta_I := |V(G)|$, $G'(L_b, \beta_I)$ and $k'(\beta_I)$ describe a blow-up SSP reduction from 3SAT to VERTEX COVER.

Proof. We show that $(g, (f_I)_{I \in \mathcal{I}}, (\beta_I)_{I \in \mathcal{I}})$ with

$$g(I) = (G'(L_b, \beta_I), k'(\beta_I)), \quad f(\ell) = v_\ell, \quad \text{and} \quad \beta_I = |V(G)|$$

is a blow-up SSP reduction from 3SAT to Π . We start by showing that the reduction is still SSP.

For this, we first make the observation that for a bipartite graph K_{β_I+1, β_I+1} between the vertex sets $V_\ell^{\beta_I}$ and $V_{\bar{\ell}}^{\beta_I}$, there are exactly two vertex covers of size $\beta_I + 1$: either one takes all v_{ℓ^i} together with v_ℓ or all $v_{\bar{\ell}^i}$ together with $v_{\bar{\ell}}$ into the vertex cover. Otherwise, there is at least one edge $\{v, w\}$ with $v \in V_\ell^{\beta_I}$ and $w \in V_{\bar{\ell}}^{\beta_I}$ not covered.

With this observation, we can follow that every vertex cover solution still consists of

- (1) exactly one of the original vertices v_ℓ and $v_{\bar{\ell}}$ for $\ell, \bar{\ell} \in L$, and
- (2) the β_I additional vertices $\{v_{\ell^i} : 1 \leq i \leq \beta_I\}$ of the blow-up gadget if and only if $v_\ell \in L_b$ is in the vertex cover solution, and
- (3) exactly two of the three vertices $v_{\ell_1^{c_j}}, v_{\ell_2^{c_j}}, v_{\ell_3^{c_j}}$ for $c_j \in C$.

Thus, the original injective SSP mapping $f(\ell) = v_\ell$ is still a valid SSP mapping for this reduction.

To show that the blow-up property holds, we have to show that for $\beta_I := |V(G)|$ and for all solutions of $g(I)$, $S_1, S_2 \in \mathcal{S}'(g(I))$,

$$f(L_b) \cap S_1 = f(L_b) \cap S_2 \Leftrightarrow \text{dist}(S_1, S_2) \leq \beta_I. \quad (7.3)$$

We begin with analyzing the maximum distances of two solutions $S_1, S_2 \in \mathcal{S}'(g(I))$ for the three distance measures if S_1 and S_2 agree on each blow-up gadget. To reach the maximum distance between two solutions S_1 and S_2 for the three distance measures, S_1 includes all the v_ℓ for all $\ell \in L \setminus L_b$, while S_2 includes all the opposite $v_{\bar{\ell}}$. Additionally, S_1 includes a different pair $v_{\ell_1^{c_j}}, v_{\ell_2^{c_j}}$ for all $c_j \in C$ in comparison to S_2 . Thus, all $v_{\bar{\ell}}$ of $\bar{\ell} \in L \setminus L_b$ and $v_{\ell_3^{c_j}}$ for all $c_j \in C$ have to be added (while deleting v_ℓ and one of $v_{\ell_1^{c_j}}, v_{\ell_2^{c_j}}$). In conclusion, $\beta_I = |V(G)|$ is an upper bound for the maximal possible κ -addition and κ -deletion distance ($|C| + |L \setminus L_b|/2$) as well as for the maximal possible *Hamming distance* (here one has to count for all deletions *and* additions such that $\beta_I = |V(G)| \geq 2|C| + |L \setminus L_b| \geq |S_1 \Delta S_2|$ is also sufficient). It follows that $f(L_b) \cap S_1 = f(L_b) \cap S_2 \Rightarrow \text{dist}(S_1, S_2) \leq \beta_I$.

We conclude with the situation that S_1 and S_2 do not agree on each blow-up gadget. Then corresponding to the analysis above for κ -addition and κ -deletion, we reach at least a distance of $|V(G)| + 1 > \beta_I$ and for the Hamming distance we reach at least $2|V(G)| + 2 > \beta_I$. It follows that $f(L_b) \cap S_1 \neq f(L_b) \cap S_2 \Rightarrow \text{dist}(S_1, S_2) > \beta_I$.

Clearly β_I and thus $G'(L_b, \beta_I)$ and $k'(\beta_I)$ are polynomial time computable. This completes the correctness proof of the reduction. \blacktriangleleft

In conclusion, we have shown that the existing reduction from satisfiability to vertex cover enhanced with the presented blow-up gadget is a blow-up SSP reduction. It follows that the combinatorial as well as the linear optimization version of recoverable robust vertex cover is Σ_3^p -complete.

7.4 The Issue of Transitivity: Preserving the Blow-up Gadget

Since blow-up SSP reductions require more structure, we unfortunately lose transitivity in comparison to normal SSP reductions, which are transitive. However, we would like to reuse an existing blow-up SSP reduction such that we do not need to start every reduction at 3SATISFIABILITY and additionally construct blow-up gadgets. For this, we introduce blow-up preserving SSP reductions, which are transitive and preserve the structure of the blow-up gadgets. The idea is that we only need to show that there is a blow-up SSP reduction for the first reduction in

the reduction chain and then the reduction chain can be prolonged by adding further problems, between which there are blow-up preserving SSP reductions.

The idea of a blow-up preserving reduction between problem Π and problem Π' is that we partition the universe into three sets, the set of elements which originate in the problem Π , a set of elements U_{off} , which are never of a solution of the instance of Π' , and a set of elements U_{on} , which are part of every solution of the instance of Π' . Therefore, every solution S in Π has a correspondent solution $S' = f(S) \cup U_{on}$ by applying the injective function f . Since the distance measures that we consider are invariant on injective functions and union, the distance between the solutions in problem Π is the same as in problem Π' . Correspondingly, the blow-up SSP reduction from 3SATISFIABILITY to Π can be extended to Π' .

From a different point of view, a blow-up preserving SSP reduction can be understood as an SSP reduction, which “adds” only two kind of new elements to the universe: Those that are trivially contained in every (optimal) solution, and those that are never contained in an (optimal) solution. It is intuitively easy to understand that compared to the old instance, the newly “added” elements cannot influence the term $\text{dist}(S_1, S_2)$. Hence Σ_3^P -hardness of recoverable problems is maintained.

Definition 7.5 (Blow-Up Preserving SSP Reduction). *Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ and $\Pi' = (\mathcal{I}', \mathcal{U}', \mathcal{S}')$ be two SSP problems. Then, there is a blow-up preserving SSP reduction from Π to Π' if there exists an SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ such that for all instances $I \in \mathcal{I}$ the following holds: For all elements $u' \in \mathcal{U}'(g(I))$, either*

- ▶ $u' \in f(\mathcal{U}(I))$ or
- ▶ for all $S' \in \mathcal{S}'$: $u' \notin S'$, i.e. $u' \in U_{off}$, or
- ▶ for all $S' \in \mathcal{S}'$: $u' \in S'$, i.e. $u' \in U_{on}$.

Remark. This definition is more restrictive than it needs to be because we do not need to force that the distance stays exactly the same. It is also possible to increase the distance while applying the reduction in a controlled manner. More precisely, we can allow γ of the elements u' to be in $U' \setminus (f(\mathcal{U}) \cup U_{off} \cup U_{on})$ and we relax the backward direction by $f(L_b) \cap S_1 = f(L_b) \cap S_2 \Rightarrow |S_1 \Delta S_2| > \beta + \gamma$. This is a more general and more complicated definition, nevertheless, it is not necessary for the problems that are presented in this chapter. For simplicity, we use the stricter variant for the rest of the chapter.

Theorem 7.4. *Let Π be an SSP-NP-complete problem for which a blow-up SSP reduction from 3SAT exists. Then every problem Π' , which is blow-up preserving SSP reducible from Π , is blow-up SSP reducible from 3SAT.*

Proof. We present a blow-up SSP reduction from 3SAT to Π' . Let $3\text{SAT} = (\mathcal{I}_{3\text{SAT}}, \mathcal{U}_{3\text{SAT}}, \mathcal{S}_{3\text{SAT}})$, $\Pi = (\mathcal{I}_{\Pi}, \mathcal{U}_{\Pi}, \mathcal{S}_{\Pi})$ and $\Pi' = (\mathcal{I}_{\Pi'}, \mathcal{U}_{\Pi'}, \mathcal{S}_{\Pi'})$. Moreover, we are given a *blow-up SSP reduction* $(g, (f_I)_{I \in \mathcal{I}_{3\text{SAT}}}, (\beta_I)_{I \in \mathcal{I}_{3\text{SAT}}})$ from 3SAT to Π and a *blow-up preserving SSP reduction* $(g', (f'_I)_{I \in \mathcal{I}_{\Pi}})$ from Π to Π' .

We concatenate the blow-up SSP reduction $(g, (f_I)_{I \in \mathcal{I}_{3\text{SAT}}}, (\beta_I)_{I \in \mathcal{I}_{3\text{SAT}}})$ and the blow-up preserving SSP reduction $(g', (f'_I)_{I \in \mathcal{I}_{\Pi}})$ to obtain a new blow-up SSP reduction $(g' \circ g, (f'_{g(I)} \circ f_I)_{I \in \mathcal{I}_{3\text{SAT}}}, (\beta_I)_{I \in \mathcal{I}_{3\text{SAT}}})$ from 3SAT to Π' . The resulting relation between the three problem universes is depicted in Figure 7.4.

Blow-up SSP reductions and blow-up preserving SSP reductions are both SSP reductions. Furthermore, SSP reductions are transitive (Lemma 3.1) and thus the concatenation of both reductions is still an SSP reduction.

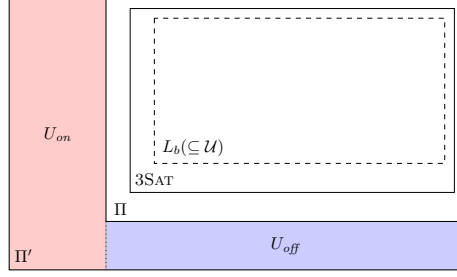


Figure 7.4: The relation between 3SAT, Π and Π' in a blow-up SSP reduction from 3SAT to Π and a blow-up preserving SSP reduction from Π and Π' . The blow-up preserving SSP reduction maps the universe of Π into the universe elements of Π' . Because the original 3SAT universe is part of the universe of Π , the blow-up literals L_b are also mapped via the functions f and f' into the universe of Π' . Furthermore, all additional elements in Π' are either part of U_{on} , which are always part of a solution in Π' , or U_{off} , which are never part of the solution in Π' .

It remains to show that the blow-up property holds. We show this by using the same polynomially computable β_I of the blow-up SSP reduction $(g, (f_I)_{I \in \mathcal{I}_{3SAT}}, (\beta_I)_{I \in \mathcal{I}_{3SAT}})$ from 3SAT to Π . In particular, we have to show that the statement

$$\forall S_1, S_2 \in \mathcal{S}_{\Pi} : f(L_b) \cap S_1 = f(L_b) \cap S_2 \Leftrightarrow \text{dist}(S_1, S_2) \leq \beta_I. \quad (7.4)$$

holds if and only if the following statement holds:

$$\forall S'_1, S'_2 \in \mathcal{S}_{\Pi'} : f' \circ f(L_b) \cap S'_1 = f' \circ f(L_b) \cap S'_2 \Leftrightarrow \text{dist}(S'_1, S'_2) \leq \beta_I. \quad (7.5)$$

We first remark that $\{S' : S' \in \mathcal{S}_{\Pi'}\} = \{f'(S) \dot{\cup} U_{on} : S \in \mathcal{S}_{\Pi}\}$. Thus, there is a bijection between solution sets $S \in \mathcal{S}_{\Pi}$ of problem Π and the solution sets $S' \in \mathcal{S}_{\Pi'}$ of problem Π' defined by $h : \mathcal{S}_{\Pi} \rightarrow \mathcal{S}_{\Pi'}, S \mapsto f'(S) \dot{\cup} U_{on}$. With the help of this bijection, we can show that the distance between the solution pair (S_1, S_2) of problem Π and the solution pair (S'_1, S'_2) of problem Π' stays the same.

$$\text{dist}(S'_1, S'_2) = \text{dist}(h(S_1), h(S_2)) \quad (7.6)$$

$$= \text{dist}(f'(S_1) \dot{\cup} U_{on}, f'(S_2) \dot{\cup} U_{on}) \quad (7.7)$$

$$= \text{dist}(f'(S_1), f'(S_2)) \quad (7.8)$$

$$= \text{dist}(S_1, S_2) \quad (7.9)$$

For Equation (7.8), we make use of the invariance of union of the distance measure. Furthermore Equation (7.9) holds, because the function f' is injective since the blow-up preserving SSP reduction is also an SSP reduction. Consequently, using the same β_I guarantees that the equivalence of Equation (7.4) and Equation (7.5) holds.

The transformation is polynomial time computable because the mappings g and g' are polynomial time computable as well as the β_I as given in the blow-up SSP reduction $(g, (f_I)_{I \in \mathcal{I}_{3SAT}}, (\beta_I)_{I \in \mathcal{I}_{3SAT}})$. This concludes the proof of the theorem. \square

Corollary 7.1. *Let Π be an SSP-NP-hard problem for which a blow-up SSP reduction from 3SAT-ISFIABILITY exists. Then for all SSP-NP-hard problems Π' that are blow-up preserving SSP reducible from Π , the recoverable robust variant COMB. RR- Π' is Σ_3^p -hard.*

We have shown that blow-up preserving SSP reductions are a possible tool to build up a reduction chain that begins at a blow-up SSP reduction. Next, we prove that those chains are extendable by adding further blow-up preserving SSP reductions by showing the transitivity of these reductions.

Lemma 7.1. *Blow-up preserving SSP reductions are transitive.*

Proof. Let Π^1 , Π^2 and Π^3 be three SSP problems such that Π^1 is blow-up preserving SSP reducible to Π^2 by (g, f) and Π^2 is blow-up preserving SSP reducible to Π^3 by (g', f') . We aim to show that Π^1 is blow-up preserving SSP reducible to Π^3 .

Consider an instance (I^1, U^1) of Π^1 , which is mapped by the blow-up preserving SSP reduction (I^2, U^2) . Then,

$$I^2 = g(I^1) \text{ and } U^2 = f(U^1) \dot{\cup} U_{on}^2 \dot{\cup} U_{off}^2.$$

We can then use the second reduction (g', f') to derive

$$I^3 = g'(g(I^1)) \text{ and } U^3 = f'(f(U^1)) \dot{\cup} f'(U_{on}^2) \dot{\cup} f'(U_{off}^2) \dot{\cup} U_{on}^3 \dot{\cup} U_{off}^3.$$

Then, we can reassign $U_{on} = f'(U_{on}^2) \dot{\cup} U_{on}^3$ and $U_{off} = f'(U_{off}^2) \dot{\cup} U_{off}^3$ as follows to derive

$$U^3 = f'(f(U^1)) \dot{\cup} U_{on} \dot{\cup} U_{off}. \quad (7.10)$$

Because g and g' are transitive as well as $f' \circ f$ is injective (since f and f' are injective), we have an SSP reduction, which is also blow-up preserving because of Equation (7.10). \square

7.4.1 Blow-up Gadgets for various Problems

In this subsection, we apply the developed framework of blow-up-preserving SSP reductions. In Subsection 7.3.1, we have seen that vertex cover, 3Sat, independent set, subset sum, directed Hamiltonian path, two directed disjoint path and Steiner tree are blow-up SSP reducible from satisfiability. Next, we show that one can find blow-up preserving reductions starting from these problems to show Σ_3^p -hardness of further recoverable robust problems. Again, we defer most of the reductions to Chapter 9. However to convey the intuition, how a blow-up preserving SSP reduction works, we present a reduction from vertex cover to dominating set. The complete tree of reductions can be found in Figure 7.5. Correspondingly, we can state that the recoverable robust problem based on the nominal problems in the reduction tree are Σ_3^p -complete.

Theorem 7.5. *The recoverable robust version of the following nominal problems is Σ_3^p -complete: SATISFIABILITY, 3SATISFIABILITY, VERTEX COVER, DOMINATING SET, SET COVER, HITTING SET, FEEDBACK VERTEX SET, FEEDBACK ARC SET, UNCAPACITATED FACILITY LOCATION, P-CENTER, P-MEDIAN, INDEPENDENT SET, CLIQUE, SUBSET SUM, KNAPSACK, PARTITION, SCHEDULING, DIRECTED HAMILTONIAN PATH, DIRECTED HAMILTONIAN CYCLE, UNDIRECTED HAMILTONIAN CYCLE, TRAVELING SALESMAN PROBLEM, TWO DIRECTED VERTEX DISJOINT PATH, k-VERTEX DIRECTED DISJOINT PATH, STEINER TREE*

We define dominating set according to the SSP framework as follows.

DOMINATING SET

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Solution set: The set of all dominating sets of size at most k .

For a reduction from VERTEX COVER to DOMINATING SET, we use a modification of a folklore reduction. The vertex cover instance, consisting of a graph $G = (V, E)$ and an integer k ,

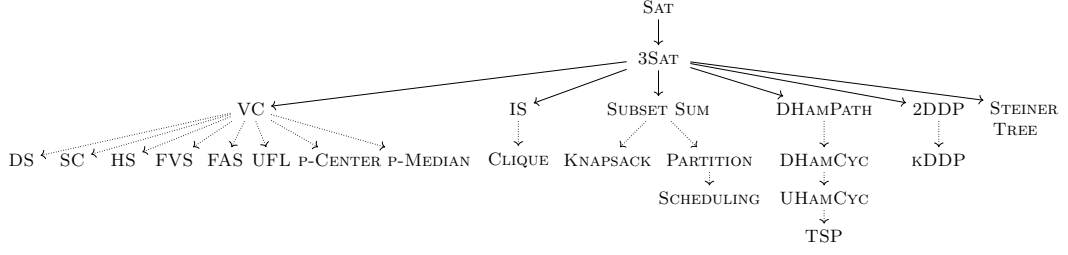


Figure 7.5: The tree of SSP reductions for all considered problems. While solid edges indicate that there is a blow-up SSP reduction between the problems, dotted edges represent a blow-up preserving SSP reduction between the problems.

is transformed to a dominating set instance, consisting of a graph $G' = (V', E')$ and an integer k' .¹ Every vertex $v \in V$ is mapped to a vertex $v' \in V'$, we denote the set of all these vertices v' by W . Every edge $\{v, w\} \in E$ is mapped to an edge $\{v', w'\} \in E'$. Furthermore for every edge $\{v, w\} \in E$, there are $|V| + 1$ additional vertices² vw_i , for $1 \leq i \leq |V| + 1$. These vertices are connected to both v' and w' , i.e. $\{\{v', vw_i\}, \{w', vw_i\} : 1 \leq i \leq |V| + 1\} \subseteq E'$. At last, we set $k' = k$. The transformation of one edge is depicted in Figure 7.6.

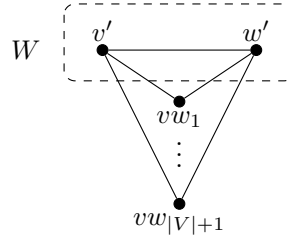


Figure 7.6: Modified reduction of VERTEX COVER to DOMINATING SET. This is the transformation for two vertices $v, w \in V$ connected by one edge $\{v, w\} \in E$.

Claim 7.3. *The reduction from above from VERTEX COVER to DOMINATING SET is an SSP reduction.*

Proof. This reduction is an SSP reduction, because every vertex $v \in V$ is mapped to a corresponding vertex $v' \in V'$, which is in the dominating solution if and only if vertex v is in the vertex cover solution. Formally,

$$\begin{aligned}
 \{f(S) : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{\text{VC}}\} &= \{\{f(v) : v \in S\} : S \subseteq V \text{ s.t. } S \in \mathcal{S}_{\text{VC}}\} \\
 &= \{\{v' : v \in S\} : S \subseteq V \text{ s.t. } S \in \mathcal{S}_{\text{VC}}\} \\
 &= \{\{v' : v' \in S' \cap f(V)\} : S' \cap f(V) \subseteq W \text{ s.t. } S' \in \mathcal{S}_{\text{DS}}\} \\
 &= \{S' \cap f(V) : S' \in \mathcal{S}_{\text{DS}}\}.
 \end{aligned}$$

The correctness proof of the reduction itself is presented in Chapter 9. ◀

¹For the sake of conciseness, we assume w.l.o.g. that the vertex cover instance is a connected graph. For a full proof for arbitrary instances we refer to Chapter 9.

²The original reduction uses only one additional vertex for each edge. However, this reduction is not an SSP reduction.

Finally, we have to show that the reduction is blow-up preserving. For this, we take a look at all vertices of V' and associate them with the sets $f(\mathcal{U})$, U_{on} and U_{off} .

Claim 7.4. *The reduction from above from VERTEX COVER to DOMINATING SET is a blow-up reduction.*

Proof. All vertices are mapped to their correspondence in V' . Moreover, all vertices of $\{vw_i : \{v, w\} \in E, 1 \leq i \leq |V| + 1\}$ are never part of a solution of size k' . Overall, it holds that $V' = W \cup \{vw_i : \{v, w\} \in E, 1 \leq i \leq |V| + 1\}$. Thus, we have

$$\begin{aligned} f(\mathcal{U}) &= f(V) = W, \\ U_{on} &= \emptyset, \\ U_{off} &= \{vw_i : \{v, w\} \in E, 1 \leq i \leq |V| + 1\}, \\ V' &= f(V) \dot{\cup} U_{on} \dot{\cup} U_{off}. \end{aligned}$$

◀

All in all, the reduction from above is a blow-up preserving SSP reduction and thus recoverable robust dominating set is Σ_3^P -complete.

Chapter 8

Minimum Cardinality Interdiction

8.1 Introduction

In this chapter, we return to interdiction problems in the form of *minimum cardinality interdiction*, which is arguably the most natural form of interdiction. We have already seen how the SSP framework can be applied to minimum cost interdiction problems in Chapter 4. In minimum cost interdiction, we are given some base problem (the so-called *nominal problem*) and we wish to find a subset of small costs such that this subset has a non-empty intersection with every optimal solution of the base problem. On the contrary, in minimum cardinality interdiction, we wish to find a *subset of minimum cardinality*. Consequently, we restrict the cost function of a minimum cost interdiction problem to be only of unit costs in the minimum cardinality interdiction case. As an example of the type of problems that this chapter is concerned with, consider the following two problems:

MIN CARDINALITY CLIQUE INTERDICTION

Input: Graph $G = (V, E)$

Task: Find a minimum-size subset $V' \subseteq V$ such that every maximum clique shares at least one vertex with V' .

MIN CARDINALITY HAMILTONIAN CYCLE INTERDICTION

Input: Graph $G = (V, E)$

Task: Find a minimum-size subset $E' \subseteq E$ such that every Hamiltonian cycle shares at least one edge with E' .

In particular, if in the above examples the set V' (respectively the set E') is deleted from the graph, the maximum clique size decreases (respectively, the graph becomes Hamiltonian-cycle-free). Hence the interdiction problem can be interpreted as the minimal effort required to destroy all optimal solutions.

Our Results. We introduce a meta-theorem that simultaneously classifies the complexity of minimum cardinality interdiction problems. This meta-theorem specifies a set of sufficient conditions for some nominal problem, which imply that the minimum cardinality interdiction problem becomes Σ_2^P -complete. For proving this meta-theorem, we extend the SSP framework presented in Chapter 3. Chapter 4 already provides such a result in the case where the action of interdicting an element is associated with so-called interdiction costs, which may be different for each element. We extend the existing ideas to the unit-cost case to derive said sufficient conditions. Concretely, we first define a general minimum cardinality interdiction problem by embedding

it in the SSP framework. Furthermore, we provide a general reduction from a minimum cost interdiction problem to a minimum cardinality interdiction problem. Thus, we show that the minimum cardinality interdiction problem is Σ_2^P -complete for the following nominal problems:

satisfiability, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median, independent set, clique, subset sum, knapsack, Hamiltonian path/cycle (directed/undirected), TSP, k -directed vertex disjoint path ($k \geq 2$), Steiner tree.

We remark that Σ_2^P -completeness was already known in the case of clique, independent set, and knapsack [CCLW13, Rut93, TCCH24]. Hence, our work is a generalization of these results.

8.2 Minimum Cardinality Interdiction Problems

In this section, we prove our Σ_2^P -completeness results regarding the minimum cardinality interdiction problem. Since we want to prove the theorem simultaneously for multiple problems at once, we require an abstract definition of the interdiction problem. For this, consider the following definition.

Definition 8.1 (Minimum Cardinality Interdiction Problem). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. The minimum cardinality interdiction problem associated to Π is denoted by MIN CARDINALITY INTERDICTION- Π and defined as follows: The input is an instance $I \in \mathcal{I}$ together with a number $k \in \mathbb{N}_0$. The question is whether*

$$\exists B \subseteq \mathcal{U}(I), |B| \leq k : \forall S \in \mathcal{S}(I) : B \cap S \neq \emptyset.$$

For the remainder of the chapter, it is helpful to imagine this problem as a game between two players: the *attacker* and the *defender*. That is, interdiction is an action performed by an attacker (or interdictor), who wishes to select a blocker of few elements to destroy all solutions. On the other hand, the defender wants to find a solution to the problem after the attacker selected a blocker. This leads to the following interpretation:

- ▶ The set $\mathcal{U}(I)$ contains all the elements the attacker is allowed to attack.
- ▶ The set $\mathcal{S}(I)$ contains all the solutions the attacker wants to destroy such that the defender is not able to find any solution. For example, this could be the set of all Hamiltonian cycles, the set of all cliques of a certain size, etc.

Therefore, the formulation of the base problem as SSP problem $(\mathcal{I}, \mathcal{U}, \mathcal{S})$ determines which elements the attacker can attack, which he cannot attack (e.g. edges/vertices of a graph), and what the attacker's goal is. We note that different formulations $(\mathcal{I}, \mathcal{U}, \mathcal{S})$ of the same problem are formally different SSP problems. They might be both SSP-NP-complete independent of each other, but require their own SSP-NP-completeness proof each. Finally, note that if the base problem is an LOP problem, then by definition $\mathcal{S}(I)$ is the set of feasible solutions below some threshold specified in the input. For example, applying Definition 8.1 to $\Pi = \text{CLIQUE}$ yields the following decision problem:

MIN CARDINALITY INTERDICTION-CLIQUE

Input: Graph $G = (V, E)$, numbers $k, t \in \mathbb{N}_0$

Question: Does there exist a subset $B \subseteq V$ of size $|B| \leq k$ such that every clique of size at least t shares at least one vertex with B ?

Some more technical details, concerning the subtle differences between different variants of interdiction referenced in the literature as well as concerning the question whether t can be chosen to be optimal are discussed in Subsection 8.2.2.

We start the complexity analysis of minimum cardinality interdiction problems by showing the containment in the class Σ_2^p , if the nominal problem is in NP. We then proceed with presenting the main result.

Theorem 8.1. *Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem in SSP-NP, then MIN CARDINALITY INTERDICTION- Π is in Σ_2^p .*

Proof. We provide a polynomial time algorithm V that verifies a specific solution y_1, y_2 of polynomial size for instance I such that

$$I \in L \Leftrightarrow \exists y_1 \in \{0, 1\}^{m_1} \forall y_2 \in \{0, 1\}^{m_2} : V(I, y_1, y_2) = 1.$$

With the \exists -quantified y_1 , we encode the blocker $B \subseteq \mathcal{U}(I)$. The encoding size of y_1 is polynomially bounded in the input size of Π because $|\mathcal{U}(I)| \leq \text{poly}(|x|)$. Next, we encode the solution $S \in \mathcal{S}(I)$ to the nominal problem Π using the \forall -quantified y_2 within polynomial space. This is doable because the problem Π is in SSP-NP. At last, the verifier V has to verify the correctness of the given solution provided by the \exists -quantified y_1 and \forall -quantified y_2 . Checking whether $|B| \leq t$ and $B \cap S \neq \emptyset$ is trivial and checking whether $S \in \mathcal{S}(I)$ is clearly in polynomial time because Π is in SSP-NP. It follows that MIN CARDINALITY INTERDICTION- Π is in Σ_2^p . \square

Next, we show the hardness of minimum cardinality interdiction problems under the condition that the nominal problem is NP-hard. For this, we introduce the concept of invulnerability reductions that helps us to grasp the problems in a unified approach. We describe this concept in the following subsection. Subsequently, we provide invulnerability reductions for a large set of problems in Section 8.3 with the goal to obtain the following main theorem of the chapter. The definitions of the following problems can be found in Chapter 9.

Theorem 8.2. *The problem MIN CARDINALITY INTERDICTION- Π is Σ_2^p -complete for all the following problems: independent set, clique, subset sum, knapsack, Hamiltonian path/cycle (directed/undirected), TSP, k -directed vertex disjoint paths ($k \geq 2$), Steiner tree, dominating set, set cover, hitting set, feedback vertex set, feedback arc set, uncapacitated facility location, p -center, p -median.*

We remark that the case of satisfiability deserves special attention, which is discussed more thoroughly in Section 8.4.

8.2.1 Invulnerability Reduction

Our proof strategy for each of the problems listed in Theorem 8.2 is essentially the same. In fact, we show that Theorem 8.2 is actually a consequence of the following, more powerful *meta-theorem*. This meta-theorem catches the essence of an invulnerability reduction.

Theorem 8.3. *Consider an SSP-NP-complete problem Π . If there exists a polynomial-time reduction g which receives as input a tuple (I, C, k) of an instance I of Π , some set $C \subseteq \mathcal{U}(I)$ and some $k \in \mathbb{N}_0$, and returns instances $I' := g(I, C, k)$ of Π , such that the following holds:*

$$\begin{aligned} \exists B \subseteq C : |B| \leq k \text{ and } B \cap S \neq \emptyset \forall S \in \mathcal{S}(I) \\ \Leftrightarrow \exists B' \subseteq \mathcal{U}(I') : |B'| \leq k \text{ and } B' \cap S' \neq \emptyset \forall S' \in \mathcal{S}(I'). \end{aligned}$$

Then MIN CARDINALITY INTERDICTION- Π is Σ_2^p -complete.

It would be nice to have Theorem 8.3 for all problems in the class SSP-NP_c, not only those who admit a function g with the properties as described above. However, we give a reasoning in Section 8.4 why such a generalization is not possible. The rest of this section is devoted to the proof of Theorem 8.3. For this proof, we start from the corresponding *combinatorial interdiction problem* as defined in Chapter 4, which we restate in the following. A combinatorial interdiction problem is a more general version of interdiction, where there is a set $C \subseteq \mathcal{U}(I)$ of so-called vulnerable elements. One can also interpret the set of vulnerable elements C as the elements that have cost of interdiction of 1 while all other elements $\mathcal{U}(I) \setminus C$ have a cost of interdiction of ∞ and a blocker of small costs is sought.

Definition 8.2 (Comb. Interdiction Problem, from Chapter 4). *Let an SSP problem $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be given. We define COMB. INTERDICTION-II as follows: The input is an instance $I \in \mathcal{I}$, a number $k \in \mathbb{N}_0$, and a set $C \subseteq \mathcal{U}(I)$. The set C is called the set of vulnerable elements. The question is whether*

$$\exists B \subseteq C, |B| \leq k : \forall S \in \mathcal{S}(I) : B \cap S \neq \emptyset.$$

It is proven in Chapter 4 that for every problem in SSP-NP_c, the combinatorial interdiction problem is Σ_2^p -complete. Now, let Π be in SSP-NP_c and g be a reduction such that

$$\begin{aligned} \exists B \subseteq C : |B| \leq k \text{ and } B \cap S \neq \emptyset \forall S \in \mathcal{S}(I) \\ \Leftrightarrow \exists B' \subseteq \mathcal{U}(I') : |B'| \leq k \text{ and } B' \cap S' \neq \emptyset \forall S' \in \mathcal{S}(I'), \end{aligned}$$

then g is a reduction from COMB. INTERDICTION-II to MIN CARDINALITY INTERDICTION-II. This is because the first line is equivalent to the statement that instance I is a YES-instance of COMB. INTERDICTION-II, and the second line is equivalent to the statement that I' is a YES-instance of MIN CARDINALITY INTERDICTION-II. It directly follows that MIN CARDINALITY INTERDICTION-II is Σ_2^p -complete. This completes the proof of Theorem 8.3.

We remark that while in some sense the proof is rather trivial, we still see a lot of value in explicitly stating a set of easy-to-check sufficient conditions that render some minimum-cardinality interdiction problem Σ_2^p -complete.

How can one find a function g with properties as described above? Oftentimes it is possible by employing the following natural idea: Given an instance of the comb. interdiction problem, let the set $D := \mathcal{U}(I) \setminus C$ be called the *invulnerable* elements. For each problem we explain separately that a gadget for the invulnerable elements in D exists. Intuitively speaking this gadget guarantees that an attacker, no matter which k elements of the universe they attack, can never render the elements of D unusable. On the other hand, we make sure that the *invulnerability gadgets* do not meaningfully change the set of solutions. In Section 8.3 many examples of such gadgets are presented. We remark that we are not the first to come up with this natural idea. For example, Zenklusen [Zen10] used the same idea in the context of matching interdiction.

8.2.2 Different Variants of Interdiction

In this section, we examine variants of interdiction problems that can be found in the literature. For this, we study the relation of our definition of a minimum cardinality interdiction problem and the existing variants. Additionally, we discuss the implications of the hardness of our minimum cardinality interdiction problems on the other variants.

1. Minimal Blocker Problem.

Input: Instance I with universe U , blocker cost function c , solution cost function d , and solution threshold τ

Task: Find the minimum-cost set $\min_{B \subseteq U} c(B)$ such that for all feasible solutions $F \in \mathcal{F}$ with $F \cap B = \emptyset$, we have $d(F) \leq \tau$.

2. Full Decision Variant of Interdiction.

Input: Instance I with universe U , blocker cost function c , blocker budget k , solution cost function d , and solution threshold τ

Task: Is there a set $B \subseteq U$ with $c(B) \leq k$ such that for all feasible solutions $F \in \mathcal{F}$ with $F \cap B = \emptyset$, we have $d(F) \leq \tau$?

3. Most Vital Elements Problem.

Input: Instance I with universe U , blocker cost function c , blocker budget k , and solution cost function d

Task: Find a set $B \subseteq U$ with $c(B) \leq k$ such that the costs of all feasible solutions $F \cap B = \emptyset$ are maximized, i.e. $\max_B \min_{F \in \mathcal{F}, F \cap B = \emptyset} d(F)$.

Note that all of these variants are based on LOP problems instead of SSP problems. An SSP problem, however, can always trivially be expressed as LOP problem by defining $d(u) = 0$ and $t = 0$ thus defining $\mathcal{S} = \mathcal{F}$. Our goal is to show that all of the variants from above are at least as hard as our formulation of *minimum cardinality interdiction* (Definition 8.1). This results in the following theorem.

Theorem 8.4. *Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be an LOP problem. Then the Most Vital Elements Problem of Π (for all problems Π in Theorem 8.2), the Minimal Blocker Problem of Π , and the Full Decision Variant of Interdiction of Π are at least as hard to compute as MIN CARDINALITY INTERDICTION-II.*

The rest of this section is devoted to the proof of this theorem. In our formulation of minimum cardinality interdiction, a set B is sought, which intersects every solution in the set \mathcal{F} as given by the corresponding LOP problem. We now have to distinguish between problems, which are naturally formulated as SSP problems (e.g. Hamiltonian cycle), and naturally formulated LOP problems (e.g. clique). For natural SSP problems, the solution set \mathcal{S} consists of all feasible solutions, i.e. there are no feasible solutions outside of \mathcal{S} due to the missing cost function d on the solution elements. Thus all of the three variants from above are generalizations of minimum cardinality interdiction:

- (1) The *minimal blocker problem* is the optimization version of the corresponding minimum cardinality interdiction problem.
- (2) The *full decision version of interdiction* is a generalization of the corresponding minimum cardinality interdiction problem because the latter assumes to have unit costs in the cost function c for all elements from U .
- (3) The *most vital elements problem* behaves the same as the full decision version of interdiction.

For natural LOP problems, basically the same holds, however, with a modified and a technically more intricate argumentation. Here we are given a feasible solution set \mathcal{F} and the solution set is defined by $\mathcal{S} = \{F \in \mathcal{F} : d(F) \leq t\}$ and we can find a reduction by generalization as follows:

- (1) For *minimal blocker problems*, we can set $\tau := t - 1$. Then, we again have that the minimal blocker problem on the SSP derived from the LOP problem is the optimization version of the corresponding minimum cardinality interdiction problem.

- (2) For the *full decision version of interdiction*, we can also set $\tau := t-1$. Then, the full decision version is again a generalization of the corresponding minimum cardinality interdiction problem due to the fact that the latter has a unit cost function c .
- (3) For *most vital elements problems*, the situation is more complicated. We first observe that the blocker part of $B \subseteq U$ with $c(B) \leq k$ is a generalization of the blocker part in minimum cardinality interdiction. The inner part on the nominal problem deserves special attention, though, due to the fact that the most vital element problem maximizes the objective while minimum cardinality interdiction blocks all solutions from the solution set \mathcal{S} . We focus on this in the next paragraph.

Reducing Minimum Cardinality Interdiction to Most Vital Elements. The concepts of minimum cardinality interdiction and most vital elements coincide if and only if the set \mathcal{S} contains exactly the optimal solutions, i.e. $\mathcal{S} = \{F \in \mathcal{F} : d(F) \leq t^*\}$, where t^* is optimal (i.e. minimal). In order to assure that \mathcal{S} captures exactly the optimal solutions, we need to include this condition into the reduction. In particular, the SSP reduction (g, f) needs to guarantee that all instances I are mapped to instances $g(I)$ such that all possible solutions are necessarily optimal. In other words, t is the optimal objective value of the LOP instance $g(I)$, since there are no feasible solutions, whose cost is even smaller than t . We call SSP reductions that fulfill this criterion *tight* and formally define them as follows.

Definition 8.3 (Tight SSP reduction). *Let Π_1 be an SSP problem and $\Pi_2 = (\mathcal{I}, \mathcal{U}, \mathcal{F}, d, t)$ be an LOP problem. Consider an SSP reduction $(g, (f_I)_{I \in \mathcal{I}})$ from Π_1 to (the SSP problem derived from) Π_2 . The reduction is called *tight* if for all YES-instances I_1 of Π_1 , the corresponding instance $I_2 = g(I_1)$ of Π_2 with the associated parameter $t := t^{(I_2)}$ and associated cost function $d := d^{(I_2)}$, the following holds:*

$$\{F \in \mathcal{F}(I_2) : d(F) \leq t\} \neq \emptyset \text{ and } \{F \in \mathcal{F}(I_2) : d(F) \leq t-1\} = \emptyset \quad (8.1)$$

All SSP reductions (to SSP problems derived from LOP problems) that can be found in Chapter 9 fulfill this definition and are thus tight. Therefore for all LOP problems (independent set, clique, knapsack, TSP, Steiner tree, dominating set, set cover, hitting set, feedback vertex set, feedback arc set), we obtain that the most vital element problem is at least as hard to compute as the minimum cardinality problem.

Vertex/Edge Deletion Problems. In this chapter, we are concerned with finding a set B such that $B \cap S \neq \emptyset$ for every solution S . Note that this definition is meaningful even if the nominal problem is not graph-based. However, in the special case where the nominal problem is graph-based, one could also consider a very related notion which is usually called *vertex deletion problem* or *edge deletion problem*. Here, the question is how many vertices (edges) need to be deleted from the graph until some desired property is met. Element deletion problems are well-studied in classical complexity theory for hereditary graph properties [LY80] and in parameterized complexity theory for properties expressible by first order formulas [BCT24]. In the general case, element deletion problems are not the same problem as our minimum cardinality interdiction problem. This is because for every set of deleted elements, the underlying instance is changed (vertices/edges are removed, which changes the graph). This is not the case for minimum cardinality interdiction problems as defined in this chapter. Thus, it is not possible to transfer the results of minimum cardinality interdiction directly to element deletion problems. Albeit for the problems of clique and independent set, the Σ_2^P -completeness results hold for both minimum cardinality interdiction as well as for vertex deletion interdiction. This is because for these problems the deletion of a vertex coincides with not taking this vertex into the solution. An analogous statement holds for edge deletions for the problems of directed/undirected Hamiltonian cycle/path, k -vertex-disjoint path, and Steiner tree.

8.3 Invulnerability Reductions for Various Problems

In this section, we show that a lot of well-known problems satisfy the assumptions of Theorem 8.3, i.e. it is possible to construct invulnerability reductions for them. Note that this proves the hardness part of Theorem 8.2. Let in the following $C \subseteq \mathcal{U}(I)$ denote the set of vulnerable elements, let $\mathcal{U}(I) \setminus C$ denote the set of invulnerable elements, and k denote the budget of the attacker. The following SSP problems are defined by using the natural choices of $(\mathcal{I}, \mathcal{U}, \mathcal{S})$ and are formally specified in Chapter 9.

Clique. We have $\mathcal{U} = V$ in this case. For a given graph $G = (V, E)$, and a set $C \subseteq V$, we explain how to make $V \setminus C$ invulnerable. We obtain a graph G' from G by replacing every vertex $v \in V \setminus C$ with an independent set X_v of size $|X_v| = k + 1$. For a vertex $v \in C$, we define $X_v := \{v\}$. For all edges uv in G , the new graph G' contains the complete bipartite graph between X_u and X_v . Note that every clique of G' contains at most one vertex from every set X_v . Hence the size of a maximum clique is the same in G and G' . Since for $v \in V \setminus C$, we have $|X_v| = k + 1$ and all vertices in X_v have the same neighborhood, the attacker is not able to attack all vertices of X_v at once because its budget of k is too small. Hence v has been made “invulnerable”. Furthermore, for every clique in G , we find a corresponding clique in G' that contains at most one vertex from each set X_v . Together, this implies that an attacker can find a set $B' \subseteq V(G')$ of size $|B'| \leq k$ interdicting all maximum cliques in G' if and only if the attacker can find a set $B \subseteq C$ of size $|B| \leq k$ interdicting all maximum cliques of G , i.e. the assumptions of Theorem 8.3 are met.

Independent Set. Analogous to clique in the complement graph.

Dominating Set. We have $\mathcal{U} = V$ in this case. To make a vertex $v \in V \setminus C$ invulnerable, we use the same construction as for the clique problem, with the only difference that X_v is a clique instead of an independent set. Every optimal dominating set takes at most one vertex from each set X_v , but all $k + 1$ vertices inside X_v are equivalent. More precisely, they have the same (closed) neighborhood. This means for an invulnerable vertex $v \in V \setminus C$, an attacker cannot attack all $k + 1$ vertices of X_v simultaneously. Furthermore, it is easily seen that on the vulnerable vertices, the attacker interdicts all optimal dominating sets in the old graph if and only if the analogous attack interdicts all optimal dominating sets in the new graph.

Hitting Set. In this case, we have some universe \mathcal{U} , sets $Y_1, \dots, Y_t \subseteq \mathcal{U}$, and the problem is to find a minimal hitting set $X \subseteq \mathcal{U}$ hitting all the sets Y_j , $j = 1, \dots, t$. To make an element $e \in \mathcal{U}$ invulnerable, simply delete it and replace it by $k + 1$ copies. We modify the sets such that every set Y_j that contained e now contains the $k + 1$ copies of e instead. It is clear that all the copies of e hit the same sets as e (i.e. taking multiple copies into the hitting set does not offer any advantage). Furthermore, it is not possible for the attacker to attack all $k + 1$ copies simultaneously. By an argument analogous to the above paragraphs, we are done.

Set cover. We have a ground set E , and a family \mathcal{F} of sets $S_1, \dots, S_n \subseteq E$ over the ground set. We let $\mathcal{U} := \{1, \dots, n\}$ and the goal is to pick a subset $I \subseteq \mathcal{U}$ of the indices such that $\bigcup_{i \in I} S_i = E$. The attacker can attack up to k of the indices $i \in I$ to forbid the corresponding sets from being picked. We can make some index $i \in \mathcal{U}$ invulnerable by simply duplicating the set S_i a total of $k + 1$ times.

Note that this satisfies the assumptions of Theorem 8.3, but modifies the family \mathcal{F} such that the same set could appear multiple times in the family. Alternatively, our construction can be adjusted such that this is avoided. For this, we introduce $k + 1$ new elements e_1, \dots, e_{k+1} and $k + 2$ new elements f_1, \dots, f_{k+2} to the ground set E . For each invulnerable index $i \in \{1, \dots, n\} \setminus C$, we substitute S_i by the $k + 1$ sets $S_i^{(j)} = S_i \cup \{e_j\}$ for $j = 1, \dots, k + 1$. Furthermore, we introduce

$k + 2$ new sets $S'_j := \{e_1, \dots, e_{k+1}\} \cup \{f_1, \dots, f_{k+2}\} \setminus \{f_j\}$ for $j = 1, \dots, k + 2$. This completes the description of the instance. Note that the following holds: The elements $\{f_1, \dots, f_{k+2}\}$ are covered by a set cover, if and only if it contains at least two sets of the form S'_j . Under this assumption, all the elements $\{e_1, \dots, e_{k+1}\}$ are already covered. Hence all the different copies $S_i^{(j)}$ for $j = 1, \dots, k + 1$ are essentially equivalent. Thus the attacker can not meaningfully attack all these copies simultaneously. Note that the attacker can also not meaningfully attack the sets S'_j , since no matter which k of them are attacked, 2 of them always remain.

Steiner tree. We have $\mathcal{U} = E$ in this case. To make an edge $uv \in E \setminus C$ invulnerable, we replace it with $k + 1$ parallel subdivided edges, i.e. we introduce vertices w_1, \dots, w_{k+1} and edges uw_i and w_iv for $i = 1, \dots, k + 1$. Every vulnerable edge uv is replaced with only a single subdivided edge, i.e. a vertex w and edges uw, wv . It is clear that the number of edges of a minimum Steiner tree in the new instance is exactly two times as big as before, and the edge uv has become effectively invulnerable.

Two vertex-disjoint path. We have $\mathcal{U} = A$ in this case. The gadget is the same as for Steiner tree, except that the construction is directed, i.e. the arc (u, v) is replaced either by the arcs $(u, w_i), (w_i, v)$ for $i = 1, \dots, k + 1$ (invulnerable case) or by the two arcs $(u, w), (w, v)$ (vulnerable case). Since the paths in this problem have to be vertex disjoint, adding additional subdivided arcs between two existing vertices does not produce additional solutions because traveling from u to v renders all other paths from u to v unusable.

Feedback arc set. We have $\mathcal{U} = A$ in this case. Note that making some arc $a = (u, v) \in A \setminus C$ invulnerable means to ensure that it can be used in a minimal feedback arc set, no matter which k arcs the attacker chooses. This can be achieved the following way: Subdivide a into $k + 1$ arcs. Clearly, the set of cycles in the new graph stays essentially the same. Furthermore, the attacker cannot block all $k + 1$ arcs from being chosen for the solution. Choosing one of the subdivided pieces of a in the new instance has the same effect as choosing a in the old instance.

Feedback vertex set. We have $\mathcal{U} = V$ in this case. To make a vertex $v \in V \setminus C$ invulnerable, we split it into two vertices v_{in} and v_{out} , put all incoming arcs of the old vertex v to v_{in} , put all outgoing arcs of the old vertex v to v_{out} , and connect v_{in} to v_{out} with a directed path P_v of $k + 1$ vertices. Note that in the new instance, a directed cycle uses one vertex of P_v if and only if the cycle uses all vertices of P_v if and only if a corresponding cycle in the old instance uses v . By an analogous to argument to the feedback arc set case, we are done.

Uncapacitated facility location. We have $\mathcal{U} = J$ in this case, where J is the set of sites for potential facilities. The attacker selects facility sites and forbids the decision maker to build a facility there. To make a facility site $j \in J \setminus C$ invulnerable, we can simply delete the site and replace it with $k + 1$ identical sites, i.e. sites which have the same facility opening cost and service cost functions as the original facility j . Clearly, this way the attacker can not stop one of the equivalent facilities to be opened. On the other hand, since the facilities are identical (and uncapacitated), the decision maker has no advantage from opening two identical copies of the same facility. Hence the new instance is identical to the old instance, with the only difference that facility site j is invulnerable.

p -median, p -center. The difference between the facility location problem and the p -center and p -median problem is that in the latter two, there are no facility opening costs, at most p facilities are allowed to be opened. The service costs in the p -center problem are calculated using a minimum, and in the p -median problem they are calculated using the sum. All of these differences do not affect the argument from above, i.e. one can still make a facility site invulnerable by creating $k + 1$ identical facility sites. Hence the same argument holds.

Subset Sum. We have $\mathcal{U} = \{1, \dots, n\}$ and are given numbers $a_1, \dots, a_n \in \mathbb{N}$ and a target value T . The question is whether there exists $S \subseteq \mathcal{U}$ with $\sum_{i \in S} a_i = T$. Consider some index $i \in \mathcal{U} \setminus C$. In order to make the index i invulnerable, the first idea is to copy the number a_i a total amount of $k + 1$ times. But there is a problem with this construction – if we do this, then the same number a_i could be picked multiple times, which is not allowed in the original instance. We need an additional gadget to make sure that a_i gets used at most once for each i . This can be done in the following way: Choose some number $B > 2k + 2$ as a basis. For each $i \in C$, it contains the single number $B^{n(k+1)}a_i$. For each $i \in \{1, \dots, n\} \setminus C$, it contains the $k + 1$ distinct numbers $c_i^{(j)} := B^{n(k+1)}a_i + B^{(i-1)(k+1)+j}$ for $j = 0, \dots, k$ as well as the $k + 1$ distinct numbers $d_i^{(j)} := \sum_{\ell=0, \ell \neq j}^k B^{(i-1)(k+1)+\ell}$ for $j = 0, \dots, k$ and the $k + 1$ distinct numbers $e_i^{(j)} := B^{(i-1)(k+1)+j}$ for $j = 0, \dots, k$. We call $d_i^{(j)}$ and $e_i^{(j)}$ the auxiliary numbers. The new instance contains a total of $|C| + 3(k + 1)(n - |C|)$ numbers. The new target value is

$$T' := B^{n(k+1)}T + \sum_{i \in \{1, \dots, n\} \setminus C} \sum_{\ell=0}^k B^{(i-1)(k+1)+\ell}.$$

Note that this has the following effect: Consider the representation of all involved numbers in base B . Let us call the digits 0 up to $n(k + 1) - 1$ the lower positions. Note that in the lower positions there can never be any carry, since for every lower position, all involved numbers have either a zero or one in that position and less than B numbers have a one in the same place. Due to that fact, in the lower positions the target T' is reached if and only if for every $i \in \{1, \dots, n\} \setminus C$, the corresponding “bitmask” is filled out (by this, we mean the positions $(i - 1)(k + 1)$ up to $i(k + 1) - 1$). This is achieved if and only if for some $j \in \{0, \dots, k\}$ both the values $c_i^{(j)}$ and $d_i^{(j)}$ or both the values $d_i^{(j)}$ and $e_i^{(j)}$ are picked. In particular, at most one of the $k + 1$ values $c_i^{(j)}$ for $j = 0, \dots, k$ is picked. In the upper positions, the target T' is reached if and only if the corresponding choice in the old instance meets the target T .

Consider an attack of $k + 1$ numbers by the attacker. For each $i \in \{1, \dots, n\} \setminus C$ it holds that there exists a j such that both $c_i^{(j)}$ and $d_i^{(j)}$ are not attacked. Likewise there exists a j' such that both $d_i^{(j')}$ and $e_i^{(j')}$ are not attacked. That means that if i is an invulnerable index, then no matter which $k + 1$ values of $c_i^{(j)}$, $d_i^{(j)}$ and $e_i^{(j)}$ are attacked, a correct solution of subset sum will take for some j either both $c_i^{(j)}$ and $d_i^{(j)}$ (which corresponds to taking a_i in the original instance) or take both $d_i^{(j)}$ and $e_i^{(j)}$ (which corresponds to not taking a_i in the original instance). It follows that it is possible to block the new instance by attacking $k + 1$ values if and only if it is possible to block the old instance by attacking $k + 1$ of the vulnerable values. This was to show. Finally, if the old numbers a_1, \dots, a_n are pairwise distinct, the new numbers are distinct as well. Hence the interdiction problem for subset sum is Σ_2^P -complete, even if all involved numbers are distinct.

Knapsack. The knapsack problem can be seen as a more general version of the subset sum problem, by creating for each i from the subset sum instance a knapsack item with both profit $p_i = a_i$ and weight $w_i = a_i$, and setting both the weight and profit threshold to T . Hence the Σ_2^P -completeness of MIN CARDINALITY INTERDICTION-KNAPSACK follows as a consequence of the Σ_2^P -completeness of MIN CARDINALITY INTERDICTION-SUBSET SUM. This holds even if all the involved knapsack items are distinct.

8.3.1 An Invulnerability Reduction for Hamiltonian Cycle

The invulnerability gadget for Hamiltonian cycle is the most involved of all our constructions, hence we devote a subsection to it. The main result in this section is that the minimum cardinality interdiction problem is Σ_2^P -complete for the nominal problems of both directed and undirected Hamiltonian cycle and path, as well as TSP.

We present our reduction for the case of undirected Hamiltonian cycle and then argue how it can be adapted to the other cases. The main idea is to consider 3-regular graphs $G = (V, E)$ as an intermediate step, and then for a subset $C \subseteq E$ show how $E \setminus C$ can be made invulnerable. To this end, consider the SSP problem 3REG HAM:

3REG HAM

Instances: Undirected, 3-regular Graph $G = (V, E)$

Universe: $\mathcal{U} := E$.

Solution set: The set of all Hamiltonian cycles in G .

In Chapter 9, it is shown that HAMILTONIAN CYCLE is SSP-NP-complete. We now require the following stronger statement.

Lemma 8.1. *3REG HAM is SSP-NP-complete.*

Proof. Garey, Johnson & Tarjan [GJT76] give a reduction from 3SAT to 3REG HAM, such that for every variable x_i in the 3SAT instance the graph G has two distinct edges $e(x_i)$ and $e(\bar{x}_i)$ (compare Figure 7 in [GJT76]). Let $E' := \bigcup_i \{e(x_i), e(\bar{x}_i)\}$ be the set of all these edges. For some assignment α of the 3SAT variables, we say that α corresponds to the edge set E_α defined by $\{e(x_i) : \alpha(x_i) = 1\} \cup \{e(\bar{x}_i) : \alpha(x_i) = 0\}$. Garey, Johnson & Tarjan show that there is a bijection between satisfying assignments and edge sets $E'' \subseteq E'$ that can be subset of a Hamiltonian cycle. More formally: 1.) For every satisfying assignment α , if one considers the set $E_\alpha \subseteq E'$ of edges corresponding to that assignment, there exists a Hamiltonian cycle H extending E_α , i.e. $H \cap E' = E_\alpha$. 2.) For every Hamiltonian cycle H , we have that $H \cap E'$ equals E_α for some satisfying assignment α . In total, 1.) and 2.) together show that the reduction in [GJT76] is an SSP reduction by defining $f(x_i) := e(x_i)$ and $f(\bar{x}_i) := e(\bar{x}_i)$. \square

We remark that it follows from [ANS80, GJT76] by the same argument that the problem is even SSP-NP-complete if restricted to 3-regular, bipartite, planar, 2-connected graphs. However, for our arguments it suffices to consider 3-regular graphs.

Consider now an instance of 3REG HAM, i.e. a 3-regular undirected graph $G = (V, E)$. Let $C \subseteq E$ be a subset of the edges and $k \in \mathbb{N}_0$ the attacker's budget. We call C the vulnerable edges. Let $D := E \setminus C$. In the remainder of this section we describe and prove a construction how to make the edges in D invulnerable. We quickly sketch the main idea: To make an edge $e = ab$ invulnerable, we enlarge it by replacing it with a large clique W'_{ab} making sure that e can be traversed no matter which k edges inside W'_{ab} are attacked. We also blow up each vertex a of the original graph into a clique W_a . However, this introduces new vertices into the instance, and we need to make sure that a Hamiltonian cycle can always trivially visit all the new vertices. At the same time however, it should still hold that a Hamiltonian cycle in the new graph should be able to enter and exit these new objects W_a and W'_{ab} at most once, since otherwise a corresponding cycle in the old graph G would visit edges or vertices twice, which is of course forbidden. We achieve this by associating to each edge $e = ab$ a star F_{ab} and argue that a Hamiltonian cycle can use at most one edge of each star F_{ab} . Furthermore, we will show that the fact that G is 3-regular implies that each clique W_a can be traversed only once.

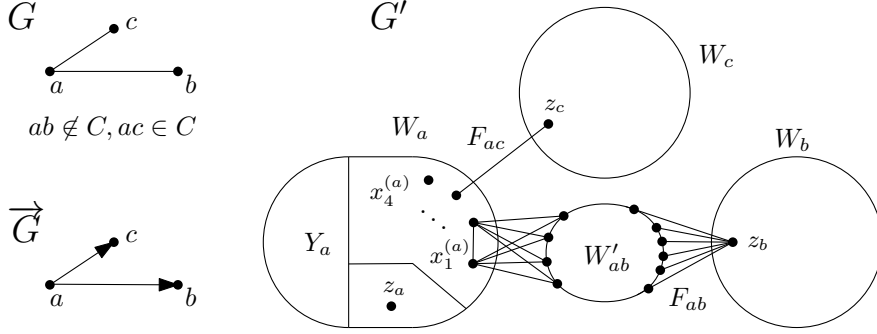


Figure 8.1: Invulnerability gadget for Hamiltonian cycle which makes the edge ab invulnerable while the edge ac remains vulnerable.

We are ready to begin with the construction. Let the directed graph \vec{G} result from G by orienting its edges arbitrarily. We construct an undirected graph $G' = (V', E')$ from \vec{G} as follows: Let $n := |V(G)|$. For each vertex $a \in V(\vec{G})$, let d_a be the out-degree of a , and let W_a be a set of $2d_a + 4k + 1$ vertices. For each invulnerable edge $ab \in D$ in the old graph G , let W'_{ab} be a set of $4k$ vertices. The vertex set $V(G')$ of the new graph G' is then defined by

$$V(G') = \bigcup_{a \in V} W_a \cup \bigcup_{ab \in D} W'_{ab}.$$

We further partition W_a into three disjoint parts $W_a = X_a \cup Y_a \cup \{z_a\}$ of size $|X_a| = 2d_a$ and $|Y_a| = 4k$ and $|\{z_a\}| = 1$. We denote the vertices of X_a by $x_1^{(a)}, \dots, x_{2d_a}^{(a)}$.

The edges of G' are defined as follows: First, we let W_a be a clique for all $v \in V$. Second, for each vertex $a \in V(\vec{G})$, let e_1, \dots, e_{d_a} be its outgoing edges. For each $i = 1, \dots, d_a$, consider the i -th outgoing edge $e_i = (a, b)$ of a , where b is the corresponding neighbor. If $e_i \in C$, i.e. e_i is vulnerable, then G' contains simply the single edge $x_{2i-1}^{(a)}z_b$. In the other case, i.e. $e_i \in D$ is invulnerable, then G' contains an invulnerability gadget as depicted in fig. 8.1 induced on the vertices $\{x_{2i-1}^{(a)}, x_{2i}^{(a)}\} \cup W'_{ab} \cup \{z_b\}$. The invulnerability gadget consists of a clique on the vertex set $\{x_{2i-1}^{(a)}, x_{2i}^{(a)}\} \cup W'_{ab}$, together with all edges from the set W'_{ab} to the vertex z_b , i.e. a star graph centered at z_b that has W'_{ab} as its leaves. Let F_{ab} denote this star graph. Finally, for all vulnerable edges $ab \in C$, we also define F_{ab} to be the single edge $x_{2i-1}^{(a)}z_b$ that connects W_a to W_b . This can be interpreted as a trivial star centered at z_b with only one leaf. This completes the description of G' .

The overall idea of this construction is that the cliques of W_a cannot be attacked because they have at least k vertices. Thus it is always possible to find a path visiting all vertices of W_a . Additionally, a star F_{ab} of size larger than k makes the edge $ab \in E$ invulnerable because at most k edges can be attacked. Thus there is always the possibility to travel over one edge of F_{ab} which corresponds to using edge ab in the original graph. On the other hand, since every edge of the star is connected to the same vertex z_b , we have that the star F_{ab} can be used (essentially) exactly once. Thus only the stars of size one (which correspond to the vulnerable edges) are attackable. We now have everything that we need to prove our main result of this section.

Theorem 8.5. *Minimum cardinality interdiction for UNDIRECTED HAMILTONIAN CYCLE is Σ_2^P -complete.*

Proof. Due to Chapter 4, and Lemma 8.1, we have that $\text{COMB. INTERDICTION-3REG HAM}$ is Σ_2^p -complete. We claim that the construction of G' yields a correct reduction from $\text{COMB. INTERDICTION-3REG HAM}$ to $\text{MIN CARDINALITY INTERDICTION-HAMCYCLE}$. Indeed, the following two Lemmas 8.2 and 8.3 show that YES-instances of one problem get transformed into YES-instances of the other problem. \square

We remark that the 3-regularity of the graph is not maintained by the reduction. Indeed, an argument similar to the arguments given later in Section 8.4 shows that the interdiction problem for Hamiltonian cycle restricted to only 3-regular graphs is likely not Σ_2^p -complete.

Lemma 8.2. *If there exists $B' \subseteq E'$ of size $|B'| \leq k$, such that $G' - B'$ has no Hamiltonian cycle, then there is $B \subseteq C$ of size $|B| \leq k$ such that $G - B$ has no Hamiltonian cycle.*

Proof. Proof by contraposition. Assume that for all $B \subseteq C$ with $|B| \leq k$ the graph $G - B$ has a Hamiltonian cycle H . Given some $B' \subseteq E'$ with $|B'| \leq k$, we have to show that the graph $G' - B'$ has a Hamiltonian cycle. Let $B \subseteq C$ be the set of vulnerable edges in G whose copies in G' are attacked by B' (i.e. $B = \{ab \in C : F_{ab} \in B'\}$). Since $B \subseteq C$ and $|B| \leq k$, by assumption $G - B$ has a Hamiltonian cycle H . We want to modify H to a Hamiltonian cycle H' of $G' - B'$. The basic idea is to follow the same route in G' as H does in G . However, we have to pay attention, because we are not allowed to use edges from B' . We call a vertex in G' *attacked*, if at least one of its incident edges is attacked by B' , and call it *free* otherwise. Note that since $|B'| \leq k$ and $|Y_a| = 4k$ and $|W'_{ab}| = 4k$ for $a \in V, ab \in E$, the vertex sets Y_a and W'_{ab} have at least $2k$ free vertices. Free vertices are good for the following reason: Whenever we plan to go from some vertex u to v in G' , but we cannot because $uv \in B'$ was attacked, then we can instead choose any free vertex f and go the route u, f, v instead. Now the plan is that H' will roughly employ the following strategy: Follow the same path in G' like H does in G . That is whenever H' enters some new set W_a for the first time, then we first visit all the sets W'_{ab} for all out-neighbors b of a in \vec{G} if and only if $ab \notin H$. Note that for such b , the set W'_{ab} has two adjacent vertices with W_a (we use these two vertices to enter and leave), and we collect all the vertices of W'_{ab} . Here, we prioritize to visit first the attacked vertices of W'_{ab} and then the remaining vertices of W'_{ab} . After that, we collect all remaining vertices of W_a (again prioritizing the attacked vertices first) before leaving W_a . (If the path on which we are leaving W_a corresponds to an invulnerable edge ab in G , we also collect all of W'_{ab} in the process of leaving W_a .)

Note that this plan might at first not be feasible, because it requires going over some edge $e' \in B'$. However note that, since H does not use any edge of B , for every such edge e' there are always at least $2k$ free vertices that are adjacent to both endpoints of e' . Hence it is possible to “repair” such an edge e' by rerouting over some free vertex instead (and later skip this free vertex). Since there are at most k defects, and there are at least $2k$ free vertices available at the end of traversing every set W_a or W'_{ab} , all defects can be repaired. Hence we can modify H' to be a Hamiltonian cycle of $G' - B'$, which was to show. \square

Lemma 8.3. *If there exists $B \subseteq C$ of size $|B| \leq k$, such that $G - B$ has no Hamiltonian cycle, then there is $B' \subseteq E'$ of size $|B'| \leq k$ such that $G' - B'$ has no Hamiltonian cycle.*

Proof. Proof by contraposition. Assume that for all $B' \subseteq E'$ of size $|B'| \leq k$ the graph $G' - B'$ has a Hamiltonian cycle. Given some $B \subseteq C$ with $|B| \leq k$, we have to show that the graph $G - B$ has a Hamiltonian cycle. Let B' be the trivial stars in G' corresponding to the edges in B (i.e. $B' = \{F_{ab} : ab \in B\}$). Since $|B'| \leq k$, by assumption there is a Hamiltonian cycle H' in $G' - B'$. Consider the set $F := \bigcup_{ab \in E} E(F_{ab})$, i.e. the union of the edge sets of all the stars, trivial or not.

We claim that w.l.o.g. we can assume that $|H' \cap F_{ab}| \leq 1$ for all $ab \in E$. Indeed, the graph $G' - F$ consists out of multiple connected components. Each of these components contains exactly one set of the form W_a , for $a \in V(G)$, and is incident to exactly three sets of the form F_e in G' (where $e \in E(G)$ is an edge that is either incoming to or outgoing from a in \vec{G}). Suppose for some F_{ab} we have $|H' \cap F_{ab}| \geq 2$. Since F_{ab} is a star graph centered around the single vertex z_b , we have $|H' \cap F_{ab}| = 2$. Consider the edge $ab \in E(G)$ such that F_{ab} connects the vertex z_b with W'_{ab} . By the observation about $G' - F$, the following is true about H' : It enters W'_{ab} in one of the two vertices attached to X_a , then traverses exactly all of $W'_{ab} \cup \{z_b\}$, then leaves through the other of the two vertices attached to X_a , and at a later point returns to collect all vertices of X_b . However, by the same observation as in Lemma 8.2, if we define a free vertex to be a vertex not adjacent to any edge in B' , then both W'_{ab} and W_b have $2k$ free vertices. Hence we can modify H' such that $H' \cap F_{ab} = \emptyset$. We thus assume that $|H' \cap F_{ab}| \leq 1$ for all $ab \in E$.

Consider again the graph $G' - F$. Since each of its component is adjacent to three sets F_e and $|H' \cap F_e| \leq 1$, we conclude that H' uses exactly two of these three sets F_e . But this implies that H' enters and exits each of the components of $G' - F$ only once and collects all of its vertices in the process. This implies that H' globally follows the same path as some Hamiltonian cycle H of G . Since $H' \subseteq E' - B'$, we conclude $H \subseteq E - B$. This was to show. \square

These two lemmas together prove Theorem 8.5. We would now like to prove Σ_2^p -completeness also for Hamiltonian cycle interdiction of directed graphs. Note that this does not follow from a trivial argument: Even though one can transform an undirected graph into a directed one, by substituting every undirected edge uv by two directed arcs $(u, v), (v, u)$, there is a problem: In the new setting the interdictor needs two attacks to separate u, v , while in the old setting the attacker only needs one.

Still, the above proof can be adapted to the case of directed Hamiltonian cycle the following way: We start with [Ple79], which provides an SSP reduction to prove that the Hamiltonian cycle problem is NP-complete even in directed graphs G such that $\text{indegree}(v) + \text{outdegree}(v) \leq 3$ for every vertex v , and such that for all pairs u, v of G at most one of the two arcs (u, v) and (v, u) is present. Given a directed graph G , we then repeat the same construction as before, with the difference that we can start directly with the directed graph G instead of obtaining an orientation \vec{G} first. This way, we can obtain an undirected graph G' in the same way as before. In a final step, we turn G' into a directed graph by substituting every undirected edge uv by a pair of two arcs $(u, v), (v, u)$. We perform this substitution for every edge of G' with the exception of the edges that are part of some star F_{ab} . Instead, for each star F_{ab} , we orient the edges of F_{ab} the same way as the original arc of G between a, b . It can be shown that all the arguments from the above construction still hold. Hence the minimum cardinality interdiction problem is Σ_2^p -complete also for directed graphs.

If one is interested in Hamiltonian paths instead of cycles, a similar modification is possible. Inspecting the proof of [GJT76] (or [Ple79], respectively) more closely, we find that in both constructions the graph G contains some edge $e = uv$ (some arc $e = (u, v)$, respectively) such that every Hamiltonian cycle uses e . We can delete e and identify the vertices s, t with the endpoints of e . Then a Hamiltonian s - t -path in the new graph corresponds to a Hamiltonian cycle in the old graph and vice versa. Note that this does not increase the degree of the graph. The rest of the proof proceeds in the same manner, both in the undirected and directed case. Finally, the proof can also easily be adapted to the TSP by a standard reduction of undirected Hamiltonian cycle to the TSP (a graph G is transformed into a TSP instance on the complete graph where the costs obey $c(uv) = 1$ if $uv \in E(G)$ and $c(uv) = n + 1$ if $uv \notin E(G)$). In conclusion, we have proven that the minimum cardinality interdiction problem is Σ_2^p -complete for the directed/undirected Hamiltonian path/cycle problem and the TSP.

8.4 Cases where the meta-theorem does not apply

It would be nice to establish a meta-theorem providing Σ_2^P -completeness of the minimum cardinality interdiction version of all nominal problems, which are SSP-NP-complete, instead of only those problems that admit an additional function g with properties as stated in Theorem 8.3. However, we show in this section that this is not possible. In order to classify problems that do not admit this function g , we provide a lemma that guarantees that the minimum cardinality version of a problem in SSP-NP is in coNP. Therefore, under the usual complexity-theoretic assumption $\text{NP} \neq \Sigma_2^P$, the interdiction problem is not Σ_2^P -complete.

In order to provide an intuition under which circumstances a minimum cardinality interdiction problem resides in the class coNP, we examine the vertex cover problem. In a vertex cover, every edge uv needs to be covered by at least one of the two incident vertices u and v . This, however, gives the attacker the opportunity to attack both u and v such that the edge uv can never be covered. Therefore, an attacker budget of at least 2 results in a clear YES-instance. On the other hand, if the attacker budget is at most 1, we can provide a certificate for NO-instances. We can summarize this observation in the following lemma.

Lemma 8.4. *Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP problem. If in each instance $I \in \mathcal{I}$ there is a subset $U' \subseteq \mathcal{U}(I)$ of constant size, i.e. $|U'| = O(1)$, such that for $U' \cap S \neq \emptyset$ for all $S \in \mathcal{S}(I)$, then MIN CARDINALITY INTERDICTION-II is contained in coNP.*

Proof. Let k be the interdiction budget. If $|U'| \leq k$, then the interdictor is able to block the whole set U' . By definition of U' , there is no solution $S \in \mathcal{S}(I)$ such that $U' \cap S \neq \emptyset$ and thus the interdictor has a winning strategy. If on the other hand $k < |U'| = O(1)$, then there is a polynomially sized certificate encoding a winning strategy of the defender, i.e. a certificate for a NO-instance of the problem. For this, we first encode the $\binom{|U(I)|}{k} = |U(I)|^{O(1)}$ possible blockers $B' \subseteq \mathcal{U}(I)$ and then the solution $S \in \mathcal{S}(I)$ such that $S \cap B' \neq \emptyset$ for all $B' \subseteq \mathcal{U}(I)$. It is possible to efficiently verify the solution by checking whether there is a solution $S \in \mathcal{S}(I)$ such that $S \cap B' \neq \emptyset$ for all $B' \subseteq \mathcal{U}(I)$ holds because the nominal problem Π is in NP. It follows that the problem lies in coNP. \square

Consider the different variants of interdiction problems introduced in Subsection 8.2.2. Since they are more general, Lemma 8.4 does not immediately imply that those variants are contained in coNP. However, if for each instance the stronger condition $U' \cap F \neq \emptyset$ for all feasible solutions $F \in \mathcal{F}(I)$ and for some constant size set $U' \subseteq \mathcal{U}(I)$ holds, then the *full decision variant of interdiction* is contained in coNP. To prove this, we can employ the same strategy as in Lemma 8.4 while substituting the solutions \mathcal{S} by the feasible solutions \mathcal{F} .

This also has direct consequences on the *minimal blocker problem* and the *most vital elements problem*. On the one hand, the minimal blocker problem can be efficiently solved given an oracle for the full decision variant. On the other hand, the *most vital elements problem* can be solved by trying out all possible blockers $B' \subseteq \mathcal{U}(I)$ as in Lemma 8.4. If $k < |U'| = O(1)$, all blockers can be searched for the optimal blocker. If $k \geq |U'|$, then all feasible solutions can be blocked resulting in $\max_B \min_{F, F \cap B = \emptyset} d(F) = \infty$ because $U' \cap F \neq \emptyset$ holds for all feasible solutions $F \in \mathcal{F}(I)$.

Besides the containment in coNP, we can also derive the following corollary pinpointing the complexity of minimum cardinality interdiction problems whose nominal problem is in SSP-NP.

Corollary 8.1. *Let $\Pi = (\mathcal{I}, \mathcal{U}, \mathcal{S})$ be an SSP-NP-complete problem. If in each instance $I \in \mathcal{I}$ there is a subset $U' \subseteq \mathcal{U}(I)$ of constant size, i.e. $|U'| = O(1)$, such that for $U' \cap S \neq \emptyset$ for all $S \in \mathcal{S}(I)$, then MIN CARDINALITY INTERDICTION-II is coNP-complete.*

Proof. There is a reduction by restriction: Setting the interdiction budget $k = 0$ results in the corresponding co-problem coII of the nominal problem II . \square

8.4.1 Applying the Lemma to Various Problems

In this section, we apply Lemma 8.4 to the problems mentioned earlier in this chapter. Some of the problems are affected in their original general form, e.g. vertex cover or satisfiability, while for others the lemma can be applied on a restricted version such as independent set on graphs with bounded minimum degree. For this, we shortly describe the problem and then give the argument on how the lemma is applicable.

Vertex Cover. An instance of the vertex cover interdiction problem consists of a graph G and numbers $t, k \in \mathbb{N}_0$. The question is if the attacker can find a set $B \subseteq V(G)$ with $|B| \leq k$ such that $B \cap S \neq \emptyset$ for every vertex cover S of size at most t . Now, observe that if $k \geq 2$ (and the graph is non-empty), the attacker can easily find such a set B by selecting two adjacent vertices. Thus, Lemma 8.4 applies by defining $U' = \{u, v\}$ for some edge $uv \in E(G)$. Observe that this not only destroys the solutions $S \in \mathcal{S}(I)$ but also all feasible solutions $F \in \mathcal{F}(I)$. Thus the minimum cardinality interdiction version and the full decision variant of interdiction of vertex cover are coNP -complete.

Satisfiability. An instance of the satisfiability interdiction problem consists out of a formula in CNF over the variables $X = \{x_1, \dots, x_n\}$, with the literal set as universe, i.e. $\mathcal{U} = X \cup \bar{X}$, and interdiction budget k . A similar issue as in vertex cover interdiction arises here: If $k \geq 2$, the interdictor can just choose for some $i \in \{1, \dots, n\}$ to attack both literals x_i, \bar{x}_i . Every satisfying assignment (of non-trivial instances) contains either x_i or \bar{x}_i , hence this is a successful attack. Thus, Lemma 8.4 applies by defining $U' = \{x, \bar{x}\}$ for some literal pair $x, \bar{x} \in \mathcal{U}$. Again, this also destroys all feasible solutions $F \in \mathcal{F}(I)$. Thus the minimum cardinality interdiction version and the full decision variant of interdiction of satisfiability are coNP -complete.

Independent Set on graphs with bounded minimum degree. An instance of the independent set interdiction problem consists of a graph $G = (V, E)$ with universe $U = V$, a threshold t and an interdiction budget k . The question of the independent set problem is if there is a set $I \subseteq V$ such that all vertices in I do not share an edge. We now take the vertex d of bounded degree into consideration. If the attacker attacks the closed neighborhood $N[d]$ of d , all (optimal) solutions $S \in \mathcal{S}$ can be interdicted and thus Lemma 8.4 is applicable. Thus minimum cardinality interdiction independent set on graphs with bounded minimum degree is coNP -complete. In contrast to the other problems, this statement is not true for general feasible solutions $F \in \mathcal{F}(I)$. Hence we do not obtain a result for the variants from Subsection 8.2.2.

Dominating Set on graphs with bounded minimum degree. An instance of the dominating set interdiction problem consists of a graph $G = (V, E)$ with universe $U = V$, a threshold t and an interdiction budget k . The question of the dominating set problem is if there is a set $D \subseteq V$ of size at most t such that D dominates all vertices of vertex set V . In other words, the union of the neighborhoods of the vertices in D is the vertex set V , i.e. $\bigcup_{v \in D} N[v] = V$. Again we consider a vertex d of bounded degree. Then, we can define the set of constant size to be $U' = N[d]$. All feasible solutions $F \in \mathcal{F}$ have to include some vertex from U' (otherwise d would not be dominated). Thus Lemma 8.4 is applicable to dominating set. Accordingly, the minimum cardinality interdiction version and the full decision variant of interdiction of dominating set on graphs with bounded minimum degree are coNP -complete.

Hitting Set with bounded minimum set size. An instance of hitting set interdiction consists of a ground set $\{1, \dots, n\}$ and m sets $S_j \subseteq \{1, \dots, n\}$ as well as a threshold t and an interdiction

budget k . The universe is defined by $\mathcal{U} = \{1, \dots, n\}$. The question of the hitting set problem is whether there is a hitting set $H \subseteq \{1, \dots, n\}$ of size at most t for the sets S_j , that is, $H \cap S_j \neq \emptyset$ for $1 \leq j \leq m$. We can apply Lemma 8.4 by defining U' to be the set of constant size $|S_c| = O(1)$. Then, the attacker is able to block all elements of the set S_c such that it is not hittable, which interdicts all feasible solutions $F \in \mathcal{F}$. Therefore the minimum cardinality interdiction version and the full decision variant of interdiction of hitting set with bounded minimum set size are coNP-complete.

Set Cover with bounded minimum coverage. An instance of the set cover interdiction problem consists of sets $S_i \subseteq \{1, \dots, m\}$ for $1 \leq i \leq n$, a threshold t and an interdiction budget k . The universe is defined as the sets $\{S_i : 1 \leq i \leq n\}$. The question of the set cover problem is whether there is selection $S \subseteq \{S_1, \dots, S_n\}$ of size at most k such that $\bigcup_{s \in S} s = \{1, \dots, m\}$. If there is an element $e \in \{1, \dots, m\}$ of bounded coverage, i.e. there is a constant number of S_i , $1 \leq i \leq n$, with $e \in S_i$, then the attacker can attack all of these sets S_i . Thus, we can apply Lemma 8.4 by choosing $U' = \{S_i \mid e \in S_i\}$ and all feasible solutions $F \in \mathcal{F}$ are blockable. Accordingly, the minimum cardinality interdiction version and the full decision variant of interdiction of set cover with bounded minimum coverage are coNP-complete.

Steiner Tree on graphs with bounded minimum degree of terminal vertices. An instance of the Steiner tree interdiction problem consists of a graph $G = (S \cup T, E)$ of Steiner vertices S and terminals T , edge weights $c : E \rightarrow \mathbb{N}$, a threshold t and a interdiction budget k . The universe is the edge set $\mathcal{U} = E$. The question of the Steiner tree problem is if there is a tree $E' \subseteq E$ of weight $c(E') \leq t$ such that all terminal vertices T are connected by E' . If there is a terminal vertex $d \in T$ of bounded degree, then all incident edges build up a set $U' = \{dv \in E\}$ on which we can apply Lemma 8.4. This blocks all feasible solutions $F \in \mathcal{F}$. Therefore, the minimum cardinality interdiction version and the full decision variant of interdiction of Steiner tree on graphs with bounded minimum degree of terminal vertices are coNP-complete.

Two Vertex-Disjoint Path on graphs with bounded degree. An instance of the two vertex-disjoint path interdiction problem consists of a directed graph $G = (V, A)$, vertices $s_1, s_2, t_1, t_2 \in V$ and interdiction budget k . The universe is the arc set $\mathcal{U} = A$. The question is if there are two paths $P_1, P_2 \subseteq A$ such that P_i starts at s_i and ends at t_i and both paths P_1 and P_2 do not share a vertex. If the the graph has bounded degree, we can choose any of the vertices that have to be included in on of the paths, e.g. s_1 , and include all the incident arcs in $U' = \{(s_1, v) \in A\}$ such that we can apply Lemma 8.4. This blocks all feasible solutions $F \in \mathcal{F}$. Accordingly, the minimum cardinality interdiction version and the full decision variant of interdiction of two vertex-disjoint path on graphs with bounded degree are coNP-complete.

Feedback Vertex Set on graphs with bounded girth. An instance of the feedback vertex set interdiction problem consists of a directed graph $G = (V, A)$, a threshold t and interdiction budget k . The universe is the vertex set $\mathcal{U} = V$. The question of feedback vertex set is if there is a set $V' \subseteq V$ such that the graph is cycle free. Accordingly, if the graph has bounded girth, there is a cycle of bounded length, which the attacker can attack or in other words, the cycle cannot be deleted by the defender by choosing a corresponding vertex to be in the feedback vertex set. Thus all feasible solutions $F \in \mathcal{F}$ are blockable by applying Lemma 8.4 with $U' = \{v \in V \mid v \text{ is part of the smallest cycle in } G\}$. Therefore, the minimum cardinality interdiction version and the full decision variant of interdiction of feedback vertex set on graphs with bounded girth are coNP-complete.

Feedback Arc Set on graphs with bounded girth. An instance of the feedback arc set interdiction problem consists of a directed graph $G = (V, A)$, a threshold t and interdiction budget k . The universe is the arc set $\mathcal{U} = A$. The question of feedback arc set is if there is an arc set $A' \subseteq A$

such that the graph is acyclic. We can use the same argument as in feedback vertex set. That is, the attacker can choose the arcs of the smallest cycle in G . Thus all feasible solutions $F \in \mathcal{F}$ are blockable by applying Lemma 8.4 with $U' = \{a \in A \mid a \text{ is part of the smallest cycle in } G\}$. Therefore, the minimum cardinality interdiction version and the full decision variant of interdiction of feedback arc set on graphs with bounded girth are coNP-complete.

Uncapacitated Facility Location, p-Center, p-Median with bounded minimum customer coverage. An instance of the minimum cardinality interdiction version of these three problems consists of a set of potential facilities F and a set of clients C together with a cost function on the facilities $f : F \rightarrow \mathbb{N}$ and a service cost function $c : F \times C \rightarrow \mathbb{N}$ as well as a threshold t and an interdiction budget k . The universe is the potential facility set $\mathcal{U} = F$ and it is asked for a set of facilities $F' \subseteq F$ not exceeding the cost threshold t . If the coverage of one customer is bounded, i.e. there is a bounded number of potential facilities that are able to serve the customer, the attacker is able to block all of these. Thus we can define U' as the set of potential facilities that are able to serve the customer of bounded coverage such that all feasible solutions $F \in \mathcal{F}$ can be interdicted. Therefore, we can apply Lemma 8.4 and the minimum cardinality interdiction version and the full decision variant of interdiction of these three facility locations problems with bounded minimum customer coverage are coNP-complete.

Hamiltonian path/cycle (directed/undirected), TSP on graphs with bounded minimum degree. An instance of the minimum cardinality interdiction version of these problems consists of a graph $G = (V, E)$ (respectively $G = (V, A)$ in the directed case) and an interdiction budget k . The universe is the set of edges $\mathcal{U} = E$ (respectively the set of arcs $\mathcal{U} = A$). The question is whether there is a Hamiltonian path or cycle in G , i.e. a path/cycle that visits every vertex exactly once. Because there is a vertex d of bounded degree which has to be visited, we can define the set of constant size $U' = \{dv \in E\}$ (respectively $U' = \{(d, v), (v, d) \in A\}$). If the set U' is blocked it is not possible to visit the vertex, thus all feasible solutions $F \in \mathcal{F}$ can be interdicted. Therefore, we can apply Lemma 8.4 and the minimum cardinality interdiction version and the full decision variant of interdiction of these five Hamiltonian problems on graphs with bounded minimum degree are coNP-complete.

8.4.2 Satisfiability with Universe over the Variables

In the previous subsection we explained why minimum cardinality interdiction-SAT is contained in coNP, hence likely not Σ_2^P -complete. Note that this is a consequence of our choice of definition of SATISFIABILITY, where we explicitly defined the universe to be the literal set $L = X \cup \bar{X}$. As a consequence, the interdictor may attack $X \cup \bar{X}$.

SATISFIABILITY ($\mathcal{U} = L$)

Instances: Literal Set $L = \{x_1, \dots, x_n\} \cup \{\bar{x}_1, \dots, \bar{x}_n\}$, Clauses $C \subseteq 2^L$

Universe: $L =: \mathcal{U}$.

Solution set: The set of all sets $L' \subseteq \mathcal{U}$ such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c_j| \geq 1$ for all $c_j \in C$.

An interesting behavior occurs, when we consider the following alternative version SATISFIABILITY ($\mathcal{U} = X$).

SATISFIABILITY ($\mathcal{U} = X$)

Instances: Variable Set $X = \{x_1, \dots, x_n\}$, Clauses $C \subseteq 2^{X \cup \bar{X}}$

Universe: $X =: \mathcal{U}$.

Solution set: The set of all sets $X' \subseteq \mathcal{U}$ such that the assignment $\alpha : X \rightarrow \{0, 1\}$ with $\alpha(x) = 1 \leftrightarrow x \in X'$ satisfies all clauses in C .

Here the universe is only the variable set X , so in the interdiction version, the interdictor may only attack X , i.e. the interdictor may target individual variables and enforce that they must be set to *false*. We show now that in contrast to the variant, where the universe is the literal set, in this new variant the interdiction problem is Σ_2^p -complete again. Since the problem SATISFIABILITY ($\mathcal{U} = X$) is not part of the original problem set of Chapter 9, we perform this proof in two steps.

Lemma 8.5. *SATISFIABILITY ($\mathcal{U} = X$) is SSP-NP-complete, even when all clauses are restricted to length at most three.*

Proof. We provide an SSP reduction from the SSP-NP-complete problem SATISFIABILITY ($\mathcal{U} = L$) to SATISFIABILITY ($\mathcal{U} = X$). Consider an instance of SATISFIABILITY ($\mathcal{U} = L$) given by a formula φ with n variables $X = \{x_1, \dots, x_n\}$ and universe/literal set $\mathcal{U} = L = X \cup \bar{X}$. SATISFIABILITY ($\mathcal{U} = L$) is SSP-NP-complete even when all clauses are restricted to length three, so let us w.l.o.g. assume that property. We have to show how to embed this universe into the universe \mathcal{U}' of some corresponding SATISFIABILITY ($\mathcal{U} = X$) instance φ' , where only positive literals are allowed in \mathcal{U}' . This can be done the following way: We introduce $2n$ new variables $X' := \{x_1^t, \dots, x_n^t\} \cup \{x_1^f, \dots, x_n^f\}$. The universe $\mathcal{U}' := X'$ consists out of the $2n$ corresponding positive literals X' . The new formula φ' is defined from φ in two steps. First a substitution process takes place: For each $i = 1, \dots, n$, the positive literal x_i is replaced by the positive literal x_i^t and each negative literal \bar{x}_i is replaced by the positive literal x_i^f . In a second step, the clauses $(x_i^t \vee \bar{x}_i^f) \wedge (\bar{x}_i^t \vee x_i^f)$ (note that these are equivalent to $x_i^t \oplus x_i^f$) are added to φ' . Formally,

$$\varphi' = \text{substitute}(\varphi) \wedge \bigwedge_{i=1}^n (x_i^t \vee x_i^f) \wedge (\bar{x}_i^t \vee \bar{x}_i^f).$$

The SSP reduction is completed by specifying the embedding function $f : \mathcal{U} \rightarrow \mathcal{U}'$ via $f(x_i) := x_i^t$ and $f(\bar{x}_i) := x_i^f$. Clearly all clauses of φ' have length at most three. Note that this reduction is a correct reduction, i.e. it transforms YES-instances into YES-instances and NO-instances into NO-instances, because the added constraints make sure that exactly one of x_i^t and x_i^f is true. Furthermore, it has the SSP property: For every solution $S \subseteq \mathcal{U}$ of SATISFIABILITY ($\mathcal{U} = L$), the “translated” set $f(S) \subseteq \mathcal{U}'$ is a solution of SATISFIABILITY ($\mathcal{U} = X$). Furthermore, for every solution $S \subseteq \mathcal{U}'$ of SATISFIABILITY ($\mathcal{U} = X$), the set $f^{-1}(S) \subseteq \mathcal{U}$ is a solution of SATISFIABILITY ($\mathcal{U} = L$). Accordingly, we have a correct SSP reduction (where the SSP mapping f is even bijective due to $f(\mathcal{U}) = \mathcal{U}'$). \square

Theorem 8.6. *MIN CARDINALITY INTERDICTION-SATISFIABILITY ($\mathcal{U} = X$) is Σ_2^p -complete.*

Proof. By the previous lemma, SATISFIABILITY ($\mathcal{U} = X$) is SSP-NP-complete, even if all clauses are restricted to length three. Due to Chapter 4, the problem COMB. INTERDICTION-SATISFIABILITY ($\mathcal{U} = X$) is Σ_2^p -complete, even if all clauses are restricted to length three. We provide a reduction from the latter problem in terms of an invulnerability gadget analogous to the gadgets presented in Section 8.3. For this, consider an instance of SATISFIABILITY ($\mathcal{U} = X$) with formula φ in CNF and every clause of length three, together with the universe $\mathcal{U} = \{x_1, \dots, x_n\}$, a set $C \subseteq \mathcal{U}$ of vulnerable literals, and interdiction budget $k \in \mathbb{N}_0$. For every variable $x_i \in \mathcal{U} \setminus C$, we explain how to make x_i invulnerable. We introduce $k + 1$ new variables $x_i^{(1)}, \dots, x_i^{(k+1)}$. Our goal is to establish the equivalence

$$x_i \equiv x_i^{(1)} \vee \dots \vee x_i^{(k+1)}.$$

We can achieve this through means of the following substitution process starting from formula φ : Every occurrence of x_i in the formula gets substituted by $x_i^{(1)} \vee \dots \vee x_i^{(k+1)}$. Every occurrence of

\bar{x}_i gets substituted (by De Morgan's law) by $(\bar{x}_i^{(1)} \wedge \dots \wedge \bar{x}_i^{(k+1)})$. Note that this has two effects: First, the length of a clause may now exceed 3. Secondly, the formula is not in CNF anymore. Note however that we can use the distributive law to expand every clause that is not in CNF. Since before each clause had a length of at most three, this results in a blow-up of the instance size of a factor at most $(k+1)^3$, i.e. at most a polynomial factor. Let φ' be the resulting formula. We can see that there is an equivalence of the satisfying assignments of φ and φ' , in the sense that x_i is true in φ if and only if $x_i^{(1)} \vee \dots \vee x_i^{(k+1)}$ is true in φ' (for all invulnerable x_i). However, since the interdiction budget is only k , the interdictor can never enforce $x_i^{(1)} \vee \dots \vee x_i^{(k+1)}$ to be false for all invulnerable variables. This shows that COMB. INTERDICTION-SATISFIABILITY ($\mathcal{U} = X$) reduces to MIN. CARDINALITY INTERDICTION-SATISFIABILITY ($\mathcal{U} = X$), hence proving its Σ_2^p -completeness. \square

Note that the reasoning presented in this proof was slightly different from Theorem 8.3, since we start with a formula where every clause has length three, but do not preserve this property during the proof. Hence Σ_2^p -completeness is only shown in the case where clauses can have unrestricted length.

We can use an argument similar to Lemma 8.4 to show the coNP-completeness of the minimum cardinality interdiction version, the full decision variant of interdiction and the most vital elements problem of 3-SATISFIABILITY ($\mathcal{U} = X$), i.e. with clauses of length bounded by 3. Indeed, it is easy to see that the interdiction problem of SATISFIABILITY ($\mathcal{U} = X$) where every clause has length 3 is coNP-complete: If $k \geq 3$ holds for the interdiction budget, the attacker distinguishes two cases: If there is a clause with three positive literals, the attacker blocks all of them and immediately wins. In the other case, every clause has at least one negative literal. Then the attacker can never win, since the defender can set every variable to false, which is a satisfying assignment that can never be blocked. By an analogous argument, we can see that for any constant $b \geq 3$, the interdiction problem of SATISFIABILITY ($\mathcal{U} = X$) with clauses restricted to length b is coNP-complete.

Finally, we remark that slightly different variants of interdiction-3-Sat have been shown to be Σ_2^p -complete. In these variants, the interdictor does not have access to all variables (see Chapter 4 or [JKZ24a, Thm. 1]).

Chapter 9

SSP Reduction Compendium

In this chapter, we present a multitude of SSP-NP-complete problems. For this, we provide SSP definitions of all the corresponding problems, i.e. for each SSP problem we describe its set \mathcal{I} of input instances, as well as the universe $\mathcal{U}(I)$ and solution set $\mathcal{S}(I)$ associated to each instance $I \in \mathcal{I}$. We show SSP reductions between all the problems, hence proving that all considered problems are contained in the class SSP-NPc. In this chapter, we distinguish between SSP problems which are derived from an LOP problem, and those which are not (see Subsection 3.2.2 for an explanation). If the SSP problem is derived from an LOP problem, we additionally provide the set $\mathcal{F}(I)$ of feasible solutions. In all of these cases, the cost function $d^{(I)}$ and the threshold $t^{(I)}$ of the original LOP problem can be derived from the context.

All reductions presented in this chapter are already known and can be found in the literature. For each existing reduction, we shortly describe the known reduction $g : \mathcal{I} \rightarrow \mathcal{I}'$ as well as the functions $(f_I)_{I \in \mathcal{I}} : \mathcal{U}(I) \rightarrow \mathcal{U}'(g(I))$. However, for the sake of conciseness, we do not always show the correctness of the original reductions explicitly and focus only on the correct embedding into the SSP framework. In Figure 9.1, the tree of all presented reductions can be found, beginning at SATISFIABILITY, which is defined as follows. We heavily use the transitivity of SSP reductions (Lemma 3.1).

SATISFIABILITY

Instances: Literal Set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$, Clauses $C \subseteq 2^L$.

Universe: $L =: \mathcal{U}$.

Solution set: The set of all sets $L' \subseteq \mathcal{U}$ such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c_j| \geq 1$ for all $c_j \in C$, $j \in \{1, \dots, |C|\}$.

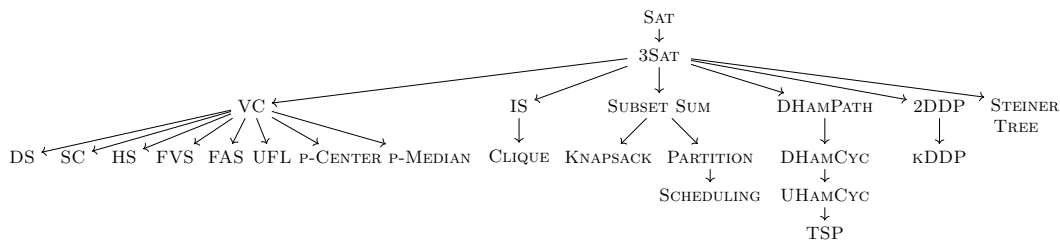


Figure 9.1: The tree of SSP reductions for all considered problems.

With the tree of SSP reductions shown in Figure 9.1, we derive the following theorem.

Theorem 9.1. *The following problems are SSP-NP-complete: SATISFIABILITY, 3SATISFIABILITY, VERTEX COVER, DOMINATING SET, SET COVER, HITTING SET, FEEDBACK VERTEX SET, FEEDBACK ARC SET, UNCAPACITATED FACILITY LOCATION, P-CENTER, P-MEDIAN, INDEPENDENT SET, CLIQUE, SUBSET SUM, KNAPSACK, PARTITION, SCHEDULING, DIRECTED HAMILTONIAN PATH, DIRECTED HAMILTONIAN CYCLE, UNDIRECTED HAMILTONIAN CYCLE, TRAVELING SALESMAN PROBLEM, TWO DIRECTED VERTEX DISJOINT PATH, k-VERTEX DIRECTED DISJOINT PATH, STEINER TREE*

Besides the standard SSP reductions, we also provide blow-up SSP reductions and blow-up preserving SSP reductions as discussed in Section 7.4. The tree of the corresponding blow-up (preserving) reductions can be found in Figure 9.2.

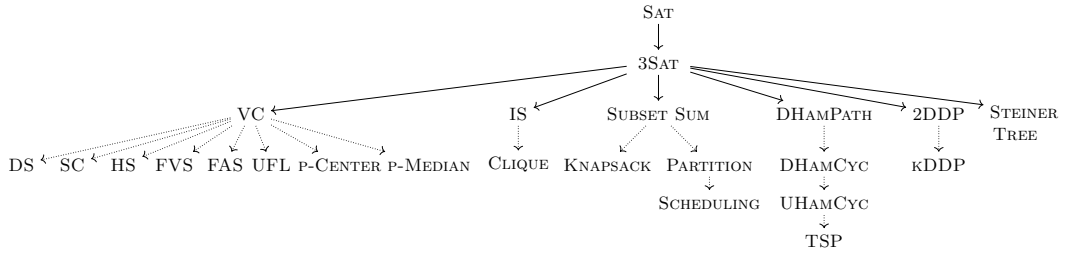


Figure 9.2: The tree of SSP reductions for all considered problems. While solid edges indicate that there is a blow-up SSP reduction between the problems, dotted edges represent a blow-up preserving SSP reduction between the problems.

The presentation of the SSP problems and reductions is structured as follows. We first state the SSP version of the problem in a box. For LOP problems, we additionally state the feasible solution set. After this, we present the SSP reduction for the corresponding problem. At last, we show that the given SSP reduction can either be extended to a blow-up SSP reduction or a blow-up preserving SSP reduction. We travel through the reduction graph (Figure 9.1) in a depth-first search manner.

3-SATISFIABILITY

Instances: Literal Set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$, Clauses $C \subseteq 2^L$ s.t. $\forall c_j \in C : |c_j| = 3$.

Universe: $L =: \mathcal{U}$.

Solution set: The set of all sets $L' \subseteq \mathcal{U}$ such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c_j| \geq 1$ for all $c_j \in C$.

We begin with Karp's reduction [Kar72] from SATISFIABILITY to 3SATISFIABILITY. We claim that this reduction is an SSP reduction. Let $I = (L, C)$ be the SAT instance and (L', C') be the corresponding 3SAT instance. The reduction maps each clause $c \in C$ of more than three literals to a set of clauses in C' of length three by introducing helper variables h_1, h_2, \dots splitting the clause into smaller clauses. Every clause $\{a, b, c, d, \dots\} \in C$ with more than three literals is recursively split until there are no more clauses with more than three literals as follows:

$$\{a, b, c, d, \dots\} \mapsto \{a, b, h_i\}, \{\bar{h}_i, c, d, \dots\}.$$

The number of splits is bounded by the length of the instance, thus it is computable in polynomial time.

The literals L of the SAT instance remain in the 3SAT instance and are one-to-one correspondent. Thus, the corresponding solutions have the SSP property by defining the functions $(f_I)_{I \in \mathcal{I}}$ by $f_I(\ell) = \ell \in L'$ for $\ell \in L$. Note that the set $f_I(L)$ is exactly the set of positive and negative literals corresponding to non-helper variables. It is easily verified that each satisfying assignment of the 3SAT instance restricted to the set $f_I(L)$ implies a satisfying assignment of the SAT instance (i.e. forgetting about the helper variables). This is property (P2) as explained in Subsection 3.1.3. Likewise, each satisfying assignment of the SAT instance can be completed to a satisfying assignment of the 3SAT instance by setting the helper variables appropriately. This is property (P1). Hence, the solutions of SAT and 3SAT correspond one-to-one to each other on the set $f_I(L)$, in the precise sense that $\{f_I(S) : S \in \mathcal{S}_{\text{Sat}}\} = \{S' \cap f_I(L) : S' \in \mathcal{S}_{3\text{Sat}}\}$. Thus this reduction is indeed an SSP reduction.

Blow-up SSP Reduction. We are able to extend the reduction with the following blow-up gadget: For a blow-up factor of β_I , we add β_I new variables $\ell^1, \dots, \ell^{\beta_I}$ for each existing variable $\ell \in B_{up} \subseteq L$. By introducing additional clauses $(\ell \vee \bar{\ell}^i), (\bar{\ell} \vee \ell^i)$ for all $i \in \{1, \dots, \beta_I\}$, we ensure that ℓ and all its corresponding blow-up literals $\ell^1, \dots, \ell^{\beta_I}$ are logically equivalent. Consequently if the literal ℓ is exchanged with $\bar{\ell}$, then all literals $\ell, \ell^1, \dots, \ell^{\beta_I}$ have to be exchanged with $\bar{\ell}, \bar{\ell}^1, \dots, \bar{\ell}^{\beta_I}$.

In order to calculate the β_I for the three distance measures, we observe that a SAT solution always includes exactly one of the literals ℓ or $\bar{\ell}$. Thus, Alice may choose all literals of $L \setminus L_b$ wrongly. Consequently, we get the following β_I :

- ▶ κ -addition: $\beta_I = |L \setminus L_b|/2$
- ▶ κ -deletion: $\beta_I = |L \setminus L_b|/2$
- ▶ Hamming distance: $\beta_I = |L \setminus L_b|$.

VERTEX COVER

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all vertex covers.

Solution set: The set of all vertex covers of size at most k .

The reduction of Garey and Johnson [GJ79] from 3SAT to VERTEX COVER is an SSP reduction. In order to show this, we reformulate the reduction and adapt it to the SSP framework. Let $I = (L, C)$ be the 3SAT instance with literals L and clauses C . We define the corresponding instance $g(L, C)$ of VERTEX COVER as the following tuple $(G', k') = ((V', E'), k')$. Each literal $\ell \in L$ is mapped to a literal vertex $v_\ell \in V'$, where v_ℓ and its negation $v_{\bar{\ell}}$ are connected by an edge $\{v_\ell, v_{\bar{\ell}}\} \in E'$. Furthermore, we introduce a 3-clique for each clause $c \in C$. Each of the three vertices $v_{\ell_{i_1}}^c, v_{\ell_{i_2}}^c, v_{\ell_{i_3}}^c$ represents a literal in the clause, and is then connected to the corresponding literal vertex, i.e. $\{v_\ell, v_{\bar{\ell}}\} \in E'$ for $\ell \in c$. Finally, we define the parameter k' by $k' := |L|/2 + 2|C|$. An example instance can be found in Figure 9.3, where the set of literal vertices is denoted by W .

The universe elements of 3SAT are injectively mapped to the literal vertices in $W \subseteq V'$, where $f_I(\ell) = v_\ell$. All valid solutions (if there are any) include exactly one of v_ℓ or $v_{\bar{\ell}} \in W$ and two additional vertices from the 3-clique corresponding to each clause. To cover a 3-clique at least two vertices of that 3-clique have to be in the vertex cover. If a clause is not satisfied, then no neighboring literal vertex is in the solution. In this case all three vertices of the 3-clique simulating the clause have to be taken into the solution (otherwise the edges connecting the

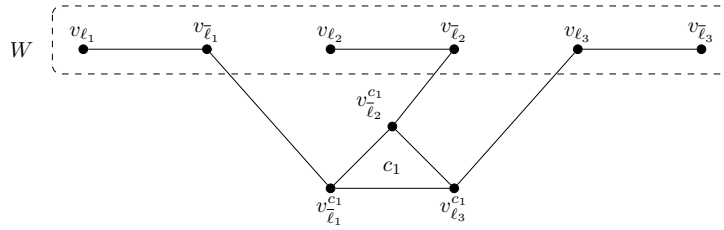


Figure 9.3: Classic reduction of 3SAT to VERTEX COVER for $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$.

literal vertices with the 3-clique are not covered). But then this vertex cover must already have size more than k' . In total, we have that every vertex cover of size at most k' restricted to the set W corresponds to a solution of 3SAT. On the other hand, every solution of 3SAT can be transferred over to the set W and be completed in at least one way to a vertex cover of size at most k' , i.e. the following equation holds true

$$\{f_I(S) : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{3\text{SAT}}\} = \{S' \cap f_I(L) : S' \in \mathcal{S}_{VC}\}.$$

Thus, the SSP reduction is indeed correct.

Blow-up SSP Reduction. This SSP reduction can also be extended to a blow-up SSP reduction. We have already shown this in Claim 7.2 by introducing a corresponding blow-up gadget.

DOMINATING SET

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all dominating sets.

Solution set: The set of all dominating sets of size at most k .

For a reduction from VERTEX COVER to DOMINATING SET, we use a (modified) folklore reduction as depicted in Figure 9.4. Let $I = ((V, E), k)$ be the VERTEX COVER instance and $((V', E'), k')$ be the DOMINATING SET instance. For every two vertices $v, w \in V$ connected by an edge $\{v, w\}$, we introduce $|V| + 1$ additional vertices vw_i for $i \in \{1, \dots, |V| + 1\}$ and connect them to v and to w . All isolated vertices $v \in V$ are mapped to itself, that is $v \in V'$. Furthermore, we introduce a star around vertex v_{iso} connected to vertices v_{iso}^i for $i \in \{1, \dots, |V| + 1\}$. Then, we connect v_{iso} to all the isolated vertices from V' . The parameter k' is set to $k' = k + 1$. With this construction, the vertex cover is directly translatable to a dominating set in the DOMINATING SET instance by leaving it as is and by taking v_{iso} into the dominating set. Note that v_{iso} dominates itself, all v_{iso}^i for $i \in \{1, \dots, |V| + 1\}$, and all originally isolated vertices. The other way around, we claim that every dominating set of size at most $k' = k + 1$ in the new graph has the property that restricted to the set W it encodes a vertex cover of size at most k of the old graph. Indeed, observe that one needs at least $|V| + 1$ vertices to dominate all vertices v, w and vw_i $i \in \{1, \dots, |V| + 1\}$ of one edge, the same holds for the star around v_{iso} . Thus for all dominating sets and for all “original” edges $\{v, w\}$, one needs to include either v or w . Note that v_{iso} is always part of the dominating set because it is the center of large star and all of the originally isolated vertices are dominated. Consequently, we have a one-to-one correspondence between the vertex cover and the dominating set and the solutions are preserved accordingly. This one-to-one correspondence with the mapping $f_I(v) = v \in V'$ for all $v \in V$ directly implies that this is an SSP reduction.

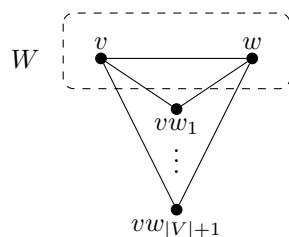


Figure 9.4: The modified reduction of VERTEX COVER to DOMINATING SET.

Blow-up preserving SSP reduction. This reduction is also a blow-up preserving SSP reduction. We have already shown this in Claim 7.4 by proving that $U_{on} = \{v_{iso}\}$ and $U_{off} = \{vw_i : \{v, w\} \in E, 1 \leq i \leq |V| + 1\}$.

SET COVER

Instances: Sets $S_i \subseteq \{1, \dots, m\}$ for $i \in \{1, \dots, n\}$, number $k \in \mathbb{N}$.

Universe: $\{S_1, \dots, S_n\} =: \mathcal{U}$.

Feasible solution set: The set of all $S \subseteq \{S_1, \dots, S_n\}$ s.t. $\bigcup_{s \in S} s = \{1, \dots, m\}$.

Solution set: Set of all feasible solutions with $|S| \leq k$.

The reduction from VERTEX COVER to SET COVER by Karp [Kar72] is an SSP reduction. VERTEX COVER and SET COVER are basically the same problem, which means that the syntax of the input is the same, however the semantics behind the encoding are different. Thus, the reduction of Karp implies a direct one-to-one correspondence not only between the universe elements but also between the edges and the sets. Let $I = ((V, E), k)$ be the VERTEX COVER instance. In the reduction, each vertex $v \in V$ is mapped to the set S_v and each edge $e \in E$ is mapped to the set $\{1, \dots, |E|\} = \{1, \dots, m\}$ according to their index. Each set S_v includes its the indices of the incident edges. Thus, if a vertex $v \in V$ is taken into the vertex cover all incident edges are covered which is equivalent to including S_v into the set cover such that all elements of S_v are covered, which are exactly the indices of all incident edges to v . Consequently, the one-to-one correspondence is defined by $f_I(v) = S_v$ for $v \in V$ as desired, which also implies that this is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the SET COVER instance universe originate from a universe element of the VERTEX COVER instance. Since vertex v is in the VERTEX COVER solution if and only if the corresponding set S_v is in the SET COVER solution, we define $f_I(v) = S_v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

HITTING SET

Instances: Sets $S_j \subseteq \{1, \dots, n\}$ for $j \in \{1, \dots, m\}$, number $k \in \mathbb{N}$.

Universe: $\{1, \dots, n\} =: \mathcal{U}$.

Feasible solution set: The set of all $H \subseteq \{1, \dots, n\}$ such that $H \cap S_j \neq \emptyset$ for all $j \in \{1, \dots, m\}$.

Solution set: Set of all feasible solutions with $|H| \leq k$.

Karp's reduction [Kar72] from VERTEX COVER to HITTING SET is an SSP reduction. Similar to SET COVER, HITTING SET is basically the same problem as VERTEX COVER. We only have

to reinterpret the semantic of the input encoding as follows. Let $I = ((V, E), k)$ be the VERTEX COVER instance. Each vertex $v \in V$ is mapped into the set $\{1, \dots, |V|\}$ by its index $id(v)$. Then, each edge $e = \{v, w\} \in E$ is mapped to the set $S_e = \{id(v), id(w)\}$ (thus $m = |E|$). It follows that each vertex is exactly one-to-one correspondent to its index and the solutions are preserved because every vertex cover covers all edges which is equivalent that the corresponding hitting set induced by this vertex cover hits all S_e . Consequently, we can define $f_I(v) = id(v)$ and we have an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the HITTING SET instance universe originate from a universe element of the VERTEX COVER instance. A vertex v is in the VERTEX COVER solution if and only if the corresponding index $id(v)$ is in the HITTING SET solution. Then, we have $f_I(v) = id(v)$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

FEEDBACK VERTEX SET

Instances: Directed Graph $G = (V, A)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all vertex sets $V' \subseteq V$ such that after deleting V' from G , the resulting graph is cycle-free (i.e. a forest).

Solution set: The set of all feasible solutions V' of size at most k .

The reduction by Karp [Kar72] from VERTEX COVER to FEEDBACK VERTEX SET is an SSP reduction. Let $I = (G, k) = ((V, E), k)$ be the VERTEX COVER instance and $(G', k') = ((V', A'), k')$ the FEEDBACK VERTEX SET instance. The transformation maps every vertex $v \in V$ to itself ($v \in V'$) and every edge $\{v, w\} \in E$ is mapped to two arcs $(v, w), (w, v) \in A'$ orienting the edge in both directions. We further set $k = k'$. We define the injective embedding function f_I by the identity on the vertices, i.e. every vertex in V is mapped onto its corresponding twin in V' . Note that the solutions are also directly one-to-one transformable and thus preserved. To see this, assume to have a vertex cover for the graph G , then the same set removes all cycles from the directed graph G' , because a vertex cover is incident to all edges E in G and thus to all arcs A' in G' resulting in an independent set, which is obviously cycle-free. On the other hand, a solution to the FEEDBACK VERTEX SET instance G' needs to remove all cycles. Each cycle is induced by two vertices connected by both arcs $(v, w), (w, v) \in A'$. Thus, at least one vertex of v and w has to be deleted such that both arcs $(v, w), (w, v) \in A'$ are also deleted and do not form a cycle. This, however, is obviously equivalent to a vertex cover in G .

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the FEEDBACK VERTEX SET instance universe originate from a universe element of the VERTEX COVER instance. A vertex v is in the VERTEX COVER solution if and only if the corresponding vertex v is in the FEEDBACK VERTEX SET solution. Then, we have $f_I(v) = v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

FEEDBACK ARC SET

Instances: Directed Graph $G = (V, A)$, number $k \in \mathbb{N}$.

Universe: Arc set $A =: \mathcal{U}$.

Feasible solution set: The set of all arc sets $A' \subseteq A$ such that after deleting A' from G , the resulting graph is cycle-free (i.e. a forest).

Solution set: The set of all feasible solutions A' of size at most k .

A modification of the reduction by Karp [Kar72] from VERTEX COVER to FEEDBACK ARC SET is an SSP reduction. Let $I = (G, k) = ((V, E), k)$ be the VERTEX COVER instance and $(G', k') = ((V', A'), k')$ the FEEDBACK ARC SET instance. This reduction is more complicated in the SSP framework in comparison to the reductions, we have seen before. Due to the fact that the universe is changed from the vertex set to the arc set, we have to be more careful in analyzing the individual mappings. First of all, we transform the vertices $v \in V$ to two vertices $v_0, v_1 \in V'$. We define the injective embedding function f_I by mapping each vertex $v \in V$ to the arc $(v_0, v_1) \in A'$, which is also the corresponding element in all solutions, that is $f_I(v) = (v_0, v_1)$. At last, we transform each edge $\{v, w\} \in E$ to $|V + 1|$ once subdivided arcs from v_1 to w_0 and to $|V + 1|$ once subdivided arcs from w_1 to v_0 . Finally, we leave the parameter $k = k'$ unchanged. This completes the description of the reduction. We denote the vertices added by the subdivision v_1^i for the arcs between v_1 to w_0 and w_1^i between w_1 to v_0 for $i \in \{1, \dots, |V| + 1\}$. Overall, one vertex pair with an edge induces more than $|V|$ cycles $(v_0, v_1), (v_1, v_1^i)(v_1^i, w_0), (w_0, w_1), (w_1, w_1^i), (w_1^i, v_0)$ of length six. By deleting the arc (v_0, v_1) , which corresponds to vertex v in G all of these induced cycles are disconnected. This implies that every vertex cover of G is translated to a feedback arc set by the function f_I . On the other hand, a feedback arc set must contain for every original edge $\{v, w\}$ either the arc (w_0, w_1) or the arc (v_0, v_1) , because otherwise a cycle remains. This shows that every feedback arc set of size at most k , when restricted to the set $f_I(V)$ encodes a vertex cover of G of size at most k . Hence we have an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . First, a vertex v is in the VERTEX COVER solution if and only if the corresponding arc (v_0, v_1) is in the FEEDBACK ARC SET solution. Thus, we have $f_I(v) = (v_0, v_1)$. To define the set U_{off} , we have to consider the additionally introduced arcs of the form $(v_1, v_1^i)(v_1^i, w_0), (w_1, w_1^i), (w_1^i, v_0)$ for an edge $e = \{v, w\}$ and $i \in \{1, \dots, |V| + 1\}$. These elements are never part of a solution and are the only elements besides the elements from $f_I(\mathcal{U}(I))$. Accordingly, we can define $U_{\text{off}} = \{(v_1, v_1^i)(v_1^i, w_0), (w_1, w_1^i), (w_1^i, v_0) \mid e = \{v, w\} \in E, i \in \{1, \dots, |V| + 1\}\}$ and $U_{\text{on}} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

UNCAPACITATED FACILITY LOCATION

Instances: Set of potential facilities $F = \{1, \dots, n\}$, set of clients $C = \{1, \dots, m\}$, fixed cost of opening facility function $f : F \rightarrow \mathbb{Z}$, service cost function $c : F \times C \rightarrow \mathbb{Z}$, cost threshold $k \in \mathbb{Z}$

Universe: Facility set $F =: \mathcal{U}$.

Solution set: The set of sets $F' \subseteq F$ s.t.

$$\sum_{i \in F'} f(i) + \sum_{j \in C} \min_{i \in F'} c(i, j) \leq k.$$

Note that we define this problem explicitly as SSP and not as LOP because in the standard interpretation, the objective function is not linear. The reduction by Cornuéjols, Nemhauser, and Wolsey [CNW83] from VERTEX COVER to UNCAPACITATED FACILITY LOCATION is an SSP reduction. Let $I = ((V, E), k)$ be the VERTEX COVER instance and (F, C, f, c) be the UNCAPACITATED FACILITY LOCATION instance. We let $F := V$ and $C := E$. The injective embedding function f_I is given by $f_I(v) = v \in F$ for $v \in V$. Further, we define $c(v, e) = 0$ if $v \in e$ and $c(v, e) = |V| + 1$ otherwise. At last, we set $f(v) = 1$ for all $v \in F$ and leave the parameter k unchanged. The one-to-one correspondence between the solutions can be explained by analyzing the correctness of the reduction. On the one hand, a vertex cover S is a solution to the facility location problem, because at most k many facilities are opened and all clients

$e \in C$ are served, which corresponds exactly that all edges $e \in E$ are covered by $v \in S$. On the other hand, if there is a facility set F' with cost k , then it has to include at most k facilities and additionally serve all clients $e \in C$, i.e. the corresponding vertex set covers all edges $e \in E$ (because of the high costs of $c(v, e) = |V| + 1$ for $v \notin e$). Thus, this is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the UNCAPACITATED FACILITY LOCATION instance universe originate from a universe element of the VERTEX COVER instance. A vertex v is in the VERTEX COVER solution if and only if the corresponding facility v is in the UNCAPACITATED FACILITY LOCATION solution. Then, we have $f_I(v) = v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

P-CENTER

Instances: Set of potential facilities $F = \{1, \dots, n\}$, set of clients $C = \{1, \dots, m\}$, service cost function $c : F \times C \rightarrow \mathbb{Z}$, facility threshold $p \in \mathbb{N}$, cost threshold $k \in \mathbb{Z}$

Universe: Facility set $F =: \mathcal{U}$.

Solution set: The set of sets $F' \subseteq F$ s.t. $|F'| \leq p$ and $\max_{j \in C} \min_{i \in F'} c(i, j) \leq k$.

Like in the previous problem, we cannot interpret this problem as an LOP problem since the objective is not linear. A modified version of the reduction by Cornuéjols, Nemhauser, and Wolsey [CNW83] from VERTEX COVER to UNCAPACITATED FACILITY LOCATION is an SSP reduction. Let $I = ((V, E), k)$ be the VERTEX COVER instance and (F, C, c, p, k') be the P-CENTER instance. We map each $v \in V$ to $v \in F$ and each $e \in E$ to $e \in C$. Further, we define $c(v, e) = 0$ if $v \in e$ and $c(v, e) = |V| + 1$ otherwise. At last, we set p equal to the size k of the vertex cover and $k' = 0$. Note that this implies that in a solution the objective has to be 0. We now argue analogous to the reduction to UNCAPACITATED FACILITY LOCATION. The embedding function f_I is given by the one-to-one correspondence between the universe elements $v \in V$ and $v \in F$. The rest of the argument is analogous to the above.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the P-CENTER instance universe originate from a universe element of the VERTEX COVER instance. A vertex v is in the VERTEX COVER solution if and only if the corresponding facility v is in the P-CENTER solution. Then, we have $f_I(v) = v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

P-MEDIAN

Instances: Set of potential facilities $F = \{1, \dots, n\}$, set of clients $C = \{1, \dots, m\}$, service cost function $c : F \times C \rightarrow \mathbb{Z}$, facility threshold $p \in \mathbb{N}$, cost threshold $k \in \mathbb{Z}$

Universe: Facility set $F =: \mathcal{U}$.

Solution set: The set of sets $F' \subseteq F$ s.t. $|F'| \leq p$ and $\sum_{j \in C} \min_{i \in F'} c(i, j) \leq k$.

Like in the previous problem, we cannot interpret this problem as an LOP problem since the objective is not linear. A modified version of the reduction by Cornuéjols, Nemhauser, and Wolsey [CNW83] from VERTEX COVER to UNCAPACITATED FACILITY LOCATION is an SSP reduction. It is the same as for P-CENTER. Let $I = ((V, E), k)$ be the VERTEX COVER instance and (F, C, c, p, k') be the P-MEDIAN instance. We map each $v \in V$ to $v \in F$ and each $e \in E$ to $e \in C$. Further, we define $c(v, e) = 0$ if $v \in e$ and $c(v, e) = |V| + 1$ otherwise. At last, we set $p = k$ equals to the size of the vertex cover and $k' = 0$. The one-to-one correspondence between

the universe elements $v \in V$ and $v \in F$ defines the embedding function f_I . The rest of the argument is analogous to the above.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the P-MEDIAN instance universe originate from a universe element of the VERTEX COVER instance. A vertex v is in the VERTEX COVER solution if and only if the corresponding facility v is in the P-MEDIAN solution. Then, we have $f_I(v) = v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

INDEPENDENT SET

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all independent sets.

Solution set: The set of all independent sets of size at least k .

For a reduction from 3SAT to INDEPENDENT SET, we use a folklore reduction, which is based on the reduction from 3SAT to VERTEX COVER by Garey and Johnson [GJ79]. Let $I = (L, C)$ be the 3SAT instance. We define a corresponding VERTEX COVER instance $((V', E'), k')$. Every literal $\ell \in L$ is transformed to a vertex $v_\ell \in V'$ and every pair $(\ell, \bar{\ell}) \in L \times L$ is again transformed to an edge $\{v_\ell, v_{\bar{\ell}}\} \in E'$ between the corresponding literal vertices. Every clause $c \in C$ is again transformed to a 3-clique, where each vertex $v_{\ell_{i_1}}^c, v_{\ell_{i_2}}^c, v_{\ell_{i_3}}^c$ represents a literal in the clause. In contrast to the VERTEX COVER reduction, the clause vertices are connected to the opposite literal vertex. For example in Figure 9.5, we have that the 3SAT clause is given by $\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3$. Hence, the clique is connected to the corresponding literal vertices ℓ_1, ℓ_2 and $\bar{\ell}_3$. Finally, we define the parameter k' by $k' := |L|/2 + |C|$.

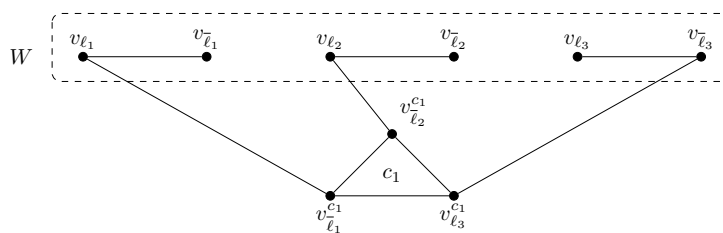


Figure 9.5: Classical reduction of 3SAT to INDEPENDENT SET for $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$.

Again, the universe elements of 3SAT are injectively mapped to the literal vertices in $W \subseteq V'$, where $f_I(\ell) = v_\ell \in V'$. All solutions included exactly one of v_ℓ or $v_{\bar{\ell}} \in W$ corresponding to ℓ and $\bar{\ell}$ in the 3SAT solution as well as one additional vertex for each clause. Note that every independent set has size at most k' , since only one vertex of every 3-clique and every 2-clique can be taken into the solution. Whenever a clause $c \in C$ is not satisfied, all three vertices in the clause c are blocked from the independent set from the opposite literals that are in the solution.

In total, we have that every independent set of size at least k' restricted to the set W corresponds to a solution of 3SAT. On the other hand, every solution of 3SAT can be transferred over to the set W and be completed in at least one way to an independent set of size at least k' , i.e. the following equation holds true

$$\{f_I(S) : S \subseteq L \text{ s.t. } S \in \mathcal{S}_{3\text{SAT}}\} = \{S' \cap f_I(L) : S' \in \mathcal{S}_{IS}\}.$$

Thus, the SSP reduction is correct.

Blow-up SSP reduction. We show that a blow-up gadget exists. This is also a complete bipartite graph K_{β_I+1, β_I+1} as in the VERTEX COVER reduction and is depicted in Figure 9.6.

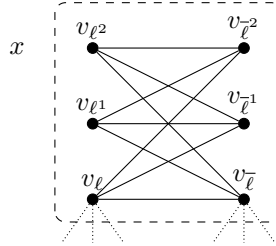


Figure 9.6: Blow-up gadget for the reduction of 3SAT to INDEPENDENT SET with blow-up factor of $\beta = 2$.

The argument is analogous to the one from the VERTEX COVER reduction in Subsection 7.3.1. That is, either all literal vertices $v_{\ell^1}, \dots, v_{\ell^\beta}$ or all literal vertices $v_{\bar{\ell}^1}, \dots, v_{\bar{\ell}^\beta}$ are in the solution since the edge between v_{ℓ^i} and $v_{\bar{\ell}^j}$ for some $i \neq j$ makes the independent set invalid. In order to calculate the β_I , we again analyze the solution structure. Exactly one of the vertices v_ℓ and $v_{\bar{\ell}}$ for $\ell, \bar{\ell} \in L \setminus L_b$ need to be included as well as exactly one of $v_{\ell_1}^{c_j}, v_{\ell_2}^{c_j}, v_{\ell_3}^{c_j}$. If we assume all of them are wrongly chosen, we receive the following β_I for the three distance measures:

- ▶ κ -addition: $\beta_I = |C| + |L \setminus L_b|/2$
- ▶ κ -deletion: $\beta_I = |C| + |L \setminus L_b|/2$
- ▶ Hamming distance: $\beta_I = 2|C| + |L \setminus L_b|$.

CLIQUE

Instances: Graph $G = (V, E)$, number $k \in \mathbb{N}$.

Universe: Vertex set $V =: \mathcal{U}$.

Feasible solution set: The set of all cliques.

Solution set: The set of all cliques of size at least k .

There is a reduction by Garey and Johnson [GJ79] from INDEPENDENT SET to CLIQUE, which is an SSP reduction. Let $I = (G, k) = ((V, E), k)$ be the INDEPENDENT SET instance and $(G', k') = ((V', E'), k')$ the CLIQUE instance. The reduction simply maps every vertex $v \in V$ to itself in V' . Furthermore every edge $\{v, w\} \in E$ mapped to a non-edge $\{v, w\} \notin E'$ and every non-edge $\{v, w\} \notin E$ is mapped to an edge $\{v, w\} \in E'$. Thus, every independent set $S \subseteq V$ in G is transformed in to a clique of the same vertices $S \subseteq V = V'$ in G' . By this transformation, the vertices are directly one-to-one correspondent with $f_I(v) = v$. Thus, this reduction is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the CLIQUE instance universe originate from a universe element of the INDEPENDENT SET instance. A vertex v is in the INDEPENDENT SET solution if and only if the corresponding vertex v is in the CLIQUE solution. Then, we have $f_I(v) = v$ and $U_{on} = U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

SUBSET SUM

Instances: Numbers $\{a_1, \dots, a_n\} \subseteq \mathbb{N}$, and target value $M \in \mathbb{N}$.

Universe: $\{a_1, \dots, a_n\} =: \mathcal{U}$.

Solution set: The set of all sets $S \subseteq \mathcal{U}$ with $\sum_{a_i \in S} a_i = M$.

The reduction by Sipser [Sip97] from 3SAT to SUBSET SUM is an SSP reduction. Let $I = (L, C)$ be the 3SAT instance. We define a SUBSET SUM instance $(\{a_1, \dots, a_n\}, M)$. We create a table as depicted in Figure 9.7 to transform each literal pair $(\ell_i, \bar{\ell}_i)$ (or variable x_i) into a number whose binary representation has length $|L|/2 + |C|$.

	x_1	x_2	x_3	$c_1 = \bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3$
s_1	1	0	0	0
s_2	1	0	0	1
s_3	0	1	0	0
s_4	0	1	0	1
s_5	0	0	1	1
s_6	0	0	1	0
s_7	0	0	0	1
s_8	0	0	0	2
Σ	1	1	1	4

Figure 9.7: Classical reduction of 3SAT to SUBSET SUM for $\varphi = (\bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3)$.

We fix an ordering on the variables and clauses to define the table. Each variable and each clause has a unique column i defining the i -th digit of each number. For each literal ℓ , we add a number that has a 1 exactly at the position of the corresponding variable and additional 1s at the positions of clauses that contain the literal. The target sum M is to be defined as 1 in each variable column and 4 in each clause column. This means that exactly one of the numbers corresponding to a literal pair can be added to the solution. Furthermore to satisfy a clause $c \in C$, the sum of the column corresponding to c has to be exactly 4. Thus, we add two numbers, a_c^1 and a_c^2 , for each clause $c \in C$: a_c^1 contains a 1 and a_c^2 contains a 2 in the column of clause c . Consequently, whenever a clause c is satisfied, i.e. the sum of the columns of c is greater than 1, the sum can be expanded to exactly 4. This reduction can be transformed into binary (and any other) encoding as well by introducing leading zeros such that no carryover occurs.

Note that the described SUBSET SUM instance in total contains the following numbers: Two numbers a_i, \bar{a}_i for every literal pair $(\ell_i, \bar{\ell}_i)$ plus some additional helper numbers. We define the injective embedding function f_I by $f_I(\ell_i) = a_i$ and $f_I(\bar{\ell}_i) = \bar{a}_i$. Note that with respect to this embedding f_I we have the SSP property, i.e. every subset of numbers with total sum M restricted to the set $f_I(L)$ encodes a correct solution of 3SAT. Therefore, this reduction is an SSP reduction.

Blow-up SSP reduction. The reduction can be extended to a blow-up SSP reduction by defining a blow-up gadget. For this, we expand the table as depicted in Figure 9.8.

Let β_I be the blow-up factor. For every literal ℓ_i , we add β_I additional columns representing a copy of ℓ_i . We then take the number a_i and add 1s to the β_I newly introduced columns. That also means that the number a_i now has a 0 in the β_I columns of $\bar{\ell}_i$. The target sum is extended with 1s for each of the $\beta_I |L|$ newly introduced columns. In order to fill up the 0s for every number, we add $\beta_I \cdot |L|$ new numbers b_i^ℓ , $1 \leq i \leq \beta_I$, having exactly one 1 in each of the newly introduced columns for every literal ℓ . The logic of the original reduction is still valid,

	x_1	ℓ_1^1	ℓ_1^2	ℓ_1^3	$\bar{\ell}_1^1$	$\bar{\ell}_1^2$	$\bar{\ell}_1^3$...	$c_1 = \bar{\ell}_1 \vee \bar{\ell}_2 \vee \ell_3$
s_1	1	0	0	0	1	1	1	...	0
s_2	1	1	1	1	0	0	0	...	1
s_1^1	0	1	0	0	0	0	0	...	0
s_1^2	0	0	1	0	0	0	0	...	0
s_1^3	0	0	0	1	0	0	0	...	0
s_2^1	0	0	0	0	1	0	0	...	0
s_2^2	0	0	0	0	0	1	0	...	0
s_2^3	0	0	0	0	0	0	1	...	0
s_3	0	0	0	0	0	0	0	...	1
s_4	0	0	0	0	0	0	0	...	2
Σ	1	1	1	1	1	1	1	...	4

Figure 9.8: Blow-up gadget for the reduction of 3SAT to SUBSET SUM with blow up $\beta_I = 3$.

that is $f(\ell) = a_i$. However, if a_i is taken into the solution, β_I many numbers b_i^ℓ , $1 \leq i \leq \beta_I$, have also taken into the solution to fill up the newly introduced 0 columns. Thus if the variable assignment shall be changed, $\beta_I + 1$ numbers (a_i and the b_i^ℓ , $1 \leq i \leq \beta_I$) have to be exchanged inducing a large distance between the solutions.

For computing the β_I for the three distance measures, we note that the literal numbers a_i might be wrongly chosen for all $L \setminus L_b$ as well as the numbers $a_{c_j}^1$ and $a_{c_j}^2$ for $c_j \in C$, where one of $a_{c_j}^1$ and $a_{c_j}^2$ is always in the solution. Thus, we have:

- ▶ κ -addition: $\beta_I = |C| + |L \setminus L_b|/2$
- ▶ κ -deletion: $\beta_I = |C| + |L \setminus L_b|/2$
- ▶ Hamming distance: $\beta_I = 2|C| + |L \setminus L_b|$.

KNAPSACK

Instances: Objects with prices and weights

$\{(p_1, w_1), \dots, (p_n, w_n)\} \subseteq \mathbb{N}^2$, and $W, P \in \mathbb{N}$.

Universe: $\{(p_1, w_1), \dots, (p_n, w_n)\} =: \mathcal{U}$.

Feasible solution set: The set of all $S \subseteq \mathcal{U}$ with

$\sum_{(p_i, w_i) \in S} w_i \leq W$.

Solution set: The set of feasible S with $\sum_{(p_i, w_i) \in S} p_i \geq P$.

There is a folklore reduction from SUBSET SUM to KNAPSACK. The SUBSET SUM instance $I = (\{a_1, \dots, a_n\}, M)$ can be transformed to a KNAPSACK instance $(\{(a_1, a_1), \dots, (a_n, a_n)\}, W, P)$ of objects of the same price and weight. Furthermore, the target value M is mapped to the weight threshold $W = M$ and price threshold $P = M$. Thus, $\sum_{(a_i, a_i) \in S} a_i \geq M$ and $\sum_{(a_i, a_i) \in S} a_i \leq M$ is equivalent to $\sum_{(a_i, a_i) \in S} a_i = M$. The one-to-one correspondence between the number a_i and the object (a_i, a_i) , i.e. $f_I(a_i) = (a_i, a_i)$, such that this reduction is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the KNAPSACK instance universe originate from a universe element of the SUBSET SUM instance. A number a is in the SUBSET SUM solution if and only if the corresponding object (a, a) is in the KNAPSACK solution. Then, we have $f(a) = (a, a)$ and $U_{\text{on}} = U_{\text{off}} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

PARTITION

Instances: Numbers $\{a_1, \dots, a_n\} \subseteq \mathbb{N}$.

Universe: $\{a_1, \dots, a_n\} =: \mathcal{U}$.

Solution set: The set of all sets $S \subseteq \mathcal{U}$ with $a_n \in S$ and

$$\sum_{a_i \in S} a_i = \sum_{a_j \notin S} a_j.$$

Note that we demand w.l.o.g. the last element to be in the solution. With this, we avoid symmetry of solutions, i.e. if S is a solution then $\mathcal{U} \setminus S$ is also a solution, which is not compatible with the SSP framework in the following reduction. The problems SUBSET SUM and PARTITION are almost equivalent such that the reduction between them is easy-to-see. We use basically the same reduction as Karp's [Kar72] from KNAPSACK to PARTITION. For this let $I = (\{a_1, \dots, a_n\}, M)$ be the SUBSET SUM instance. We map each number a_i to itself in the PARTITION instance and add additional numbers $M + 1$ and $\sum_i a_i + 1 - M$, whereby we set $a_n = \sum_i a_i + 1 - M$. Thus, the first $n - 2$ a_i in PARTITION are one-to-one correspondent with the a_i from SUBSET SUM, i.e. $f_I(a_i) = a_i$, such that this reduction is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . Besides the elements $\sum_i a_i + 1 - M$ and $M + 1$, all elements in the PARTITION instance universe originate from a universe element of the SUBSET SUM instance. A number a is in the SUBSET SUM solution if and only if the corresponding number a is in the PARTITION solution. We can define $f(a) = a$, $U_{\text{on}} = \{\sum_i a_i + 1 - M\}$, and $U_{\text{off}} = \{M + 1\}$. It follows that this SSP reduction is also blow-up preserving.

TWO MACHINE SCHEDULING

Instances: Jobs with processing time $\{t_1, \dots, t_n\} \subseteq \mathbb{N}$, threshold $T \in \mathbb{N}$.

Universe: The set of jobs $\{t_1, \dots, t_n\} =: \mathcal{U}$.

Solution set: The set of all $J_1 \subseteq \mathcal{U}$ such that $t_n \in J_1$ and $\sum_{t_i \in J_1} t_i \leq T$ and $\sum_{t_j \in J_2} t_j \leq T$ with $J_2 = \mathcal{U} \setminus J_1$, i.e. both machines finish in time T .

Again, we demand w.l.o.g. the last element to be in the solution for the first machine as in PARTITION. With this, we break the symmetry of solutions, which is not compatible with the SSP framework in the following reduction. The reduction from PARTITION to TWO-MACHINE-SCHEDULING is a folklore reduction, which exploits the equivalence of the problems and is therefore easy-to-see. For this let $I = \{a_1, \dots, a_n\}$ be the PARTITION instance. We transform each number a_i in the PARTITION instance to a job with processing time a_i in the TWO-MACHINE-SCHEDULING instance and set the threshold $T = \frac{1}{2} \sum_i a_i$.

Because for both sets J_1 and J_2 holds $\sum_{a_i \in J_1} a_i \leq T$ and $\sum_{a_j \in J_2} a_j \leq T$ as well as that $\sum_{a_i \in J_1} a_i + \sum_{a_j \in J_2} a_j = 2T$. We can transform the constraints above to the equivalent constraints $\sum_{t_i \in J_1} t_i = T$ and $\sum_{t_j \in J_2} t_j = T$. Therefore, we can interpret the two sets of the partition as the two machines in TWO-MACHINE-SCHEDULING and have a direct one-to-one correspondence between the solutions with $f_I(a_i) = a_i$. Thus, this reduction is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . All elements in the TWO-MACHINE-SCHEDULING instance universe originate from a universe element of the PARTITION instance. A number a is in the PARTITION solution if and only if the corresponding job with processing time a is in the TWO-MACHINE-SCHEDULING solution. Then, we have $f(a) = a$, $U_{\text{on}} = U_{\text{off}} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

DIRECTED HAMILTONIAN PATH

Instances: Directed Graph $G = (V, A)$, Vertices $s, t \in V$.

Universe: Arc set $A =: \mathcal{U}$.

Solution set: The set of all sets $C \subseteq A$ forming a Hamiltonian path going from s to t .

The reduction from 3SAT to DIRECTED HAMILTONIAN PATH from Arora and Barak [AB09] is an SSP reduction. Let $I = (L, C)$ be the 3SAT instance that we transform to the DIRECTED HAMILTONIAN CYCLE instance $G = (V, A)$. An example of the transformation can be found in Figure 9.9. First, we introduce two additional vertices $s, t \in V$. For each literal pair $(\ell_i, \bar{\ell}_i)$ (or variable x_i), we introduce a path with $4|C|$ vertices, where we denote the vertices along the path with $v_i^1, \dots, v_i^{4|C|}$. The path is directed in both ways such that v_i^1 is reachable from $v_i^{4|C|}$ and vice versa. The direction from v_i^1 to $v_i^{4|C|}$ encodes that ℓ is taken into the solution and the other direction encodes that $\bar{\ell}$ is taken into the solution. Additionally, we add arcs (s, v_1^1) and $(s, v_1^{4|C|})$, $(v_{|L|/2}^1, t)$ and $(v_{|L|/2}^{4|C|}, t)$ as well as $(v_i^1, v_{i+1}^{4|C|})$ and $(v_i^{4|C|}, v_{i+1}^1)$ for all $i \in \{1, \dots, |L|/2 - 1\}$. At last, we need to simulate the clauses. For this, we add a vertex for each clause $c_j \in C$ and connect them to the variable paths by introducing two arcs for each literal ℓ_i in the clause c_j . If ℓ_i is the non-negated literal of variable x_i , then we add the arcs (x_i^{4j-1}, c_j) and (c_j, x_i^{4j-2}) . Otherwise, we add the arcs (x_i^{4j-2}, c_j) and (c_j, x_i^{4j-1}) . Thus, one can satisfy the clause c_j , i.e. traveling over the vertex c_j , if and only if by traveling in the correct direction, i.e. whenever a literal in the clause is taken into the solution. Overall, a Hamiltonian path from s to t includes all vertices that is all clause vertices, i.e. all clauses are satisfied, and all vertices defined by literals. Consequently, each variable is assigned a value by the direction of the taken path.

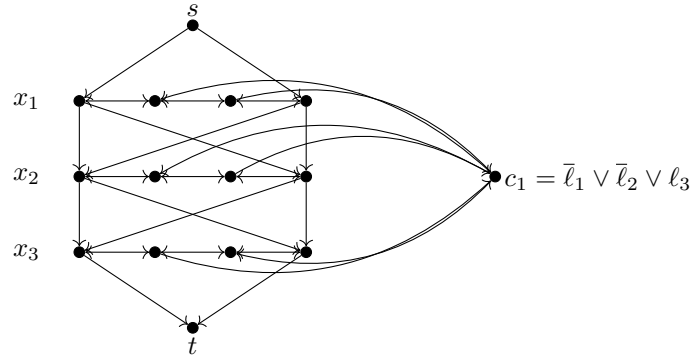


Figure 9.9: Classical reduction of 3SAT to DIRECTED HAMILTONIAN PATH for $\varphi = (\bar{\ell}_1 \wedge \bar{\ell}_2 \wedge \ell_3)$.

For this reduction, it is not directly obvious, how we find a one-to-one correspondence, because a whole path corresponds to one literal. However, we can use exactly one arc of that path to act as representative. We define the function f_I by $f_I(\ell_i) = (x_i^1, x_i^2)$ and $f_I(\bar{\ell}_i) = (x_i^2, x_i^1)$. Thus, we have a one-to-one correspondence between the literals and the arcs. Furthermore by correctness of the reduction, we have a one-to-one correspondence between the solutions to (L, C) and $G = (V, A)$ by Hamiltonian path using the either one of the arcs (x_i^1, x_i^2) and (x_i^2, x_i^1) for each $i \in \{1, \dots, |L|/2\}$ and including each clause vertex c_j for $j \in \{1, \dots, |C|\}$.

Blow-up SSP reduction. We can also find a blow-up gadget for this reduction. The idea is to lengthen the path by β_I additional vertices. Thus, we also receive β_I additional arcs. In

Figure 9.10, an example can be found, where b_1^1, b_1^2 and b_1^3 are the blow-up vertices, which also introduce the additional arcs.

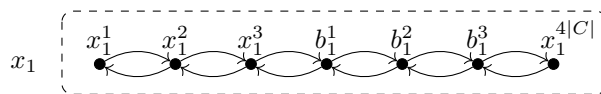


Figure 9.10: Blow-up gadget for the reduction of 3SAT to DIRECTED HAMILTONIAN PATH with blow up $\beta_I = 3$.

For computing the β_I for the three distance measures, the solution can be wrong on all literals in $L \setminus L_b$ and in this case the clause vertices c_j need to be included in the Hamiltonian path via a different literal path. Furthermore, the arcs connecting the literals as well as vertices s and t need to be changed for the literals from $L \setminus L_b$. Thus, we get the following β_I :

- ▶ κ -addition: $\beta_I = 2|C| + (4|C| + 2) \cdot |L \setminus L_b|/2$
- ▶ κ -deletion: $\beta_I = 2|C| + (4|C| + 2) \cdot |L \setminus L_b|/2$
- ▶ Hamming distance: $\beta_I = 4|C| + (8|C| + 4) \cdot |L \setminus L_b|/2$.

DIRECTED HAMILTONIAN CYCLE

Instances: Directed Graph $G = (V, A)$.

Universe: Arc set $A =: \mathcal{U}$.

Solution set: The set of all sets $C \subseteq A$ forming a Hamiltonian cycle.

We extend the reduction from 3SAT to DIRECTED HAMILTONIAN CYCLE from Arora and Barak [AB09] by simply adding an arc from t to s . Obviously, all possible cycles have to go through s and t . This has no influence on the rest of the reduction, especially on the solutions and the one-to-one correspondence of the literals and the arcs. Consequently, the reduction is still an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . Besides arc (t, s) , all elements in the DIRECTED HAMILTONIAN CYCLE instance universe originate from a universe element of the DIRECTED HAMILTONIAN PATH instance. An arc $a \in A$ is in the solution DIRECTED HAMILTONIAN PATH if and only if the corresponding arc is in the DIRECTED HAMILTONIAN CYCLE solution. Then, we have $f(a) = a$, $U_{on} = \{(t, s)\}$, and $U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

UNDIRECTED HAMILTONIAN CYCLE

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

Universe: Edge set $E =: \mathcal{U}$.

Solution set: The set of all sets $C \subseteq E$ forming a Hamiltonian cycle.

Karp's reduction [Kar72] from DIRECTED HAMILTONIAN CYCLE to UNDIRECTED HAMILTONIAN CYCLE is an SSP reduction. Let $I = (V, A)$ be the DIRECTED HAMILTONIAN CYCLE and (V', E') be the UNDIRECTED HAMILTONIAN CYCLE instance. The reduction replaces each vertex v with three vertices v'_{in}, v', v'_{out} and adds edges $\{v'_{in}, v'\}, \{v', v'_{out}\}$ to connect the three vertices to a path. All arcs $(v, w) \in A$ are replaced by one edge $\{v'_{out}, w'_{in}\}$ essentially preserving the one-to-one correspondence between the elements to the corresponding unique edge, i.e. $f_I(v, w) = \{v'_{out}, w'_{in}\}$. The solutions are also preserved because no additional solutions are added and all

original solutions are preserved (every Hamiltonian cycle has to run through each v'_{in}, v', v'_{out} exactly once in the specified order).

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . Besides the edges from $\{\{v'_{in}, v'\}, \{v', v'_{out}\} \mid v \in V\}$, all elements in the UNDIRECTED HAMILTONIAN CYCLE instance universe correspond directly to a universe element of the DIRECTED HAMILTONIAN CYCLE instance. An arc $a = (v, w) \in A$ is in the solution DIRECTED HAMILTONIAN CYCLE if and only if the corresponding edge $\{v'_{out}, w'_{in}\} \in E$ is in the UNDIRECTED HAMILTONIAN CYCLE solution. Since the edges $\{\{v'_{in}, v'\}, \{v', v'_{out}\} \mid v \in V\}$ are always in the Hamiltonian cycle, we have $f((v, w)) = \{v'_{out}, w'_{in}\}$, $U_{on} = \{\{v'_{in}, v'\}, \{v', v'_{out}\} \mid v \in V\}$, and $U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

TRAVELING SALESMAN PROBLEM

Instances: Complete Graph $G = (V, E)$, weight function $w : E \rightarrow \mathbb{Z}$, number $k \in \mathbb{N}$.

Universe: Edge set $E =: \mathcal{U}$.

Feasible solution set: The set of all TSP tours $T \subseteq E$.

Solution set: The set of feasible T with $w(T) \leq k$.

There is an easy-to-see folklore reduction from UNDIRECTED HAMILTONIAN CYCLE to TRAVELING SALESMAN PROBLEM, which is an SSP reduction. Let $I = (V, E)$ be the UNDIRECTED HAMILTONIAN CYCLE instance and (V', E', w', k') the TRAVELING SALESMAN PROBLEM instance. Every vertex $v \in V$ is mapped to itself $v \in V'$. Furthermore, we map each edge $e \in E$ to itself in E' and add additional edges to form a complete graph. The weight function $w' : E' \rightarrow \mathbb{Z}$ is defined for all $e' \in E'$ as

$$w(e') = \begin{cases} 0, & \text{if } e' \in E \\ 1, & \text{if } e' \notin E \end{cases}$$

At last, we set $k' = 0$ resulting that only the edges from E are usable. Thus, we preserve the one-to-one correspondence between the edges with $f_I(e) = e$. Consequently, this is an SSP reduction.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . Besides the additional edges $\{\{v, w\} \mid \{v, w\} \notin E\}$ of weight 1, all elements in the TRAVELING SALESMAN PROBLEM instance universe originate from a universe element of the UNDIRECTED HAMILTONIAN PATH instance. An edge $\{v, w\} \in E$ is in the solution UNDIRECTED HAMILTONIAN PATH if and only if the corresponding edge $\{v, w\} \in E'$ is in the TRAVELING SALESMAN PROBLEM solution. All edges $\{\{v, w\} \mid \{v, w\} \notin E\}$ are never in a solution, otherwise the weight threshold would be violated. Then, we have $f(e) = e$, $U_{on} = \emptyset$ and $U_{off} = \{\{v, w\} \mid \{v, w\} \notin E\}$. Thus this SSP reduction is also blow-up preserving.

DIRECTED TWO VERTEX DISJOINT PATH

Instances: Directed graph $G = (V, A)$, $s_i, t_i \in V$ for $i \in \{1, 2\}$.

Universe: Arc set $A =: \mathcal{U}$.

Solution set: The set of all sets set $A' \subseteq A$ such that $A' = A(P_1) \cup A(P_2)$, where P_1 and P_2 are some vertex-disjoint paths s.t. P_i goes from s_i to t_i for $i \in \{1, 2\}$.

The reduction by Fortune, Hopcroft and Wyllie [FHW80] is an SSP reduction. The reduction makes extensive use of a switch gadget, which is depicted in Figure 9.11. The gadget has four input arcs, B, C, W and Y , and four output arcs, A, D, X and Z . The idea of this switch gadget

is that if you have two disjoint paths going through the gadget entering at B and C , then the path entering at B must leave at D and the one entering at C must leave at A , and additionally either a path from W to X exists or a path from Y to Z exists. We can then use the first of the two paths to run first through the switches and then to the rest of the construction and the second path to run through the switches as in visualized in Figure 9.13. In doing so, the second path running only through the switches controls that the first path running through the construction is only able to satisfy the clauses according to the assignment of the variables.

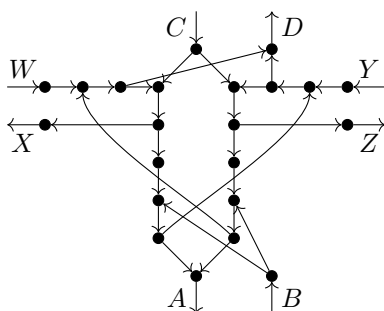


Figure 9.11: The switch gadget.

Let $I = (L, C)$ be the 3SAT instance and $(V, A, s_1, t_1, s_2, t_2)$ be the DIRECTED TWO DISJOINT PATH instance. First, we introduce four vertices s_1, t_1, s_2, t_2 representing the start and ends of the two disjoint paths. For every literal $\ell \in L$, we create a path $\ell^1, \dots, \ell^{4|C|}$ of $4|C|$ vertices. For every literal pair $\ell_i, \bar{\ell}_i$ (or variable x_i), we connect the paths by introducing two additional vertices x_i^s and x_i^t with arcs $(x_i^s, \ell_i^1), (x_i^s, \bar{\ell}_i^1)$ and $(\ell_i^{4|C|}, x_i^t), (\bar{\ell}_i^{4|C|}, x_i^t)$. We connect the literal gadgets by adding the arcs (x_i^t, x_{i+1}^s) for all $i \in \{1, \dots, |L|/2 - 1\}$. For each clause $c_j \in C$, we add two vertices c_j^1 and c_j^2 and connect them by three arcs. We connect these clause vertices with the arcs (c_j^2, t_1) as well as (c_j^2, c_{j+1}^1) for $j \in \{1, \dots, |C| - 1\}$. At last, we connect the literal paths with the clause path with an arc $(x_{|L|/2}^t, c_1^1)$.

Now, we have everything to introduce the switches into the construction. We stack the switches one after another by merging the C and D input arcs and the A and B input arcs, respectively. Then, we connect s_2 to the input arc C of the last switch in the stack and t_2 to the output arc A of the first switch of the complete switch stack. We do this analogously for s_1 , which we connect to the input arc B of the first switch of the stack and the rest of the construction with the output arc D of the last switch of the stack. Thus both path run through the switch stack as described above. At last, we use the switches to check whether the 3SAT assignment is correct. For this, we use the schematic description of a switch as depicted in Figure 9.12.

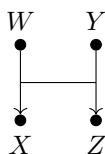


Figure 9.12: The schematic switch gadget.

That is, the arc $(\bar{\ell}_i^{4j-2}, \bar{\ell}_i^{4j-1})$ is connected to the arc (c_j^1, c_j^2) if and only if the corresponding literal $\ell_i \in L$ is in clause $c_j \in C$. More precisely, the arc $(\bar{\ell}_i^{4j-2}, \bar{\ell}_i^{4j-1})$ is substituted

by using the input arc W from $\bar{\ell}_i^{A_j-2}$ and output arc X to $\bar{\ell}_i^{A_j-1}$ and for the clause vertices c_j^1 is incident to input arc Y and output arc Z is incident to c_j^2 . Because only one path either from W to X or from Y to Z is usable, the path has to go from s_1 through the switch stack, then over the literal paths of the literals that are in the solution and at last over the clause vertices to t_1 . If for a clause there is no literal satisfying it, the path in the switch is blocked. The full construction can be found in Figure 9.13.

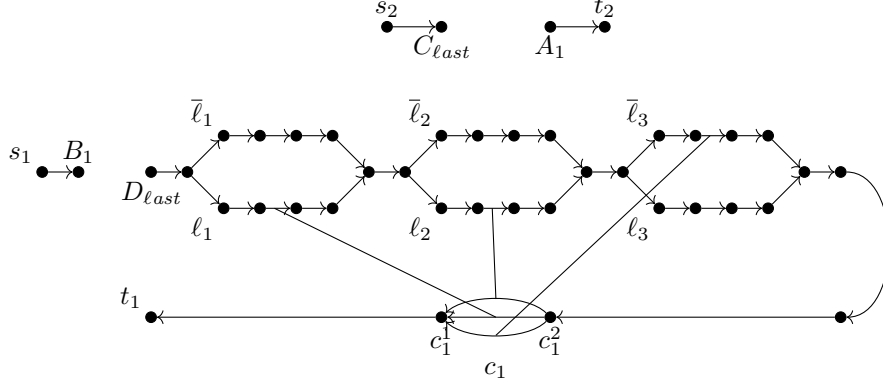


Figure 9.13: Classical reduction of 3SAT to DIRECTED TWO DISJOINT PATH for $\varphi = (\bar{\ell}_1 \wedge \bar{\ell}_2 \wedge \ell_3)$.

There is a one-to-one correspondence between the literals and the arcs of the path from s_1 to t_1 . We can define $f_I(\ell_i) = (x_i^s, \ell_i^1)$ because the path over (x_i^s, ℓ_i^1) is taken if and only if ℓ_i is in the 3SAT solution.

Blow-up SSP reduction. The blow-up gadget for this reduction works analogously as the one in the DIRECTED HAMILTONIAN CYCLE. We introduce β_I additional vertices (and therefore arcs) to each path corresponding to a literal $\ell \in L$. Thus, the functionality of the gadgets is not impaired and at least $\beta_I + 1$ many arcs have to be exchanged to achieve a reassignment of literal ℓ to $\bar{\ell}$. An example can be found in Figure 9.14. For an analysis of the β_I , we need to closely consider both paths from s_1 to t_1 and s_2 to t_2 as well as the switch gadgets. Again all literals in $L \setminus L_b$ might be chosen incorrectly such that Alice needs to be able to recover from this. It is easy to see that the path corresponding to that literal need to be changed, these are $4|C|$ arcs. Additionally, the switch gadgets need to be run through differently, which are up to $5|C|$ arcs for each literal. Furthermore, the clauses may need to be passed over different arcs, which are actually 5 arcs in the switch gadgets. These are again up to $5|C|$ arcs. Now, we need to consider the switch gadgets in the path from s_2 to t_2 and in the first half of the path from s_1 to t_1 . If a literal needs to be changed or a clause needs to be passed on different way (the path from W to X is exchanged with Y to Z), the switch gadgets need to be run through in a different way. Thus, the path from B to D and the path from A to C needs to be mirrored and thus exchanged completely. For every literal from $L \setminus L_b$, this might happen $|C|$ times and 12 arcs need to be exchanged. On the other hand for every clause $c_j \in C$, this might happen two times that 12 arcs need to be exchanged. Overall, we get the following β_I :

- ▶ κ -addition: $\beta_I = (12 + 5)|C| + (12|C| + 5|C| + 4|C|) \cdot |L \setminus L_b|/2$
- ▶ κ -deletion: $\beta_I = 17|C| + 21|C| \cdot |L \setminus L_b|/2$
- ▶ Hamming distance: $\beta_I = 34|C| + 42|C| \cdot |L \setminus L_b|/2$.

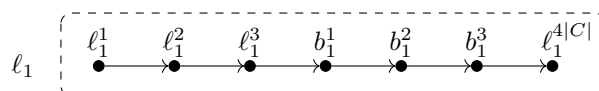


Figure 9.14: Blow-up gadget for the reduction of 3SAT to DIRECTED TWO DISJOINT PATH with blow up $\beta_I = 3$.

DIRECTED k -VERTEX DISJOINT PATH

Instances: Directed graph $G = (V, A)$, $s_i, t_i \in V$ for $i \in \{1, \dots, k\}$.

Universe: Arc set $A =: \mathcal{U}$.

Solution set: The sets of all sets $A' \subseteq A$ such that $A' = \bigcup_{i=1}^k A(P_i)$, where all P_i are pairwise vertex-disjoint paths from s_i to t_i for $1 \leq i \leq k$.

The following reduction from DIRECTED TWO VERTEX DISJOINT PATH to DIRECTED k -VERTEX DISJOINT PATH is an easy-to-see SSP reduction. We introduce $k - 2$ additional vertex pairs s_i, t_i for $i \in \{3, \dots, k\}$, which we connect by adding arcs (s_i, t_i) for all $i \in \{3, \dots, k\}$. Thus, the original DIRECTED TWO VERTEX DISJOINT PATH reduction still works for itself, while we added the necessary additional paths. The SSP properties of the DIRECTED TWO VERTEX DISJOINT PATH reduction are obviously not compromised.

Blow-up preserving SSP reduction. For a blow-up preserving SSP reduction, we have to partition a solution into the sets $f_I(\mathcal{U}(I))$, U_{off} , and U_{on} . Besides the arcs from $\{(s_i, t_i) \mid i \in \{3, \dots, k\}\}$, all elements in the DIRECTED k -VERTEX DISJOINT PATH instance universe originate from a universe element of the DIRECTED TWO VERTEX DISJOINT PATH instance. An arc $a \in A$ is in the solution DIRECTED TWO VERTEX DISJOINT PATH if and only if the corresponding arc a is in the DIRECTED k -VERTEX DISJOINT PATH solution. The additional arcs $\{(s_i, t_i) \mid i \in \{3, \dots, k\}\}$ are always part of a solution. Then, we have $f(a) = a$ and $U_{on} = \{(s_i, t_i) \mid i \in \{3, \dots, k\}\}$, and $U_{off} = \emptyset$. Thus this SSP reduction is also blow-up preserving.

STEINER TREE

Instances: Undirected graph $G = (SUT, E)$, set of Steiner vertices S , set of terminal vertices T , edge weights $c : E \rightarrow \mathbb{N}$, number $k \in \mathbb{N}$.

Universe: Edge set $E =: \mathcal{U}$.

Feasible solution set: The set of all sets $E' \subseteq E$ such that E' is a tree connecting all terminal vertices from T .

Solution set: The set of feasible solutions E' with $\sum_{e' \in E'} c(e') \leq k$.

There is a folklore reduction from 3SAT to STEINER TREE, which is an SSP reduction, and which is depicted in Figure 9.15. First, there are designated terminal vertices s and t . For every literal $\ell \in L$, there is a Steiner vertex ℓ . Additionally for every literal pair $(\ell_i, \bar{\ell}_i)$, $1 \leq i \leq |L|/2 - 1$, we add a Steiner vertex v_i . We define $v_0 := s$. Then all of the above vertices are connected into a “diamond chain”, where we begin with s connected to both ℓ_1 and $\bar{\ell}_1$. Both vertices ℓ_1 and $\bar{\ell}_1$ are connected to v_1 . This vertex v_1 is then connected to vertices ℓ_2 and $\bar{\ell}_2$ and so on. At last, $\ell_{|L|}$ and $\bar{\ell}_{|L|}$ are connected to t .

Furthermore, for every clause $c_j \in C$, we add a corresponding terminal vertex c_j . The vertex c_j is then connected its corresponding literals $\ell \in c_j$ via a path of Steiner vertices of length $|L| + 1$. The costs of every edge is set to 1 and the threshold is set to $k = |L| + |C| \cdot (|L| + 1)$.

For the correctness, observe that every solution of Steiner tree includes a path from s to t over the literal vertices because all paths over a clause vertex are longer than $|L|$. This path

encodes the set of literals included in a corresponding 3SAT solution, where a positive literal ℓ_i for $1 \leq i \leq |L|/2$ is included in the 3SAT solution if and only if the edge $\{v_{i-1}, \ell_i\}$ is in the Steiner tree solution. The analogous statement holds for negative literals. We therefore define the embedding function f_I in the above fashion, i.e. for all $\ell \in L$ we have $f_I(\ell) = \{v_{i-1}, \ell\}$.

Next, for every clause c , the path from literal ℓ to terminal vertex c for one $\ell \in c$ is included in the solution as well. Thus, $|C|$ paths of length $|L| + 1$ are included. If a clause c_j is not satisfied, then a path of length of at least $|L| + 2$ is needed to include the terminal vertex c_j , which violates the threshold. Thus, the reduction is correct.

The SSP property holds, because every correct 3SAT solution can be translated with the function f_I and be completed to a Steiner tree with at most k edges. On the other hand, every Steiner tree with at most k edges restricted to the set $f_I(L)$ encodes a 3SAT solution.

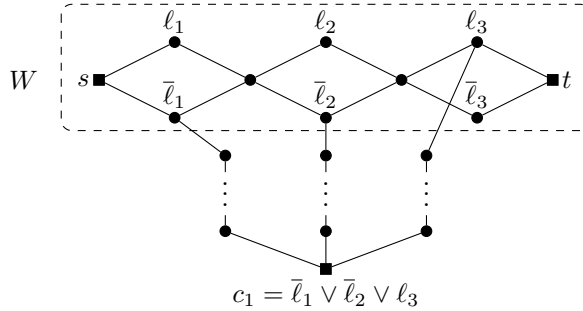


Figure 9.15: Classical reduction of 3SAT to STEINER TREE.

Blow-up SSP reduction. For the blow-up gadget, we add β_I new vertices ℓ^j , $1 \leq j \leq \beta_I$, for each literal vertex ℓ , $\ell \in L_b$ to the graph. Moreover, we connect all of these vertices ℓ^j to the existing ℓ with the edges $\{\{\ell, \ell^j\} : 1 \leq j \leq \beta_I\}$. At last, we add an edge between each pair of ℓ^j and $\bar{\ell}^j$, for $1 \leq j \leq \beta_I$, which is subdivided by one terminal vertex $t_{\ell, \bar{\ell}}^j$. The construction is depicted in Figure 9.16. The threshold for the Steiner tree is increased by $2\beta_I$ for each pair $\ell, \bar{\ell} \in L_b$ of blow-up literals. This blow-up gadget is correct because if vertex ℓ is connected to the Steiner tree, we are able to include the edges $\{\{\ell, \ell^j\}, \{\ell^j, t_{\ell, \bar{\ell}}^j\} : 1 \leq j \leq \beta_I\}$, which needs $2\beta_I$ additional edges and thus lies within the threshold. On the other hand, if an edge from $\{\{\bar{\ell}, \bar{\ell}^j\}, \{\bar{\ell}^j, t_{\ell, \bar{\ell}}^j\} : 1 \leq j \leq \beta_I\}$ is used this edge has to be connected to the Steiner tree. This is only doable by connecting $\bar{\ell}$ to the Steiner tree, which introduces additional cost of one or by connecting it over one of the terminal vertices $t_{\ell, \bar{\ell}}^j$. However in the last case, this implies that $t_{\ell, \bar{\ell}}^j$ is already connected to the Steiner tree. Thus, it is an unnecessary edge that does not connect any terminal vertex of the Steiner tree and additional costs of one are introduced. Since each solution is an optimal Steiner tree, this is a contradiction. Because the blow-up gadget does not interfere with the functionality of the original reduction, this blow-up gadget is correct.

For computing the β_I for the three distance measures, we again consider the wrongly chosen literals $L \setminus L_b$ from which Alice has to recover. A wrongly chosen literal induces that the two edges $\{v_{i-1}, \ell\}$ and $\{\ell, v_i\}$ have to be exchanged by the edges $\{v_{i-1}, \bar{\ell}\}$ and $\{\bar{\ell}, v_i\}$. This literal also may induce $|C| \cdot (|L| + 1)$ additional edges because clauses might be connected to it. Thus, we obtain the following β_I :

- ▶ κ -addition: $\beta_I = |C| \cdot (|L| + 1) + 2|L \setminus L_b|$
- ▶ κ -deletion: $\beta_I = |C| \cdot (|L| + 1) + 2|L \setminus L_b|$

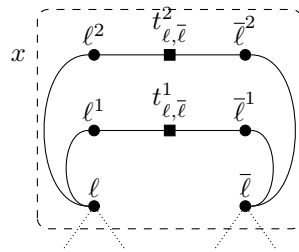


Figure 9.16: Blow-up gadget for the reduction of 3SAT to STEINER TREE with blow up $\beta = 2$.

► Hamming distance: $\beta_I = 2|C| \cdot (|L| + 1) + 4|L \setminus L_b|$.

Chapter 10

Conclusion

In this part, we have shown that for a large number of NP-complete problems, their min-max variant (min-max-min variant, respectively) is automatically Σ_2^P -complete (Σ_3^P -complete, respectively). We first showcased this behavior in the areas of network interdiction, min-max regret robust optimization, and two-stage adjustable robust optimization. In these three areas, we are able to derive completeness results if the problems are SSP-NP-complete. Moreover, we are able to extend the SSP framework to work with recoverable robust optimization and more general forms of interdiction.

In the context of recoverable robustness, we have shown that the recoverable robust version for a large class of NP-complete problems is Σ_3^P -complete for several different distance measures. For this, we have introduced two new types of reductions: blow-up SSP reductions and blow-up preserving SSP reductions. Blow-up SSP reductions are the basis to show Σ_3^P -completeness of a recoverable robust problem by a reduction from 3SAT, and further problems can be appended to the reduction chain by using the transitive blow-up preserving SSP reductions. In particular, we are able to show that 24 recoverable robust problems are Σ_3^P -complete with the ability to apply the framework to further problems.

For minimum cardinality interdiction, which is the most natural variant of interdiction, most vital vertex, and blocker problems, we have shown that for a large class of NP-complete problems, the corresponding minimum cardinality interdiction problem is Σ_2^P -complete. With that, we have also shown the hardness of several different variants of interdiction that can be found in the literature, including minimum blocker and most vital element problems. For this, we have introduced a new type of reduction, namely invulnerability reductions. This reduction uses the corresponding minimum cost interdiction problem as a basis and ensures that non-blockable elements are effectively not attackable. The hardness of the minimum cost interdiction problem is provable via an SSP reduction. Additionally, we have shown that for some problems (e.g., vertex cover, satisfiability), the Σ_2^P -completeness cannot be derived despite the fact that the minimum cost interdiction problem is Σ_2^P -complete. Overall, we have shown for 23 minimum cardinality interdiction problems that they are either Σ_2^P -complete or coNP-complete, with the ability to apply the framework to further problems.

Our findings constitute a leap in the understanding of the basic behavior of such problems. However, many questions still remain unanswered. First, it is of interest to find more problems for which the SSP framework and its derivatives are applicable.

Second, we would like to understand if our theorem can also be adapted to work with other popular areas of robust or multi-level optimization, for example, to Stackelberg games, to attacker-defender games, or to the area of computational social choice. Secondly, we have restricted our attention in this article only to such problems which can be expressed as finding a

certain subset with a nice property. It would be insightful to understand if similar meta-theorems can be made about natural variants of problems that look for a nice partition, a nice assignment function, a nice permutation, etc. For space reasons, we have restricted our focus in this part mainly to multi-stage problems with two or three stages. It seems natural to extend our arguments to multi-stage problems with an arbitrary number of stages.

Since all three main results of our paper have been proven in an essentially analogous way, it seems intriguing to consider a potential meta-meta-theorem. Which properties does a min-max *modification scheme* (such as the interdiction-modification, the regret-modification, the two-stage modification) need to possess, such that a similar meta-theorem applies?

Finally, our framework only applies to nominal problems, which are NP-complete in the first place. However, researchers in the area often are interested in robust variants of nominal problems in P. Our framework cannot say anything about these problems – for NP-complete problems, a vast amount of existing completeness reductions between them exist, which are upgraded by our meta-theorem. Is there some notion of reduction between problems in P, which supports our framework (i.e., for problems admitting such reductions, their robust variant is automatically NP-hard)?

Part II

Universe Gadget Reduction Framework

Chapter 11

Universe Gadget Reduction Framework

11.1 Introduction

We begin this part with a short view back to the past. The idea from the beginning of this doctoral project was to establish a meta-theorem on the complexity of problems in the realm of optimization under uncertainty. We began to analyze recoverable robust problems with elemental uncertainty. For this, it was necessary to define recoverable robust problems by introducing a general definition that captures most, if not all, problems existing in this regime.

Besides a general definition of recoverable robust problems, one had to define a general notion of reductions that is also directly applicable to the general definition of a recoverable robust problem. For this, gadget reductions seemed to be a suitable concept since they are prevalent all over the reduction landscape due to their divide-and-conquer paradigm.

At first, gadget reductions are just an informal concept to derive reductions for various problems. To explain the idea shortly, consider two problems Π and Π' . A gadget is a specialized device that has the task of simulating a particular element from problem Π in the instance of problem Π' . From a different point of view, a gadget is a subinstance of Π' , and the resulting reduction instance of Π' based on the instance of problem Π is the union of all its gadgets. If a problem can be divided into many small subinstances, we are able to argue easily why such subinstances can be efficiently computed and also why such a subinstance behaves like the element it originates from. In conclusion, it is easy to reason that the whole reduction instance that is built out of these gadgets is correct and is easy to compute.

Indeed, when we considered recoverable robust problems with elemental uncertainty, this concept seemed to be a good choice: If an element in problem Π' is not part of a scenario, we can just eliminate the gadget from the corresponding scenario in problem Π' . Of course, this “deletion operation” of a gadget had to be supported by the reduction. In the end, this idea worked out for certain more peculiar definitions of uncertainty scenarios: *xor*-dependencies and Γ -set scenarios.

Furthermore, we used a version of these gadget reductions to show PSPACE-completeness of online problems that try to find a vertex subset in the neighborhood reveal model while the online decision maker possesses an isomorphic copy of the underlying graph (which we also call unlabeled map). In both recoverable robustness and online vertex subset problems, the instance is not completely known to the decision maker when the first decisions have to be made, since the adversary controls which of the uncertainty scenarios is chosen. The key difference in

online vertex subset problems is that the adversary and the decision maker have to play multiple alternating moves instead of three alternating moves as in recoverable robustness.

However, there was still a long way to go to reach the goal of capturing many popular problems under different popular forms of uncertainty since we did not succeed in applying it directly, e.g., to the concept of budgeted uncertainty. The trend was clear: the whole idea needed to be simplified. Consequently, a distilled version of these gadget reductions was developed to eliminate many of the unnecessary assumptions, which in the end led to the notion of SSP problems and reductions.

Nevertheless, the following results deserve to be presented in this thesis not only because of their documentational value but also since they have their own merits. On the one hand, we are able to showcase a reduction framework for elemental uncertainty in robust optimization. This form of uncertainty is frequently studied in graph theoretical environments [LY80, BCT24]. A challenging aspect of problems involving elemental uncertainty is that the underlying instance changes, which makes it more complicated to model these problems. Often, elemental uncertainties can be expressed by cost uncertainties, for example, in the problem clique interdiction [Rut93]; however, this is not generally the case. For example, in vertex cover interdiction, eliminating a vertex is not equivalent to increasing the costs of a vertex since eliminating a vertex (and all its incident edges) is equivalent to taking the vertex into the solution (see also Chapter 8). Therefore, it remains an important open question how elemental uncertainty can be managed from a complexity theoretical point of view. In conclusion, the work on recoverable robustness with elemental uncertainty indicates that such a framework might indeed exist, even though more work has to be put into understanding the structure of different forms of elemental uncertainty. On the other hand, we show that reduction frameworks for online problems behave similarly to robust optimization problems. This indicates that we are able to transfer the results from the SSP framework to online optimization problems with different forms of cost uncertainty and maybe also to different forms of elemental uncertainty.

Related Work. This underlying idea of a gadget reduction was studied in the following two papers in different contexts. Agrawal et al. [AAI⁺01] define AC^0 -computable gadget reductions for NP-completeness, mapping one bit of the input of one problem to a bounded number of bits in the other problem. A further form of gadget reduction was introduced by Trevisan et al. [TSSW96]. They formalize gadgets with constraint families to compute optimal gadgets via linear programming for gap-preserving reductions.

11.1.1 Universe Gadget Reductions

The goal of this part is to establish a formal definition of gadget reductions that can be applied to combinatorial problems in order to derive general complexity results for the corresponding versions that include uncertainty. The name stems from the universe U , which is the ground set of the combinatorial problem (compare LOP problems from Chapter 1).

Since each gadget should simulate a part of the instance of problem A in the instance of problem B , and the union of all gadgets should be a correct instance for B , the universe in itself does not contain enough information to compute a correct transformation. Therefore, we need to take more information into account. As stated in Chapter 1, an LOP problem consists of instances $I \in \mathcal{I}$, each consisting of a universe $\mathcal{U}(I)$ together with further information. These universe elements are the building blocks of the solutions $\mathcal{F}(I)$, which are mere subsets of the universe $\mathcal{F}(I) \subseteq \mathcal{U}(I)$. Therefore, we call it *solution universe* or *solution ground set*. In contrast to LOP problems, we now assume that we have a universe that contains the building blocks of the instance and additional information in the form of (nested) relations on the universe elements. Accordingly, we call it *instance universe* or, in the following, simply *universe*. Indeed,

in combinatorial optimization, this is a typical behavior, and many problems can be modeled this way. Let us take a look at the following prototypical problem.

3SATISFIABILITY

Instances: Literal set $L = \{\ell_1, \dots, \ell_n\} \cup \{\bar{\ell}_1, \dots, \bar{\ell}_n\}$, Clauses $C \subseteq L^3$.

Instance universe: Literals L .

Solution universe: Literals L .

Solution set: The set of all sets $L' \subseteq L$ such that for all $i \in \{1, \dots, n\}$ we have $|L' \cap \{\ell_i, \bar{\ell}_i\}| = 1$, and such that $|L' \cap c_j| \geq 1$ for all $c_j \in C$, $j \in \{1, \dots, |C|\}$.

In 3SATISFIABILITY, the atomic building blocks are the literals. We are able to model each instance by a universe together with relations over that universe as follows. A pair of literals ℓ and $\bar{\ell}$ builds up a variable and is thus in a variable relation. Further, we are able to model clauses by a relation over the literals $C \subseteq L^3$. We are also able to define a literal-clause relation defined by $\{(\ell, c) \mid \ell \in c\}$. And, we can also define a literal-literal-clause relation that contains pairs of literals that are part of the same clause, i.e. $\{(\ell, \ell', c) \mid \ell \in c \text{ and } \ell' \in c \text{ for } c \in C\}$.

Another example are graph problems. Underlying each graph is a vertex set, which can be put into a relation of edges. Moreover, we are able to define, for example, an incidence relation as a relation between vertices and edges. In consequence, we assume that an instance consists of an instance universe U , which contains the building blocks of the instance. The additional information of each instance I is given as nested relations over U that define the instance. In order to define the corresponding solutions, we assume that the solutions to the problems can be defined by subsets of one of the relations (or the solution universe). The instance universe and the solution universe may coincide. For example, this is the case for the problem VERTEX COVER, which we can define in the following way.

VERTEX COVER

Instances: Vertex set V , edge relation $E \subseteq V \times V$, number $k \in \mathbb{N}$.

Instance universe: Vertex set V .

Solution universe: Vertex set V .

Solution set: The set of all vertex covers $V' \subseteq V$ of G of size at most k .

Consequently, the instance universe is the vertex set V of the corresponding graph. However, the vertex set V is also the solution universe because each vertex cover can be defined by a subset of vertices. On the other hand, the instance universe of the problem HAMILTONIAN CYCLE differs from the solution universe.

HAMILTONIAN CYCLE

Instances: Vertex set V , edge relation $E \subseteq V \times V$.

Instance universe: Vertex set V .

Solution universe: Edge set E .

Solution set: The set of all Hamiltonian cycles $T \subseteq E$.

While HAMILTONIAN CYCLE is again a graph problem and thus the instance universe is the vertex set, the solution universe is defined over the edge relation, specifically each Hamiltonian cycle is an edge subset.

11.1.2 Technical Overview

As described above, we want to express a problem by an (instance) universe, which contains the atomic building blocks, (nested) relations over the universe, and the solutions, which are subsets

of one of the relations. Let U be the instance universe. Then, we only consider relations from $\mathcal{R}(U)$ over the set U defined by

$$\begin{aligned} U &\in \mathcal{R}(U) \\ A &\in \mathcal{R}(U), & \text{if } A \subseteq B \text{ for some } B \in \mathcal{R}(U) \\ \bigtimes_i A_i &\in \mathcal{R}(U), & \text{if for all } i, A_i \in \mathcal{R}(U). \end{aligned}$$

We can now define a combinatorial decision problem as a tuple (U, R, F) , where U is the universe, R are the relations, and F are the solutions. We express the tuple (U, R, F) as a decision problem by defining the input as R and the question is whether there is a solution, i.e., $F \neq \emptyset$.

How does a gadget reduction from 3SATISFIABILITY to VERTEX COVER work? As an example, we use the reduction by Garey and Johnson [GJ79]. The idea is to introduce a vertex v_ℓ for each of the literals $\ell \in L$. For each pair of a positive and negative literal $\ell, \bar{\ell}$ corresponding to a variable, an edge $\{\ell, \bar{\ell}\}$ is introduced. This forms a gadget for the variable corresponding to literals $\ell, \bar{\ell}$. In order to cover the edge, one of the vertices v_ℓ and $v_{\bar{\ell}}$ needs to be taken into the solution. A clause c is simulated by a 3-clique of vertices $\{v_{\ell,c} \mid \ell \in c\}$. At last, we connect the 3-cliques with the variable gadgets if the literal is in the clause by introducing the edges $\{v_\ell, v_{\ell,c} \mid \ell \in c\}$. The threshold k for the VERTEX COVER instance is set to $k := |L|/2 + 2|C|$. A sample instance of this reduction for clauses $C = \{\{\ell_1, \ell_2, \ell_3\}, \{\bar{\ell}_1, \bar{\ell}_2, \ell_3\}\}$ can be found in Figure 11.1.

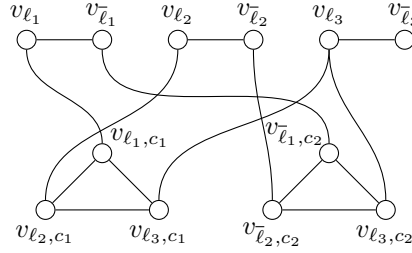


Figure 11.1: The reduction graph for 3SATISFIABILITY formula $C = \{\{\ell_1, \ell_2, \ell_3\}, \{\bar{\ell}_1, \bar{\ell}_2, \ell_3\}\}$.

It is now possible to divide the instance into gadgets. We present the different gadgets in Figure 11.2, where dashed elements are part of a different gadget. First of all, we have a vertex v_ℓ for each literal $\ell \in L$. This is the literal gadget, which can be found in Figure 11.2a. For each of the elements in the relation between a positive and negative literal that form a variable, we have introduced an edge $\{v_\ell, v_{\bar{\ell}}\}$ as in Figure 11.2b. Furthermore, we have introduced the 3-clique for each of the clauses. The vertices of the 3-clique $v_{\ell,c}$ together with the edge connecting the vertices v_ℓ and $v_{\ell,c}$ are induced by the literal-clause relation $\{(\ell, c) \mid \ell \in c\}$, see Figure 11.2c. The edges of the 3-clique are generated by the elements of the literal-literal-clause relation $\{(\ell, \ell', c) \mid \ell \in c \text{ and } \ell' \in c \text{ for } c \in C\}$, see Figure 11.2d. Thus, we have disjoint gadgets that together build the reduction instance for VERTEX COVER.

We now have everything to define a first version of universe gadget reductions. Let (U_A, R_A, F_A) and (U_B, R_B, F_B) be a combinatorial decision problems and let I_A (respectively I_B) be an index set for R_A (respectively R_B). A Universe Gadget Reduction $f_{\underline{A}}$ is composed of mappings:

$$f_{R_A, R_B}^i : R_A^i \rightarrow 2^{R_B^j} \text{ for all } (i, j) \in I_A \times I_B.$$

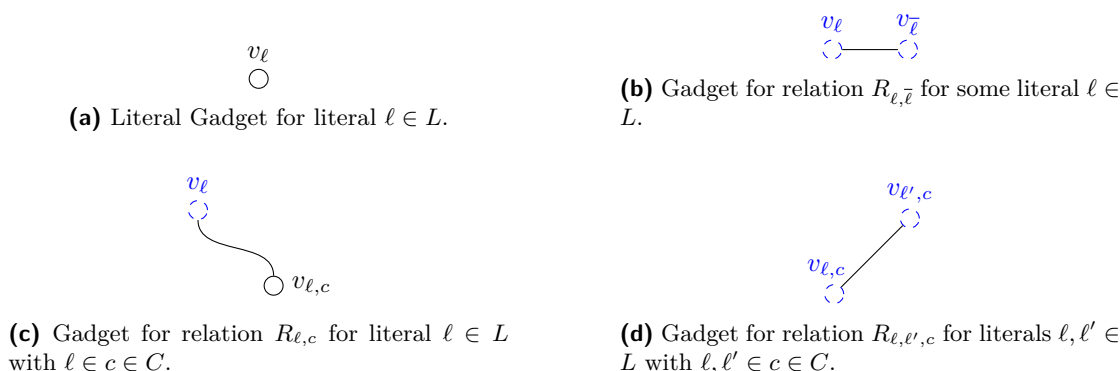


Figure 11.2: The gadgets for the universe and all relations for the reduction from 3SATISFIABILITY to VERTEX COVER.

We, then, call the substructure

$$Y_x = \bigcup_{(i,j) \in I_A \times I_B} f_{R_A^i, R_B^j}(x)$$

the gadget for the specific universe element or relation element $x \in \bigcup_i R_A^i$. We further demand that the gadgets are pre-image unique, i.e., there are no elements that belong to more than one gadget: Let $y \in R_B^j$ for some $j \in I_B$, then there is exactly one $(i, j) \in I_A \times I_B$ and exactly one $x \in R_A^i$ such that $y \in f_{R_A^i, R_B^j}(x)$. This definition of a gadget reduction for combinatorial decision problems ensures that the gadgets are uniquely relatable to the generating combinatorial elements.

We consider only gadget reductions that start at one of the first problems that were shown to be NP-complete: 3SATISFIABILITY. Its logical substructure enables relatively easy gadget constructions for a vast number of combinatorial problems. The building blocks of 3SATISFIABILITY or, in general, Boolean circuits are the variables and the logical operators. 3SATISFIABILITY has the variables at its core and three logical operators that have to be simulated: \neg , \vee , and \wedge . Accordingly, one can split the instance into literals to simulate both states of a variable (*true* and *false*) and into the logical operators. If we additionally assume that the 3SATISFIABILITY is in CNF, we can also construct gadgets for the clauses instead of gadgets for each of the logical operators. We thus have *variable gadgets* for simulating a variable and *clause gadgets* for simulating the relation between the variables induced by the logical operators. The simulation of a variable gadget is typically performed by a one-to-one correspondence between the variable assignment and the solutions on the variable gadget. That is, there is one solution on the variable gadget that corresponds to the assignment true and one solution that corresponds to the assignment false. Moreover, a clause is simulated to be satisfied if a solution on the clause gadget exists, depending on the solution on the variable gadgets.

We extend the definition of a gadget reduction from above to accommodate recoverable robust problems and online graph problems with a map. Both areas need different extensions because of their underlying different nature. However, the basic definition of a gadget reduction is at the heart of the corresponding complexity results. We proceed with the technical overview of both subfields.

Recoverable Robustness. For recoverable robust problems with elemental uncertainty, we introduce an additional property to universe gadget reductions, which we call modularity. To

explain the concept of modularity, we assume to have two problems $P_A = (U_A, R_A, F_A)$ and $P_B = (U_B, R_B, F_B)$ such that P_A is universe gadget reducible to P_B . Then modularity demands that if we remove an element $x \in R_A$ from the instance of P_A , we are able to remove the corresponding gadget Y_x and obtain the corresponding instance of problem P_B . Additionally, we have to take the distance between the solutions of the recoverable robust problem into account. For this, we demand that the solution size is altered accordingly in an efficiently computable way. Concretely, we assume that each gadget induces a constant solution size. Hence, we only have to count the number of the corresponding gadgets, and we obtain an efficiently computable solution size for the instance. With this definition of an extended universe gadget reduction, we are able to show that recoverable robust problems with so-called *xor*-dependencies or Γ -set scenarios are Σ_3^p -complete if the corresponding nominal problem is NP-complete.

The key idea for a general reduction is to show that there is a reduction from $\exists\forall\exists$ SATISFIABILITY to the recoverable robust version of 3SATISFIABILITY with the given type of uncertainty. Then, we can use the universe gadget reduction from 3SATISFIABILITY to a problem P_B to substitute each variable by its variable gadget and each clause by its clause gadget to derive a reduction from $\exists\forall\exists$ SATISFIABILITY to the recoverable robust version of P_B . At last, the solution size function transfers the chosen Hamming distance of the reduction from $\exists\forall\exists$ SATISFIABILITY to the recoverable robust version of 3SATISFIABILITY to the correct Hamming distance for the reduction to the recoverable robust version of P_B .

Online graph games with a map. In online graph games with a map, we are given a graph G and a threshold $k \in \mathbb{N}$ on the size of the vertex subset. The question is whether there is a vertex subset of size k in the graph G . This description includes popular problems such as vertex cover, independent set, and dominating set. Here, we use the basic universe gadget reduction definition and additionally include so-called *self-contained extension gadgets*. The basic function of the extension gadgets is to add neighbors to all vertices such that the degree of each vertex can be partitioned into equivalence classes. Furthermore, the gadgets need to be self-contained, i.e., the gadgets do not influence the solution on the rest of the graph. Therefore, we are able to add these gadgets to the graph without altering the underlying solution and thus the semantics of the existing gadget reduction. Based on these equivalence classes and the provided map, the online decision maker is either able to recognize the vertices or they seem isomorphic, such that he is not able to distinguish them. The resulting graph then simulates a game of TRUTH QUANTIFIED BOOLEAN FORMULA between the online decision maker (\exists -player) and the adversary (\forall -player). The online decision maker can associate each \exists -quantified variable with a vertex, choosing whether to include it in the solution at his discretion, and the adversary is able to force the online decision maker to assign the \forall -quantified variables to his preferences. Accordingly, we are able to show that online graph games with a map are PSPACE-complete.

Chapter 12

Recoverable Robustness

12.1 Introduction

The concept of *robustness* in the field of optimization problems comprises a collection of models that consider uncertainties in the input. These uncertainties may for example arise from faulty or inaccurate sensors or from a lack of knowledge. Robustness measures can model these types of uncertainty that occur in practical optimization instances into an *uncertainty set*. The goal is to find solutions that are stable over all possible *scenarios* in the uncertainty set. That is, these solutions remain good but not necessarily optimal regardless what the uncertainties turn out to be in reality.

One specific robustness concept is *recoverable robustness*, which is a recently introduced concept [LLMS09] by Liebchen et al. The input of a recoverable robust version of a problem P is a *base scenario* σ_0 , which is an instance of problem P , as well as a set of *uncertainty scenarios* S , whose members are again instances of P . The set of uncertainty scenarios S is the uncertainty set of the problem. We are asked to compute a base solution \mathbf{s}_0 to the base scenario σ_0 and to compute recovery solutions \mathbf{s} to all members of the uncertainty scenarios $\sigma \in S$ such that \mathbf{s}_0 and \mathbf{s} are not too far away from each other according to a distance measure. The solution on the base scenario does not directly include the uncertainties but needs to include the potential to adapt the base solution \mathbf{s}_0 to solutions \mathbf{s} within the given distance between the solutions. Thus, the base solution \mathbf{s}_0 may be restricted by these possibly harmful scenarios.

From a worst-case-analysis point of view, we assume that the uncertainty scenarios are chosen by an adversary. The algorithm computes a base solution with the potential to adapt to all scenarios. Then, the adversary chooses the most harmful scenario based on the base solution. Finally, the algorithm computes a recovery solution to adapt to the chosen scenario.

Related Work. Recoverable robustness is used in many practical settings such as different optimization areas in air transport [DSP19, FMW14, MDS14] or in railway optimization, for which a survey can be found in [LLB18]. Considered problems in railway optimization are to be found on all stages of railway operation, such as network design [TA18, CM12], rolling stock planning [CCG⁺08, CCG⁺12], shunting [CDS⁺09b] and timetabling [CDS⁺09c, CDS⁺09a, DSNP11, DSN09, GHM⁺13, Bös09]. Our focus lies on the complexity of recoverable robust problems. In parallel to writing the paper, on which this chapter is based on, Goerigk et al. [GLW24] analyzed the Hamming distance recoverable robust independent set, TSP and vertex cover. Hamming distance means that at most k elements may be added to or deleted from the base solution in total to obtain a recovery solution. They showed the Σ_3^P -hardness of the variant with discrete budgeted uncertainty over the costs of the elements. To the best of the author's knowledge, this is the only

contribution investigating the complexity within the polynomial hierarchy beyond NP-hardness. All other contributions study primarily algorithms and analyze the problems only on their NP-hardness or their approximability, where different distance measures between the solutions are of interest. The concept of *k-dist recoverable robustness*, allowing at most k new elements in recovery solutions, was introduced in [Büs12] but was also used in [HKZ17a]. Besides the k -dist measures, there are also measures which limit the number of deleted elements [BKK11a] or exchanged [CG16] elements. Furthermore, combinations of these distance measures are analyzed as well in the literature [BKK11b]. Further usages of Hamming distance recoverable robustness can be found in [DMP⁺15]. Among the studied recoverable robust problems is Knapsack, which is NP-hard for different distance measures between the solutions [BGKK19, BKK11a, BKK11b]. Recoverable robust versions of problems that are in PTIME are shown to be NP-complete as well such as Shortest Path, which is NP-hard for k -dist [Büs12], or Matching [DMP⁺15]. Furthermore, the recoverable robust Single Machine Scheduling problem is 2-approximable [BG22] and the recoverable robust TSP is 4-approximable [CG16]. Moreover, a recoverable robust version of Spanning Tree [HKZ17a] is shown to be in PTIME.

Contribution. We study Hamming distance recoverable robust problems with different forms of elemental uncertainty. That is, it is uncertain whether an element (e.g. a vertex or object) is included in a scenario or not. This form of uncertainty is different to cost uncertainty, where all elements are present in all scenarios but the costs of the elements are uncertain. We show that recoverable robust versions of typical NP-complete combinatorial problems with *xor*-dependencies or Γ -set scenarios are Σ_3^p -complete.

We do this by defining a gadget reduction framework, which uses a specific definition of combinatorial problems. These problems are defined over *combinatorial elements*, which are defined over a *universe* U , and *nested relations* $R(U)$ over that universe. We show that this framework is able to “upgrade” many already existing NP-hardness reductions by applying it to over 20 well-known problems. Thus, we expect that the results are easily extendable beyond those problems.

12.2 Combinatorial Problem Framework

In theoretical computer science, problems are defined as languages, which consist of all Yes-instances of the problem. The instances are encoded as words from $\{0, 1\}^*$. For combinatorial problems, we may assume that an instance contains a universe $U = \{1, \dots, n\}$, which consists of the encoding atoms of the instance. Furthermore, an instance includes (nested) relations between these atoms. To encode the relations, the atoms are used together with a delimiter symbol.

One example of such a problem is the problem UNDIRECTED S-T-CONNECTIVITY (USTCON). Its input is an undirected graph $G = (V, E)$ together with two vertices $s, t \in V$. The corresponding instance is then encoded by the vertices $V = U$ as universe and three relations $s, t \subseteq V$ and $E \subseteq V \times V$. The instance is a YES-instance iff there is path from s to t in G . Another example is the problem VERTEX COVER. Again, the vertices $V = U$ are the universe and $E \subseteq V \times V$ is a relation. The instance is a YES-instance iff there is a small vertex cover in G .

In mathematical optimization, a problem is often defined over its feasible solutions F together with a cost function c . The goal is then to find a solution that achieves the minimum (resp. maximum) costs of all feasible solutions. Oftentimes, an additional ground set of combinatorial elements X is given. For simplicity, the feasible solutions are then combinations of that ground set, that is $F(X) \subseteq 2^X$. We apply this to USTCON by interpreting the edges as the ground set $X = E$ and all paths $F(X) \subseteq 2^E$ from s to t as the feasible solutions. For VERTEX COVER, we define the vertices as ground set $X = V$ and the feasible solutions $F(X) \subseteq 2^V$ are all small

vertex covers in the graph. For simplicity, we ignore cost or weight functions and ask for the mere existence of a solution (here: a path, a small vertex cover).

While this is not a general definition, many typical combinatorial problems can be defined this way such as INDEPENDENT SET (an independent set is a subset of vertices), HAMILTONIAN PATH (a Hamiltonian path is a subset of edges), SUBSET SUM (a solution for subset sum is a subset of numbers).

We distinguish the natural encoding universe U from the solution ground set X over which the solutions are defined. With that, we reach a larger class of problems. In VERTEX COVER, the encoding universe $U = V$ is the same as the solution ground set $X = V$, because a vertex cover is a set of vertices and a graph is a set of vertices which are in relation via edges. In contrast, the instances of USTCON are still graphs while the solutions are subsets of edges. Thus for USTCON, the solution ground set and the universe do not coincide.

We begin with the definition of nested relations in order to define the instances of combinatorial problems. With these nested relations, we are able to define all possible associations of universe elements as well as between universe elements and relational elements. Thus in a graph $G = (V, E)$, we are not only able to for example encode edges $E \subseteq V \times V$ but also an incidence relation $I \subseteq V \times E$ or the neighborhood relation $N \subseteq V \leq^{|V|}$.

Definition 12.1 (Nested Relations). *Let U be a set. Then $\mathcal{R}(U)$ is the set of nested relations over U defined by the smallest set fulfilling:*

$$U \in \mathcal{R}(U) \tag{12.1}$$

$$A \in \mathcal{R}(U), \quad \text{if } A \subseteq B \text{ for some } B \in \mathcal{R}(U) \tag{12.2}$$

$$\bigtimes_i A_i \in \mathcal{R}(U), \quad \text{if for all } i, A_i \in \mathcal{R}(U) \tag{12.3}$$

We denote the set of relation elements that include $r \in A \in \mathcal{R}(U)$ by $R(r)$.

With access to all nested relations over the universe, we are able to define not only a variety of problems but we are also able to meaningfully define gadget reductions between problems. The solution ground set $X = R$ is then a subset of relational elements of one (nested) relation $R \in \mathcal{R}(U)$ over the gadget reduction universe U . Thus the solutions are of the form $F(R) \subseteq 2^R$.

Definition 12.2 (Combinatorial Decision Problem). *A combinatorial decision problem P_A is a set of tuples $(U_A, R_A, F_A(R_A))$ with the set of universe elements U_A , relations $R_A \in \mathcal{R}(U_A)^r, r \in \mathbb{N}$, and the set of feasible solutions $F_A(R_A) \subseteq 2^{R_A}$ for some $1 \leq i \leq r$. We assume that $R_A^1 = U_A$. We call R_A the instance of the problem and R_A is a YES-instance if and only if $F_A(R_A) \neq \emptyset$. We use an index set I_A to easily address the members of the tuple R_A .*

For simplicity, we may omit the problem in the index of U_A, R_A and $F_A(R_A)$ as well as the dependence of the feasible solutions $F(R)$ on the relations R and write F . For a better understanding, we again use USTCON as an example.

Example 12.1 (Undirected s - t -Connectivity Problem). *The input of USTCON is a graph $G = (V, E)$ and two vertices $s, t \in V$. A feasible solution is a path from s to t in G . This translates to the following tuple (U, R, F) . The universe U consists of the vertices V . The relations in R are the edges E and the vertices s and t , that is, $R = (V, E, s, t)$. The feasible solutions are all s - t -paths $p \in F \subseteq 2^E$ in G defined as subsets of edges.*

Observe that for combinatorial problems, the encoding of the input and the solutions depends only on the universe of elements. Thus, the universe elements in U build the atoms of the problem. The (nested) relations R model the relations between these atoms. The feasible solutions F model all possible combinations of solution elements that are feasible.

12.2.1 Scenarios for Robust Problems

Before we are able to define recoverable robust problems, we need to define scenarios. Scenarios are a central concept in robust optimization, which model the uncertainty. A Hamming distance recoverable robust problem P_A^{HDDR} is based on a combinatorial problem P_A . We then define a scenario as follows.

Definition 12.3 (Scenarios). *A scenario of the Hamming distance recoverable robust problem P_A^{HDDR} is a problem instance $(U_A, R_A, F_A(R_A))$ of the base problem P_A .*

Encoding of Scenarios. For scenarios, we use explicit encodings, implicit encodings or succinct encodings. We consider elemental uncertainty, for which it is uncertain whether a combinatorial element is part of a scenario or not. Thus, all of these encodings are based on combinatorial elements of an instance, which include the universe and all relation elements. This is different to uncertainty over the costs of elements, where the underlying combinatorial elements remain the same for all scenarios. If a combinatorial element is not part of a scenario, then all relation elements that include this combinatorial element are discarded as well in the scenario. For example, if a vertex v in a graph problem is discarded, then all edges incident to v are discarded, too. We denote this removal of combinatorial elements with $U \setminus \{r\}$ and $R \setminus R(r)$, where the removal of r removes all relation elements $R(r)$ that contain r . We call the elements that are part of the current scenario the *active* elements, otherwise we call the elements *inactive*.

First, we will use explicit encodings by providing the complete instance encoding over the base problem P_A . Additionally, we use implicit encodings by providing a set of all elements that are different from base scenario σ_0 . Furthermore, we address succinct encodings of scenarios as well. These encodings usually encode an exponential number of scenarios in polynomial space. The popular concept of discrete budgeted uncertainty, also known as Γ -scenarios, [BS04b] falls into this last category as well as later defined *xor*-dependencies, which use logical operators between the elements to encode which element is active, i.e. part of a scenario.

12.2.2 Hamming Distance Recoverable Robust Problems

Now, we define Hamming distance recoverable robust problems. For this, we need a definition of the Hamming distance over a set.

Definition 12.4 (Hamming Distance of Sets). *Let A, B be two sets. Then, we define the Hamming distance $H(A, B)$ of set A and B to be*

$$H(A, B) := |A \Delta B| = |\{x \mid \text{either } x \in A \text{ or } x \in B\}|$$

Intuitively, a Hamming distance recoverable robust problem P_A^{HDDR} is based on a nominal combinatorial decision problem P_A , e.g. USTCON. We distinguish the *base scenario* from *uncertainty scenarios*. The base scenario σ_0 is the instance on which the first solution \mathbf{s}_0 has to be computed. The uncertainty scenarios $\sigma \in \mathbf{S}$ are the scenarios for which the solution \mathbf{s} , that has to be adapted from \mathbf{s}_0 , have to be computed. All scenarios of a problem may share universe elements or relation elements. In conclusion, we not only have to find a solution for one instance, but for one base scenario σ_0 and for all uncertainty scenarios in \mathbf{S} . That is, we can recover from every possible scenario with a new solution to the problem. The solutions to the uncertainty scenarios, nonetheless, may have a Hamming distance of at most κ to the solution of the base scenario. We always define the Hamming distance over the solution ground set X between the solutions from $F(X) \subseteq 2^X$. Formally, we obtain the following definition.

Definition 12.5 (Hamming Distance Recoverable Robust Problem). *A Hamming distance recoverable robust problem P_A^{HDDR} is a combinatorial problem based on a combinatorial problem P_A . P_A^{HDDR} is defined as a set of tuples $(U, R, F(R))$ with*

$U = U_0 \cup \bigcup_{\sigma \in S} U_\sigma$ is the universe. The universe is the union over all universe elements that occur in the scenarios.

$R = (R_0, (R_\sigma)_{\sigma \in S}) = ((U_0, R_0^2, \dots, R_0^r), (U_\sigma, R_\sigma^2, \dots, R_\sigma^r)_{\sigma \in S})$ are the relations. The relations are separate for each scenario.

$F(R) = \{(s_0, (s_\sigma)_{\sigma \in S}) \in F_0(R_0) \times (F_\sigma(R_\sigma))_{\sigma \in S} \mid H(s_0, s_\sigma) \leq \kappa \text{ for all } \sigma \in S\}$ are the feasible solutions. The Hamming distance $H(s, s')$ is defined over the elements in the solutions s, s' .

The feasible solutions are not subsets of some relation R but consist of tuples including the solution for each scenario in F , which also adhere to the Hamming distance. In general, we assume that the bound on the Hamming distance κ is part of the input.

Observe that the specifications are no restriction because every decision problem can be formulated as one base scenario and no uncertainty scenarios, that is $S = \emptyset$. On the other hand, the base problem P_A is a restriction of P_A^{HDDR} by setting $S = \emptyset$. Furthermore, the base scenario is defined by $\sigma_0 = (U_0, R_0, F_0)$ and all uncertainty scenarios $\sigma \in S$ are defined by $\sigma = (U_\sigma, R_\sigma, F_\sigma)$. Again, we provide an example for a better understanding of the definition and again, we use the problem USTCON.

Example 12.2 (Hamming Distance Recoverable Robust USTCON). *Let $G = (V, E)$ be a graph, $s, t \in V$ and $\kappa \in \mathbb{N}$. $USTCON^{HDDR}$ is a Hamming distance recoverable robust problem with feasible solutions $F \subseteq 2^E$. Thus, the Hamming distance is defined over the edges. The start and end vertices s and t remain the same for all scenarios. The input R contains the following: Each scenario $\sigma \in S$ encodes the set of active vertices V_σ and edges E_σ . The feasible solutions F consists of all s - t -paths $(p_0, p_\sigma) \in 2^{E_{\sigma_0}} \times 2^{E_{\sigma \in S}}$ such that $H(p_0, p_\sigma) \leq \kappa$, for all $\sigma \in S$. In other words, the question is*

$$\exists p_0 \in 2^{E_{\sigma_0}} : \forall \sigma \in S : \exists p_\sigma \in 2^{E_\sigma} : p_0 \in F_0, p_\sigma \in F_\sigma \text{ and } H(p_0, p_\sigma) \leq \kappa.$$

12.2.3 Combinatorial Problems with Partitions as Solutions

As already stated in the introduction of this section, Definition 12.2 is not a general definition for combinatorial decision problems. For example coloring (which asks for an independent set cover) or clique cover as well as many other problems are not covered by this definition, because these have partitions as solutions and not subsets of some relation. In order to meaningfully integrate these kind of problems into this framework, we need to adapt the definition of combinatorial decision problems as well as the definition for the Hamming distance between solutions because the solutions are not sets but partitions.

Definition 12.6 (Combinatorial Decision Problem with Partition Solutions). *A combinatorial decision problem with partition solutions P_A is a set of tuples $(U_A, R_A, F_A(R_A))$ with the set of universe elements U_A , relations $R_A \in \mathcal{R}(U_A)^r, r \in \mathbb{N}$, and the set of feasible solutions $F_A(R_A) \subseteq (2^{R_A^i})^k$ which are k -partitions of R_A^i for some $1 \leq i \leq r$. We assume that $R_A^1 = U_A$. We call R_A the instance of the problem and R_A is a YES-instance if and only if $F_A(R_A) \neq \emptyset$. We use an index set I_A to easily address the members of the tuple R_A .*

The only change in the definition in comparison to Definition 12.2 is that the set of feasible solutions is defined as $F_A(R_A) \subseteq (2^{R_A^i})^k$ such that $F_A(R_A)$ consists of k -partitions. We then

define the Hamming distance between two k -partitions to be the sum of the Hamming distances of the sets of the two partitions.

Definition 12.7 (Hamming Distance of Partitions). *Let $A, B \subseteq S^k$ be two k -partitions of the set S . Then, we define the Hamming distance $H_P(A, B)$ of partitions A and B to be*

$$H_P(A, B) := \sum_{i=1}^k H(A_i, B_i),$$

where $H(A_i, B_i)$ is the Hamming distance over the sets A_i, B_i .

Accordingly, we also define Hamming distance recoverable robust versions of combinatorial problems with partitions as solutions.

Definition 12.8 (Hamming Distance Recoverable Robust Problem with partition solutions).

A Hamming distance recoverable robust problem P_A^{HDRR} is a combinatorial problem based on a combinatorial problem P_A . P_A^{HDRR} is defined as a set of tuples $(U, R, F(R))$ with

$U = U_0 \cup \bigcup_{\sigma \in \mathcal{S}} U_\sigma$ is the universe. The universe is the union over all universe elements that occur in the scenarios.

$R = (R_0, (R_\sigma)_{\sigma \in \mathcal{S}}) = ((U_0, R_0^2, \dots, R_0^r), (U_\sigma, R_\sigma^2, \dots, R_\sigma^r)_{\sigma \in \mathcal{S}})$ are the relations. The relations are separate for each scenario.

$F(R) = \{(\mathbf{s}_0, (\mathbf{s}_\sigma)_{\sigma \in \mathcal{S}}) \in F_0(R_0) \times (F_\sigma(R_\sigma))_{\sigma \in \mathcal{S}} \mid H_P(\mathbf{s}_0, \mathbf{s}_\sigma) \leq \kappa \text{ for all } \sigma \in \mathcal{S}\}$ are the feasible solutions. The Hamming distance $H_P(\mathbf{s}, \mathbf{s}')$ is defined over the solution partitions \mathbf{s}, \mathbf{s}' .

All of the following results on Hamming distance recoverable robust problems also hold for these kinds of problems.

12.3 Recoverable Robust Problems and the Polynomial Hierarchy

In this section, we investigate the connection between Hamming distance recoverable robust problems and the polynomial hierarchy. For this, we introduce two succinct encodings: *xor*-dependencies and Γ -set scenarios. We first prove that the Hamming distance recoverable robust version of problems, which are in NP, are in Σ_3^p for both encodings. Then, we prove Σ_3^p -hardness of the Hamming distance recoverable robust 3SATISFIABILITY for both encodings.

Definition 12.9 (Hamming Distance Recoverable Robust 3SATISFIABILITY). *The problem 3SATISFIABILITY^{HDRR} with Hamming distance over the literals L is defined as follows.*

Input: *Literals L , clauses C , base scenario $\sigma_0 \subseteq L$, uncertainty scenarios $\mathcal{S} \subseteq 2^L$, $\kappa \in \mathbb{N}$*

Question: *Are there solutions $\mathbf{s}_0 \subseteq \sigma_0$ and $\mathbf{s}_\sigma \subseteq \sigma$ for all $\sigma \in \mathcal{S}$ such that $H(\mathbf{s}_0, \mathbf{s}_\sigma) \leq \kappa$ for all $\sigma \in \mathcal{S}$ and setting \mathbf{s}_0 and \mathbf{s}_σ to true, all corresponding formulae of clauses $C|_{\sigma_0}$ and $C|_\sigma$ are satisfied?*

We begin with *xor*-dependency scenarios.

Definition 12.10 (*xor*-Dependency Scenarios). *Let σ_0 be the base scenario. The encoding of *xor*-dependencies is a tuple $(E', \{(E_{1,1}, E_{1,2}), \dots, (E_{n,1}, E_{n,2})\})$, where E' and all $E_{i,j}$ are pairwise disjoint sets of combinatorial elements for all $i \in \{1, \dots, n\}, j \in \{1, 2\}$. Then the scenario set \mathcal{S} includes all σ of the form $\sigma = \sigma_0 \Delta (E' \cup E_1 \cup \dots \cup E_n)$ with either $(E_i = E_{i,1})$ or $(E_i = E_{i,2})$ for all $i \in \{1, \dots, n\}$.*

Observe that with a linear sized encoding, exponentially many scenarios may be encoded. We study this combinatorial explosion with the result that it introduces more complexity for Hamming distance recoverable robust problems in comparison to the base problem. Concretely, we use 3SATISFIABILITY as base problem and show the Σ_3^P -hardness of 3SATISFIABILITY^{HDRR} with a linear number of *xor*-dependencies. From that point on, we can derive hardness results for further problems. Before we start the analysis of the hardness, we shall show that if $P_A \in \text{NP}$, then P_A^{HDRR} with a linear number of *xor*-dependencies is in Σ_3^P .

Theorem 12.1. *If $P_A \in \text{NP}$, then P_A^{HDRR} with *xor*-dependencies is in Σ_3^P .*

Proof. We present a polynomial time verifier that receives an (\exists -quantified) string y_1 , a (\forall -quantified) string y_2 , and an (\exists -quantified) string y_3 as input together with the instance. The first string y_1 encodes the solution \mathbf{s}_0 to the base scenario. The second string y_2 encodes the scenario σ for all $\sigma \in \mathbf{S}$. The third string encodes the solution \mathbf{s}_σ for the selected scenario σ .

The solution to the scenarios \mathbf{s}_0 and $(\mathbf{s}_\sigma)_{\sigma \in \mathbf{S}}$ are encoded as a subset of active elements in the corresponding scenario. The scenarios σ_0 and $\sigma \in \mathbf{S}$ can be computed in polynomial time from the input encoding encoded as sets, because the number of *xor*-dependencies is limited by the input length. Furthermore, the solutions \mathbf{s}_0 and $(\mathbf{s}_\sigma)_{\sigma \in \mathbf{S}}$ are subsets of σ_0 and $\sigma \in \mathbf{S}$ correspondingly. Consequently, the length of the input to the verifying algorithm is at most polynomial in the input length.

We can now construct the following algorithm that runs in polynomial time to verify the correctness of the strings. First we compute the explicit encodings of the base scenario and the scenario $\sigma \in \mathbf{S}$ encoded in y_2 in polynomial time. We then verify whether the solution \mathbf{s}_0 encoded by y_1 is a solution to σ_0 and whether \mathbf{s}_σ encoded by y_3 is a solution to scenario σ . This is doable by using the existing verifier for the base problem that exists because the problem is in NP. At last, we check $H(\mathbf{s}_0, \mathbf{s}_\sigma) \leq \kappa$. \square

Theorem 12.2. *3SATISFIABILITY^{HDRR} with *xor*-dependency scenarios is Σ_3^P -hard.*

Proof. We reduce $\exists\forall\exists$ 3SATISFIABILITY to 3SATISFIABILITY^{HDRR}. For this, let (X, Y, Z, C) be the $\exists\forall\exists$ 3SATISFIABILITY instance, where $\exists X\forall Y\exists Z C(X, Y, Z)$ is the formula with clauses $C(X, Y, Z)$. We denote the 3SATISFIABILITY^{HDRR} instance as I .

Variables We modify the variable set as follows. The variable set X remains the same. We substitute Y by $\{y_i^t, y_i^f \mid y_i \in Y\} =: Y'$. At last, we define $Z' := Z \cup \{y_{i,0}^t, y_{i,1}^t, y_{i,0}^f, y_{i,1}^f \mid y_i \in Y\}$.

Clauses The clauses are then modified as follows. For all $y_i \in Y$, we add $y_i^f \leftrightarrow 0$ and $y_i^t \leftrightarrow 1$ to the formula. Furthermore for all $y_i \in Y$, we add $y_i^t \leftrightarrow y_{i,1}^t$, $y_i^t \leftrightarrow \bar{y}_{i,0}^t$, $y_i^f \leftrightarrow y_{i,0}^f$, $y_i^f \leftrightarrow \bar{y}_{i,1}^f$ to the formula. At last, we do the following substitutions: For every clause $c = (a, b, y_i) \in C$ with $a, b \in X \cup Y \cup Z$, we substitute c by the clauses $(a, b, y_{i,1}^t)$ and $(a, b, y_{i,0}^f)$ and for clauses $c = (a, b, \bar{y}_i) \in C$ with $a, b \in X \cup Y \cup Z$ we substitute c by the clauses $(a, b, y_{i,0}^t)$ and $(a, b, y_{i,1}^f)$. We denote the set of modified clauses from C by C' . This is possible in polynomial time because we have a 3SATISFIABILITY instance and we are introducing at most eight new clauses per existing clause.

Scenarios In the base scenario of I only the variables from X are active. The uncertainty scenarios are encoded with *xor*-dependencies. For this, we introduce *xor*-dependencies on the variables and clauses from y_i^t and y_i^f for all $i \in \{1, \dots, |Y|\}$. Concretely, we define the set $E' = Z' \cup C'$ and for each $i \in \{1, \dots, n\}$, we define $E_{i,1} = \{y_i^t, (y_i^t \leftrightarrow y_{i,1}^t), (y_i^t \leftrightarrow \bar{y}_{i,0}^t), (y_i^t \leftrightarrow 1)\}$ as well as $E_{i,2} = \{y_i^f, (y_i^f \leftrightarrow y_{i,0}^f), (y_i^f \leftrightarrow \bar{y}_{i,1}^f), (y_i^f \leftrightarrow 0)\}$. At last we set the maximum Hamming distance between the literals to $\kappa = |Y| + |Z'|$.

Polynomial Time This transformation is computable in polynomial time because for each literal and each clause in (X, Y, Z, C) a fixed amount of literals and clauses in I are created. Furthermore, the formula can be transformed into CNF by substituting $a \leftrightarrow b$ with clauses $(\bar{a} \vee b)$ and $(a \vee \bar{b})$.

Correctness For the correctness, we have to prove that the constructed instance over the variable sets X , Y' , and Z' together with the *xor*-dependency scenarios are logically equivalent to the $\exists\forall\exists$ SATISFIABILITY formula. First, we focus on the $\exists X$ part. Any assignment to the variables from X is a valid solution to the base scenario. Because $\kappa = |Y| + |Z'|$ and $|Y| + |Z'|$ new variables appear in all of the uncertainty scenarios, the decision on the variables from X is made while choosing a solution to the base scenario and cannot be changed in any uncertainty scenario. Thus the decision on the variables from X are the same in both the base scenario and the chosen uncertainty scenario.

Next, we concentrate on the $\forall Y$ part. First for all $i \in \{1, \dots, |Y|\}$, the clauses $1 \leftrightarrow y_i^t$ and $0 \leftrightarrow y_i^f$, force the variable y_i^t to be always true and the variable y_i^f to be always false if they are active. The *xor*-dependencies activate exactly one of y_i^t and y_i^f for all $i \in \{1, \dots, |Y|\}$. Furthermore, if y_i^t is active, then $y_{i,0}^t$ evaluates to 0 and $y_{i,1}^t$ evaluates to 1, and if y_i^f is active, then $y_{i,0}^f$ evaluates to 0 and $y_{i,1}^f$ evaluates to 1. Thus the clauses containing $y_{i,0}^t$ and $y_{i,1}^t$ (resp. $y_{i,0}^f$ and $y_{i,1}^f$) have the same satisfaction behavior than the clauses that contain y_i (resp. \bar{y}_i) in the $\exists\forall\exists$ SATISFIABILITY formula. If on the other hand, $y_i^t \in E_{i,1}$ is inactive, then also the clauses $y_i^t \leftrightarrow y_{i,1}^t$ and $y_i^t \leftrightarrow \bar{y}_{i,0}^t$ are deactivated such that both $y_{i,0}^t$ and $y_{i,1}^t$ can be set to 1. This allows all clauses containing $y_{i,0}^t$ or $y_{i,1}^t$ to be trivially fulfilled, whenever y_i^t is inactive. The same argument holds for $y_i^f \in E_{i,2}$, i.e. the clauses $y_i^f \leftrightarrow y_{i,0}^f$ and $y_i^f \leftrightarrow \bar{y}_{i,1}^f$ are deleted and both $y_{i,0}^f$ and $y_{i,1}^f$ can be set to 1. Because the combinations allowed by the *xor*-dependencies are all $2^{|Y|}$ possible truth assignments to variables Y , the *xor*-dependency scenarios are equivalent to a $\forall Y$ for the variables Y . Thus, we also have a one-to-one correspondence between the variables in Y and Y' in both instances.

At last, we have to consider the $\exists Z'$ part. All variables from the sets X and Y' in the instance of $3\text{SATISFIABILITY}^{HDDR}$ are already assigned equivalently to the assignment of the variables from the sets X and Y in the $\exists\forall\exists$ SATISFIABILITY formula. The variables from the set $\{y_{i,0}^t, y_{i,1}^t, y_{i,0}^f, y_{i,1}^f \mid y_i \in Y\}$ are assigned according to Y' . All variables of the $3\text{SATISFIABILITY}^{HDDR}$ instance that are not yet assigned are free variables from Z . The clauses C' , however, are equivalent to the clauses from the $\exists\forall\exists$ SATISFIABILITY formula. Thus the rest of the variables (in both instances these are the variables from Z) is one-to-one correspondent.

In conclusion, the instance from $\exists\forall\exists$ SATISFIABILITY is equisatisfiable to the constructed instance of $3\text{SATISFIABILITY}^{HDDR}$ because the assignments on the set X , Y , and Z correspondent to the assignments in X , Y' , and Z' .

□

While the other parts of the chapter were developed independent from Goerigk et al. [GLW24], the results for Γ -set scenarios build upon it. The results based on *xor*-dependencies are adaptable to the Γ -set scenarios as described in this section. For the Γ -set scenarios, we use the definition over sets instead of elements as in Γ -scenarios, which is defined as follows.

Definition 12.11 (Γ -set Scenarios). *Let σ_0 be the base scenario. The encoding of Γ -set scenarios is a tuple $(E', \{E_1, E_2, \dots, E_n\})$, where E' and all E_i are pairwise disjoint sets of combinatorial*

elements for all $i \in \{1, \dots, n\}$. Then, the corresponding scenario set \mathcal{S} includes all σ of the form $\sigma = \sigma_0 \Delta (E' \cup \bigcup_{E \in \mathcal{E}} E)$ with $\mathcal{E} \subseteq \{E_1, E_2, \dots, E_n\}, |\mathcal{E}| \leq \Gamma$.

Again, with a linear sized encoding, exponentially many scenarios may be encoded. We show Σ_3^p -hardness of $3\text{SATISFIABILITY}^{\text{HDRR}}$ with Γ -set scenarios. A proof on the so-called ROBUST ADJUSTABLE SAT was already conducted by Goerigk et al. [GLW24]. This version of 3SATISFIABILITY uses uncertainties over the costs instead of the elements as in Σ_3^p -hardness of $3\text{SATISFIABILITY}^{\text{HDRR}}$ with Γ -set scenarios. Thus, the proof is not analogous as it is different in technicalities, nevertheless, we reuse their basic idea of introducing the cheat detection gadget (modeled by the s -variables) for our proof. Furthermore, we show also that if $P_A \in \text{NP}$, then P_A^{HDRR} with Γ -set scenarios is in Σ_3^p .

Theorem 12.3. *If $P_A \in \text{NP}$, then P_A^{HDRR} with Γ -set scenarios is in Σ_3^p .*

Proof. Each scenario from the Γ -set scenarios is encodable in polynomial space because the number of sets in \mathcal{E} from Definition 12.11 is limited by the input length. Thus, this proof is analogous to the proof for xor -dependencies. \square

Theorem 12.4. *$3\text{SATISFIABILITY}^{\text{HDRR}}$ with Γ -set scenarios is Σ_3^p -hard.*

Proof. We heavily reuse the transformation for xor -dependencies. Nevertheless, we have to introduce a mechanism to accommodate the less structured Γ -set scenarios in comparison to xor -dependencies. At last, the scenarios have to be adapted to Γ -set scenarios.

We reduce $\exists\forall\exists 3\text{SATISFIABILITY}$ to $3\text{SATISFIABILITY}^{\text{HDRR}}$. For this, let (X, Y, Z, C) be the $\exists\forall\exists 3\text{SATISFIABILITY}$ -instance, where $\exists X \forall Y \exists Z C(X, Y, Z)$ is the formula with clauses $C(X, Y, Z)$. We denote the $3\text{SATISFIABILITY}^{\text{HDRR}}$ instance as I .

Variables We modify the variable set as follows. The variable set X remains the same. We substitute the set Y by $\{y_i^t, y_i^f \mid y_i \in Y\} =: Y'$. Moreover, we define set $Z' := Z \cup \{y_{i,0}^t, y_{i,1}^t, y_{i,0}^f, y_{i,1}^f \mid y_i \in Y\} \cup \{s, s_i \mid y_i \in Y\}$. The added variables s_i for each $y_i \in Y$ and the additional variable s fulfill the same function as in the proof of Goerigk, Lendl and Wulf [GLW24].

Clauses The clauses are then modified as follows. For all $y_i \in Y$, we add $y_i^f \leftrightarrow 0$ and $y_i^t \leftrightarrow 1$ to the formula. Furthermore for all $y_i \in Y$, we add $y_i^t \leftrightarrow y_{i,1}^t, y_i^t \leftrightarrow \bar{y}_{i,0}^t, y_i^f \leftrightarrow y_{i,0}^f, y_i^f \leftrightarrow \bar{y}_{i,1}^f$ to the formula. Then, we do the following substitutions: For every clauses $c = (a, b, y_i) \in C$ with $a, b \in X \cup Y \cup Z$ we substitute c by the clauses $(a, b, y_{i,1}^t)$ and $(a, b, y_{i,0}^f)$ and for clauses $c = (a, b, \bar{y}_i) \in C$ with $a, b \in X \cup Y \cup Z$ we substitute c by the clauses $(a, b, y_{i,0}^t)$ and $(a, b, y_{i,1}^f)$. This is possible in polynomial time because we have a 3SATISFIABILITY instance and we are introducing at most eight clauses per clause. Moreover, we add \bar{s} to all clauses $c \in C$, such that we obtain a formula equivalent to $s \rightarrow C(X, Y, Z)$. We denote the set of modified clauses from C by C' . At last, we add $\bar{y}_i^t \vee s_i$ and $y_i^f \vee s_i$ as well as $s \vee \bar{s}_1 \vee \bar{s}_2 \vee \dots \vee \bar{s}_{|Y|}$ to the clauses.

Scenarios The first scenario of I consists only of the variables from X . Based on this, we encode the uncertainty scenarios with Γ -set scenarios. For this, we include the variable y_i^t (respectively y_i^f) together with its clauses in one of the E_i . Concretely, we define $E' = Z' \cup C' \cup \{(s \vee \bar{s}_1 \vee \dots \vee \bar{s}_{|Y|})\}$. Furthermore, we define $E_{2i-1} = \{y_i^t, (y_i^t \leftrightarrow y_{i,1}^t), (y_i^t \leftrightarrow \bar{y}_{i,0}^t), (y_i^t \leftrightarrow 1), (\bar{y}_i^t \vee s_i)\}$ and $E_{2i} = \{y_i^f, (y_i^f \leftrightarrow y_{i,0}^f), (y_i^f \leftrightarrow \bar{y}_{i,1}^f), (y_i^f \leftrightarrow 0), (y_i^f \vee s_i)\}$ for $i \in \{1, \dots, |Y|\}$. At last, set $\kappa = |Y| + |Z'|$ and $\Gamma = |Y|$.

Polynomial Time This transformation is computable in polynomial time because for each literal and each clause in (X, Y, Z, C) a fixed amount of literals and clauses in I are created. Furthermore, the formula can be transformed into 3CNF by substituting $a \leftrightarrow b$ with clauses $(\bar{a} \vee b)$ and $(a \vee \bar{b})$ and using Karp's reduction from SAT to 3SAT [Kar72].

Correctness For the correctness, we have to prove that the Γ -set scenarios within the construction are logically equivalent to *xor*-dependencies. Indeed the introduction of the cheat detection gadget, i.e. the s -variables, ensures this. For this, observe that whenever the set of uncertain elements \mathcal{E} is smaller than $\Gamma = |Y|$, there is a pair of variables y_i^t, y_i^f that is not active. Consequently, the clauses $\bar{y}_i^t \vee s_i$ and $y_i^f \vee s_i$ are inactive and s_i can be assigned to 0. It follows that \bar{s}_i satisfies the clause $s \vee \bar{s}_1 \vee \bar{s}_2 \vee \dots \vee \bar{s}_{|Y|}$ such that s can be assigned 0. Then all clauses are fulfilled by the addition of \bar{s} to all clauses from C . This also holds, whenever there is an active pair of y_i^f and y_i^t because by the pigeonhole principle there is a $j \in \{1, \dots, |Y|\}$ such that neither y_j^f nor y_j^t is active such that s_j can be assigned to 0. Therefore, all non-trivial cases require exactly one of y_i^f and y_i^t to be active, which is equivalent to *xor*-dependencies. □

12.4 Classes of Recoverable Robust Problems

We have shown that 3SATISFIABILITY^{HDRR} is the canonical Σ_3^P -complete Hamming distance recoverable robust problem. The goal is to “upgrade” the existing reductions on the NP-level to reduce the corresponding Hamming distance recoverable robust problems to each other. If we are additionally able to guarantee transitivity, we are also able to easily achieve complexity results for a large class of problems. Essentially, the reduction between Hamming distance recoverable robust problems needs to preserve the structure of the scenarios. For this, consider problems P_A and P_B . We need to achieve that a combinatorial element e_A in P_A is active if and only if the combinatorial elements E_B , to which e_A is mapped in P_B , are active. Then, we can use this one-to-many correspondence to (de)activate the corresponding elements in the instance of P_B .

Many of the properties from above are already constituted by the informal concept of gadget reductions. Gadget reductions describe that each part of the problem P_A is mapped to a specified part of the problem P_B that inherits the behavior in problem P_A . We adjust this concept to combinatorial elements, that is universe elements and relation elements, for our purpose. The goal is that a gadget is a subset of combinatorial elements in P_B for every combinatorial element in P_A . Furthermore, we preserve the (in)activeness of elements in a scenario. We call reductions that fulfill this property *modular* in the sense that all gadgets are easily (de)activatable. Furthermore, the solution size, which is the number of universe elements in a solution, has to adapt accordingly while being easy to compute in order to define the Hamming distance in the reduction correctly. We approach this later by demanding that the solution size of every gadget has to be a constant, i.e. it does not change when (de)activating other gadgets.

12.4.1 Universe Gadget Reduction

Let P_A be a combinatorial decision problem with instance tuples (U_A, R_A, F_A) and P_B a combinatorial decision problem with instance tuples (U_B, R_B, F_B) . A Universe Gadget Reduction f_{\rightarrow} that many-one-reduces P_A to P_B is composed of a (possibly empty) constant gadget Y_{const} , which is the same for every instance, and of the independent mappings: $f_{R_A^i, R_B^j} : R_A^i \rightarrow 2^{R_B^j}$ for all $(i, j) \in$

$I_A \times I_B$. We, then, call the substructure

$$Y_x = f_{\leq}(x) = \bigcup_{(i,j) \in I_A \times I_B} f_{R_A^i, R_B^j}(x)$$

the gadget for the specific universe element or relation element $x \in \bigcup_i R_A^i$. Additionally, we denote the set of all gadgets by $\Upsilon(R_A) = \{Y_r \mid r \in R_A^i \text{ with } i \in I_A\} \cup \{Y_{const}\}$ for the instance R_A . The mappings must fulfill the following properties.

- (1) Pre-image uniqueness: Let $y \in R_B^j$ for some $j \in I_B$, then either $y \in Y_{const}$ or there is exactly one $(i, j) \in I_A \times I_B$ and exactly one $x \in R_A^i$ such that $y \in f_{R_A^i, R_B^j}(x)$.
- (2) Modularity: If a combinatorial element $r \in R_A^i$ from (U_A, R_A, F_A) is removed to form a new instance (U'_A, R'_A, F'_A) , the removal of the gadget of r in (U_B, R_B, F_B) induces a correct reduction instance (U'_B, R'_B, F'_B) . A removal of $r \in R_A^i$ corresponds to the substitution by a (possibly empty) removal gadget Y_r^{rem} in P_B :

$$f_{\leq}(R_A \setminus R(r)) = (R_B \setminus f_{\leq}(R(r))) \cup Y_r^{rem}.$$

If the removal gadget is empty for all combinatorial elements, we call the modularity *strong*, otherwise *weak*. We substitute the gadgets Y_x , for $x \in R(r)$, with the removal gadget Y_r^{rem} in $\Upsilon(R_A)$ correspondingly. We consider the elements of a removal gadget to be disjoint from the elements of the original gadgets in order to guarantee pre-image uniqueness.

This definition of a gadget reduction for combinatorial decision problems ensures that the gadgets are uniquely relatable to the generating combinatorial elements and every element is easily deactivatable. Note that only combinatorial elements from P_A can be removed such that the new instance P'_A is a validly encoded instance. That is, combinatorial elements cannot be removed in general as this may void the validity of the instance, e.g. in USTCON the universe elements s and t cannot be deleted.

For the sake of simplicity, we only use gadget reductions originating from 3SATISFIABILITY. Therefore, we consider the following properties of solutions in a gadget reduction from 3SATISFIABILITY. These have to be proven individually for each reduction from 3SATISFIABILITY. For this, let (L, C) be a 3SATISFIABILITY instance that consists of literals L and clauses C . We introduce *variable gadgets* and *clause gadgets*. 3SATISFIABILITY has literals as universe elements. Furthermore, it includes the following relations not exclusively:

literals and negated literals	$\{(\ell, \bar{\ell}) \mid \ell \in L\}$
clauses	$\{(\ell^i, \ell^j, \ell^k) \mid (\ell^i, \ell^j, \ell^k) = c \in C \subseteq L^3\}$
literal and clause	$\{(\ell, c) \mid \ell \in c \in C\}$
negated literal and clause	$\{(\bar{\ell}, c) \mid \ell \in c \in C\}$

A *variable gadget* exists for each literal pair $\ell, \bar{\ell}$ and consists of the literal gadgets of ℓ and $\bar{\ell}$ as well as the gadget for the relation element $(\ell, \bar{\ell})$ of the literals and negated literals relation. A *clause gadget* simulates a clause. For this, all gadgets for relations that include a clause (clause, literal and clause, negated literal and clause, literals in clause, negated literals in clause) build up the clause gadget.

We first assume that the solution on the literals, i.e. the variable assignment is one-to-one correspond to the local solution on the variable gadget. More precisely, let $\ell_i, \bar{\ell}_i$ be the literals corresponding variable x_i , then there is exactly one local solution on the variable gadget of x_i that corresponds to the assignment of true to variable x_i and exactly one that corresponds to the assignment of false to the variable x_i . Furthermore, the local solution of the constant gadget is always the same. For weakly modular reductions, we additionally assume the following solution

extension property. Consider Y_x^{rem} for variable x and all Y_z^{rem} for variables $z \in Z$ such that x and z share a clause. Then, for each assignment to the variables in Z , there needs to be a local solution on Y_x^{rem} and Y_z^{rem} for all $z \in Z$ such that if Y_z^{rem} is deactivated and Y_x is activated for all $z \in Z$, while Y_x^{rem} stays active, the following holds: For all extending solutions to the assignment to Z in the 3SAT-instance, there is an extending solution to the corresponding local solutions on Y_z and the fixed local solution on Y_x^{rem} in the reduction instance. Additionally, the solution size has to adapt to the modularity of the gadgets in the universe gadget reduction. That is, if a combinatorial element in P_A is removed such that the corresponding gadgets in P_B are removed, the solution size of the instance of P_B is well-defined.

Solution Size. In order to correctly define the Hamming distance κ for a reduction from a problem P_A^{HDDR} to P_B^{HDDR} based on a universe gadget reduction from P_A to P_B , we need to find a solution size function. We demand that each gadget $Y \in \Upsilon$ has a constant *local solution size*, which is defined by the universe gadget reduction. A YES-instance has a solution size, which is defined by the sum of all local solution sizes defined as follows.

Definition 12.12 (3SATISFIABILITY-Reduction Solution Size Function). *Let P_B be a problem such that a universe gadget reduction f from 3SATISFIABILITY to P_B exists. Let (L, C) be a 3SATISFIABILITY-instance. The gadgets have a local solution size of $size(Y)$ for each $Y \in \Upsilon(L, C)$. The function*

$$size_f : 3SAT \rightarrow \mathbb{N} : (L, C) \mapsto \sum_{Y \in \Upsilon(L, C)} size(Y)$$

describes the target solution size over universe elements of $f(L, C) = R_B$ for R_B to be a YES-instance of P_B .

In the following, we only consider universe gadget reductions that have such a solution size function. We assume that the local solution size of each gadget is a constant independent of the generating combinatorial element and which combinatorial elements are active. That is, all variable gadgets and each gadget of a k -clause gadget have the same solution size. Thus, the solution size function is computable in polynomial time. While this is necessary, it is not a serious restriction as we see later. All of the reductions that we present later inherently have this property.

12.4.2 Properties of Universe Gadget Reductions

The definitions of universe gadget reductions and its solutions size function imply the following three properties, which are specifically desired as illustrated before.

Lemma 12.1. *A universe gadget reduction is total and one-to-many. The inverse to a universe gadget reduction is many-to-one.*

Proof. Let P_A and P_B combinatorial problems with $P_A \preceq^{UGR} P_B$. For every relation element $x \in \bigcup_i R_A^i$, the mappings $f_{R_A^i, R_B^j}(x)$ map to corresponding relation elements of P_B . By definition of a universal gadget reductions every relation element of P_B is generated by such a mapping or is part of the constant gadget Y_{const} such that universal gadget reductions are total. By the definition of the mappings and the constant gadget, universe gadget reductions are one-to-many because a relation element $y \in \bigcup_j R_B^j$ of P_B can be only mapped by one mapping from a relation element $x \in \bigcup_i R_A^i$ or is part of Y_{const} . Analogously, the inverse mapping of the universal gadget reduction is many-to-one. \square

Thus by definition, it is ensured that each element $y \in Y_{const} \cup \bigcup_j R_B^j$ of P_B is left unique and thus belongs to exactly one gadget. Another desirable property is transitivity. While strongly modular universe gadget reductions are transitive, we have to pay more attention to weakly modular reductions. This is due to the introduced removal gadgets. In the case that a strongly modular reduction is chained after a weakly modular reduction, the removal gadget can be transformed again into a removal gadget, making the resulting reduction weakly modular. In the case that two weakly modular reductions are chained together, the removal gadgets of both reductions may interact with each other. Then, however, it is not clear how to transform the removal gadgets into a working removal gadget in general. Thus in general, we do not reach transitivity for weakly modular reductions.

Lemma 12.2. *Polynomial universe gadget reductions are transitive in the following sense:*

- (1) *strongly modular universe gadget reductions are transitive*
- (2) *a strongly modular reduction followed by a weakly modular reduction results in a weakly modular reduction*
- (3) *a weakly modular reduction followed by a strongly modular reduction results in a weakly modular reduction*

Proof. Let P_A be a combinatorial decision problem with relations R_A , P_B a combinatorial decision problem with relations R_B and P_C a combinatorial decision problem with relations R_C . Firstly, we prove that the pre-image uniqueness is upheld. Formally, the concatenation of the mappings $f_{R_A^i, R_B^j} : R_A^i \rightarrow R_B^j$ and $f_{R_B^j, R_C^k} : R_B^j \rightarrow R_C^k$ has to preserve the following property: Let $z \in R_C^k$, for some $k \in I_C$, then either $z \in Y_{const}^{A \rightarrow C}$ or there is exactly one $(i, k) \in I_A \times I_C$ and exactly one $x \in R_A^i$ such that $z \in f_{R_A^i, R_C^k}(x)$.

Let $z \in R_C^k$ for some $k \in I_C$.

Case 1 $z \in Y_{const}^{B \rightarrow C}$. Then z is generated as part of the constant gadget of the reduction from P_B to P_C . Thus, $z \in Y_{const}^{A \rightarrow C}$.

Case 2 $z \notin Y_{const}^{B \rightarrow C}$. There is exactly one $(j, k) \in I_B \times I_C$ and exactly one $y \in R_B^j$ such that $z \in f_{R_B^j, R_C^k}(y)$. Then, $y \in R_A^j$ for some $j \in I_B$.

Case 2.1 $y \in Y_{const}^{A \rightarrow B}$. Then z is generated by exactly one element of $y \in Y_{const}^{A \rightarrow B}$. Thus, $z \in Y_{const}^{A \rightarrow C}$.

Case 2.2 There is exactly one $(i, j) \in I_A \times I_B$ and exactly one $x \in R_A^i$ such that $y \in f_{R_A^i, R_B^j}(x)$. Thus by definition, of the universe gadget reduction, z is generated by exactly on $(i, j, k) \in I_A \times I_B \times I_C$ and exactly on x with $z = f_{R_B^j, R_C^k}(f_{R_A^i, R_B^j}(x))$.

It follows that all gadgets of relation $r \in P_A$ elements are pre-image uniquely mapped in the instance $f_{P_B, P_C} \circ f_{P_A, P_B}(R_A)$. Furthermore, the modularity of the gadgets is preserved. For this, we have to consider the following three cases:

Case 1 For strongly modular reductions f_{P_A, P_B} and f_{P_B, P_C} , the concatenation of f_{P_A, P_B} and f_{P_B, P_C} is still strongly modular. Specifically if a relation element r in P_A is deleted, its gadgets are deleted from P_B according to the reduction f_{P_A, P_B} and the instance of P_B is the correct instance. Because the elements are deleted in P_B , the reduction f_{P_B, P_C} continues to delete the corresponding gadgets in P_C , whereby the the instance in P_C stays correct for all deletions. Accordingly, strongly modular universe gadget reductions are transitive.

Case 2 Let f_{P_A, P_B} a strongly modular universe gadget reduction and f_{P_B, P_C} a weakly modular universe gadget reduction. If a relation element r in P_A is deleted, then its gadgets are deleted from P_B as well according to the reduction f_{P_A, P_B} . The deletion of the gadgets of r in the instance of P_B results in the introduction of (potentially empty) removal gadgets in P_C according to the weakly modular reduction f_{P_B, P_C} . This still yields a correct universe gadget reduction, which is weakly modular.

Case 3 Let f_{P_A, P_B} a weakly modular universe gadget reduction and f_{P_B, P_C} a strongly modular universe gadget reduction. The deletion of a relation element r in the instance of P_A results in the introduction of (potentially empty) removal gadgets in P_B . By definition of weakly universe gadget reductions, this results in a correct instance to which we can apply the strongly modular reduction f_{P_B, P_C} . The resulting reduction of the concatenation $f_{P_A, P_B} \circ f_{P_B, P_C}$ is weakly modular, where the removal gadgets are defined by

$$f_{P_B, P_C}(Y_{rem}^r) = \bigcup_{\substack{(i,j) \in I_B \times I_C \\ y \in Y_{rem}^r}} f_{R_B^i, R_C^j}(y)$$

It follows that the concatenation of two strongly modular universe gadget reductions fulfill the pre-image uniqueness and strong modularity. Furthermore, the concatenation of a weakly modular gadget reduction with a strongly modular gadget reduction (independent of the order of concatenation) results in a reduction that is pre-image unique and fulfills weak modularity. \square

Furthermore, the solution size function adheres to the modularity of the universe gadget reduction.

Lemma 12.3. *The solution size function adheres to modularity. In other words, let (L, C) and (L', C') be instances of 3SATISFIABILITY with $L' \subseteq L$ and $C' \subseteq C$. Furthermore, let f be a universe gadget reduction from 3SATISFIABILITY to P_B such that $f(L', C')$ results from $f(L, C)$ by removing the corresponding gadgets. Then,*

$$size_f(L', C') = \sum_{Y \in \Upsilon(L', C')} size(Y).$$

Proof. If a gadget Y_r is removed, the solution is decreased by $size(Y_r)$ and increased by the local solution size of the removal gadget $size(Y_r^{rem})$. Because a solution size function is the sum of the local solutions size of each gadget, the following holds:

$$\begin{aligned} size_f(L', C') &= \sum_{Y \in \Upsilon(L', C')} size(Y) \\ &= size(Y_{const}) + \sum_{x \in (L', C')} size(Y_x) + \sum_{x \in ((L, C) \setminus (L', C'))} size(Y_x^{rem}). \end{aligned}$$

Accordingly, by the definition of the solution size function to be the sum of the constant local solution sizes of the gadgets, it adheres to modularity. \square

Now, we present a general reduction from $\exists \exists 3$ SATISFIABILITY to the Hamming distance recoverable robust P_B^{HRR} based on the structure that a universe gadget reduction provides. That is, if there is a polynomial time universe gadget reduction f from 3SATISFIABILITY to P_B such that the solution properties hold and a corresponding polynomial time solution size function $size_f$ exists, then there is a polynomial time reduction for the Hamming distance recoverable robust version of P_B with Hamming distance over the universe elements, transforming the scenarios accordingly.

Theorem 12.5. *If 3SATISFIABILITY is universe gadget reducible to P_B in polynomial time such that there is a corresponding solution size function, and the solution properties hold, then there is a polynomial time reduction from $\exists\forall\exists$ 3SATISFIABILITY to P_B^{HDRR} , where the Hamming distance is defined over the solution ground set and the scenario encodings are xor-dependency scenarios.*

Proof. In the following, we prove that $\exists\forall\exists$ 3SATISFIABILITY is reducible to P_B^{HDRR} . For this, we reuse the reduction from $\exists\forall\exists$ 3SATISFIABILITY to 3SATISFIABILITY^{HDRR} together with the universe gadget reduction from 3SATISFIABILITY to P_B . The basic idea is to substitute the variables and clauses by the corresponding gadgets.

Now, let $\exists X\forall Y\exists Z C(X, Y, Z)$ be the $\exists\forall\exists$ 3SATISFIABILITY instance of variable sets X , Y , and Z as well as clauses C . In order to construct the instance of P_B , we store the gadgets defined by each mapping $f_{R_{3SAT}^i, R_B^j}(r)$ in a table, for every element $r \in R_{3SAT}^i$ of all relations R_{3SAT}^i residing in the 3SATISFIABILITY input. We can compute this table, because we have a polynomial time universe gadget reduction between 3SATISFIABILITY and P_B . With this table, we can now compute the scenarios in polynomial time with the following principle. The idea is to activate the variable and clause gadgets, whenever the variable or clause is active. As universe gadget reductions are modular, the (de)activation of a variable or clause is easily translatable into the instance of P_B : We remove the corresponding gadgets (and introduce necessary removal gadgets). While the (de)activation of variables from X and Z is straightforward, we have to take care about the (de)activation of the gadgets for variables from Y according to the given structure of uncertainty. More precisely for each variable $y_i \in Y$, we deactivate the variable gadget for y_i^t or y_i^f and thus also the corresponding clauses as in Table 12.1. We model this operation with the xor-dependencies by adding

- ▶ the gadget of variable y_i^t and its clauses together with the variable removal gadget of y_i^t into the corresponding set $E_{i,1}$
- ▶ the gadget of variable y_i^f and its clauses together with the variable removal gadget of y_i^f into the corresponding set $E_{i,2}$.

	$y_i = y_i^t$	$y_i = y_i^f$
xor-dependencies	$y_i^t \leftrightarrow 1$	$y_i^f \leftrightarrow 0$
	$y_i^t \leftrightarrow y_{i,1}^t$	$y_i^f \leftrightarrow y_{i,0}^f$
	$y_i^t \leftrightarrow \bar{y}_{i,0}^t$	$y_i^f \leftrightarrow \bar{y}_{i,1}^f$

Table 12.1: The clauses to (de)activate for xor-dependencies.

These (de)activations are possible because the reduction is modular. In the base scenario, the variable gadgets of y_i^t and y_i^f are inactive, while the removal gadgets of them are active. Consequently if the variable y_i^t (respectively y_i^f) is activated by the set $E_{i,1}$ (respectively $E_{i,2}$) in an uncertainty scenario, the removal gadget of y_i^t (respectively y_i^f) is removed and the variable gadget is added. Additionally, the corresponding clauses that contain y_i^t (respectively y_i^f) are activated and we obtain the instance where the gadgets of y_i^t (respectively y_i^f) are active simulating the variable accordingly. Note that these are exactly the clauses from Table 12.1, which also have the same structure (six clauses of two variables each by substituting $a \leftrightarrow b$ with clauses $(\bar{a} \vee b)$ and $(a \vee \bar{b})$). Therefore, the variable gadgets respectively the variable removal gadgets of y_i^t and y_i^f have the same overall local solution size for each $i \in \{1, \dots, |Y|\}$. By the correctness of the reduction from $\exists\forall\exists$ 3SATISFIABILITY to 3SATISFIABILITY^{HDRR} and the universe gadget reduction

from 3SATISFIABILITY to P_B , the reduction for each scenario remains correct and the one-to-one correspondence of the activeness between the variable and the variable gadget is upheld.

In order to describe the scenarios formally, we summarize the reduction from $\exists\forall\exists$ 3SATISFIABILITY to 3SATISFIABILITY^{HDRR} with *xor*-dependencies. First, we substituted all variables $y_i \in Y$ by two variables y_i^t and y_i^f . We then replaced the occurrences of the literal y_i in all clauses by $y_{i,1}^t, y_{i,0}^t, y_{i,1}^f, y_{i,0}^f$ resulting in a duplication of clauses. The additional variables $y_{i,1}^t, y_{i,0}^t, y_{i,1}^f, y_{i,0}^f$ were added to the set Z' , which was then defined by $Z' = Z \cup \{y_{i,1}^t, y_{i,0}^t, y_{i,1}^f, y_{i,0}^f \mid y_i \in Y\}$. The *xor*-dependencies were defined over the variables y_i^t and y_i^f and their clauses (compare Table 12.1).

We are now ready to describe the *xor*-dependencies formally. For this, we have to define the base scenario σ_0 and a set E' together with pairs of sets $(E_{i,1}, E_{i,2})$. For the base scenario, we compute the reduction instance of all variables and clauses that are available in any of the scenarios. We now deactivate the necessary gadgets to obtain an instance that corresponds to the base scenario. The base scenario contains only the variable gadgets from X . All other gadgets are removed and replaced by the corresponding removal gadgets, where the variable removal gadgets of $Y' = Y^t \cup Y^f$ are deactivated first. Accordingly, we define $\sigma_0 = \{Y_x \cup Y_a^{rem} \mid x \in X \text{ and } a \in Y' \cup Z' \cup C'|_X\}$. To construct the set E' , we observe that the variables of X and Z' and all clauses that contain variables from X or Z' are available in all uncertainty scenarios. Thus, the set E' contains the variable gadgets of variables from Z' as well as the clause gadgets for clauses containing X and Z' . Remember that variables $y_{i,1}^t, y_{i,0}^t, y_{i,1}^f, y_{i,0}^f$, for $1 \leq i \leq |Y|$, were added to set Z' and are thus active in exactly all uncertainty scenarios. Accordingly, we define $E' = \{Y_a^{rem} \cup Y_b \mid a \in Z' \cup C'|_X \text{ and } b \in Z', C'|_{X,Z'}\}$.

At last, we define the pairs of sets $(E_{i,1}, E_{i,2})$, where either $E_{i,1}$ or $E_{i,2}$. For this, we consider a variable $y_i \in Y$. Each of these variables was split into two variables y_i^t and y_i^f and the corresponding clauses from Table 12.1 were introduced. On the one hand, we define

$$E_{i,1} = \{Y_{y_i^t}^{rem}, Y_{y_i^t}, Y_{(y_i^t \leftrightarrow y_{i,1}^t)}, Y_{(y_i^t \leftrightarrow \bar{y}_{i,0}^t)}, Y_{(y_i^t \leftrightarrow 1)}\}$$

to include the variable gadget of y_i^t and its (possibly empty) removal gadget. On the other hand, we define

$$E_{i,2} = \{Y_{y_i^f}^{rem}, Y_{y_i^f}, Y_{(y_i^f \leftrightarrow y_{i,0}^f)}, Y_{(y_i^f \leftrightarrow \bar{y}_{i,1}^f)}, Y_{(y_i^f \leftrightarrow 0)}\}$$

to include the variable gadget of y_i^f as well as its (possibly empty) removal gadget. Note that in the base scenario the removal gadget of y_i^t (respectively y_i^f) was active and the variable gadget inactive. Thus if y_i^t (respectively y_i^f) is activated, then the variable gadget of y_i^t (respectively y_i^f) is activated and the removal gadget is deactivated.

As we have considered f to be a modular reduction based on the $\exists\forall\exists$ 3SATISFIABILITY instance (L, C) , we can set

$$\kappa = size_f(X \cup \bar{X}, \emptyset) + size_f(X \cup \bar{X} \cup Y' \cup Z' \cup \bar{Z}', C') - 2 \sum_{x \in X} size(Y_x).$$

This is correct, because the gadgets of variables from X as well as the removal gadgets of $|Y|$ many variables from Y' are present in all scenarios while all other variables are only present either in the base scenario or the uncertainty scenarios. Furthermore for each variable y_i , there is one variable gadget active (e.g. of y_i^t) and one variable removal gadget (e.g. of y_i^f) active. This guarantees that the solution size for all uncertainty scenarios is the same because there are always $|Y|$ variable gadgets active and $|Y|$ variable removal gadgets active and we assume the gadgets to have a constant local solution size. Now consider the solution on the variables

on X . The base scenario activates only the variable gadgets of the set X . The solution size of $size_f(X \cup \bar{X}, \emptyset) + size_f(X \cup \bar{X} \cup Y' \cup Z' \cup \bar{Z}', C')$ is exactly the sum of the solution sizes of the base scenario and any of the uncertainty scenarios without considering $|Y|$ of the gadgets corresponding to variables from Y' . Since we subtract $2 \sum_{x \in X} size(Y_x)$ to define κ and all elements but the elements from X are Y many variable gadgets from Y' , the partial solution to the variables of X has to stay the same by switching from the base scenario to any of the uncertainty scenarios. In conclusion, the partial solution on X is the same for all of the uncertainty scenarios.

By the correctness of the underlying gadget reduction, we obtain the corresponding reduction instance for the base scenario and for each of the uncertainty scenarios. We can now use the same argumentation as in the reduction from $\exists \forall \exists$ SATISFIABILITY to 3 SATISFIABILITY^{HDRR}. First in the base scenario, a partial solution to the gadgets of the \exists -quantified variables X is fixated because the Hamming distance is chosen accordingly. Therefore, the partial solution on the gadgets of X is the same for the base scenario and all uncertainty scenarios such that the order of quantification is followed. This solution corresponds to an assignment of the variables X . In the uncertainty scenarios, the partial solution on the variable gadgets of X has to be extended to a complete solution of the reduction instance. Furthermore, each of the possible truth assignments on the variables Y is simulated by exactly the corresponding uncertainty scenario, in which the corresponding gadgets are activated. Again the local solutions on the variable gadgets of Y' correspond to an assignment of the variables of Y . At last, we need to extend the solution on the variable gadgets of X and Y' to a complete solution. For this, a local solution to the variable gadgets of Z' and the clause gadgets need to be found. Note that this can be freely chosen because the Hamming distance κ is large enough. If and only if this is possible, we have a YES-instance for the corresponding uncertainty scenario, where the assignment on the variable gadgets of X, Y' , and Z' in P_B^{HDRR} correspond to an assignment to the variables X, Y , and Z in $\exists \forall \exists$ SATISFIABILITY. Note that in the case of weak modularity, the removal gadgets in $E_{i,1}$ for y_i^t (or in $E_{i,2}$ for y_i^f , analogously) might stay active in the base and uncertainty scenarios. Since the variables $y_{i,0}^t$ and $y_{i,1}^t$ are assigned to true if y_i^t is inactive, an existing local solution on the variable removal and clause removal gadgets can be extended in the uncertainty scenarios due to the solution extension property. For this, the local solution corresponding to the assignment to true is applied to the gadgets for $y_{i,0}^t$ and $y_{i,1}^t$ if $E_{i,1}$ is inactive in the uncertainty scenarios. Otherwise if the removal gadgets of $E_{i,1}$ are deactivated and the actual variable and clause gadgets activated, no solution on the now active variable and clause gadgets is fixed and thus can be freely assigned. It follows that, the $\exists \forall \exists$ SATISFIABILITY is satisfiable if and only if the base scenario as well as all uncertainty scenarios of the P_B^{HDRR} instance are YES-instances such that the Hamming distance is at most κ . \square

We can derive a similar result for Γ -set scenarios by reusing the construction from above.

Theorem 12.6. *If 3 SATISFIABILITY is universe gadget reducible to P_B in polynomial time such that there is a corresponding solution size function, and the solution properties hold, then there is a polynomial time reduction from $\exists \forall \exists$ SATISFIABILITY to P_B^{HDRR} , where the Hamming distance is defined over the solution ground set and the scenario encodings are Γ -set scenarios.*

Proof. We modify the construction from Theorem 12.5 by introducing gadgets for the s -variables as in Theorem 12.4. Let $E_i^t = E_{i,1}$ and $E_i^f = E_{i,2}$. Instead of having one s_i for the pair of variables y_i^t and y_i^f , we split the s_i into two variables s_i^t and s_i^f to add the corresponding clauses into the sets E_i^t corresponding to activate y_i^t (i.e. setting y_i to 1) and E_i^f corresponding to activate y_i^f (i.e. setting y_i to 0). Specifically, we add the gadgets for the variables $s, s_1^t, s_i^f, \dots, s_{|Y|}^t, s_{|Y|}^f$ and the corresponding clauses for $(s \vee (\bar{s}_1^t \wedge \bar{s}_1^f) \vee \dots \vee (\bar{s}_{|Y|}^t \wedge \bar{s}_{|Y|}^f))$, and $(\bar{y}_i^t \vee s_i^t)$ and $(y_i^f \vee s_i^f)$

for all $i \in \{1, \dots, |Y|\}$. Because the variables $s, s_1, \dots, s_{|Y|}$ are part of Z' in Theorem 12.4, we add the variable gadgets for $s, s_1^t, s_i^f, \dots, s_{|Y|}^t, s_{|Y|}^f$ to the set E' . We additionally add the gadget of clause $(\bar{y}_i^t \vee s_i^t)$ to E_i^t , where all other gadgets (variable, clause, and variable removal) of y_i^t still reside and we add analogously the gadget of clause $(y_i^f \vee s_i^f)$ to E_i^f . Accordingly if E_i^t (respectively E_i^f) is not activated, s_i^t (respectively s_i^f) can be set to 0 because the clause $(\bar{y}_i^t \vee s_i^t)$ (respectively $(y_i^f \vee s_i^f)$) is removed. Furthermore, the subformula $(s \vee (\bar{s}_1^t \wedge \bar{s}_1^f) \vee \dots \vee (\bar{s}_{|Y|}^t \wedge \bar{s}_{|Y|}^f))$ works equivalently to the clause $(s \vee \bar{s}_1 \vee \dots \vee \bar{s}_{|Y|})$ in Theorem 12.4 because both s_i^t and s_i^f need to be set to 0 such that s can be set to 0. We have to analyze the following three cases.

If exactly one of E_i^t and E_i^f is active, then we have the same situation as in the proof of Theorem 12.5 for *xor*-dependencies. Accordingly, we still have to prove the correctness for the other two cases.

If there is an $i \in \{1, \dots, |Y|\}$ such that neither of E_i^t and E_i^f are active, then the removal gadgets of both y_i^t and y_i^f are still active as in the base scenario. Thus, a correct instance of P_B is induced. Because the 3SATISFIABILITY formula is trivially solvable by setting $s = 0$, the resulting instance of P_B is a YES-instance. The Hamming distance of κ is sufficient because at most $\Gamma = |Y|$ many gadgets of variables from Y' are activated to which the solution has to be switched.

If there is an $i \in \{1, \dots, |Y|\}$ such that both E_i^t and E_i^f are active, the variable and clause gadgets of both y_i^t and y_i^f are active while the removal gadgets of both y_i^t and y_i^f are inactive. Again a correct instance of P_B is induced, where both y_i^t and y_i^f are simulated to be active. The instance is, however, also a YES-instance due to the pigeonhole principle and $\Gamma \leq |Y|$. That is, there is a $j \in \{1, \dots, |Y|\}$ such that neither E_j^t nor E_j^f are active. Accordingly, the corresponding 3SATISFIABILITY instance is satisfiable by setting $s = 0$ and a correct YES-instance of P_B is induced as in the case above.

Observe that the subformula $(s \vee (\bar{s}_1^t \wedge \bar{s}_1^f) \vee \dots \vee (\bar{s}_{|Y|}^t \wedge \bar{s}_{|Y|}^f))$ can be transformed into a CNF by Tseitin's transformation [Tse83]. Furthermore all clauses of length greater than four can be transformed into clauses of length three by Karp's reduction from CNF-SAT to 3SAT [Kar72]. The newly introduced variables can be added to the set E' . \square

With these structural properties in mind, we can construct a whole set of Hamming distance recoverable robust problems. Note that the transitivity of the universe gadget reduction can be used to deduce further reductions.

12.4.3 Gadget Reductions for Various Combinatorial Decision Problems

In this section, we examine various but not all problems that are universe gadget reducible from 3SATISFIABILITY. The reductions are all well-known results or modifications of well-known results. We adapt these results to the universe gadget reduction framework to indicate that Theorems 12.5 and 12.6 are general statements. We prove the following theorem by showing that a universe gadget reduction from 3SATISFIABILITY exists for all the problems. For this, we use the transitivity of the reductions as illustrated in Figure 12.1.

Theorem 12.7. *The Hamming distance recoverable robust version of the following problems are Σ_3^P -complete with xor-dependency scenarios or Γ -set-scenarios: VERTEX COVER, DOMINATING SET, FEEDBACK ARC SET, FEEDBACK VERTEX SET, HITTING SET, INDEPENDENT SET, CLIQUE, SUBSET SUM, KNAPSACK, PARTITION, TWO MACHINE SCHEDULING, (UN)DIRECTED HAMILTONIAN CYCLE, (UN)DIRECTED HAMILTONIAN PATH, TRAVELING SALESMAN, 3DIMEN-*

SIGNAL MATCHING, EXACT COVER BY 3-SETS, k DISJOINT DIRECTED PATH ($k \geq 2$), 3COLORING, k COLORING, CLIQUE COVER.

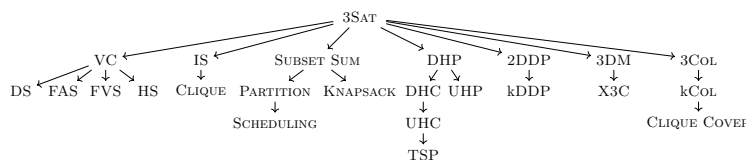


Figure 12.1: The tree of gadget reductions for all considered problems.

Vertex Cover

As an introductory example, we take a close look at a universe gadget reduction of 3SATISFIABILITY to VERTEX COVER, which was initially developed by Garey and Johnson [GJ79]. This example directly proves Lemma 12.4. For the VERTEX COVER reduction we use the very fine-grained universe gadget reduction for each combinatorial element. In the following reductions, however, we directly use variable and clause gadgets as described in Subsection 12.4.1, to shorten our argumentation.

Lemma 12.4. *3SATISFIABILITY is strongly modular universe gadget reducible to VERTEX COVER such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The problem 3SATISFIABILITY consists of the universe L for the literals and the relations

- ▶ $R_{\ell, \bar{\ell}}$ that relates a literal ℓ with its negation $\bar{\ell}$,
- ▶ $R_{\ell, c}$ that relates a literal ℓ to a clause c , iff $\ell \in c$ and
- ▶ $R_{\ell, \ell', c}$ that relates literals ℓ and ℓ' , iff $\ell, \ell' \in c$.

The problem VERTEX COVER, on the other hand, consists of vertices V and edges E that form a graph $G = (V, E)$. Based on these universe and relations, the gadgets as in Figure 12.2 can be found. Therefore, we define the mappings:

$$f_{L, V}, f_{L, E}, f_{R_{\ell, \bar{\ell}}, V}, f_{R_{\ell, \bar{\ell}}, E}, f_{R_{\ell, c}, V}, f_{R_{\ell, c}, E}, f_{R_{\ell, \ell', c}, V}, f_{R_{\ell, \ell', c}, E}$$

The dashed vertices in Figure 12.2 indicate that these are part of a different gadget.

A complete example can be found in Figure 12.3. On the other hand, the reduction based on variable and clause gadgets can also be established. For this, the relations from above are combined in the gadgets.

- ▶ The universe L is combined with relation $R_{\ell, \bar{\ell}}$ to a *variable gadget* for variable $x \in X$.
- ▶ The relations $R_{\ell, c}$ and $R_{\ell, \ell', c}$ are combined to one clause gadget that connects the corresponding variable gadget correctly to a clause $c \in C$.

These gadgets are depicted in Figure 12.4, in which the dashed vertices indicate that these are part of a different gadget. Observe that the gadgets only combine the more fine-grained relations and the overall reduction stays the same. That is, the reduction is overall the same for both views and can be found in Figure 12.3 as well. The existence of these variable gadgets and a clause gadgets also shows the strong modularity of this reduction: One can easily remove the variable

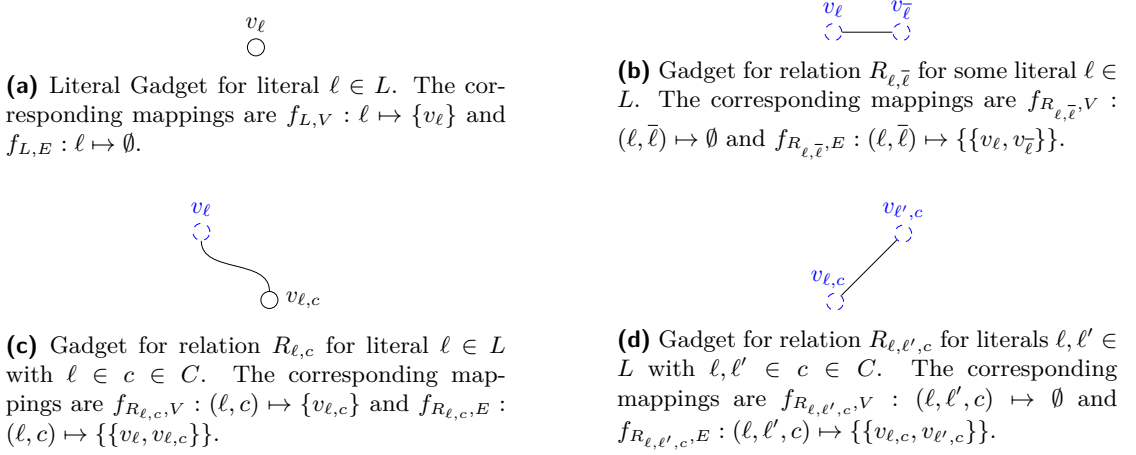


Figure 12.2: The gadgets for the universe and all relations for the reduction from 3SATISFIABILITY to VERTEX COVER.

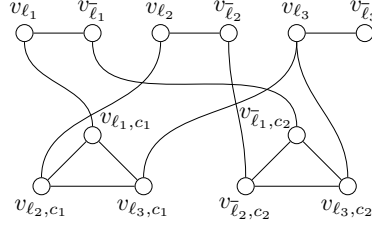


Figure 12.3: The reduction graph for 3SATISFIABILITY formula $C = \{\{l_1, l_2, l_3\}, \{\bar{l}_1, \bar{l}_2, \bar{l}_3\}\}$.

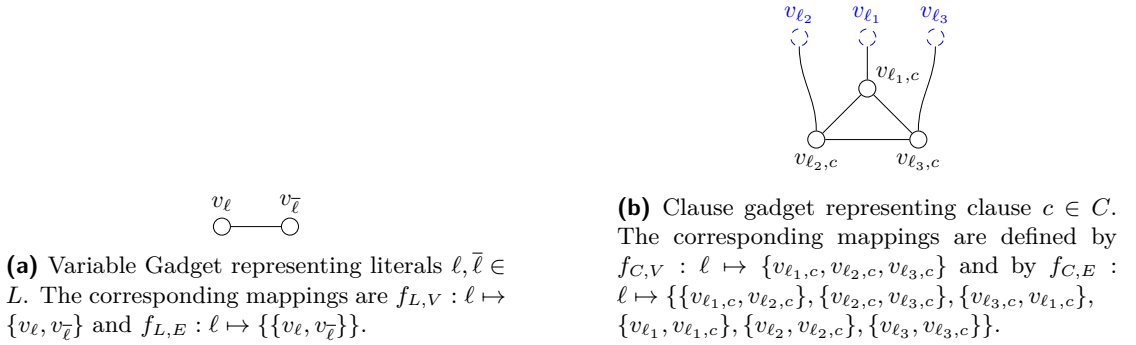


Figure 12.4: Gadgets for universe and relations for the 3SATISFIABILITY-VERTEX COVER reduction

gadget and all clause gadgets containing that variable or removing just one clause gadget. The resulting graph is the correct reduction graph of the corresponding 3SAT instance.

The solution size function for each gadget can be defined as follows. A solution includes one vertex for each literal in the solution of the 3SAT instance: v_{ℓ_i} is included if and only if x_i is assigned to true, and $v_{\bar{\ell}_i}$ is included if and only if x_i is assigned to false. Additionally to satisfy

the clause gadgets, two of the vertices of the 3-clique need to be included. If a vertex v_ℓ is in the solution, then the edge $\{v_\ell, v_{\ell,c}\}$ is covered. Therefore, the vertices $v_{\ell',c}$ and $v_{\ell'',c}$ can be taken into the solution in order to cover the 3-clique and the incident edges. If the clause is not satisfied, then non of v_ℓ , $v_{\ell'}$, and $v_{\ell''}$ are in the solution and all three vertices $v_{\ell,c}, v_{\ell',c}$, and $v_{\ell'',c}$ need to be included to cover all edges invalidating the solution. Thus, $size_f(L, C) = |L|/2 + 2|C|$. This also fulfills the necessary conditions by the modularity of the gadget reduction. Correspondingly, the feasible solutions of the scenarios are defined with the help of the solution size function. Concretely, a solution to the VERTEX COVER instance is feasible if the vertex cover has size at most $size_f(L, C)$. \square

Lemma 12.5. *VERTEX COVER is strongly modular universe gadget reducible to DOMINATING SET such that the solution properties hold and a solution size function for this reduction exists.*

Proof. There is a folklore reduction that is a universe gadget reduction. For every edge $\{u, v\}$ between vertices $u, v \in V$, the reduction adds two vertices uv^1 and uv^2 together with edges $\{uv^i, u\}, \{uv^i, v\}, i \in \{1, 2\}$. The universe elements of both problems are the vertices V and the relations are the edges E . Thus, there are vertex gadgets, see Figure 12.5a, defined by

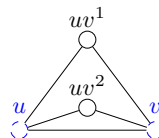
$$f_{V,V}, f_{V,E}$$

and the edge gadgets, see Figure 12.5b,

$$f_{E,V}, f_{E,E}.$$



(a) Vertex Gadget for $v \in V$. The corresponding mappings are $f_{V,V} : v \mapsto \{v\}$ and $f_{V,E} : \{u, v\} \mapsto \emptyset$.



(b) Edge Gadget for $v \in V$. The corresponding mappings are $f_{E,V} : \{u, v\} \mapsto \{\{uv^1\}, \{uv^2\}\}$ and $f_{E,E} : \{u, v\} \mapsto \{\{u, v\}, \{u, uv^1\}, \{uv^1, v\}, \{u, uv^2\}, \{uv^2, v\}\}$.

It is easy to see that both properties of a universe gadget reduction are fulfilled. The gadgets are disjoint. Furthermore, removing a vertex (and its incident edges) results in removing the corresponding vertex gadget and edge gadgets. Accordingly, the resulting instance remains correct for this reduction. Removing only an edge results in removing the edge gadget, which is also correct. At last, we consider the solution. The solution size function remains $size_f(L, C) = |L|/2 + 2|C|$ as in the VERTEX COVER reduction because the solution of the DOMINATING SET and VERTEX COVER build up a one-to-one correspondence. That is, a vertex v is part of a vertex cover if and only if v is part of a dominating set. The feasible solutions are accordingly defined by the dominating sets of size at most $size_f(L, C)$. \square

Lemma 12.6. *VERTEX COVER is strongly modular universe gadget reducible to FEEDBACK ARC SET such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction of Karp [Kar72] is a universe gadget reduction. VERTEX COVER consists of vertices V and edges E . FEEDBACK ARC SET consists of vertices V' and arcs A' . The reduction

maps every vertex $v \in V$ to two vertices $v_0, v_1 \in V'$ and one arc $(v_0, v_1) \in A'$. Furthermore, we map each edge $\{u, v\} \in E$ to two arcs $(u_1, v_0), (v_0, u_1) \in A'$. Thus, each edge $\{u, v\} \in E$ induces a cycle of four arcs, which has to be disconnected by removing one of the arcs. Observe that the arc $(v_0, v_1) \in A'$ is contained in all cycles induced by incident edges $e \in E$ of $v \in V$. Thus, it is always favorable to remove arcs $(v_0, v_1) \in A'$ which corresponds to taking $v \in V$ into the vertex cover. Because of this one-to-one correspondence between an arc (v_0, v_1) in FEEDBACK ARC SET and a vertex v in VERTEX COVER in the solution, the solution size remains $size_f(L, C) = |L|/2 + 2|C|$. Furthermore, the one-to-one correspondence between the elements and their gadgets guarantees the modularity and that all pre-images of all gadgets are unique. The feasible solutions are accordingly defined by the feedback arc sets of size at most $size_f(L, C)$. \square

Lemma 12.7. *VERTEX COVER is strongly modular universe gadget reducible to FEEDBACK VERTEX SET such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction of Karp [Kar72] is a universe gadget reduction. Again, VERTEX COVER consists of vertices V and edges E . On the other hand, FEEDBACK VERTEX SET consists of vertices V' and arcs A' . The reduction maps every vertex $v \in V$ to vertex $v \in V'$ and every edge $\{u, v\} \in E$ is mapped to two arcs (u, v) and (v, u) in A' . Because of the one-to-one correspondence in the solution between vertex $v \in V$ in VERTEX COVER and vertex $v \in V'$ in FEEDBACK VERTEX SET, the solution size remains $size_f(L, C) = |L|/2 + 2|C|$. Furthermore, the correspondence between the elements and their gadgets guarantees modularity and that all pre-images of all gadgets are unique. The feasible solutions are accordingly defined by all feedback vertex sets of size at most $size_f(L, C)$. \square

Lemma 12.8. *VERTEX COVER is strongly modular universe gadget reducible to HITTING SET such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction of Karp [Kar72] from VERTEX COVER to HITTING SET is a universe gadget reduction. VERTEX COVER consists of vertices V and edges E . The universe of HITTING SET is a set U and the relations are subsets $s_i \subseteq U$ for $1 \leq i \leq r$. Every vertex $v \in V$ is mapped to a corresponding element $v \in U$ and every edge $(u, v) \in E$ is mapped to a subset $s = \{u, v\} \subseteq U$. By the one-to-one correspondence of vertex v and element v in the universe of HITTING SET as well as the edge $\{u, v\}$ and the subset $s = \{u, v\}$, the gadgets are disjoint, uniquely retraceable to their origin, and the reduction is modular. At last, we consider the solution. The solution size function remains $size_f(L, C) = |L|/2 + 2|C|$ because the elements of HITTING SET and the vertices in VERTEX COVER build up a one-to-one correspondence. The feasible solutions are accordingly defined by the hitting sets of size at most $size_f(L, C)$. \square

Independent Set

Lemma 12.9. *3SATISFIABILITY is strongly modular universe gadget reducible to INDEPENDENT SET such that the solution properties hold and a solution size function for this reduction exists.*

Proof. For INDEPENDENT SET, we reuse the reduction from 3SATISFIABILITY to VERTEX COVER by Garey and Johnson [GJ79]. For 3SATISFIABILITY, we use the literals as universe elements and the relations $R_{\ell, \bar{\ell}}$, which relates a literal and its negation, $R_{\bar{\ell}, c}$, which relates a clause with the negation of the its literals, $R_{\ell, \bar{\ell}, c}$, which relates the literal and its negation with the clauses

the literal is in. INDEPENDENT SET, on the other side, consists of vertices V and edges E . This results in the mappings for the variable gadget, see Figure 12.6a,

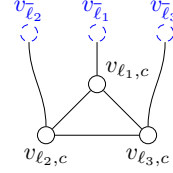
$$f_{L,V}, f_{L,E}, f_{R_{\ell,\bar{\ell}},V}, f_{R_{\ell,\bar{\ell}},E},$$

and the clause gadget, see Figure 12.6b,

$$f_{R_{\bar{\ell},c},V}, f_{R_{\bar{\ell},c},E}, f_{R_{\ell,\ell',c},V}, f_{R_{\ell,\ell',c},E}.$$



(a) Variable Gadget representing literals $\ell, \bar{\ell} \in L$. The corresponding mappings are $f_{L,V} : \ell \mapsto \{v_\ell, v_{\bar{\ell}}\}$ and $f_{L,E} : \ell \mapsto \{\{v_\ell, v_{\bar{\ell}}\}\}$.



(b) Clause gadget representing clause $c \in C$. The corresponding mappings are defined by $f_{C,V} : \ell \mapsto \{v_{\ell_1,c}, v_{\ell_2,c}, v_{\ell_3,c}\}$ and by $f_{C,E} : \ell \mapsto \{\{v_{\ell_1,c}, v_{\ell_2,c}\}, \{v_{\ell_2,c}, v_{\ell_3,c}\}, \{v_{\ell_3,c}, v_{\ell_1,c}\}, \{v_{\bar{\ell}_1}, v_{\ell_1,c}\}, \{v_{\bar{\ell}_2}, v_{\ell_2,c}\}, \{v_{\bar{\ell}_3}, v_{\ell_3,c}\}\}$.

Figure 12.6: Gadgets for universe and relations for the reduction from 3SATISFIABILITY to INDEPENDENT SET

Analogously to the VERTEX COVER reduction, this reduction is a universe gadget reduction. Furthermore, the solution size function includes one vertex for each variable gadget and one vertex for each clause gadget. A vertex v_{ℓ_i} is included in the solution if and only if x_i is assigned to true, and $v_{\bar{\ell}_i}$ is included in the solution if and only if x_i is assigned to false. Because v_ℓ is in the solution and thus not $v_{\bar{\ell}}$, the vertex $v_{\ell,c}$ can be included simulating the satisfaction of the clause. If a clause $c \in C$ is not satisfied, then all vertices $v_{\bar{\ell}}, v_{\bar{\ell}'},$ and $v_{\bar{\ell}''}$ for $\ell, \ell', \ell'' \in c$ are in the independent set such that non of $v_{\ell,c}, v_{\ell',c}, v_{\ell'',c}$ can be taken into the solution. With the same arguments as for the VERTEX COVER reduction, the solution size function is modular. We define the feasible solutions by the independent sets of size at least $size_f(L, C) = |L|/2 + |C|$. \square

Lemma 12.10. *INDEPENDENT SET is strongly modular universe gadget reducible to CLIQUE such that the solution properties hold and a solution size function for this reduction exists.*

Proof. For CLIQUE, we reuse the duality between VERTEX COVER, INDEPENDENT SET, and CLIQUE as described by Garey and Johnson [GJ79]. The problem INDEPENDENT SET consists of a graph with vertices V and edges E . On the other hand, we define CLIQUE with vertices V' as universe but a different relation $\bar{E} \subseteq V' \times V'$, the set of non-edges. This definition of CLIQUE allows us to use the equivalence as universe gadget reduction.

For the reduction, we map every vertex $v \in V$ to the vertex $v' \in V'$ and we map every edge $e \in E$ to a non-edge $\bar{e} \in \bar{E}$. Thus, we have a one-to-one correspondence between the vertices and the edges and non-edges. This one-to-one correspondence also holds for the solution. That is, every solution of one problem is also a solution to the other problem. By this one-to-one correspondence, the modularity, the pre-image uniqueness and the solution size of $size_f(L, C) = |L|/2 + |C|$ remains. The feasible solutions are accordingly defined by cliques of size at least $size_f(L, C)$. \square

Subset Sum

Lemma 12.11. *3SATISFIABILITY is weakly modular universe gadget reducible to SUBSET SUM such that the solution properties hold and a solution size function for this reduction exists.*

Proof. We use a modification of the reduction by Sipser [Sip97] from 3SATISFIABILITY to SUBSET SUM. For 3SATISFIABILITY, we use the literals as universe elements and the relations $R_{\ell, \bar{\ell}}$, which relates a literal and its negation, the clause relation R_c , which is a unary relation on the clauses, and $R_{\ell, c}$, which relates a clause with the negation of its literals.

SUBSET SUM, on the other side, consists of binary numbers of $\{0, 1\}^t$. For the sake of simplicity, the reduction description uses non-binary numbers. Numbers that are bigger than one are easily translatable in corresponding binary numbers with an offset such that a possible carry has no influence. This results in the mappings for the variable gadget, see Figure 12.7,

$$f_{L, \{0, 1\}^t},$$

and the clause gadget, see both Figures 12.8 and 12.9,

$$f_{R_c, \{0, 1\}^t}, f_{R_{\ell, c}, \{0, 1\}^t}.$$

ℓ_1	$\bar{\ell}_1$...	ℓ_n	$\bar{\ell}_n$	x_1	x_2	...	x_n	c_1	c_2	...	c_m
1	0	...	0	0	1	0	...	0	0	0	...	0
0	1	...	0	0	1	0	...	0	0	0	...	0

Figure 12.7: Variable Gadget representing literals $\ell, \bar{\ell} \in L$ (here for ℓ_1 and $\bar{\ell}$).

ℓ_1	$\bar{\ell}_1$...	ℓ_n	$\bar{\ell}_n$	x_1	x_2	...	x_n	c_1	c_2	...	c_m
0	0	...	0	0	0	0	...	0	11	0	...	0
0	0	...	0	0	0	0	...	0	12	0	...	0
0	0	...	0	0	0	0	...	0	13	0	...	0

Figure 12.8: Clause Gadget for $c \in C$ (here for c_1).

ℓ_1	$\bar{\ell}_1$...	ℓ_n	$\bar{\ell}_n$	x_1	x_2	...	x_n	c_1	c_2	...	c_m
1	0	...	0	0	0	0	...	0	1	0	...	0

Figure 12.9: Literal Clause Gadget for $\ell \in c \in C$ (here for ℓ_1 with $\ell_1 \in c_1$ and $\ell_1 \notin c_2, c_m$).

The target sum in SUBSET SUM plays a crucial role to simulate the satisfaction of the clause correctly. In Figure 12.10, the target sum is depicted. In a solution, the row corresponding to $\bar{\ell}_i$ is taken into the solution if and only if variable x_i is assigned true. This enables the solution to include the literal clause gadget for ℓ_i . Otherwise, the row corresponding to ℓ_i is part of the solution and the literal clause gadget for $\bar{\ell}_i$. Thus all clauses that include ℓ_i have at least 1 in their row. After this assignment, the suitable row from the clause gadget is included such that the sum of the clause column reaches 14. If a clause is not satisfied, the corresponding column cannot reach 14.

The modularity of this reduction is weak. The removal gadget of a clause is the addition of a number that simulates the satisfaction of that clause. This gadget is depicted in Figure 12.11.

$$\sum \left\| \begin{array}{c|c|c|c|c} \ell_1 & \bar{\ell}_1 & \dots & \ell_n & \bar{\ell}_n \\ \hline 1 & 1 & \dots & 1 & 1 \end{array} \right\| \left\| \begin{array}{c|c|c|c} x_1 & x_2 & \dots & x_n \\ \hline 1 & 1 & \dots & 1 \end{array} \right\| \left\| \begin{array}{c|c|c|c} c_1 & c_2 & \dots & c_m \\ \hline 14 & 14 & \dots & 14 \end{array} \right\|$$

Figure 12.10: The target value k of the sum.

Note that we leave the numbers from Figure 12.8 available. We can use this clause gadget to also construct a literal removal gadget for ℓ or $\bar{\ell}$. This gadget simulates the fulfillment of the clauses that contain ℓ or $\bar{\ell}$. Consequently, we add the clause removal gadget for clauses that contain ℓ or $\bar{\ell}$. Additionally, we leave the literal gadget within the instance. Then together with the clause gadget from Figure 12.8, the sum of the clause column containing ℓ or $\bar{\ell}$ can be set to 14, depending on the other variables. The partial solution in the base scenario is extendable to a full solution in the uncertainty scenarios because the clause removal gadget only fixes the solution on the literals that are part of that clause.

$$\left\| \begin{array}{c|c|c|c|c} \ell_1 & \bar{\ell}_1 & \dots & \ell_n & \bar{\ell}_n \\ \hline 0 & 0 & \dots & 0 & 0 \end{array} \right\| \left\| \begin{array}{c|c|c|c} x_1 & x_2 & \dots & x_n \\ \hline 0 & 0 & \dots & 0 \end{array} \right\| \left\| \begin{array}{c|c|c|c} c_1 & c_2 & \dots & c_m \\ \hline 14 & 0 & \dots & 0 \end{array} \right\|$$

Figure 12.11: The Removal Gadget for $c_1 \in C$.

At last, we describe the solution size function over the numbers. Due to weak modularity, we use a reduction f from an instance (L', C') with $L \subseteq L'$ and $C \subseteq C'$ such that

$$size_f(L, C) = |L'| + |C'|.$$

Overall, we include one number for each literal and one number for each clause. If a variable is removed, we include the removal gadgets for the clauses containing this variable and leave the gadgets for the literals and the literal clause relation in the instance. The removal of a clause (without also removing a variable) does not change the number of elements in the solution because 11, 12, 13, or 14 still need to be added such that the clause column adds up to 14. Accordingly the solution size of a variable gadget is the same as for the variable removal gadget. \square

Lemma 12.12. *SUBSET SUM is strongly modular universe gadget reducible to KNAPSACK such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction from SUBSET SUM to KNAPSACK is by generalization. The numbers in SUBSET SUM are mapped to objects of the weight and price corresponding to the value of the number. By setting the knapsack capacity and the price threshold to the target sum of SUBSET SUM, the reduction is complete.

Overall, this is a one-to-one correspondence between all combinatorial elements and the solutions. Thus, the modularity, the pre-image uniqueness, the solution properties and the solution size function of SUBSET SUM directly applies to KNAPSACK as well. \square

Lemma 12.13. *SUBSET SUM is strongly modular universe gadget reducible to PARTITION such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction from SUBSET SUM to PARTITION by Karp [Kar72] is a universe gadget reduction. The numbers A in SUBSET SUM are transferred to the PARTITION instance and remain unchanged. Furthermore, let k be the target sum of SUBSET SUM, then $k+1$ and $1-k+\sum_{a \in A} a$

are added to the PARTITION instance as well. These two numbers build up the constant gadget. W.l.o.g. we assume that $1 - k + \sum_{a \in A} a$ is in the first set of the partition.

Overall, the numbers from SUBSET SUM and PARTITION are one-to-one correspondent. Thus modularity and pre-image uniqueness hold. Furthermore, note that the solution of SUBSET SUM is the same as the set of the partition that additionally includes the element $1 - k + \sum_{a \in A} a$. Thus, we also have a one-to-one correspondence between the elements in the solutions. \square

Lemma 12.14. *PARTITION is strongly modular universe gadget reducible to TWO MACHINE SCHEDULING such that the solution properties hold and a solution size function for this reduction exists.*

Proof. PARTITION is a special case of TWO MACHINE SCHEDULING. By interpreting the numbers in the PARTITION instance to be the job times in TWO MACHINE SCHEDULING and by interpreting the sets of the partition as two identical machines, we have a one-to-one correspondence between the combinatorial elements and the solutions as well. Thus, pre-image uniqueness, modularity, and solution properties hold and the solution size function remains the same. \square

Hamiltonian Path

Lemma 12.15. *3SATISFIABILITY is weakly modular universe gadget reducible to DIRECTED HAMILTONIAN PATH such that the solution properties hold and a solution size function for this reduction exists.*

Proof. A modification of the reduction by Arora and Barak [AB09] is a universe gadget reduction. For 3SATISFIABILITY, we use the literals as universe elements and the relations $R_{\ell, \bar{\ell}}$, which relates a literal and its negation, and $R_{\ell, c}$, which relates a clause with the negation of its literals.

HAMILTONIAN CYCLE, on the other side, consists of vertices V and arcs A . This results in the mappings for the variable gadget, see Figure 12.13,

$$f_{L, V}, f_{L, A}, f_{R_{\ell, \bar{\ell}}, V}, f_{R_{\ell, \bar{\ell}}, A},$$

and the clause gadget, see Figure 12.14a and Figure 12.14b,

$$f_{R_{\ell, c}, V}, f_{R_{\ell, c}, A}.$$

In order to connect the variable gadgets, we also need a constant gadget defined by mappings $f_{const, V}$ and $f_{const, E}$, see Figure 12.12 .

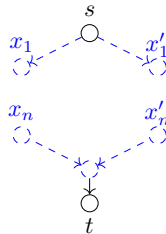


Figure 12.12: Constant Gadget for the reduction.

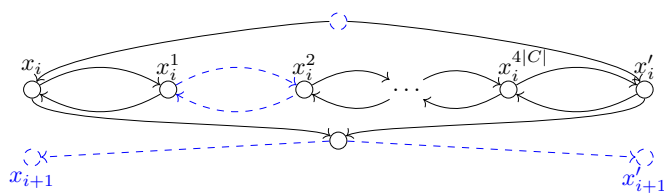


Figure 12.13: Variable Gadget representing literals $\ell, \bar{\ell} \in L$.



(a) If the literals ℓ and $\bar{\ell}$ are not in the j -th clause $c_j \in C$, we merge the vertices x_i^{4j-2} and x_i^{4j-1} .

(b) Clause Gadget for $\ell \in c \in C$.

Figure 12.14: Clause Gadget for $c \in C$.

If we include the path from x_i to x_i' , we simulate that the variable is assigned true. Vice versa, if the path from x_i' to x_i is included, the variable x_i is simulated to be false. Therefore, it is possible to include the vertices of clauses satisfied by the assignment into the Hamiltonian cycle.

This reduction is only weakly modular because removing a variable x_i results in a disconnected graph. For this, we can employ a clause removal gadget, which can be found in Figure 12.15. Then for a variable removable gadget of variable x_i , we leave the variable gadget in the instance, while introducing the clause removal gadget for all clauses that contain the variable x_i . A solution in the base scenario is extendable to a full solution because the clause removal gadget only fixes the solution on the literals that are part of the clause.

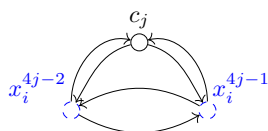


Figure 12.15: Clause Removal Gadget for clause $c \in C$.

At last, we describe the solution size function over the arcs. Due to weak modularity, we use a reduction f from an instance (L', C') with $L \subseteq L'$ and $C \subseteq C'$ such that

$$size_f(L, C) = 1 + (1 + 3 \cdot |L'|/2) \cdot |C'| + \sum_{c \in C'} (1 + |c|).$$

Overall, we include one arc for the constant gadget. Furthermore for each variable, we include $3|C'|$ arcs. For all clauses $c \in C$ of size $|c|$, we need to include $1 + |c|$ arcs. The size of a clause gadget and clause removal gadget are the same. Thus we do not have to change the solution size function in this regard. \square

Lemma 12.16. *DIRECTED HAMILTONIAN PATH is strongly modular universe gadget reducible to DIRECTED HAMILTONIAN CYCLE such that the solution properties hold and a solution size function for this reduction exists.*

Proof. Adding an arc from t to s as constant gadget closes the cycle. This reduction is a universe gadget reduction because the combinatorial elements are mapped one-to-one such that the solutions are one-to-one translatable as well. This directly proves the pre-image uniqueness, the modularity, and the solution properties. The solution size function includes an additional term of one for the arcs (t, s) , to be correct. \square

Lemma 12.17. *DIRECTED HAMILTONIAN PATH is strongly modular universe gadget reducible to UNDIRECTED HAMILTONIAN PATH such that the solution properties hold and a solution size function for this reduction exists.*

Proof. Karp's reduction [Kar72] is a universe gadget reduction. It triples the vertices and connects the triplets as depicted in Figure 12.16. Furthermore, each arc (u, v) in the graph is mapped to an edge $\{u^{out}, v^{in}\} \in E'$.

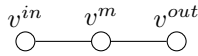


Figure 12.16: The vertex gadget for the reduction from DIRECTED HAMILTONIAN PATH to UNDIRECTED HAMILTONIAN PATH.

The pre-image uniqueness and the modularity remain. The solutions on arcs (u, v) and edges $\{u^{out}, v^{in}\}$ are one-to-one correspondent, while all edges $\{v^{in}, v^m\}$ and $\{v^m, v^{out}\}$ have to be in the solution. Thus, the solution properties still hold, while the solution size function needs to take the two edges from v^{in} over v^m to v^{out} into account for every vertex. That is, the number of used edges in a solution of the variable is tripled. Overall, we get

$$size_f(L, C) = 6 + 9|L'|/2 \cdot |C'| + \sum_{c \in C'} (1 + 3|c|).$$

\square

Lemma 12.18. *DIRECTED HAMILTONIAN CYCLE is strongly modular universe gadget reducible to UNDIRECTED HAMILTONIAN CYCLE such that the solution properties hold and a solution size function for this reduction exists.*

Proof. This reduction is completely analogous to Karp's reduction from DIRECTED HAMILTONIAN PATH to UNDIRECTED HAMILTONIAN PATH (Lemma 12.17). \square

Lemma 12.19. *UNDIRECTED HAMILTONIAN CYCLE is strongly modular universe gadget reducible to TRAVELING SALESMAN such that the solution properties hold and a solution size function for this reduction exists.*

Proof. We consider TRAVELING SALESMAN to be defined over an undirected weighted graph. This graph does not have to be complete. Then, the graph $G = (V, E)$ of the UNDIRECTED HAMILTONIAN CYCLE instance can be mapped to a weighted graph $G' = (V', E', w')$, where $V = V'$ and $E = E'$. The weights are set to 0 and the weight threshold is set to 0.

The reductions yields a one-to-one correspondence between the vertices and edges and thus between the solutions. It follows that the solution size function remains the same and the solution properties, the pre-image uniqueness as well as the modularity hold. \square

2-Disjoint Path

Lemma 12.20. *3SATISFIABILITY is weakly modular universe gadget reducible to 2DISJOINT DIRECTED PATH such that the solution properties hold and a solution size function for this reduction exists.*

Proof. A modification of the reduction by Fortune, Hopcroft and Wyllie [FW80] is a universe gadget reduction. This reduction is much more complex than earlier reductions such that we first explain the construction and then explain how this construction can be divided into variable and clause gadgets.

First of all, the reduction introduces a so-called switch gadget, which is visualized in Figure 12.17. The idea of this gadget is that two vertex-disjoint paths entering at vertex B (re-

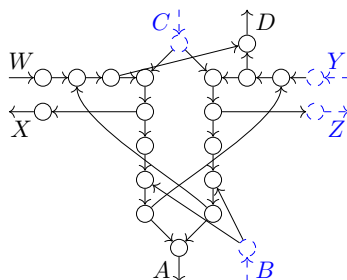


Figure 12.17: The switch gadget. The black solid elements are always part of the corresponding clause gadget. The vertices Y and Z are part of the gadget of the variable that belongs to the clause. The vertices B and C are part of the previous clause gadget if it is the first switch gadget of the clause.

spectively leaving at vertex A) need to leave through vertex D (respectively need to enter at vertex C) such that either the path from X to Z or the path from W to X is still usable without violating the vertex-disjoint path constraint. In order to integrate this gadget into the complete construction, we use the schematic view on the gadget as depicted in Figure 12.18.

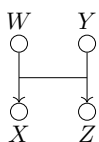


Figure 12.18: The schematic switch gadget.

We now start the description of the actual construction. Let (L, C) be the 3SAT instance of literal L and clauses C . On the other hand, let $(V, A, s_1, t_1, s_2, t_2)$ be the DIRECTED TWO DISJOINT PATH instance with vertex set V , arc set A , and the start and end vertices of the two disjoint paths s_1, t_1, s_2, t_2 . First, we introduce the four start and end vertices s_1, t_1, s_2, t_2 . Second for every literal $\ell \in L$, we create a path $\ell^1, \dots, \ell^{4|C|}$. In the original reduction, four vertices and three arcs are inserted for each clause. We modify the original reduction by adding the left and right arc to the variable gadget first. Then, the middle arc is part of the clause gadget if and only if the corresponding literal is in the clause, otherwise the middle of the two vertices are merged together such that only the left and right arc remain. Let $\ell_i, \bar{\ell}_i$ the literals corresponding to variable x_i . Then we connect the paths of literals ℓ_i and $\bar{\ell}_i$ by introducing two

vertices x_i^s and x_i^t together with arcs $(x_i^s, \ell_i^1), (x_i^s, \bar{\ell}_i^1)$ and $(\ell_i^{4|C|}, x_i^t), (\bar{\ell}_i^{4|C|}, x_i^t)$. These are part of the variable gadget. We further connect these gadgets by adding the arcs (x_i^t, x_{i+1}^s) for all $i \in \{1, \dots, |X|\}$. We call this the lobe for variable x_i . Third for each clause $c_j \in C$, we add two vertices c_j^1 and c_j^2 . We connect these two vertices by three arcs of the form (c_j^1, c_j^2) . Additionally, we connect these vertex pairs by adding the arc (c_j^2, c_{j+1}^1) for each $j \in \{1, \dots, |C| - 1\}$. These elements are also part of the clause gadget for clause c_j . At last, we connect the variable lobes with the clause path with an arc $(x_{\lfloor L/2 \rfloor}^t, c_1^1)$ and we add the arc $(c_{|C|}^2, t_1)$. These elements are part of the constant gadget. This summarizes the overall structure of the reduction.

We are now ready to introduce switch gadgets into the construction. For each k -clause $c \in C$, we add k switch gadgets to the construction. These k switch gadgets are part to the clause gadget of the corresponding clause $c \in C$. All of the switch gadgets are now stacked by merging the vertex C of one switch gadget with vertex A of the following switch gadget and by merging vertex B of one switch gadget with vertex D of the following switch gadget. This leaves the vertices W, X, Y and, Z unconnected. We connect these to the graph in the following way. Consider the j -th clause c_j that contains the literal ℓ . We identify W and X with the existing vertices ℓ^{4j-2} and ℓ^{4j-1} in the lobes of the variables. Note that we merge the vertices ℓ^{4j-2} and ℓ^{4j-1} if $\ell \notin c_j$. Additionally, the vertices Y and Z are identified with the vertices c_j^1 and c_j^2 induced by clause c_j . Note that if there are two consecutive switch gadgets belonging to two different clauses c_j and c_{j+1} , then we define that vertex A (merged with the following C) and D (merged with the following B) belong to c_j . To finish the construction, we add arcs $(s_2, C_{last}), (A_1, t_2), (s_1, B_1)$, and (D_{last}, x_1^s) , which are part of the constant gadget. The whole construction can be found in Figure 12.19 in which we symbolize the usage of a switch gadget as depicted in Figure 12.18. The construction as described above is pre-image unique because each vertex and arc is induced by one variable or clause.

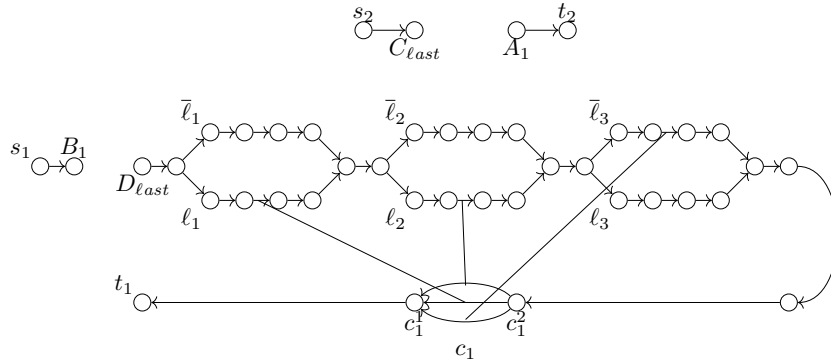


Figure 12.19: Classical reduction of 3SAT to DIRECTED TWO DISJOINT PATH for $\varphi = (\bar{\ell}_1 \wedge \bar{\ell}_2 \wedge \ell_3)$.

There is a correspondence between the assignment of variables and the lobes of the variables and thus the variable gadgets: If on the one hand the path corresponding to literal ℓ_i is part of the solution, then x_i is assigned to false; if on the other hand the path corresponding to literal $\bar{\ell}_i$ is part of the solution, then x_i is assigned to true. Furthermore observe that if the path of ℓ_i is part of the solution, all arcs of $\bar{\ell}_i$ are still unused. Accordingly, while traveling through the clause gadget of clause c_j , which contains literal $\bar{\ell}_i$, these arcs can be used. On the other hand, a clause gadget of clause c_j containing ℓ_i cannot make use of the arcs of the path of ℓ_i .

This reduction is only weakly modular because removing a variable x_i results in a disconnected graph. However, there is a corresponding removal gadget for clauses mitigating this problem.

Thus the removal gadget of a variable is the variable gadget itself together with the clause removal gadgets. In order to deactivate a clause, one can add a path from c_j^1 to c_j^2 of length 5 such that the number of used arcs remains the same. If this new path is used the clause is simulated to be satisfied. A solution in the base scenario is extendable to a full solution because the clause removal gadget only fixes the solution on the literals that are part of the clause.

At last, we describe the solution size function over the arcs. Due to weak modularity, we use a reduction f from an instance (L', C') with $L \subseteq L'$ and $C \subseteq C'$ such that

$$size_f(L, C) = 5 + \frac{2|L'|}{2} + 2|C'| |L'| + \sum_{c \in C'} \left(\frac{11|c|}{2} + \frac{5|c|}{2} + 5 \right).$$

Overall, we include 5 arcs for the constant gadget, which are the arcs (s_1, B_1) , (s_2, C_{last}) , $(c_{|C|}^1, t_1)$, (A_1, t_2) , and $(x_{|X|}^t, c_1^2)$. For each clause, we need to travel the C - A path and the B - D path, which are 11 arcs for each variable in the clause. Furthermore, we have to travel over the W - X path of all switch gadgets for each clause. Accordingly for each clause $c \in C$, we have to include $5|c|/2$ arcs. In order to travel from c_j^1 to c_j^2 for each clause c_j , the Y - Z path has to be used, which are 5 arcs. At last for each variable x_i , we have to use one of the arcs (x_i^s, ℓ_i) , $(x_i^s, \bar{\ell}_i)$ and one of the arcs (ℓ_i, x_i^t) , $(\bar{\ell}_i, x_i^t)$. Additionally, we need to include two arcs per clause to travel through the lobe. This completes the description of the solution size function. Note that the clause removal gadgets were designed such that the number of solution elements remains the same. \square

Lemma 12.21. *2DISJOINT DIRECTED PATH is strongly modular universe gadget reducible to k DISJOINT DIRECTED PATH such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction from k DISJOINT DIRECTED PATH to $k + 1$ DISJOINT DIRECTED PATH for $k \geq 2$ works as follows. The reduction consists only of a constant gadget, which adds an additional path from s_{k+1} to t_{k+1} over the single arc (s_{k+1}, t_{k+1}) . The solution size functions needs to include this additional arc. Because, the rest of the instance remains the same, we have a one-to-one correspondence between all combinatorial elements. Thus, modularity and pre-image uniqueness remain. \square

3-Dimensional Matching

Lemma 12.22. *3SATISFIABILITY is weakly modular universe gadget reducible to 3DIMENSIONAL MATCHING such that the solution properties hold and a solution size function for this reduction exists.*

Proof. A modification of the reduction from 3SATISFIABILITY to 3DIMENSIONAL MATCHING by Garey and Johnson [GJ79] is a universe gadget reduction. For 3SATISFIABILITY, we use the literals as universe elements and the relations $R_{\ell, \bar{\ell}}$, which relates a literal and its negation, $R_{\ell, c}$, which relates the literals with the clauses, and $R_{\bar{\ell}, c}$, which relates the negated literals with the clauses. 3DIMENSIONAL MATCHING consists of a ground set U including all elements of the triples. Additionally, there is a 3-ary relation between the triples $T \subseteq U_1 \times U_2 \times U_3$ with $U_1 \dot{\cup} U_2 \dot{\cup} U_3 = U$ and $|U_1| = |U_2| = |U_3|$. A solution is a perfect matching $M \subseteq T$ of U .

We describe the construction and explain how to divide the elements in variable and clause gadgets. In the original reduction the sets T_i^t , T_i^f , and G of triples were introduced. The semantics are to include the set T_i^t in the solution if variable x_i was set to true and T_i^f if the

variable x_i is set to false. The set G is a garbage collection set that has the task to collect all non-matched elements in the sets T_i^t and T_i^f .

For the modified version, we introduce a variable gadget for the literal pair $\ell_i, \bar{\ell}_i$. Such a variable gadget consists of four triples $(\bar{\ell}_i[0], a_i[0], b_i[0])$ (belonging to the set T_i^t), $(\ell_i[0], a_i[1], b_i[0])$ (belonging to the set T_i^f), $(\bar{\ell}_i[0], g_1[i], g_2[i])$, and $(\ell_i[0], g_1[i], g_2[i])$ (both belonging to the garbage collection set G). For each clause, we introduce the following of triples as clause gadget. We assume that the clauses C are ordered and let $|c|$ denote the number of literals in clause $c \in C$. Further, let γ_i be the number of clauses containing ℓ_i or $\bar{\ell}_i$.

$$\begin{aligned} & (\bar{\ell}_i[j], a_i[j], b_i[j]), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c \\ & (\ell_i[j], a_i[j + 1 \bmod \gamma_i + 1], b_i[j]), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c \\ & (\bar{\ell}_i[j], g_1^c[k], g_2^c[k]), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c, \text{ and } 1 \leq k \leq |c| - 1 \\ & (\ell_i[j], g_1^c[k], g_2^c[k]), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c, \text{ and } 1 \leq k \leq |c| - 1 \\ & (\ell_i[j], s_1^c, s_2^c), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c, \text{ and } \ell_i \in c \\ & (\bar{\ell}_i[j], s_1^c, s_2^c), \text{ if } c \text{ is the } j\text{-th clause with } \ell_i \text{ or } \bar{\ell}_i \in c, \text{ and } \bar{\ell}_i \in c \end{aligned}$$

The element $(\bar{\ell}_i[j], a_i[j], b_i[j])$ is part of the set T_i^t , the element $(\ell_i[j], a_i[j + 1], b_i[j])$ is part of the set T_i^f , and the elements $(\bar{\ell}_i[j], g_1^c[k], g_2^c[k])$ and $(\ell_i[j], g_1^c[k], g_2^c[k])$ are part of the set G . In comparison to the original reduction, we leave out all elements $(\bar{\ell}_i[j], a_i[j], b_i[j])$, and $(\ell_i[j], a_i[j + 1], b_i[j])$ in T_i^t and T_i^f as well as the additional garbage collection element from G if the literal is not part of the clause. For each clause $c_j \in C$, we have added the triples

$$C_j = \{(\ell_i[j], s_1^c, s_2^c) \mid \ell_i \in c_j\}.$$

A triple from C_j is taken into the solution if ℓ_i is able to satisfy the clause c_j . This is only possible if T_i^t and T_i^f are chosen correspondingly.

The variable removal gadget is the variable gadget itself together with the clause removal gadgets. The removal gadget for clause $c_j \in C$ is

$$\bar{C}_j = \{(\ell_i[j], s_1^c, s_2^c), (\bar{\ell}_i[j], s_1^c, s_2^c) \mid \ell_i \in c_j\}.$$

If a clause C_j is unsatisfied, then one of $\ell_i[j]$ and $\bar{\ell}_i[j]$ is not matched for some $\ell_i \in C_j$ because G is only able to match at most $k - 1$ many $\ell_i[j]$ for a clause of size k , while $2k$ many $\ell_i[j]$ are introduced and k being matched by the elements from T_i^t or T_i^f . In other words, the set \bar{C}_j simulates that C_j is satisfied and also includes the element $\ell_i[j]$, which cannot be matched by G . In conclusion, the solution size stays the same if a clause is removed. Thus we get

$$size_f(L, C) = |L'| + \sum_{c \in C'} 2|c|,$$

and the reduction is modular by the given one-to-many correspondence of the literals and clauses on the one side and their gadgets on the other side. A solution in the base scenario is extendable to a full solution because the clause removal gadget only fixes the solution on the literals that are part of the clause. \square

Lemma 12.23. *3DIMENSIONAL MATCHING is strongly modular universe gadget reducible to EXACT COVER BY 3-SETS such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The reduction from 3DIMENSIONAL MATCHING to EXACT COVER BY 3-SETS by Garey and Johnson [GJ79] is a universe gadget reduction. Because EXACT COVER BY 3-SETS is a generalization of 3DIMENSIONAL MATCHING, the 3DIMENSIONAL MATCHING instance is just reinterpreted as an instance of EXACT COVER BY 3-SETS. This yields a direct one-to-one correspondence between the 3-tuples and 3-sets. Thus, the solution size function remains and all necessary properties still hold. \square

Coloring (Partition Problems)

Lemma 12.24. 3SATISFIABILITY is strongly modular universe gadget reducible to 3COLORING such that the solution properties hold and a solution size function for this reduction exists.

Proof. The reduction from 3SATISFIABILITY to COLORING by Garey et al. [GJS76] is a universe gadget reduction. COLORING has the vertices of the graph V as universe elements and the edges E as relation over the universe elements. We therefore have the mappings $f_{const,V}, f_{const,E}, f_{L,V}, f_{L,E}, f_{C,V}, f_{C,E}$.

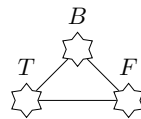


Figure 12.20: Constant Gadget for the reduction. The corresponding mappings are $f_{const,V} : \emptyset \mapsto \{B, F, T\}$ and $f_{const,E} : \emptyset \mapsto \{\{B, F\}, \{B, T\}, \{F, T\}\}$

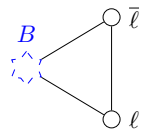


Figure 12.21: Variable Gadget representing literals $l, \bar{l} \in L$. The corresponding mappings are $f_{L,V} : (l, \bar{l}) \mapsto \{v_l, v_{\bar{l}}\}$ and $f_{L,E} : (l, \bar{l}) \mapsto \{\{v_l, v_{\bar{l}}\}, \{v_l, B\}, \{v_{\bar{l}}, B\}\}$

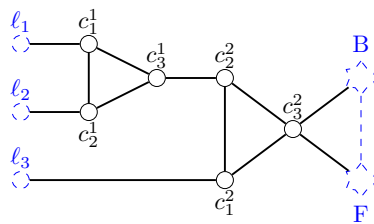


Figure 12.22: The clause gadget representing clause $c \in C$. The corresponding mappings are $f_{C,V} : c \mapsto \{c_1^1, c_2^1, c_3^1, c_1^2, c_2^2, c_3^2\}$ and $f_{C,E} : c \mapsto \{\{c_1^1, c_2^1\}, \{c_1^1, c_3^1\}, \{c_2^1, c_3^1\}, \{c_1^2, c_2^2\}, \{c_1^2, c_3^2\}, \{c_2^2, c_3^2\}, \{l_1, c_1^1\}, \{l_2, c_2^1\}, \{l_3, c_3^1\}, \{c_1^1, c_2^1\}, \{c_1^2, c_2^2\}, \{c_3^2, B\}, \{c_3^2, F\}\}$

The constant gadget is a 3-clique, see Figure 12.20. W.l.o.g we assume that T is always in the first set, F is always in the second set, and B is always in the third set of the partition. For the

literals, the mapping $f_{L,V}$ maps a literal $\ell \in L$ to two vertices. The mapping $f_{L,E}$, on the other hand, maps a literal $\ell \in L$ to three edges connecting the two vertices ℓ and $\bar{\ell}$ and vertex B of the constant gadget, which is generated by the constant mapping f_{const} , visualized in Figure 12.21. At last, we have the clause gadget. The mapping $f_{C,V}$ maps the clause to six vertices, which are depicted as circles in Figure 12.22. The mapping $f_{C,E}$ maps the clause to the edges as shown as solid edges in Figure 12.22. The dashed vertices are part of different literal gadgets and the vertices F and B and the three dashed edges are from the constant gadget.

A vertex v_{ℓ_i} is assigned the color T if and only if the variable x_i is assigned true. One can verify that if one of the literal vertices v_{ℓ_1} , v_{ℓ_2} , and v_{ℓ_3} is assigned color T , the vertex c_3^2 is assigned color T . Accordingly this clause gadget does not violate the coloring constraint. If on the other hand, v_{ℓ_1} , v_{ℓ_2} , and v_{ℓ_3} are assigned color F , then c_3^2 has to be assigned F violating the coloring constraint.

Overall, all vertices and edges are either generated by the constant function or are attributable to exactly one literal or one clause of the 3SATISFIABILITY-instance. Furthermore, deleting a variable gadget or a clause gadget results in the correct reduction such that we have strong modularity. Thus, the reduction fulfills the universal gadget reduction properties. The solution size function includes all vertices in one of the partitions (the colors). Thus, $size_f(L, C) = 2|L| + 6|C| + 3$ because every variable introduces two vertices and every clause introduces 6 vertices. The 3 additional vertices result from the constant gadget. \square

Lemma 12.25. *3COLORING is strongly modular universe gadget reducible to kCOLORING such that the solution properties hold and a solution size function for this reduction exists.*

Proof. The graph $G = (V, E)$ for kCOLORING remains, but a vertex v_{new} is added and connected to all existing vertices V . Thus, v_{new} needs to have a different color than all existing vertices in V . This is a universe gadget reduction because the vertex v_{new} is a constant gadget and every edge to v_{new} is part of the vertex gadget of v together with v itself. The pre-image uniqueness and the modularity results from the one-to-two correspondence of vertex v and the vertex gadget consisting of v and the edge $\{v, v_{new}\}$. The solution size function needs to include the additional vertex v_{new} , thus 1 is added. \square

Lemma 12.26. *kCOLORING is strongly modular universe gadget reducible to CLIQUE COVER such that the solution properties hold and a solution size function for this reduction exists.*

Proof. This reduction is analogous to the reduction from INDEPENDENT SET to CLIQUE due to the fact that a coloring of a graph is a partition into independent sets while a clique cover is a partitions into cliques. \square

Chapter 13

Online Graph Problems

13.1 Introduction

Online computation is an intuitive concept to model real time computation where the full instance is not known beforehand. In this setting, the instance is revealed piecewise to the online algorithm and each time a piece of information is revealed, an irrevocable decision by the online algorithm is required. To analyze the worst-case performance of an algorithm solving an online problem, a malicious adversary is assumed.

The adversary constructs the instance while the online algorithm has to react and compute a solution. This setting is highly asymmetric in favor of the adversary. Thus, for most decision problems, the adversary is able to abuse the imbalance of power to prevent the online algorithm from finding a solution that is close to the optimal one. To overcome the imbalance, there are different extensions of the online setting in which the online algorithm is equipped with some form of a priori knowledge about the instance. In this work, we analyze the influence of knowing an isomorphic copy of the input instance, which is also called *unlabeled map*. With the unlabeled map, the algorithm is able to recognize unique structures while the online instance is revealed – like a vertex with unique degree – but it cannot distinguish isomorphic vertices or subgraphs.

The relation between the online algorithm and the adversary corresponds to players in an asymmetric two-player game, in which the algorithm wants to maximize its performance and the adversary’s goal is to minimize it. The unlabeled map can be considered as the game board. One turn of the game consists of a move by the adversary followed by a move of the online algorithm. In such moves, the adversary reveals a vertex together with its neighbors and the online algorithm has to irrevocably decide whether to include this vertex in the solution or not. The problem is to evaluate whether the online algorithm has a *winning strategy*, that is, it is able to compute a solution of size smaller/greater or equal to the desired solution size k , for all possible adversary strategies.

Papadimitriou and Yannakakis make use of the connection between games and online algorithms for analyzing the Canadian traveler problem in [PY91], which is an online problem where the task is to compute a shortest s - t -path in an a priori known graph in which certain edges can be removed by the adversary. They showed that the computational problem of devising a strategy that achieves a certain competitive ratio is PSPACE-complete by giving a reduction from TRUE QUANTIFIED BOOLEAN FORMULA, short TQBF.

Independently, Halldórsson [Hal00] introduced the problems online coloring and online independent set on a priori known graphs, which is equivalent to having an unlabeled map. He studies how the competitive ratios improve compared to the model when the graph is a priori not known. Based on these results, Halldórsson et al. [HIMT02] continued the work on the online

independent set problem without a priori knowing the graph. These results are then applied by Boyar et al. [BFKM17] to derive a lower bound for the advice complexity of the online independent set problem. Furthermore, they introduce the class of asymmetric online covering problems (AOC) containing ONLINE VERTEX COVER, ONLINE INDEPENDENT SET, ONLINE DOMINATING SET and others. Boyar et al. [BK18] analyze the complexity of the corresponding graph property versions, namely the online vertex cover number, online independence number and online domination number, by showing their NP-hardness.

Moreover based on the work by Halldórsson [Hal00], Kudahl [Kud15] shows PSPACE-completeness of the decision problem ONLINE CHROMATIC NUMBER WITH PRECOLORING on an a priori known graph, which asks whether some online algorithm is able to color G with at most k colors for every possible order in which G is presented while having a precolored part in G . This approach is then improved by Böhm and Veselý [BV18] by showing that ONLINE CHROMATIC NUMBER is PSPACE-complete by giving a reduction from TQBF.

Our Contribution. We analyze the computational complexity of online graph problems where the solution is a subset of the vertices. Similar to the problem ONLINE CHROMATIC NUMBER, we equip the online algorithm with an unlabeled map in order to apply and formalize the ideas of Böhm and Veselý. We call these problems *online vertex subset games* due to their relation to two-player games. While symmetrical combinatorial two-player games are typically PSPACE-complete [FG87], this principle does not apply to our asymmetrical setting. We are still able to prove PSPACE-completeness for the online vertex subset games based on VERTEX COVER, INDEPENDENT SET, CLIQUE, DOMINATING SET and FEEDBACK VERTEX SET by designing reductions such that the adversary’s optimal strategy corresponds to the optimal strategy of the \forall -player in TQBF.

In order to derive reductions from TQBF to online vertex subset games, we identify properties describing the revelation or concealment of information to correctly simulate the \forall - and \exists -decisions as well as the evaluation of the quantified Boolean formula in the online vertex subset game. This simulation is modeled by disjoint and modular gadgets, which form a so-called gadget reduction – similar to already known reductions between NP-complete problems. Different forms of gadget reductions are described by Agrawal et al. [AAI⁺01] who formalize AC^0 -gadget-reductions in the context of NP-completeness and by Trevisan et al. [TSSW96] who describe gadgets in reductions that are formalized as linear programs. By formalizing gadgets capturing the above mentioned properties, we provide a framework to derive reductions for other online vertex subset games, which are based on problems that are gadget-reducible from 3SATISFIABILITY.

13.2 Online Vertex Subset Game Framework

The goal of this chapter is to introduce a reduction framework for online vertex subset games that are based on NP-complete graph problems. For this, we introduce a general definition for online vertex subset games and show that these problems are in PSPACE if the nominal problem is in NP. As main contribution of the chapter, we further present a gadget reduction framework with that we are able to “upgrade” existing gadget reductions for NP-hard vertex subset problems to PSPACE-hardness reductions. We start by defining the reveal model of the underlying online model and then proceed with the definition of online vertex subset games.

Neighborhood Reveal Model. Each request of the adversary in the online problem reveals information about the instance to the online algorithm. The amount of information in each step is based on the reveal model. For an online problem with an unlabeled map, the subgraph that arrives in one request is called *revelation subgraph*.

The neighborhood reveal model, which we use in this chapter, was introduced by Harutyunyan et. al. [HPR21]. Within that model, the online algorithm gains information about the complete neighborhood of the requested vertex. Nevertheless, the online algorithm has to make a decision on the current requested vertex only but not on the revealed neighborhood vertices. All revealed but not yet requested neighborhood vertices have to be requested in the process of the online problem such that a decision can be made upon them. We denote the closed neighborhood of v with $N[v]$, that is, the set of v and all vertices adjacent to v .

Definition 13.1 (Neighborhood Reveal Model). *The neighborhood reveal model is defined by an ordering of graphs $(V_i, E_i)_{i \leq |V|}$. The reveal order of the adversary is defined by $adv \in S_{|V|}$, where $S_{|V|}$ is the symmetric group of size $|V|$. The graph G_i is defined by*

$$\begin{aligned} V_0 &= E_0 = \emptyset, \\ V_i &= V_{i-1} \cup N[v_{adv(i)}], && \text{for } 0 < i \leq |V| \\ E_i &= E_{i-1} \cup \{(v_{adv(i)}, w) \in E \mid w \in V_i\}, && \text{for } 0 < i \leq |V|. \end{aligned}$$

The revelation subgraph G' in the neighborhood reveal model is the subgraph of G_i defined by $G' = (V', E')$ with $V' = N[v_{adv(i)}]$ and $E' = \{(v_{adv(i)}, w) \in E\}$. The online algorithm has to decide whether $v_{adv(i)}$ is in the solution or not.

Online Vertex Subset Games. Throughout the chapter, we consider a special class of combinatorial graph problems. The question is to find a vertex subset, where the size should be either smaller or equals, for minimization problems, or greater or equals, for maximization problems, some k , which is part of the input. Thereby, the vertex set needs to fulfill some constraints based on the specific problem. We call these problems vertex subset problems. Well-known problems like VERTEX COVER, INDEPENDENT SET, CLIQUE, DOMINATING SET and FEEDBACK VERTEX SET are among them. We denote the online version with a map of a vertex subset problem P^{VS} with P_o^{VS} and define them as follows.

Definition 13.2 (Online Vertex Subset Game). *An online vertex subset game P_o^{VS} has a graph G and a $k \in \mathbb{N}$ as input. The question is, whether the online algorithm is able to find a vertex set of size smaller (resp. greater) or equals k , which fulfills the constraints of P^{VS} for all strategies of the adversary. Thereby, the online algorithm has access to an isomorphic copy of G and the adversary reveals the vertices according to the neighborhood reveal model.*

13.3 Gadget Reductions

Gadget reductions are a concept to reduce combinatorial problems in a modular and structured way. For the context of the chapter, we define gadget reductions from 3SATISFIABILITY to vertex subset problems. The 3SATISFIABILITY instances $\varphi = (L, C)$ are defined by their literals L and their clauses C . We use a literal vertex v_ℓ for all $\ell \in L$ to represent a literal in the graph. There are implicit relations over the literals besides the explicit relation C , in that the reduction may be decomposed. For example, the relation between a literal and its negation, which is usually implicitly used to build up a *variable gadget*. These variable gadgets are connected by graph substructures that assemble the clauses as *clause gadgets*.

Definition 13.3 (Gadget Reduction from 3SATISFIABILITY to Vertex Subset Problems). *A gadget reduction $R_{gadget}(P^{VS})$ from a 3SATISFIABILITY formula $\varphi = (L, C)$ to a vertex subset problem with graph $G_\varphi = (V, E)$ is a tuple containing functions from the literal set and all relations of the 3SATISFIABILITY formula to the vertex set and all relations of the vertex subset problem. In the*

following, we denote the gadget based on element x to be $G_x := (V_x, E_x)$ with V_x being a set of vertices and E_x a set of edges, where the edges are potentially incident to vertices of a different gadget.

The literal set of a 3SATISFIABILITY formula $\ell_1, \ell_2, \dots, \ell_{|L|-1}, \ell_{|L|}$ is mapped to the vertex set of the graph problem. Thereby, each literal is mapped to exactly one vertex:

$$R_{\text{gadget}}^{L \rightarrow V}(P^{VS}) : L \rightarrow V, \ell \mapsto G_\ell$$

The following relations on the literals are mapped as well.

- (1) Literal - Negated Literal: $R_{\text{gadget}}^{L, \bar{L}}(P^{VS}) : R(L, \bar{L}) \rightarrow (V, E), (\ell, \bar{\ell}) \mapsto G_{\ell, \bar{\ell}}$
- (2) Clause: $R_{\text{gadget}}^C(P^{VS}) : R(C) \rightarrow (V, E), C_j \mapsto G_{C_j}$
- (3) Literal - Clause: $R_{\text{gadget}}^{L, C}(P^{VS}) : R(L, C) \rightarrow (V, E), (\ell, C_j) \mapsto G_{\ell, C_j}$
- (4) Negated Literal - Clause: $R_{\text{gadget}}^{\bar{L}, C}(P^{VS}) : R(\bar{L}, C) \rightarrow (V, E), (\bar{\ell}, C_j) \mapsto G_{\bar{\ell}, C_j}$

Additionally, the following mapping allows for constant parts that do not change depending on the instance: $R_{\text{gadget}}^{\text{const}}(P^{VS}) : \emptyset \rightarrow (V, E), \emptyset \mapsto G_{\text{const}}$. Thereby, the vertices and edges of all gadgets are pairwise disjoint.

We use the more coarse grained view of *variable gadgets* as well. These combine the mappings $R_{\text{gadget}}^{L \rightarrow V}$ and $R_{\text{gadget}}^{L, \bar{L}}$ to *variable gadgets* R_{gadget}^X , and $R_{\text{gadget}}^{L, C}$ and $R_{\text{gadget}}^{\bar{L}, C}$ to *clause gadgets* $R_{\text{gadget}}^{X, C}$, where X is the set of n variables.

The important function of the variable gadget is to ensure that only one of the literals of $\ell, \bar{\ell} \in L$ is chosen. On the other hand, the function of the clause gadget is to ensure together with the constraints of the vertex subset problem P^{VS} that the solution encoded on the literals fulfill the 3SATISFIABILITY-formula if and only if the literals induce a correct solution. These functionalities are utilized in the correctness proof of the reduction by identifying the logical dependencies between the literal vertices v_ℓ for $\ell \in L$ and all other vertices based on the graph and the constraints of P^{VS} together with combinatorial arguments on the solution size. We denote these logical dependencies as *solution dependencies* as they are logical dependencies on the solutions of P^{VS} . Due to the asymmetric nature of the online problems, the adversary can reveal a solution dependent vertex before revealing the corresponding literal vertex. Thus, a decision on the solution dependent vertex is implicitly also a decision on the literal vertex, although it has not been revealed. We address this specific problem later in the description of the framework.

Definition 13.4 (Solution dependent vertices). *Given a gadget reduction, the following vertices of the reduction graph are solution dependent:*

- (1) All literal vertices are solution dependent on their respective variable.
- (2) For a literal ℓ (resp. its negation $\bar{\ell}$), we denote the set of vertices that need to be part of the solution if v_ℓ (resp. $v_{\bar{\ell}}$) is part of the solution with V_ℓ (resp. $V_{\bar{\ell}}$). Then the vertices, that are in one but not both of these sets, i.e. $V_\ell \Delta V_{\bar{\ell}}$, are solution dependent on the corresponding variable.

All vertices that are not solution dependent on any variable are called *solution independent*.

In order to understand solution dependencies thoroughly, we describe the solution dependencies in a folklore reduction from 3SATISFIABILITY to DOMINATING SET, which we also use later to show the PSPACE-completeness of the corresponding online game version of DOMINATING SET.

Example 13.1. Let φ be the 3SATISFIABILITY-formula, let X be set of n variables and let C be the set of m clauses of φ . We construct the following graph $G_\varphi = (V, E)$, which is also depicted in Figure 13.1: For each variable x_i , $i \in \{1, \dots, n\}$, we introduce a triangle as its variable gadget. Two of the vertices represent the literals, which we call literal vertices. The third vertex ensures that always one vertex of the variable gadget has to be chosen. For each clause C_j , $j \in \{1, \dots, m\}$, we construct a clause gadget consisting of the single vertex. Each clause vertex is connected to the literal vertices of the literals the clause contains. The size of the dominating set that should be found in G_φ is set to $k = n$.

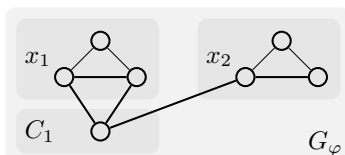


Figure 13.1: The reduction graph for the reduction from 3SATISFIABILITY to DOMINATING SET for instance $\varphi = (x_1 \vee \bar{x}_1 \vee x_2)$.

The logical dependencies in G_φ are of the type if the literal vertex representing the literal ℓ is not contained in a solution, then the literal vertex representing $\bar{\ell}$ must be contained. Furthermore, for each clause containing ℓ , one of the remaining literal vertices must be contained since a solution of size k never contains clause vertices. As any truth assignment assigns each literal either true or false, the third vertex of the variable gadget is solution independent. Therefore, only the literal vertices in G_φ are solution dependent.

13.4 A Reduction Framework for Online Vertex Subset Games

In this section, we present a general framework for reducing TQBF GAME to an arbitrary online vertex subset game P_o^{VS} , which proves its PSPACE-hardness. The TQBF GAME is played on a fully quantified Boolean formula, where one player decides the \exists -variables and the other decides the \forall -variables, in the order they are quantified. We assume that the TQBF GAME instance is in CNF with clauses of at most three literals.

Before we describe the reduction, we prove that the online game version of each vertex covering graph problem in NP is in PSPACE.

Theorem 13.1. *If P^{VS} is in NP, then P_o^{VS} is in PSPACE.*

Proof. First of all, we are able to encode the instance graph in linear space of the input. Further, we are able to encode polynomial verifiable solutions, which are subsets of vertices, in at most linear space because the base problem P^{VS} is in NP. Thus, all states of the game, which are a partial instance together with a partial solution, are encodable in linear space. Consequently, we are able to store the state of each move of both players in linear space. Lastly, the number of moves are bounded in the number of vertices which is also linear in the input. After a complete move sequence, the game state is verifiable in polynomial time (and thus polynomial space) because P^{VS} is in NP. Consequently, the problem P_o^{VS} is in PSPACE. \square

This framework uses an (existing) gadget reduction of the vertex subset problem P^{VS} from 3SATISFIABILITY and extends it in order to give the online algorithm the ability to recognize the current revealed vertex. Due to the quantification of variables, we call the variable gadget of a \forall -variable a \forall -gadget (resp. \exists -gadget for an \exists -variable). Based on this, the online algorithm can

use a one-to-one correspondence between the solution of the TQBF GAME instance and the P_o^{VS} instance. The one-to-one correspondence between the \forall -variables and the \forall -gadgets is ensured by the knowledge of the adversary about the deterministic online algorithm. It simulates the response of the algorithm on the \forall -gadget.

Extension Gadgets. We extend the reduction graph G_φ of the offline problem with gadgets to a reduction graph for the online problem. These gadgets extend G_φ by connecting to a subset of its vertices. We denote these gadgets G_{ext} as *extension gadgets*.

Definition 13.5 (Graph Extension). A graph extension of a graph $G = (V, E)$ by an extension gadget $G_{ext} = (V_{ext}, E_{ext}, E_{con})$ with the set of connecting edges $E_{con} \subseteq V \times V_{ext}$ is defined as $H = G \circ G_{ext}$, where

$$\begin{aligned} V(H) &= V \cup V_{ext}, \\ E(H) &= E \cup E_{ext} \cup E_{con}. \end{aligned}$$

We further define $G \circ_{i \in I} G_{ext}^i := (\dots ((G \circ G_{ext}^{i_1}) \circ G_{ext}^{i_2}) \circ \dots) \circ G_{ext}^{i_{|I|}}$.

We also need the notion of *self-contained* gadgets. These do not influence the one-to-one correspondence between solutions of the online vertex subset game P_o^{VS} and TQBF GAME. In other words, optimal solutions on the graph and the extension gadget can be disjointly merged to obtain an optimal solution on the extended graph. Due to this independence, we are able to provide local information to the online algorithm via the map without changing the underlying formula. An example for self-contained extension gadgets is provided in Figure 13.2. Note that, it can occur that self-containment depends on the extended graph.

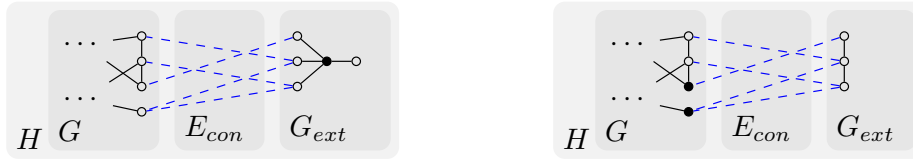


Figure 13.2: On the left, there is an example for an extension gadget that is self-contained w.r.t. the dominating set problem: No matter the solution on G , at least one vertex of G_{ext} has to be chosen. Additionally, choosing the black vertex of G_{ext} dominates all vertices attached to G , and thus any solution on G remains valid. On the right, there is an example for an extension gadget that is not self-contained w.r.t. the dominating set problem: If the solution on G contains the black vertices, it is also a solution for H , but the optimal solution on G_{ext} contains one vertex.

For our reduction framework, we introduce three types of self-contained extension gadgets: fake clause gadgets, dependency reveal gadgets and ID gadgets. The goal of these gadgets is that it is optimal for the adversary to reveal variables in the order of quantification, and that the online algorithm is able to assign the value of the \exists -variables, while the adversary is able to assign the value of the \forall -variables.

Fake Clause Gadgets. The number of occurrences of a certain literal in clauses or different literals occurring together in one clause is information that may allow the online algorithm to distinguish the literals of some \forall -variables, allowing the online algorithm to decide the assignment instead of the adversary. To avoid this information leak, we add gadgets for all possible non-existing clauses to the reduction graph. A *fake clause gadget* is only detectable if and only if a vertex, which is part of that clause gadget, is revealed by the adversary. The gadget needs to

be self-contained such that the one-to-one correspondence between the solutions of the P_o^{VS} and TQBF GAME is not affected.

Definition 13.6 (Fake Clause Gadget). *A fake clause gadget $G_{fc}(C'_j)$ for a non-existing clause $C'_j \notin C$ is an extension gadget that is self-contained. The fake clause gadgets are connected to the variable gadgets like the clause gadgets are to the variable gadgets according to the original gadget reduction, see Definition 13.3.*

All fake clause gadgets are pairwise disjoint. Let G_φ be the gadget reduction graph and

$$G'_\varphi := G_\varphi \bigcirc_{C'_j \notin C} G_{fc}(C'_j).$$

After adding fake clause gadgets for all clauses $C'_j \notin C$ to G_φ , the revelation subgraphs of all vertices $v \in V(G'_\varphi)$, which are part of a literal gadget, are pairwise isomorphic.

Dependency Reveal Gadgets. The two functions of the dependency reveal gadgets are that the adversary chooses the reveal order to be the order of quantification and the online algorithm knows the decision on the \forall -variables after the decision is made by the adversary. If the adversary deviates from the quantification order, the \forall -decision degenerates to an \exists -decision for the online algorithm. On the other hand, since the adversary forces the online algorithm to blindly choose the truth value of a \forall -quantified variable, the online algorithm does not know the chosen truth value. Thus, we need to reveal the truth value to the online algorithm whenever a solution dependent vertex of the next variables is revealed.

Definition 13.7 (Dependency Reveal Gadget). *A dependency reveal gadget $G_{dr}(x_i)$ for \forall -variable x_i is an extension gadget that is self-contained with the property: Let $\ell, \bar{\ell}$ be the literals of x_i . If a solution dependent vertex of x_j with $j \geq i$ is revealed to the online algorithm, the online algorithm is able to uniquely identify the vertices v_ℓ and $v_{\bar{\ell}}$.*

ID Gadgets. At last, the online algorithm needs information on the currently revealed vertex to identify it with the help of the map. For this, we introduce ID gadgets, which make the revelation subgraph of vertices distinguishable to a certain extent. Thus, the online algorithm is able to correctly encode the TQBF solution into the solution of the vertex subset game.

Definition 13.8 (ID Gadget). *An identification gadget $G_{id}(v)$ is a self-contained extension gadget connected to v such that the revelation subgraph of v is isomorphic to revelation subgraphs of vertices within a distinct vertex set $V' \subseteq V$.*

Note that for any reduction, where two different literal vertices cannot be connected to the same vertex in a clause gadget, the fake clause gadget is technically not needed and its task can also be accomplished by ID gadgets. However, to keep the framework more general, we include a dedicated fake clause gadget.

The General Reduction for Online Vertex Subset Games. With the gadget schemes defined above, we are able to construct a gadget reduction from TQBF GAME to P_o^{VS} . The idea of the reduction is to construct the optimal game strategy for the online algorithm to compute the solution to the TQBF GAME formula. Furthermore, encoding the solution to the TQBF GAME into the P_o^{VS} instance is a winning strategy by using the equivalence of the \exists - and \forall -gadgets to the \exists - and \forall -variables. At last, there is a one-to-one correspondence between the reduction graph solution of P^{VS} and the 3SATISFIABILITY-solution.

A gadget reduction from 3SATISFIABILITY to vertex subset problem P^{VS} can be extended such that P_o^{VS} is reducible from TQBF GAME as follows. Recall that G_φ is the gadget reduction graph of a fixed but arbitrary instance of P^{VS} .

- (1) Add fake clause gadgets for all clauses that are not in the TQBF GAME instance

$$G'_\varphi = G_\varphi \bigcirc_{c' \notin C} G_{fc}(c') .$$

- (2) Add dependency reveal gadgets for all \forall -variables x

$$G''_\varphi = G'_\varphi \bigcirc_{\substack{x \in X \\ x \text{ is } \forall}} G_{dr}(x) .$$

- (3) Add ID gadgets to all vertices

$$G'''_\varphi = G''_\varphi \bigcirc_{v \in V(G''_\varphi)} G_{id}(v) .$$

Then, if all gadgets can be constructed in polynomial time, G'''_φ is the corresponding reduction graph of P_o^{VS} . The gadget reduction also implies the following gadget properties, which individually have to be proven for a specific problem.

- (1) The fake clause gadgets are self-contained.
- (2) The dependency reveal gadgets are self-contained.
- (3) The ID gadgets are self-contained.
- (4) In G'''_φ , each solution dependent vertex which is not in a literal gadget of a \forall -variable has a unique revelation subgraph.
- (5) In G'''_φ , the two literal vertices of a \forall -variable have the same revelation subgraph, but different from vertices of any other gadget.
- (6) In G'''_φ , each vertex that is solution independent or part of an extension gadget has a revelation subgraph that allows for an optimal decision.

From the above construction, the following Lemmas 13.1 to 13.3, are fulfilled such that P_o^{VS} is proven to be PSPACE-hard in the following Theorem 13.2.

Lemma 13.1. *In the construction of the reduction, there is a one-to-one correspondence between the solution of the problem P_o^{VS} and TQBF GAME, if there is a one-to-one correspondence between the solutions in the gadget reduction from P^{VS} and 3SATISFIABILITY. The equivalence is computable in PTIME.*

Proof. The one-to-one-correspondence is preserved by the definition of self-contained extension gadgets. All graph extensions are based on self-contained gadgets. Thus, the original solution is preserved and only complemented by the disjoint partial solution on all extension gadgets.

The ID gadgets ensure that the optimal decision for vertices of the same degree is unique (by Gadget Property 4-6), except for literal vertices of \forall -variables. Thus, a one-to-one correspondence between vertices of the map and the actual online game instance is easy to find by the online algorithm. Therefore, the online algorithm is able to decide whether to put a vertex in the solution or not by vertex degree for all vertices, except for literal vertices of \forall -variables. On the other hand, the adversary can decide the assignment of \forall -variables, as it is able to simulate the online algorithm and predict its decision. \square

In the following, we show that the adversary has to reveal one literal vertex of each variable gadget before revealing vertices of other gadgets (except ID gadgets). Furthermore, the adversary has to adhere to the quantification order of the variables when revealing the first literal vertices of each gadget. If the adversary deviates from this strategy, it may allow the online algorithm to decide the truth assignment of \forall -variables. This may allow the algorithm to win a game based on an unsatisfiable formula. Thus, an optimal adversary strategy always follows the quantification order.

Lemma 13.2. *Every optimal game strategy for the adversary adheres to the reveal ordering*

$$G_{\ell_1} \text{ or } G_{\bar{\ell}_1} < G_{\ell_2} \text{ or } G_{\bar{\ell}_2} < \dots < G_{\ell_n} \text{ or } G_{\bar{\ell}_n}, \quad (13.1)$$

$$G_\ell \text{ or } G_{\bar{\ell}} < G_c(C_j), \quad \text{for all } \ell \in C_j \in C, \quad (13.2)$$

$$G_\ell \text{ or } G_{\bar{\ell}} < G_{fc}(C'_j), \quad \text{for all } \ell \in C'_j \notin C, \quad (13.3)$$

$$G_\ell \text{ or } G_{\bar{\ell}} < G_{dr}(x), \quad \text{for all } x \in X. \quad (13.4)$$

Proof. We prove each proposition one after another.

- (1) Assume a vertex of the variable gadget of x_j is revealed before any vertex of the variable gadget of x_i is revealed for $i < j$. The following cases may apply:
 - (1.1) x_i and x_j are \exists -variables
Then, a \forall -variable x_k exists, $i < k < j$, for which the dependency reveal gadget is revealed before the vertices of its variable gadget are revealed. Therefore, the variable x_k degenerates to an \exists -variable.
 - (1.2) x_i is an \exists -variable and x_j is an \forall -variable
Case 1: $j > i + 1$ Then, a \forall -variable x_k exists, $i < i + 1 \leq k < j$, for which the dependency reveal gadget is revealed before the vertices of its variable gadget are revealed. Therefore, the variable x_k degenerates to an \exists -variable.
Case 2: $j = i + 1$ Then, the decision on x_j happens before x_i . However, the online algorithm is not able to detect which decision took place. Consequently, it has no additional information for the \exists -variable x_i . This does not change the game at all.
 - (1.3) x_i is an \forall -variable and x_j is an \exists -variable
Then, the dependency reveal gadget for x_j is revealed and x_j degenerates to an \exists -variable.
 - (1.4) x_i and x_j are \forall -variables
Then, the dependency reveal gadget for x_j is revealed and x_j degenerates to an \exists -variable.
- (2) By revealing a clause gadget before the corresponding literal gadgets, it is revealed which literals are in the clause and whether the clause is a fake clause or not. Thus, it is dominant to reveal that information after revealing the literals.
- (3) By revealing a fake clause gadget before the corresponding literal gadgets, it is revealed which literals are in the fake clause and whether the fake clause is a fake clause or not. Thus, it is dominant to reveal that information after revealing the literals gadget.
- (4) The dependency reveal gadgets connected to a literal gadget reveal the information which literal gadget represents the negated literal. Thus, the \forall -variable degenerates to an \exists -variable.

The degeneration of a \forall -variable to an \exists -variable gives the online algorithm the possibility to satisfy a possibly unsatisfiable formula. \square

Lemma 13.3. *The vertex assignments of an \exists -variable gadget (resp. \forall -variable gadget) are equivalent to the decision of the \exists -player (resp. \forall -player) on an \exists -quantifier (resp. \forall -quantifier) in the TQBF GAME. The equivalence is computable in PTIME.*

Proof. By the dominating strategy of the adversary described in Lemma 13.2, for all $i, 1 \leq i \leq n$, the adversary reveals all vertices of the variable gadget of variable x_i . Thereby, the online algorithm has to take a decision after each revealed vertex.

- (\exists) Due to Gadget Properties 4 and 6, the online algorithm is able to detect which exact vertex of the variable gadget is revealed. Thus, the online algorithm is able to decide which vertices to take into the solution to encode both of the truth values of the variable into the solution. This implies that the online algorithm takes the decision on the variable as in the TQBF GAME.
- (\forall) Because of Lemma 13.2, the adversary will reveal the literal gadgets first. The literal gadgets of different variables may only be connected by a path of length ≥ 2 , if there is a connection via a clause gadget or dependency reveal gadget. These do not reveal information as every possible clause is covered either by a clause gadget or a fake clause, but it is not revealed whether the connection is established by clause or fake clause gadget unless a vertex of a clause gadget or fake clause gadget is revealed. The dependency reveal gadget reveals only additional information if a reveal ordering which does not correspond to Lemma 13.2 is used. Due to Gadget Property 5, the online algorithm is not able to detect whether a vertex that encodes an assignment to true or false is revealed over the degree. Thus, the online algorithm is not able to detect which literal gadget resembles the true or false assignment.

Therefore, a reveal ordering of the variable gadgets of the \forall -variables exists that forces every fixed deterministic online algorithm to choose the options that prevent the online algorithm from winning if and only if the TQBF GAME formula is unsatisfiable.

The computation is in PTIME because only the degree of the vertex needs to be checked. \square

Therefore, the solutions to the formula in the TQBF GAME and the solutions to the online vertex subset game are equivalent. Thus, the reduction graph G_φ''' is a valid reduction from TQBF GAME because the one-to-one correspondence between solutions is preserved, which concludes the proof of Theorem 13.2. At last, the online algorithm is able to win the game if and only if the TQBF GAME is winnable.

Theorem 13.2. *If P^{VS} is gadget reducible from 3SATISFIABILITY and Gadget Properties 1 to 6 hold, then P_o^{VS} is PSPACE-complete.*

13.5 Applying the Framework to Vertex Cover

In this section, we use our reduction framework to show that the online vertex subset game based on the VERTEX COVER problem, the ONLINE VERTEX COVER GAME, is PSPACE-complete. VERTEX COVER was originally shown to be NP-complete by Karp [Kar72] with a reduction from CLIQUE. However, since our reduction framework extends reductions from 3SATISFIABILITY, we use an alternative reduction from Garey and Johnson [GJ79], which we modify accordingly.

For this, let φ be the 3SATISFIABILITY-formula, let X be the set of n variables and let C be the set of m clauses of φ . We construct a graph $G_\varphi = (V, E)$: For each variable x_i , we introduce a variable gadget consisting of two vertices, connected by an edge. One of these vertices represents the positive literal, while the other represents the negative literal. Thus we refer to these vertices as literal vertices. For each clause C_j , $j \in \{1, \dots, m\}$, we construct a clause gadget, which is a triangle of vertices, where each vertex represents one of the literals in C_j . Finally, each vertex of a clause is connected to the literal it represents. An example of this construction is shown in Figure 13.3.

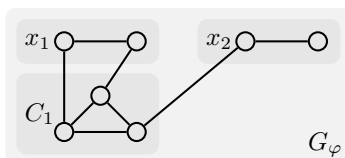


Figure 13.3: The reduction graph for the reduction from 3SATISFIABILITY to VERTEX COVER for instance $\varphi = (x_1 \vee \bar{x}_1 \vee x_2)$.

The dependencies in G_φ are of the type that if a literal vertex is not contained in a solution, then all clause vertices representing the same literal must be contained in that solution. Therefore, all vertices in G_φ are solution dependent.

The ONLINE VERTEX COVER GAME has a graph G and a number $k \in \mathbb{N}$ as input. It asks whether there is a winning strategy for the online algorithm, that is, it finds a vertex cover of size at most k for every reveal order while knowing an isomorphic copy of G .

Theorem 13.3. *The ONLINE VERTEX COVER GAME is PSPACE-complete.*

The containment of ONLINE VERTEX COVER GAME in PSPACE is already established by Theorem 13.1. To show hardness, we extend the above reduction for VERTEX COVER according to our framework. Therefore, we need to introduce fake clause gadgets, dependency reveal gadgets, and ID gadgets and prove that they fulfill the gadget properties, required by Lemmas 13.1 to 13.3.

Fake Clause Gadget. The fake clause gadget exists to avoid leaking information about the literals by revealing which clauses contain them. Since the reduction from 3SATISFIABILITY to VERTEX COVER from [GJ79] never connects a vertex in a clause gadget to more than one literal vertex, the only information that can be revealed about a literal is how many clauses contain it. Whether it appears in a clause together with specific other literals is already concealed by the construction of the clause gadget. Therefore, as mentioned earlier, the ID gadget alone can already prevent this information leak. To adhere to our framework, we still define the fake clause gadget explicitly, however it uses the same construction as the ID gadget.

Definition 13.9 (Self-contained fake clause gadget for VERTEX COVER). *The fake clause gadget, for non-existing clause $C'_j \notin C$, consists of three stars with two leaves each, one for each literal contained in C'_j . Each center of a star is connected to the literal vertex it represents.*

An example for a fake clause gadget is shown in Figure 13.4. Any optimal vertex cover on the fake clause gadget has size 3 and contains exactly the center vertices of the three stars. In the neighborhood reveal model, fake clause gadgets can not be distinguished from real clause gadgets, as long as only vertices of variable gadgets are revealed by the adversary. However, as soon as a vertex of the fake clause gadget is revealed, it can be distinguished from a real clause gadget, as the vertex degrees are different due to the ID gadgets which are described later.

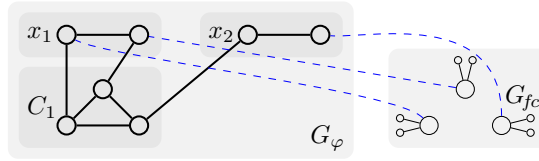


Figure 13.4: The reduction graph for the reduction from 3SATISFIABILITY to VERTEX COVER for instance $\varphi = (x_1 \vee \bar{x}_1 \vee x_2)$. The clause $(x_1 \vee \bar{x}_1 \vee \bar{x}_2)$ does not exist and is represented by a fake clause gadget G_{fc} . The blue dashed edges are the set E_{con} for the fake clause gadget.

Lemma 13.4 (Gadget Property 1). *The fake clause gadget (Definition 13.9) is self-contained for VERTEX COVER.*

Proof. Let $C'_j \notin C$ be an arbitrary non-existing clause with $G_{fc}(C'_j)$ being its fake clause gadget. Any optimal vertex cover on $G_{fc}(C'_j)$ has size 3 and contains exactly the three center vertices. For a contradiction, assume there is a vertex cover S' of size at most 3 on $G_{fc}(C'_j)$, that does not contain at least one of those three vertices. However, each of them has two neighbors that are not connected to any other vertices, which would then need to be part of the vertex cover instead, while not covering any additional edges. Additionally, S' needs to contain at least one vertex of the other two stars to cover their edges. This is a contradiction to S' containing at most three vertices.

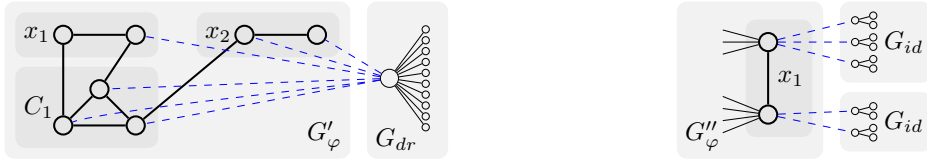
By the same argument, none of the center vertices can be replaced their neighbors in the reduction graph. Additionally, as the center vertices are already part of the vertex cover, none of their neighbors in the reduction graph are forced to be in the vertex cover to cover the edge between them.

Let $G_\varphi^0 = G_\varphi$ and let G_φ^γ be the graph that has been extended with γ many fake clause gadgets. Then given the graph $G_\varphi^{\gamma-1}$, the graph $G_{fc}(C'_j)$ and the graph $G_\varphi^\gamma = G_\varphi^{\gamma-1} \circ G_{fc}(C'_j)$, the following holds: There exists a solution of size 3 for $G_{fc}(C'_j)$, such that for any optimal solution of size k^* on $G_\varphi^{\gamma-1}$ the disjoint union of those two solutions is an optimal solution of size $k^* + 3$ for G_φ^γ . Therefore the fake clause gadget from Definition 13.9 is self-contained. \square

Dependency Reveal Gadget. Since the vertices of a clause gadget are solution dependent on the literal vertices they represent, the dependency reveal gadget needs to account for that. Our dependency reveal gadget for vertex cover gives all solution-dependent vertices a common neighbor. The presence of this common neighbor then allows the online algorithm to recognize the truth assignment of the \forall -variable, or if the adversary revealed a solution-dependent vertex too early, to distinguish the literals of the \forall -variable such that the \forall -decision degenerates into an \exists -decision. An example of a dependency reveal gadget is depicted in Figure 13.5a.

Definition 13.10 (Self-contained dependency reveal gadget for VERTEX COVER). *The dependency reveal gadget for a \forall -variable x_i is a star. Its center vertex is connected to the literal vertices of all variables with index at least i (except the true literal of variable x_i), and all vertices representing them in clauses (including the true literal of variable x_i). The number of leaves is such that, together with the connecting edges, the degree of the center vertex equals $3\binom{2n}{3} + 2n + 1$.*

The number of leaves of a dependency reveal gadget is always at least 2. Given a formula with n variables, there are exactly $2n$ literals, and $\binom{2n}{3}$ possible clauses with three literals. A dependency reveal gadget can target all three vertices of a clause and all literal vertices, however the true literal of variable x_i is not targeted. Thus, the number of connecting edges is at most



(a) The dependency reveal gadget for the \forall -variable x_1 , with only one variable of higher index, is depicted. The fake clause gadgets of G'_φ are omitted.

(b) The ID gadgets for the \forall -variable x_1 are shown. Both literal vertices have the same degree. For each gadget, the blue dashed edges are the set E_{con} .

Figure 13.5: Dependency reveal gadget and ID gadget for VERTEX COVER.

$3\binom{2n}{3} + 2n - 1$. Therefore, the optimal solution for the dependency reveal gadget always contains exactly the center vertex of the star by the same argument as for the fake clause gadget.

Lemma 13.5 (Gadget Property 2). *The dependency reveal gadget (Definition 13.10) is self-contained for VERTEX COVER.*

Proof. Let x_i be any \forall -variable of the TQBF GAME-instance. Since $G_{dr}(x_i)$ is a star with at least 2 leaves, the optimal vertex cover on $G_{dr}(x_i)$ contains exactly the center of the star, and no other vertices. As covering any edges incident to target vertices by those target vertices has no effect on the edges in $G_{dr}(x_i)$, no solution on the reduction graph can make the optimal solution on $G_{dr}(x_i)$ smaller. This also means that the optimal vertex cover on $G_{dr}(x_i)$ already covers all edges connecting it to its target vertices, thus no vertices of the reduction graph are forced to be in the vertex cover by attaching $G_{dr}(x_i)$.

Remember that G'_φ is the graph that resulted from G_φ by adding a fake clause gadget for every possible clause $C'_j \notin C$ with three literals. Let $G_\varphi^0 = G'_\varphi$ and let G_φ^γ be the graph that has been extended with γ many dependency reveal gadgets. Then given the graph $G_\varphi^{\gamma-1}$, the graph $G_{dr}(x_i)$ and the graph $G_\varphi^\gamma = G_\varphi^{\gamma-1} \circ G_{dr}(x_i)$, the following holds: There exists a solution of size 1 for $G_{dr}(x_i)$, such that for any optimal solution of size k^* on $G_\varphi^{\gamma-1}$ the disjoint union of those two solutions is an optimal solution of size $k^* + 1$ for G_φ^γ . Therefore the dependency reveal gadget from Definition 13.10 is self-contained. \square

ID Gadgets for Literal and Clause Vertices. Since both the literal vertices and the vertices of clause gadgets are solution dependent, the online algorithm needs to be able to identify which variable they correspond to, and in the case of \exists -variables also which literal they correspond to. For that, we look at the degrees of all vertices in the graph G''_φ . The leaves in the stars of the dependency reveal gadgets and fake clause gadgets have degree 1, and the center vertices in fake clause gadgets have degree 3. The center vertices of the dependency reveal gadgets have degree $3\binom{2n}{3} + 2n$. Therefore, any vertex that was not present in G_φ already has a degree that is either smaller than $\binom{2n-1}{2} + 4$ or larger than $\binom{2n-1}{2} + 4n + 3$ for $n \geq 2$ (which is necessary for three different literals per clause). Note that the latter inequality holds due to the following:

$$\binom{2n}{3} > \binom{2n-1}{2} \quad n \geq 2 \quad (13.5)$$

$$\binom{2n}{3} \geq 2n \quad n \geq 2 \quad (13.6)$$

$$3\binom{2n}{3} + 2n \geq \binom{2n}{3} + 6n > \binom{2n-1}{2} + 4n + 3 \quad n \geq 2 \quad (13.7)$$

Therefore, we use that range of degrees for our literal vertices and clause vertices.

Let $d_{<}^{\forall}(i)$ (resp. $d_{\leq}^{\forall}(i)$) be the number of \forall -variables with index smaller (resp. smaller or equal) than i . Since the formula of the TQBF GAME always alternates between \exists - and \forall -variables, $d_{<}^{\forall}(i) = \lfloor \frac{i}{2} \rfloor$ (resp. $d_{\leq}^{\forall}(i) = \lceil \frac{i}{2} \rceil$) if variable x_1 is \forall -quantified and $d_{<}^{\forall}(i) = \lfloor \frac{i-1}{2} \rfloor$ (resp. $d_{\leq}^{\forall}(i) = \lceil \frac{i-1}{2} \rceil$) otherwise. There are $\binom{2n-1}{2}$ possible different clauses with three literals that contain one specific literal. Thus in G''_{φ} , the literal vertices have degree $\binom{2n-1}{2} + d_{<}^{\forall}(i) + 1$, or $\binom{2n-1}{2} + d_{<}^{\forall}(i) + 2$ in case of the false literal of a \forall -variable that is not the last variable. With this, we can define the literal ID gadgets. As mentioned above, the fake clause gadget uses the same construction as the ID gadgets. That is, the ID gadgets are a collection of stars with two leaves, where the center vertices are connected to the target vertex. An example for an ID gadget can be seen in Figure 13.5b.

Definition 13.11 (Self-contained literal ID gadget for VERTEX COVER). *Let ℓ be some literal and x_i its corresponding variable. Let $d_{<}^{\forall}(i)$ be defined as above. Further let*

$$\begin{aligned} d(\ell) &= 4i - d_{<}^{\forall}(i) - 1 && \text{if } \ell \text{ is positive} \\ d(\ell) &= 4i - d_{<}^{\forall}(i) - 2 && \text{if } \ell \text{ is negative and } x_i \text{ is } \forall\text{-quantified} \\ d(\ell) &= 4i - d_{<}^{\forall}(i) && \text{if } \ell \text{ is negative and } x_i \text{ is } \exists\text{-quantified} \end{aligned}$$

Then the literal ID gadget for the literal vertex representing ℓ consists of $d(\ell)$ stars with two leaves, where each center vertex is connected to the identified literal vertex.

It is necessary that the online algorithm can recognize which literal a clause vertex represents, as the adversary could choose to reveal a clause vertex before a literal vertex of the corresponding variable gadget.

Definition 13.12 (Self-contained clause ID gadget for VERTEX COVER). *Let ℓ be a literal with $\ell \in C_j$ for some clause $C_j \in C$ and let x_i be the variable ℓ belongs to. Let $d_{\leq}^{\forall}(i)$ be defined as above. Further let*

$$\begin{aligned} d(\ell) &= \binom{2n-1}{2} + 4i - d_{\leq}^{\forall}(i) - 1 && \text{if } \ell \text{ is positive} \\ d(\ell) &= \binom{2n-1}{2} + 4i - d_{\leq}^{\forall}(i) && \text{if } \ell \text{ is negative.} \end{aligned}$$

Then the clause ID gadget for the clause vertex representing ℓ consists of $d(\ell)$ stars with two leaves, where each center vertex is connected to the identified clause vertex.

The following lemma follows directly from the same arguments as in Lemma 13.4 because they are also a collection of stars with two leaves.

Lemma 13.6 (Gadget Property 3). *The ID gadgets (Definitions 13.11 and 13.12) for VERTEX COVER are self-contained.*

In our framework, we also add ID gadgets to the fake clause gadgets and dependency reveal gadgets, however the next lemma and Table 13.1 show that those vertices can already be recognized by the online algorithm. Therefore, their ID gadgets are simply the empty graph, which trivially fulfills the property of self-containment. In Table 13.1, the vertex degrees after adding all ID gadgets are shown.

Lemma 13.7 (Gadget Properties 4-6). *In G'''_{φ} , each solution dependent vertex which is not in a literal gadget of a \forall -variable has a unique revelation subgraph. Further, the two literal vertices of*

Vertex degree	Type of vertices	Strategy
3	center vertices of ID gadgets and fake clause gadgets	accept
5	triangle vertices of fake clause gadgets	accept
$\binom{2n-1}{2} + 4i$	true literal of x_i , also false literal of x_i if it is \forall -quantified	depends on φ
$\binom{2n-1}{2} + 4i + 1$	false literal of variable x_i if it is \exists -quantified	depends on φ
$\binom{2n-1}{2} + 4i + 2$	vertex representing true literal of variable x_i in a clause	depends on φ
$\binom{2n-1}{2} + 4i + 3$	vertex representing false literal of variable x_i in a clause	depends on φ

Table 13.1: List of all vertex degrees in the final reduction from TQBF GAME to ONLINE VERTEX COVER NUMBER, where n is the number of variables. For any vertex that is not part of a literal or clause gadget, the optimal solution can be deduced just from its degree.

a \forall -variable have the same revelation subgraph, but different from vertices of any other gadget. Finally, each vertex that is solution independent or part of an extension gadget has a revelation subgraph that allows for an optimal decision.

Proof. All solution dependent vertices that are not part of the literal gadget of a \forall -variable have a unique revelation subgraph, since they have unique degree, as seen in Table 13.1.

A literal vertex of a \forall -variable shares its degree with exactly one other vertex, that is the vertex representing the negated literal. Thus as long as the adversary does not reveal any (fake) clause vertices, dependency reveal vertices or literal vertices of variables with higher index, the two literal vertices of a \forall -variable have exactly the same revelation subgraph, but different from any other vertex.

All solution-independent vertices and vertices of extension gadgets that have a degree larger than 1 are always contained in an optimal solution, by construction of these gadgets. Since their degrees are also always different from solution dependent vertices, the online algorithm can always make an optimal decision for them. Finally, by construction of the extension gadgets, no vertex of degree 1 is contained in any optimal solution. Therefore, the online algorithm can also always make an optimal decision on them based on their revelation subgraph. \square

Polynomial Time Reduction. All our gadgets can be constructed in polynomial time, as they contain at most $\mathcal{O}\left(\binom{2n}{2}\right) = \mathcal{O}(n^2)$ many vertices. Furthermore, the number of gadgets is also polynomial in the number of variables, as the number of possible clauses with three literals is bounded by $\binom{2n}{3} \in \mathcal{O}(n^3)$. Finally, the solution size k can also be computed in polynomial time.

- (1) For the base reduction, $n + 2m$ vertices are part of the optimal vertex cover.
- (2) For the fake clause gadgets, $3 \cdot \left(\binom{2n}{3} - m\right)$ vertices are part of the optimal vertex cover.
- (3) Recall that $d_{<}^{\forall}(i)$ is the number of \forall -variables with an index lower than i and therefore computable in polynomial time. Then the number of vertices in the optimal vertex cover that are part of dependency reveal gadgets is $d_{<}^{\forall}(n)$.
- (4) For the ID gadgets, we distinguish between literal and clause ID gadgets. For the literal ID gadgets,

$$x' + \sum_{i=1}^n 2 \cdot (4i + d_{<}^{\forall}(i) - 1)$$

vertices are part of the optimal vertex cover, where $x' = 1$ iff both x_1 and x_n are \exists -quantified, $x' = -1$ iff both x_1 and x_n are \forall -quantified, and $x' = 0$ otherwise.

For a variable x_i we denote with ℓ_{2i-1} its positive literal and with ℓ_{2i} its negative literal. For the clause ID gadgets, let $\#(\ell_a)$ be the number of times the literal ℓ_a , $a \in \{1, \dots, 2n\}$ appears in a clause. Then $\sum_{i=1}^n (\#(\ell_{2i-1}) + \#(\ell_{2i})) = 3m$ and therefore the size of all $\#(\ell_a)$ is polynomially bounded. The number of vertices in the optimal vertex cover, that are part of a clause ID gadget, is given by

$$\sum_{i=1}^n \#(\ell_{2i-1}) \cdot \left(\binom{2n-1}{2} + 4i - d_{\leq}^{\forall}(i) - 1 \right) + \sum_{i=1}^n \#(\ell_{2i}) \cdot \left(\binom{2n-1}{2} + 4i - d_{\leq}^{\exists}(i) \right)$$

Since all constructions are polynomial time computable, we established the requirements for Theorem 13.2, thus Theorem 13.3 is proven. The full construction of G_{φ}''' is shown in Figure 13.6.

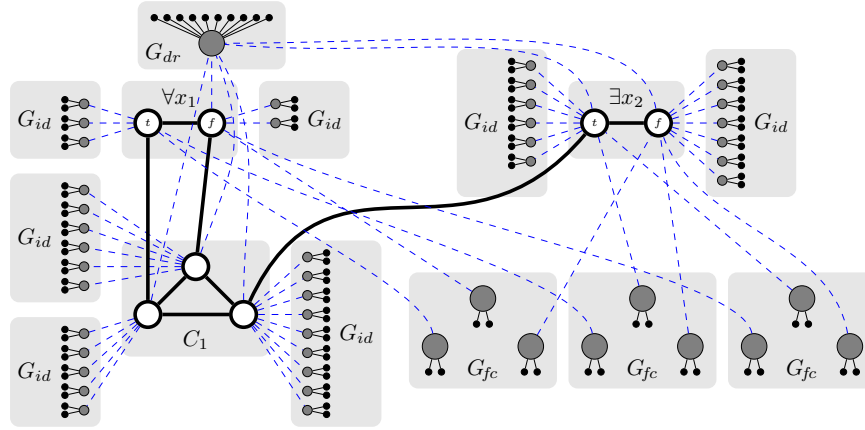


Figure 13.6: Complete view on the reduction for the TQBF-instance $\forall x_1 \exists x_2 (x_1 \vee \bar{x}_1 \vee x_2)$ to ONLINE VERTEX COVER GAME. The thick vertices and edges represent the original reduction. The blue dashed edges are the connecting edges of the extension gadgets. There are optimal solutions that contain all the gray vertices and none of the black vertices. Whether the white vertices are contained depends on the feasible solutions for the TQBF-formula.

13.6 More Vertex Subset Problems

In this section, we apply Theorem 13.2 to the more vertex subset graph problems: ONLINE INDEPENDENT SET GAME, ONLINE CLIQUE GAME, ONLINE DOMINATING SET GAME and ONLINE FEEDBACK VERTEX SET GAME. Like the ONLINE VERTEX COVER GAME, they take a graph G and a number $k \in \mathbb{N}$ as input. They ask whether there is a winning strategy for the online algorithm, that is, it finds a subset of vertices of size at least (resp. most) k fulfilling certain conditions for every reveal order while knowing an isomorphic copy of G .

Independent Set. We start with the reduction for ONLINE INDEPENDENT SET GAME, which uses the same construction as the vertex cover reduction with a slight modification. Again, we use our framework to derive the following result.

Theorem 13.4. *The ONLINE INDEPENDENT SET GAME is PSPACE-complete.*

We extend an existing reduction and apply Theorem 13.2. For the base reduction from 3SATISFIABILITY to INDEPENDENT SET, we use a slight modification of the reduction from

3SATISFIABILITY to VERTEX COVER given in [GJ79]. Instead of connecting each clause vertex to the literal vertex it represents, we connect it to its negation. The size of the independent set that should be found in G_φ is then $k = |X| + |C|$. The correctness argument works analogously to the reduction for VERTEX COVER.

For the fake clause gadget, we use the same construction as for VERTEX COVER, but the target vertices are adjusted in the same way as for the clause gadgets. In case of the dependency reveal gadget and ID gadgets we use exactly the same constructions as for VERTEX COVER. The full construction of the reduction is shown in Figure 13.7. Recall that for any vertex cover $S \subseteq V$, the set $V \setminus S$ forms an independent set. Since the optimal solution of any of our extension gadgets for VERTEX COVER are the vertices incident to E_{con} of that gadget, the optimal solution of these gadgets for INDEPENDENT SET contains all vertices not incident to E_{con} . Thus, their optimal solution is not influenced by the solution on G_φ and vice versa. Therefore, they are self-contained for INDEPENDENT SET. Gadget Properties 4-6 hold for INDEPENDENT SET by the same arguments as for VERTEX COVER.

Since this modified reduction is obviously still computable in polynomial time, Theorems 13.1 and 13.2 prove our claim.

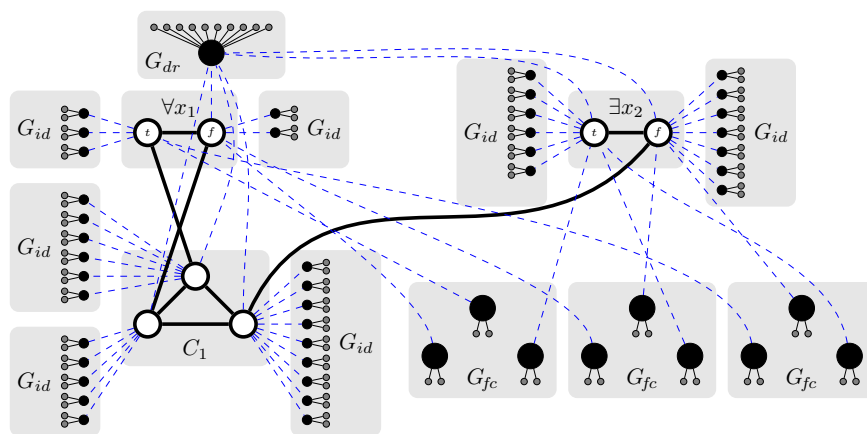


Figure 13.7: Complete view on the reduction for the TQBF-instance $\forall x_1 \exists x_2 (x_1 \vee \bar{x}_1 \vee x_2)$ to ONLINE INDEPENDENT SET GAME. The thick vertices and edges represent the original reduction. The blue dashed edges are the connecting edges of the extension gadgets. There are optimal solutions that contain all the gray vertices and none of the black vertices. Whether the white vertices are contained depends on the feasible solutions for the TQBF-formula.

Clique. Next, we show that ONLINE CLIQUE GAME is PSPACE-complete. For this, we make use of the strong connection to the independent set problem. The idea is to use the complement graph, which is already constituting the reduction on the NP-level between INDEPENDENT SET and CLIQUE.

Theorem 13.5. *The ONLINE CLIQUE GAME is PSPACE-complete.*

The reduction is the complement graph of the reduction graph for ONLINE INDEPENDENT SET GAME with the same target size k . For the correctness, observe that revealing the neighborhood yields the same information as not revealing the neighborhood but the complement of the neighborhood. The only difference is that the online algorithm receives the labels of the neighborhood in the independent set game and in the clique game it receives the labels of the complement of

the neighborhood. Thus, the only algorithm is able to distinguish the non-neighbors in the clique game which is one-to-one correspondent to distinguishing the neighbors in the independent set game and vice versa. With this observation, we analyze all gadgets and show that the reduction is correct.

We begin with the original reduction on the NP-level. The reduction is correct because if there are no edges between a set of vertices, the complement graph has a clique with exactly these vertices. We now extend the reduction as described in the framework. We first add all fake clause gadgets to the base reduction. Secondly, we add all dependency reveal gadgets. At last, we add the ID gadgets to the graph.

The fake clause gadget is the complement graph of the fake clause gadget of INDEPENDENT SET. Furthermore, we define the connecting edges to be the complement of the connecting edges in the INDEPENDENT SET fake clause gadget as well. Especially, there are connecting edges to all extension gadgets. The fake clause gadget remains self-contained due to the fact that the vertices in the INDEPENDENT SET solution, which are all leaves of the three stars, are also the vertices in the solution of CLIQUE. These six vertices can again be disjointly merged to the solution of the existing graph. For this, observe that the former leaves of the three stars are now connected to all vertices of the base graph such that a clique in the base graph can be easily extended with these six vertices.

Secondly, we define the dependency reveal gadget to be the complement graph and the complement of the connecting edges of those from the INDEPENDENT SET reduction. These gadgets are also self-contained because the optimal solution are also the former leaves of the star which can be disjointly merged with the rest of the graph.

At last, the ID gadgets are identical to the fake clause gadgets in the sense that we add the complement graph of a star with two leaves. The connecting edges are again the complement edges of the ID gadget in the INDEPENDENT SET reduction. Thus, the degrees are the number of all vertices of the resulting graph minus one minus the values from Table 13.1. With the same argument for the fake clause gadget and dependency reveal gadget, the gadgets are self-contained.

After adding all gadgets, the resulting graph is exactly the complement graph to the corresponding reduction graph from 3SATISFIABILITY to INDEPENDENT SET. Thus, every clique in the reduction graph from 3SATISFIABILITY to CLIQUE corresponds one-to-one to an independent set in the reduction graph from 3SATISFIABILITY to INDEPENDENT SET. Furthermore, the game decisions by the online algorithm and the adversary are one-to-one correspondent to the decisions in the game of ONLINE INDEPENDENT SET due to the observation from above that revealing the labels of all neighbors yields the same information than revealing the labels of all non-neighbor but not the neighbors. This concludes the description of the reduction. The reduction is computable in polynomial time because only predefined gadgets of polynomial size are added to a polynomial number of vertices.

Dominating Set. Next, we use our framework to derive the completeness result for ONLINE DOMINATING SET GAME. For this, we use a direct reduction from 3SATISFIABILITY to DOMINATING SET instead of reusing the reduction from 3SATISFIABILITY to VERTEX COVER and a reduction from VERTEX COVER to DOMINATING SET by making use of transitivity.

Theorem 13.6. *The ONLINE DOMINATING SET GAME is PSPACE-complete.*

For the base reduction, we use the folklore reduction, which we described as example for solution dependent vertices in Example 13.1.

We define the following extension gadgets for DOMINATING SET. The fake clause gadget for non-existing clause $C'_j \notin C$, is a star with $2n - 2$ leaves. Its center is connected to the literal vertices representing the literals contained in the clause. The dependency reveal gadget for a \forall -variable x_i is a star. The target vertices are the literal vertices of all variables with index at

Vertex degree	Type of vertices	Strategy
1	leaves of any fake clause / dependency reveal gadget, leaves of literal ID gadgets not adjacent to literal vertex	reject
2	leaves of literal ID gadgets adjacent to literal vertex, third vertex of variable gadgets	reject
3	clause vertices	reject
$2n + 1$	center vertex of fake clause / dependency reveal gadgets	accept
$\binom{2n-1}{2} + 4(n+1)$	center vertex of any literal ID gadget	accept
$\binom{2n-1}{2} + 4i$	true literal of x_i , false literal of x_i if it is \forall -quantified	depends on φ
$\binom{2n-1}{2} + 4i + 1$	false literal of variable x_i if it is \exists -quantified	depends on φ

Table 13.2: List of all vertex degrees in the final reduction from TQBF GAME to ONLINE DOMINATING SET GAME. For any vertex that is not part of a literal gadget, the optimal solution can be deduced just from its degree.

least i , except the true literal of variable x_i . The number of leaves is such that, together with the connecting edges, the degree of the center vertex equals $2n + 1$. At last, we define the ID gadgets. For this, let ℓ be some literal, and x_i the corresponding variable. Let $d_{<}^{\forall}(i)$ be defined as above. Further let

$$\begin{aligned}
 d(\ell) &= 4i - d_{<}^{\forall}(i) - 2 && \text{if } \ell \text{ is positive} \\
 d(\ell) &= 4i - d_{<}^{\forall}(i) - 3 && \text{if } \ell \text{ is negative and } x_i \text{ is } \forall\text{-quantified} \\
 d(\ell) &= 4i - d_{<}^{\forall}(i) - 1 && \text{if } \ell \text{ is negative and } x_i \text{ is } \exists\text{-quantified}
 \end{aligned}$$

Then the literal ID gadget for the literal vertex representing ℓ is a star with $\binom{2n-1}{2} + 4(n+1)$ leaves, where $d(\ell)$ of those leaves are also connected to the literal vertex.

Our fake clause gadget and dependency reveal gadget for DOMINATING SET have the same structure, as they are stars. Any optimal dominating set on those gadgets contains exactly the center vertex of the star. The connection of those gadgets to the reduction graph is done only by edges from the centers of the respective stars to literal vertices. Thus, neither can make the solution on the reduction graph smaller, as the literal vertices of each variable are in a triangle together with a third vertex that is not connected to any other vertex. At the same time, dominating the center vertex of one of those gadgets by one of the connected literal vertices never removes it from an optimal dominating set, as it has at least two leaves. This proves the Gadget Properties 1 and 2.

The literal ID gadget is also a star. Since there are always at least 2 of its leaves that are not connected to any other vertex, an optimal solution on the ID gadget always contains the center vertex. Thus the vertices that are connected to the reduction graph are dominated, but not contained in any optimal dominating set. Since they are all connected to the same vertex, this proves Gadget Property 3.

After extending G_{φ} with the fake clause gadgets, dependency reveal gadgets and literal ID gadgets (in that order), Gadget Properties 4-6 already hold as shown in Table 13.2. Thus we can use the empty graph as an ID gadget for the clauses, fake clause gadgets and dependency reveal gadgets to obtain a reduction according to our framework. Since the reduction can be computed in polynomial time by an analogous argument to the reduction for ONLINE VERTEX COVER GAME, Theorem 13.2 proves PSPACE-hardness of ONLINE DOMINATING SET GAME. Thus together with Theorem 13.1, it is also PSPACE-complete.

Feedback Vertex Set. The last problem on which we apply the framework is undirected FEED-

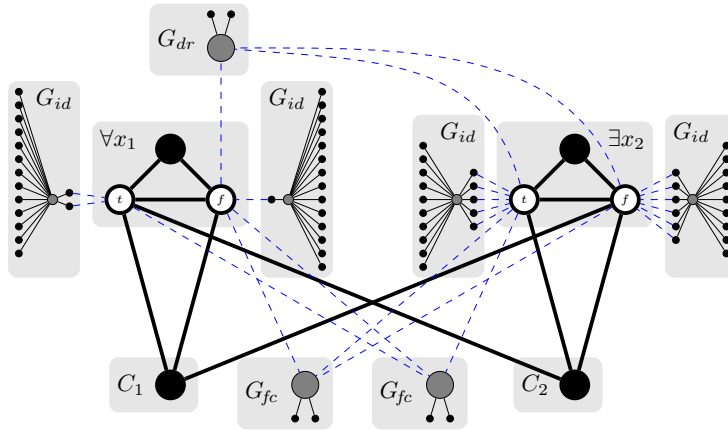


Figure 13.8: Complete view on the reduction for the TQBF-instance $\forall x_1 \exists x_2 (x_1 \vee \bar{x}_1 \vee \bar{x}_2) \wedge (x_1 \vee x_2 \vee \bar{x}_2)$ to ONLINE DOMINATING SET GAME. The thick vertices and edges represent the original reduction. The blue dashed edges are the connecting edges of the extension gadgets. There are optimal solutions that contain all the gray vertices and none of the black vertices. Whether the white vertices are contained depends on the feasible solutions for the TQBF-formula.

BACK VERTEX SET. The FEEDBACK VERTEX SET problem is again strongly related to VERTEX COVER such that we use the reduction from 3SATISFIABILITY to VERTEX COVER and then we use a folklore reduction from VERTEX COVER to FEEDBACK VERTEX SET as base reduction. We then define the extension gadgets accordingly to this new reduction.

Theorem 13.7. *The ONLINE FEEDBACK VERTEX SET GAME is PSPACE-complete.*

We first start with the base reduction from 3SATISFIABILITY to FEEDBACK VERTEX SET. For this, we use the reduction from 3SATISFIABILITY to VERTEX COVER. Then for each edge in the reduction graph, we add a fresh vertex and connect it to the incident vertices of the edge. Thus, every edge in the original graph is part of a 3-cycle, which has to be eliminated by the feedback vertex set.

For the correctness, let us assume we have a vertex cover in the original graph. This vertex cover can be translated one-to-one to the new reduction graph and all cycles are eliminated due to the fact that every edge induces exactly one 3-cycle over its incident vertices over a fresh vertex. Observe that a vertex cover already eliminates all other cycles in the graph. On the other hand, if there is a feedback vertex set, all 3-cycles are eliminated. That is, for every edge one vertex of the corresponding 3-cycle has to be included. If only original vertices are included, we have a one-to-one correspondence between the solutions. Otherwise if one of the fresh vertices is included, we can choose to take either one of the original vertices adjacent to that vertex into the vertex cover solution. Then for each edge, either endpoint is in the feedback vertex set. Thus, the same set of vertices is also a vertex cover in the original graph.

As we use the VERTEX COVER reduction, all original vertices of that reduction remain solution-dependent. The fresh vertices inducing a 3-cycle, however, are not solution-dependent because they are easily identifiable by the online algorithm (after adding all extension gadgets) and it is always at least as good to include the corresponding adjacent vertices into the solution as described above.

With this, we have a base reduction from 3SATISFIABILITY to FEEDBACK VERTEX SET, we

only need to define the extension gadgets. As before, we begin with the fake clause gadgets. The fake clause gadgets are three independent vertex pairs. The literal vertices are then connected one by one to one of the three pairs. If a literal vertex is revealed, the online algorithm is not able to distinguish between a clause gadget and a fake clause gadget because it only sees two independent vertices. Furthermore, the gadget is self-contained because two independent vertices do not induce a cycle and cannot induce a cycle when connected to only one vertex each. Therefore, the solution on the fake clause gadget is empty. Thus, we can easily merge the solutions of the fake clause gadgets with the solution of G_φ .

Next, we construct the dependency reveal gadget. It is basically a star, where we connect the first two pairs to a 3-cycle each. Furthermore, the center vertex is connected to the literal vertices of all variables with larger index i (except the true literal of x_i) and to all solution dependent vertices representing them in the clauses. Consequently the dependency reveal gadget is connected to the same vertices as in the VERTEX COVER reduction. Additionally, the center vertex has to be part of a solution due to the two 3-cycles. Furthermore, all cycles that are created by attaching the dependency reveal gadget are also eliminated by deleting this center vertex. Thus we are able to disjointly merge the solutions of the dependency reveal gadget and the rest of the reduction graph.

At last, we have to define the ID gadgets. These are independent single vertices connected to the vertex to identify (like the fake clause gadgets). Thus they are also self-contained (like the fake clause gadgets) because their solution is empty. The target degrees of every vertex can be found in Table 13.3.

This concludes the description of the reduction. The full reduction graph from TQBF to ONLINE FEEDBACK VERTEX SET GAME is depicted in Figure 13.9. The reduction is computable in polynomial time because only predefined gadgets of polynomial size are added to a polynomial number of vertices.

Vertex degree	Type of vertices	Strategy
1	leaves of any extension gadget	reject
2	vertices introduced for an edge, triangles of dependency reveal gadget	reject
$3\binom{2n}{3} + 2(n+2)$	center vertex of dependency reveal gadget	accept
$\binom{2n-1}{2} + 4i$	true literal of x_i , false literal of x_i if it is \forall -quantified	depends on φ
$\binom{2n-1}{2} + 4i + 1$	false literal of variable x_i if it is \exists -quantified	depends on φ
$\binom{2n-1}{2} + 4i + 2$	vertex representing true literal of variable x_i in a clause	depends on φ
$\binom{2n-1}{2} + 4i + 3$	vertex representing false literal of variable x_i in a clause	depends on φ

Table 13.3: List of all vertex degrees in the final reduction from TQBF GAME to ONLINE FEEDBACK VERTEX SET GAME, where n is the number of variables. For any vertex that is not part of a literal or clause gadget, the optimal solution can be deduced just from its degree.

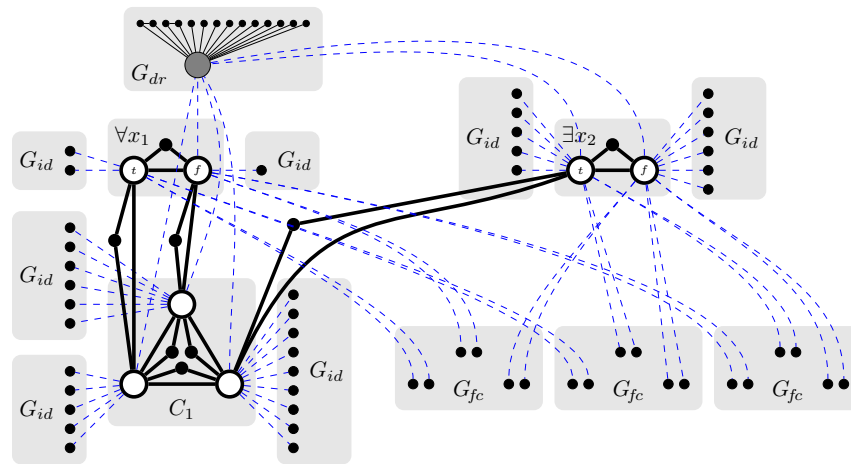


Figure 13.9: Complete view on the reduction for the TQBF-instance $\forall x_1 \exists x_2 (x_1 \vee \bar{x}_1 \vee x_2)$ to ONLINE FEEDBACK VERTEX SET GAME. The thick vertices and edges represent the original reduction. The blue dashed edges are the connecting edges of the extension gadgets. There are optimal solutions that contain all the gray vertices and none of the black vertices. Whether the white vertices are contained depends on the feasible solutions for the TQBF-formula. Note that the leftmost vertices of the dependency reveal gadget form triangles with its center, forcing it to be part of the solution.

Chapter 14

Conclusion

In this part, we have shown that for many NP-complete problems, their recoverable robust version with elemental uncertainty is Σ_3^P -complete. Furthermore, we have shown that online vertex subset problems based on NP-complete problems are PSPACE-complete.

Concretely, we have defined Hamming distance recoverable robust problems with elemental uncertainty, where the elemental uncertainty can be expressed by *xor*-dependencies or Γ -set scenarios. Then, we have defined universe gadget reductions to build a framework for a class of Hamming distance recoverable robust problems. The complexity results are that the Hamming distance recoverable robust versions of NP-complete problems are Σ_3^P -complete for *xor*-dependency scenarios and Γ -set scenarios if 3SATISFIABILITY is universe gadget reducible to them and a corresponding solution size function exists. Remaining interesting questions are whether there is a (light-weight) reduction framework for problems in robust, bilevel, and online optimization with elemental uncertainty to derive completeness for higher levels in the polynomial hierarchy than NP. The SSP framework already answered many open questions concerning whether there is a framework for problems in robust, bilevel, and online optimization for cost uncertainty.

In the context of online optimization, we have defined online vertex subset games. These problems include a map, which is an unlabeled copy of the underlying graph. Thus, the adversary is only able to decide on the reveal order of the vertices. We have developed a gadget reduction framework for online versions of vertex subset problems under the neighborhood reveal model that allows reductions from TQBF GAME to show that these are PSPACE-complete. We have shown particularly that the online versions of VERTEX COVER, INDEPENDENT SET, CLIQUE, DOMINATING SET, and FEEDBACK VERTEX SET with the neighborhood reveal model are PSPACE-complete. The question arises if the five problems VERTEX COVER, INDEPENDENT SET, CLIQUE, DOMINATING SET, and FEEDBACK VERTEX SET are actually PSPACE-complete under the vertex arrival model as described in [Kud15, BV18]. One way to show the PSPACE-completeness might be by using our reduction framework together with a type of error correction gadget. However, the missing knowledge in the vertex arrival model might increase the asymmetry in favor of the adversary, such that the complexity decreases and it remains NP-hard. Additionally, the presented framework may be extended to more general subset problems where the solution is not a vertex subset.

Bibliography

- [AAI⁺01] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Comput. Complex.*, 10(2):117–138, 2001.
- [AB00] Igor Averbakh and Oded Berman. Minmax regret median location on a network under uncertainty. *INFORMS J. Comput.*, 12(2):104–110, 2000.
- [AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009.
- [ABV08] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Complexity of the min-max (regret) versions of min cut problems. *Discret. Optim.*, 5(1):66–73, 2008.
- [ABV09] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min-max and min-max regret versions of combinatorial optimization problems: A survey. *Eur. J. Oper. Res.*, 197(2):427–438, 2009.
- [AL04] Igor Averbakh and Vasilij Lebedev. Interval data minmax regret network optimization problems. *Discret. Appl. Math.*, 138(3):289–301, 2004.
- [ANS80] Takanori Akiyama, Takao Nishizeki, and Nobuji Saito. NP-completeness of the hamiltonian cycle problem for bipartite graphs. *Journal of Information processing*, 3(2):73–76, 1980.
- [Ave03] Igor Averbakh. Complexity of robust single facility location problems on networks with uncertain edge lengths. *Discret. Appl. Math.*, 127(3):505–522, 2003.
- [Bar25] Celina Janet Bartlett. A compendium of subset search problems and reductions relating to the parsimonious property. *CoRR*, abs/2506.12255, 2025.
- [BBdHH22] Luce Brotcorne, Christoph Buchheim, Dick den Hertog, and Dorothee Henke. Optimization at the second level (dagstuhl seminar 22441). *Dagstuhl Reports*, 12(10):207–224, 2022.
- [BBPR15] Cristina Bazgan, Cédric Bentz, Christophe Picouleau, and Bernard Ries. Blockers for the stability number and the chromatic number. *Graphs Comb.*, 31(1):73–90, 2015.
- [BCT24] Max Bannach, Florian Chudigiewitsch, and Till Tantau. On the descriptive complexity of vertex deletion problems. In Rastislav Kráľovic and Antonín Kucera, editors, *49th International Symposium on Mathematical Foundations of Computer Science, MFCS 2024, August 26-30, 2024, Bratislava, Slovakia*, volume 306 of *LIPICs*, pages 17:1–17:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.
- [BFKM17] Joan Boyar, Lene M. Favrholdt, Christian Kudahl, and Jesper W. Mikkelsen. The advice complexity of a class of hard online problems. *Theory Comput. Syst.*, 61(4):1128–1177, 2017.
- [BG22] Matthew Bold and Marc Goerigk. Investigating the recoverable robust single machine scheduling problem under interval uncertainty. *Discret. Appl. Math.*, 313:99–114, 2022.
- [BGGN04] Aharon Ben-Tal, A. P. Goryashko, E. Guslitzer, and Arkadi Nemirovski. Adjustable robust solutions of uncertain linear programs. *Math. Program.*, 99(2):351–376, 2004.
- [BGKK19] Christina Büsing, Sebastian Goderbauer, Arie M. C. A. Koster, and Manuel Kutschka. Formulations and algorithms for the recoverable Γ -robust knapsack problem. *EURO J.*

- Comput. Optim.*, 7(1):15–45, 2019.
- [BGN09] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*, volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009.
- [BK18] Joan Boyar and Christian Kudahl. Adding isolated vertices makes some greedy online algorithms optimal. *Discret. Appl. Math.*, 246:12–21, 2018.
- [BKK11a] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: Γ -scenarios. In Julia Pahl, Torsten Reiners, and Stefan Voß, editors, *Network Optimization - 5th International Conference, INOC 2011, Hamburg, Germany, June 13-16, 2011. Proceedings*, volume 6701 of *Lecture Notes in Computer Science*, pages 583–588. Springer, 2011.
- [BKK11b] Christina Büsing, Arie M. C. A. Koster, and Manuel Kutschka. Recoverable robust knapsacks: the discrete scenario case. *Optim. Lett.*, 5(3):379–392, 2011.
- [BLS⁺12] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE transactions on power systems*, 28(1):52–63, 2012.
- [BNKS98] Amotz Bar-Noy, Samir Khuller, and Baruch Schieber. The complexity of finding most vital arcs and nodes. 1998.
- [BS03] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Math. Program.*, 98(1-3):49–71, 2003.
- [BS04a] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Oper. Res.*, 52(1):35–53, 2004.
- [BS04b] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization under ellipsoidal uncertainty sets. *Manuscript, MIT*, 9, 2004.
- [BTT10] Cristina Bazgan, Sonia Toubaline, and Zsolt Tuza. Complexity of most vital nodes for independent set in graphs related to tree structures. In Costas S. Iliopoulos and William F. Smyth, editors, *Combinatorial Algorithms - 21st International Workshop, IWOCA 2010, London, UK, July 26-28, 2010, Revised Selected Papers*, volume 6460 of *Lecture Notes in Computer Science*, pages 154–166. Springer, 2010.
- [BTT11] Cristina Bazgan, Sonia Toubaline, and Zsolt Tuza. The most vital nodes with respect to independent set and vertex cover. *Discret. Appl. Math.*, 159(17):1933–1946, 2011.
- [BTV10] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Complexity of determining the most vital elements for the 1-median and 1-center location problems. In Weili Wu and Ovidiu Daescu, editors, *Combinatorial Optimization and Applications - 4th International Conference, COCOA 2010, Kailua-Kona, HI, USA, December 18-20, 2010, Proceedings, Part I*, volume 6508 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 2010.
- [BTV13] Cristina Bazgan, Sonia Toubaline, and Daniel Vanderpooten. Complexity of determining the most vital elements for the p-median and p-center location problems. *J. Comb. Optim.*, 25(2):191–207, 2013.
- [Büs09] Christina Büsing. The exact subgraph recoverable robust shortest path problem. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *Lecture Notes in Computer Science*, pages 231–248. Springer, 2009.
- [Büs11] Christina Büsing. *Recoverable robustness in combinatorial optimization*. Cuvillier Verlag, 2011.
- [Büs12] Christina Büsing. Recoverable robust shortest path problems. *Networks*, 59(1):181–189, 2012.
- [BV18] Martin Böhm and Pavel Veselý. Online chromatic number is pspace-complete. *Theory Comput. Syst.*, 62(6):1366–1391, 2018.
- [BYFL95] André Blais, Robert Young, Christopher Fleury, and Miriam Lapp. Do people vote on the basis of minimax regret? *Political Research Quarterly*, 48(4):827–836, 1995.

- [CCG⁺08] Valentina Cacchiani, Alberto Caprara, Laura Galli, Leo G. Kroon, and Gábor Maróti. Recoverable robustness for railway rolling stock planning. In Matteo Fischetti and Peter Widmayer, editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems, Karlsruhe, Germany, September 18, 2008*, volume 9 of *OASICS*. Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), Schloss Dagstuhl, Germany, 2008.
- [CCG⁺12] Valentina Cacchiani, Alberto Caprara, Laura Galli, Leo G. Kroon, Gábor Maróti, and Paolo Toth. Railway rolling stock planning: Robustness against large disruptions. *Transp. Sci.*, 46(2):217–232, 2012.
- [CCLW13] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J. Woeginger. A complexity and approximability study of the bilevel knapsack problem. In Michel X. Goemans and José R. Correa, editors, *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013, Valparaíso, Chile, March 18-20, 2013. Proceedings*, volume 7801 of *Lecture Notes in Computer Science*, pages 98–109. Springer, 2013.
- [CCLW14] Alberto Caprara, Margarida Carvalho, Andrea Lodi, and Gerhard J. Woeginger. A study on the computational complexity of the bilevel knapsack problem. *SIAM J. Optim.*, 24(2):823–838, 2014.
- [CDS⁺09a] Serafino Cicerone, Gianlorenzo D’Angelo, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. Recoverable robust timetabling for single delay: Complexity and polynomial algorithms for special cases. *J. Comb. Optim.*, 18(3):229–257, 2009.
- [CDS⁺09b] Serafino Cicerone, Gianlorenzo D’Angelo, Gabriele Di Stefano, Daniele Frigioni, and Alfredo Navarra. Recoverable robustness for train shunting problems. *Algorithmic Oper. Res.*, 4(2):102–116, 2009.
- [CDS⁺09c] Serafino Cicerone, Gianlorenzo D’Angelo, Gabriele Di Stefano, Daniele Frigioni, Alfredo Navarra, Michael Schachtebeck, and Anita Schöbel. Recoverable robustness in shunting and timetabling. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *Lecture Notes in Computer Science*, pages 28–60. Springer, 2009.
- [CG16] André B. Chassein and Marc Goerigk. On the recoverable robust traveling salesman problem. *Optim. Lett.*, 10(7):1479–1492, 2016.
- [CGH⁺06] Robert D Carr, Harvey J Greenberg, William E Hart, Goran Konjevod, Erik Lauer, Henry Lin, Tod Morrison, and Cynthia A Phillips. Robust optimization of contaminant sensor placement for community water systems. *Mathematical programming*, 107:337–356, 2006.
- [CGKZ18] André B. Chassein, Marc Goerigk, Adam Kasperski, and Pawel Zielinski. On recoverable and two-stage robust selection problems with budgeted uncertainty. *Eur. J. Oper. Res.*, 265(2):423–436, 2018.
- [CM12] Luis Cadarso and Ángel Marín. Recoverable robustness in rapid transit network design. *Procedia - Social and Behavioral Sciences*, 54:1288–1297, 2012. Proceedings of EWGT2012 - 15th Meeting of the EURO Working Group on Transportation, September 2012, Paris.
- [CNW83] Gérard Cornuéjols, George Nemhauser, and Laurence Wolsey. The uncapacitated facility location problem. Technical report, Cornell University Operations Research and Industrial Engineering, 1983.
- [Con14] Eduardo Conde. A MIP formulation for the minmax regret total completion time in scheduling with unrelated parallel machines. *Optim. Lett.*, 8(4):1577–1589, 2014.
- [Coo71] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranjan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.
- [CSN22] Amadeu Almeida Coco, Andréa Cynthia Santos, and Thiago F. Noronha. Robust min-max regret covering problems. *Comput. Optim. Appl.*, 83(1):111–141, 2022.
- [Den11] Scott Denegre. *Interdiction and Discrete Bilevel Linear Programming*. PhD thesis, USA,

2011. AAI3456385.
- [DL15] Arianna Degan and Ming Li. Psychologically-based voting with uncertainty. *European Journal of Political Economy*, 40:242–259, 2015.
- [DLY⁺15] Tao Ding, Shiyu Liu, Wei Yuan, Zhaohong Bie, and Bo Zeng. A two-stage robust reactive power optimization considering uncertain wind power integration in active distribution networks. *IEEE Transactions on Sustainable Energy*, 7(1):301–311, 2015.
- [DMP⁺15] Mitre Costa Dourado, Dirk Meierling, Lucia Draque Penso, Dieter Rautenbach, Fábio Protti, and Aline Ribeiro de Almeida. Robust recoverable perfect matchings. *Networks*, 66(3):210–213, 2015.
- [DSN09] Gianlorenzo D’Angelo, Gabriele Di Stefano, and Alfredo Navarra. Evaluation of recoverable-robust timetables on tree networks. In Jirí Fiala, Jan Kratochvíl, and Mirka Miller, editors, *Combinatorial Algorithms, 20th International Workshop, IWOCA 2009, Hradec nad Moravicí, Czech Republic, June 28–July 2, 2009, Revised Selected Papers*, volume 5874 of *Lecture Notes in Computer Science*, pages 24–35. Springer, 2009.
- [DSNP11] Gianlorenzo D’Angelo, Gabriele Di Stefano, Alfredo Navarra, and Maria Cristina Pinotti. Recoverable robust timetables: An algorithmic approach on trees. *IEEE Trans. Computers*, 60(3):433–446, 2011.
- [DSP19] Bert Dijk, Bruno Filipe Santos, and João P. Pita. The recoverable robust stand allocation problem: a GRU airport case study. *OR Spectr.*, 41(3):615–639, 2019.
- [DW10] Vladimir G. Deineko and Gerhard J. Woeginger. Pinpointing the complexity of the interval min-max regret knapsack problem. *Discret. Optim.*, 7(4):191–196, 2010.
- [DZ20] Stephan Dempe and Alain Zemkoho. Bilevel optimization. In *Springer optimization and its applications*, volume 161. Springer, 2020.
- [FG87] Aviezri S. Fraenkel and Elisheva Goldschmidt. Pspace-hardness of some combinatorial games. *J. Comb. Theory A*, 46(1):21–38, 1987.
- [FGJ22] Janosch Fuchs, Christoph Grüne, and Tom Janßen. The complexity of online graph games. *CoRR*, abs/2210.01694, 2022.
- [FGJ23] Janosch Fuchs, Christoph Grüne, and Tom Janßen. The complexity of graph exploration games. *CoRR*, abs/2302.08420, 2023.
- [FGJ24] Janosch Fuchs, Christoph Grüne, and Tom Janßen. The complexity of online graph games. In Henning Fernau, Serge Gaspers, and Ralf Klasing, editors, *SOFSEM 2024: Theory and Practice of Computer Science - 49th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2024, Cochem, Germany, February 19–23, 2024, Proceedings*, volume 14519 of *Lecture Notes in Computer Science*, pages 269–282. Springer, 2024.
- [FGJ25] Janosch Fuchs, Christoph Grüne, and Tom Janßen. The complexity of graph exploration games. In Rastislav Královic and Vera Kurková, editors, *SOFSEM 2025: Theory and Practice of Computer Science - 50th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2025, Bratislava, Slovak Republic, January 20–23, 2025, Proceedings, Part II*, volume 15539 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2025.
- [FHLW21] Dennis Fischer, Tim A. Hartmann, Stefan Lendl, and Gerhard J. Woeginger. An investigation of the recoverable robust assignment problem. In Petr A. Golovach and Meirav Zehavi, editors, *16th International Symposium on Parameterized and Exact Computation, IPEC 2021, September 8–10, 2021, Lisbon, Portugal*, volume 214 of *LIPICs*, pages 19:1–19:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [FHW80] Steven Fortune, John E. Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theor. Comput. Sci.*, 10:111–121, 1980.
- [FLMS19] Fabio Furini, Ivana Ljubic, Sébastien Martin, and Pablo San Segundo. The maximum clique interdiction problem. *Eur. J. Oper. Res.*, 277(1):112–127, 2019.

- [FMW14] Gary Froyland, Stephen J. Maher, and Cheng-Lung Wu. The recoverable robust tail assignment problem. *Transp. Sci.*, 48(3):351–372, 2014.
- [FR21] Nicolas Fröhlich and Stefan Ruzika. On the hardness of covering-interdiction problems. *Theor. Comput. Sci.*, 871:1–15, 2021.
- [GH24] Marc Goerigk and Michael Hartisch. *An Introduction to Robust Combinatorial Optimization*, volume 361 of *International Series in Operations Research & Management Science*. Springer, 2024.
- [GHM⁺13] Marc Goerigk, Sacha Heße, Matthias Müller-Hannemann, Marie Schmidt, and Anita Schöbel. Recoverable robust timetable information. In Daniele Frigioni and Sebastian Stiller, editors, *13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems, ATMOS 2013, September 5, 2013, Sophia Antipolis, France*, volume 33 of *OASICS*, pages 1–14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2013.
- [GJ79] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [GJS76] M. R. Garey, David S. Johnson, and Larry J. Stockmeyer. Some simplified np-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.
- [GJT76] M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is NP-complete. *SIAM J. Comput.*, 5(4):704–714, 1976.
- [GLR21] Esther Galby, Paloma T. Lima, and Bernard Ries. Reducing the domination number of graphs via edge contractions and vertex deletions. *Discret. Math.*, 344(1):112169, 2021.
- [GLW22a] Marc Goerigk, Stefan Lendl, and Lasse Wulf. Recoverable robust representatives selection problems with discrete budgeted uncertainty. *Eur. J. Oper. Res.*, 303(2):567–580, 2022.
- [GLW22b] Marc Goerigk, Stefan Lendl, and Lasse Wulf. Two-stage robust optimization problems with two-stage uncertainty. *Eur. J. Oper. Res.*, 302(1):62–78, 2022.
- [GLW24] Marc Goerigk, Stefan Lendl, and Lasse Wulf. On the complexity of robust multi-stage problems with discrete recourse. *Discret. Appl. Math.*, 343:355–370, 2024.
- [GMP23] Arun Ganesh, Bruce M. Maggs, and Debmalya Panigrahi. Robust algorithms for TSP and steiner tree. *ACM Trans. Algorithms*, 19(2):12:1–12:37, 2023.
- [GP25] Christoph Grüne and Femke Pfaue. A compendium of reductions: reductions.network, 2025. <https://reductions.network>.
- [Grü22] Christoph Grüne. The complexity classes of hamming distance recoverable robust problems. *CoRR*, abs/2209.06939, 2022.
- [Grü24] Christoph Grüne. The complexity classes of hamming distance recoverable robust problems. In José A. Soto and Andreas Wiese, editors, *LATIN 2024: Theoretical Informatics - 16th Latin American Symposium, Puerto Varas, Chile, March 18-22, 2024, Proceedings, Part I*, volume 14578 of *Lecture Notes in Computer Science*, pages 321–335. Springer, 2024.
- [GW23] Christoph Grüne and Lasse Wulf. Completeness in the polynomial hierarchy for many natural problems in bilevel and robust optimization. *CoRR*, abs/2311.10540, 2023.
- [GW24] Christoph Grüne and Lasse Wulf. On the complexity of recoverable robust optimization in the polynomial hierarchy. *CoRR*, abs/2411.18590, 2024.
- [GW25a] Christoph Grüne and Lasse Wulf. Completeness in the polynomial hierarchy for many natural problems in bilevel and robust optimization. In Nicole Megow and Amitabh Basu, editors, *Integer Programming and Combinatorial Optimization - 26th International Conference, IPCO 2025, Baltimore, MD, USA, June 11-13, 2025, Proceedings*, volume 15620 of *Lecture Notes in Computer Science*, pages 256–269. Springer, 2025.
- [GW25b] Christoph Grüne and Lasse Wulf. The complexity of blocking all solutions. *CoRR*, abs/2502.05348, 2025.
- [GW25c] Christoph Grüne and Lasse Wulf. On the complexity of recoverable robust optimization in the polynomial hierarchy. In Pawel Gawrychowski, Filip Mazowiecki, and Michal Skrzypczak, editors, *50th International Symposium on Mathematical Foundations of Com-*

- puter Science, *MFCS 2025, August 25-29, 2025, Warsaw, Poland*, volume 345 of *LIPICs*, pages 52:1–52:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.
- [GZ22a] Christoph Grüne and Stephan Zieger. Demand-responsive scheduling in railway transportation. In Jeroen Ploeg, Markus Helfert, Karsten Berns, and Oleg Gusikhin, editors, *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2022, Online Streaming, April 27-29, 2022*, pages 239–248. SCITEPRESS, 2022.
- [GZ22b] Christoph Grüne and Stephan Zieger. Solving the dial-a-ride problem for railway traffic by means of heuristics. In Cornel Klein, Matthias Jarke, Jeroen Ploeg, Markus Helfert, Karsten Berns, and Oleg Gusikhin, editors, *Smart Cities, Green Technologies, and Intelligent Transport Systems - 11th International Conference, SMARTGREENS 2022, and 8th International Conference, VEHITS 2022, Virtual Event, April 27-29, 2022, Revised Selected Papers*, volume 1843 of *Communications in Computer and Information Science*, pages 93–133. Springer, 2022.
- [Hal00] Magnús M. Halldórsson. Online coloring known graphs. *Electron. J. Comb.*, 7, 2000.
- [HIMT02] Magnús M. Halldórsson, Kazuo Iwama, Shuichi Miyazaki, and Shiro Taketomi. Online independent sets. *Theor. Comput. Sci.*, 289(2):953–962, 2002.
- [HKZ17a] Mikita Hradovich, Adam Kasperski, and Pawel Zielinski. Recoverable robust spanning tree problem under interval uncertainty representations. *J. Comb. Optim.*, 34(2):554–573, 2017.
- [HKZ17b] Mikita Hradovich, Adam Kasperski, and Pawel Zielinski. The recoverable robust spanning tree problem with interval costs is polynomially solvable. *Optim. Lett.*, 11(1):17–30, 2017.
- [HLW23] Hung P. Hoang, Stefan Lendl, and Lasse Wulf. Assistance and interdiction problems on interval graphs. *Discret. Appl. Math.*, 340:153–170, 2023.
- [HMMP23] Felix Hommelsheim, Nicole Megow, Komal Muluk, and Britta Peis. Recoverable robust optimization with commitment. *CoRR*, abs/2306.08546, 2023.
- [HMU07] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation, 3rd Edition*. Pearson international edition. Addison-Wesley, 2007.
- [HPR21] Hovhannes A. Harutyunyan, Denis Pankratov, and Jesse Racicot. Online domination: The value of getting to know all your neighbors. In Filippo Bonchi and Simon J. Puglisi, editors, *46th International Symposium on Mathematical Foundations of Computer Science, MFCS*, volume 202 of *LIPICs*, pages 57:1–57:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
- [HZ23] Kyle Hunt and Jun Zhuang. A review of attacker-defender games: Current state and paths forward. *European Journal of Operational Research*, 2023.
- [IS95] Masahiro Inuiguchi and Masatoshi Sakawa. Minimax regret solution to linear programming problems with an interval objective function. *European Journal of Operational Research*, 86(3):526–536, 1995.
- [Jer85] Robert G. Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Math. Program.*, 32(2):146–164, 1985.
- [JKZ24a] Marcel Jackiewicz, Adam Kasperski, and Pawel Zielinski. Computational complexity of the recoverable robust shortest path problem with discrete recourse. *CoRR*, abs/2403.20000, 2024.
- [JKZ24b] Marcel Jackiewicz, Adam Kasperski, and Pawel Zielinski. Recoverable robust shortest path problem under interval uncertainty representations. *CoRR*, abs/2401.05715, 2024.
- [Joh11] Berit Johannes. New classes of complete problems for the second level of the polynomial hierarchy. 2011.
- [Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer*

- Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972.*
- [KBB⁺08] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, Gábor Rudolf, and Jihui Zhao. On short paths interdiction problems: Total and node-wise limited interdiction. *Theory Comput. Syst.*, 43(2):204–233, 2008.
- [KKKS15] Dennis Komm, Rastislav Královic, Richard Královic, and Jasmin Smula. Treasure hunt with advice. In Christian Scheideler, editor, *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, volume 9439 of *Lecture Notes in Computer Science*, pages 328–341. Springer, 2015.
- [KLLS21] Thomas Kleinert, Martine Labbé, Ivana Ljubic, and Martin Schmidt. A survey on mixed-integer programming techniques in bilevel optimization. *EURO J. Comput. Optim.*, 9:100007, 2021.
- [KN16] Sascha Kurz and Stefan Napel. Dimension of the lisbon voting rules in the EU council: a challenge and new world record. *Optim. Lett.*, 10(6):1245–1256, 2016.
- [Kud15] Christian Kudahl. Deciding the on-line chromatic number of a graph with pre-coloring is pspace-complete. In Vangelis Th. Paschos and Peter Widmayer, editors, *Algorithms and Complexity - 9th International Conference, CIAC 2015, Paris, France, May 20-22, 2015. Proceedings*, volume 9079 of *Lecture Notes in Computer Science*, pages 313–324. Springer, 2015.
- [KY13] Panos Kouvelis and Gang Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 2013.
- [KZ15] Adam Kasperski and Pawel Zielinski. Robust two-stage network problems. In Karl F. Doerner, Ivana Ljubic, Georg Pflug, and Gernot Tragler, editors, *Operations Research Proceedings 2015, Selected Papers of the International Conference of the German, Austrian and Swiss Operations Research Societies (GOR, ÖGOR, SVOR/ASRO), University of Vienna, Austria, September 1-4, 2015*, Operations Research Proceedings, pages 35–40. Springer, 2015.
- [KZ16] Adam Kasperski and Pawel Zieliński. Robust discrete optimization under discrete and interval uncertainty: A survey. *Robustness analysis in decision aiding, optimization, and analytics*, pages 113–143, 2016.
- [KZ17] Adam Kasperski and Pawel Zielinski. Robust recoverable and two-stage selection problems. *Discret. Appl. Math.*, 233:52–64, 2017.
- [LA06] Vasilij Lebedev and Igor Averbakh. Complexity of minimizing the total flow time with interval data and minmax regret criterion. *Discrete Applied Mathematics*, 154(15):2167–2177, 2006.
- [LC93] Kao-Chêng Lin and Maw-Sheng Chern. The most vital edges in the minimum spanning tree problem. *Inf. Process. Lett.*, 45(1):25–31, 1993.
- [LG16] Nikolaos H Lappas and Chrysanthos E Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016.
- [LLB18] Richard Martin Lusby, Jesper Larsen, and Simon Bull. A survey on robustness in railway planning. *Eur. J. Oper. Res.*, 266(1):1–15, 2018.
- [LLMS09] Christian Liebchen, Marco E. Lübbecke, Rolf H. Möhring, and Sebastian Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In Ravindra K. Ahuja, Rolf H. Möhring, and Christos D. Zaroliagis, editors, *Robust and On-line Large-Scale Optimization: Models and Techniques for Transportation Systems*, volume 5868 of *Lecture Notes in Computer Science*, pages 1–27. Springer, 2009.
- [LLW21] Thomas Lachmann, Stefan Lendl, and Gerhard J. Woeginger. A linear time algorithm for the robust recoverable selection problem. *Discret. Appl. Math.*, 303:94–107, 2021.

- [LPT22] Stefan Lendl, Britta Peis, and Veerle Timmermans. Matroid bases with cardinality constraints on the intersection. *Math. Program.*, 194(1):661–684, 2022.
- [LR24] Felicia Lucke and Bernard Ries. On blockers and transversals of maximum independent sets in co-comparability graphs. *Discret. Appl. Math.*, 356:307–321, 2024.
- [LS17] Tao Li and Suresh P Sethi. A review of dynamic stackelberg game models. *Discrete & Continuous Dynamical Systems-B*, 22(1):125, 2017.
- [LY80] John M. Lewis and Mihalis Yannakakis. The node-deletion problem for hereditary properties is np-complete. *J. Comput. Syst. Sci.*, 20(2):219–230, 1980.
- [MDS14] Stephen J. Maher, Guy Desaulniers, and François Soumis. Recoverable robust single day aircraft maintenance routing problem. *Comput. Oper. Res.*, 51:130–145, 2014.
- [MG75] Lawrence S Mayer and IJ Good. Is minimax regret applicable to voting decisions? *American Political Science Review*, 69(3):916–917, 1975.
- [MMG89] Kavindra Malik, Ashok K Mittal, and Santosh K Gupta. The k most vital arcs in the shortest path problem. *Operations Research Letters*, 8(4):223–227, 1989.
- [NCH22] Adel Nabli, Margarida Carvalho, and Pierre Hosteins. Complexity of the multilevel critical node problem. *J. Comput. Syst. Sci.*, 127:122–145, 2022.
- [Paj20] Foad Mahdavi Pajouh. Minimum cost edge blocker clique problem. *Ann. Oper. Res.*, 294(1):345–376, 2020.
- [PBP14] Foad Mahdavi Pajouh, Vladimir Boginski, and Eduardo L. Pasiliao. Minimum vertex blocker clique problem. *Networks*, 64(1):48–64, 2014.
- [Pfa24] Femke Pfaue. Exploring the reductions between ssp- np -complete problems and developing a compendium website displaying the results. *CoRR*, abs/2411.05796, 2024.
- [Ple79] Ján Plesník. The NP-completeness of the hamiltonian cycle problem in planar digraphs with degree bound two. *Inf. Process. Lett.*, 8(4):199–201, 1979.
- [PPR16] Daniël Paulusma, Christophe Picouleau, and Bernard Ries. Reducing the clique and chromatic number via edge contractions and vertex deletions. In Raffaele Cerulli, Satoru Fujishige, and Ali Ridha Mahjoub, editors, *Combinatorial Optimization - 4th International Symposium, ISCO 2016, Vietri sul Mare, Italy, May 16-18, 2016, Revised Selected Papers*, volume 9849 of *Lecture Notes in Computer Science*, pages 38–49. Springer, 2016.
- [PPR17] Daniël Paulusma, Christophe Picouleau, and Bernard Ries. Blocking independent sets for H-free graphs via edge contractions and vertex deletions. In T. V. Gopal, Gerhard Jäger, and Silvia Steila, editors, *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 470–483, 2017.
- [PWBP15] Foad Mahdavi Pajouh, Jose L. Walteros, Vladimir Boginski, and Eduardo L. Pasiliao. Minimum edge blocker dominating set problem. *Eur. J. Oper. Res.*, 247(1):16–26, 2015.
- [PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theor. Comput. Sci.*, 84(1):127–150, 1991.
- [Rut93] Vladislav Rutenburg. Propositional truth maintenance systems: Classification and complexity analysis. *Ann. Math. Artif. Intell.*, 10(3):207–232, 1993.
- [Sip97] Michael Sipser. *Introduction to the theory of computation*. PWS Publishing Company, 1997.
- [SPG13] J Cole Smith, Mike Prince, and Joseph Geunes. Modern network interdiction problems and algorithms. In *Handbook of combinatorial optimization*, pages 1949–1987. Springer New York, 2013.
- [SSM⁺21] Mohammad H Shams, Majid Shahabi, Mohammad MansourLakouraj, Miadreza Shafiekhah, and João PS Catalão. Adjustable robust optimization approach for two-stage operation of energy hub-based microgrids. *Energy*, 222:119894, 2021.
- [Sto76] Larry J. Stockmeyer. The polynomial-time hierarchy. *Theor. Comput. Sci.*, 3(1):1–22, 1976.

- [SU08] Marcus Schaefer and Christopher Umans. Completeness in the polynomial-time hierarchy a compendium. 2008.
- [TA18] D.D. Tönissen and J.J. Arts. Economies of scale in recoverable robust maintenance location routing for rolling stock. *Transportation Research Part B: Methodological*, 117:360–377, 2018.
- [TCCH24] Alberto Boggio Tomasaz, Margarida Carvalho, Roberto Cordone, and Pierre Hosteins. On the completeness of several fortification-interdiction games in the polynomial hierarchy. *CoRR*, abs/2406.01756, 2024.
- [Tra84] Boris A. Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *IEEE Ann. Hist. Comput.*, 6(4):384–400, 1984.
- [Tse83] Grigori S Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning: 2: Classical papers on computational logic 1967–1970*, pages 466–483. Springer, 1983.
- [TSSW96] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming (extended abstract). In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 617–626. IEEE Computer Society, 1996.
- [Tur37] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proc. London Math. Soc.*, s2-42(1):230–265, 1937.
- [WF24] Noah Weninger and Ricardo Fukasawa. A fast combinatorial algorithm for the bilevel knapsack problem with interdiction constraints. *Mathematical Programming*, pages 1–33, 2024.
- [WHM06] Jean-Paul Watson, William E Hart, and Regan Murray. Formulation and optimization of robust sensor placement problems for contaminant warning systems. In *Water Distribution Systems Analysis Symposium 2006*, pages 1–13, 2006.
- [Woe21] Gerhard J. Woeginger. The trouble with the second quantifier. *4OR*, 19(2):157–181, 2021.
- [Woo93] R. Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.
- [Yan78] Mihalis Yannakakis. Node- and edge-deletion np-complete problems. In Richard J. Lipton, Walter A. Burkhard, Walter J. Savitch, Emily P. Friedman, and Alfred V. Aho, editors, *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, May 1-3, 1978, San Diego, California, USA*, pages 253–264. ACM, 1978.
- [YGdH19] İhsan Yanıkoğlu, Bram L Gorissen, and Dick den Hertog. A survey of adjustable robust optimization. *Eur. J. Oper. Res.*, 277(3):799–813, 2019.
- [YT09] Shangyao Yan and Ching-Hui Tang. Inter-city bus scheduling under variable market share and uncertain market demands. *Omega*, 37(1):178–192, 2009.
- [Zen10] Rico Zenklusen. Matching interdiction. *Discret. Appl. Math.*, 158(15):1676–1690, 2010.