Christina Büsing

Recoverable Robustness in Combinatorial Optimization

Cuvillier Verlag Göttingen
Internationaler wissenschaftlicher Fachverlag

Recoverable Robustness in Combinatorial Optimization

vorgelegt von
Diplom-Mathematikerin
Christina Büsing
aus Bonn

Von der Fakultät II - Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
Dr. rer. nat.

genemigte Dissertation

Vorsitzender: Prof. Dr. Fredi Tröltzsch
Berichter: Prof. Dr. Rolf H. Möhring
Prof. Dr. Peter Widmayer

Tag der wissenschaftlichen Aussprache: 10. Dezember 2010

Berlin, 2010
D 83

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

1. Aufl. - Göttingen: Cuvillier, 2011
Zugl.: (TU) Berlin, Univ., Diss., 2010

978-3-86955-771-7

Nonnenstieg 8, 37075 Göttingen
Telefon: 0551-54724-0
Telefax: 0551-54724-21
www.cuvillier.de

1. Auflage, 2011
Gedruckt auf säurefreiem Papier

978-3-86955-771-7

Preface

This work would not have been possible without the support of several thoughtful and generous people. First of all I very much like to thank my supervisor Prof. Dr. Rolf Möhring for trusting me in choosing my research topic, for discussing problem settings and first approaches whenever I needed some feedback and for giving me the opportunity to visit workshops, conferences and other research groups for further inspirations. Financially this research was supported by the research training group "Methods for Discrete Structures" and the Berlin Mathematical School.

I am very grateful for all my colleagues at TU Berlin. Without the constant encouragement by Wiebke Höhn and Dr. Mareike Massow, the cheerful nature of Jens Schulz and the many discussions with Dr. Tobias Harks and Max Klimm, I would never have finished this thesis. Furthermore, I like to thank Prof. Dr. Andreas Bley, Dr. Janina Brenner, Martin Fuchsberger, Elisabeth Günther, Martin Groß, Ronald Koch, Prof. Dr. Ekkehard Köhler, Dr. Felix König, Dr. Nicole Megow, Daria Schymura, Madeleine Theile, José Verschae and Andreas Wiese for various inspirations, cups of coffee and proofreading of my thesis.

In particular, I wish to thank all the people I did intensive research with. Jens Maue from the ETH Zürich introduced me to the topic of train classification and we spent several days and nights in his office to tackle this problem. Prof. Dr. Peter Widmayer welcomed me in his research group and gave important feedback on these results. Prof. Dr. Arie Koster and Manuel Kutschka from the RWTH Aachen provided me with their knowledge of the knapsack problem and with the implementations of our theoretical results. Last December I spent one wonderful week in Zürich working with Dr. Rico Zenklusen who brought me, among other things, to the topic of Matroids.

Finally, I thank all of my friends and my family for their support and endless discussions, in particular, Henrik Büsing who always listened to my complains but also shared many happy moments with me.

Christina Büsing
Berlin, October 2010

Contents

Introduction 1

1. Dealing with Uncertainties in a Robust Way 9
- 1.1. Robustness . . . 10
  - 1.1.1. Robust Linear Programming . . . 10
  - 1.1.2. Robust Combinatorial Optimization . . . 12
- 1.2. Extending Robustness to Recoverable Robustness . . . 15

2. k-Distance Recoverable Robustness 19
- 2.1. Introduction . . . 19
- 2.2. Discrete Scenarios . . . 23
  - 2.2.1. Weighted Disjoint Hitting Set Problem . . . 29
- 2.3. Interval Scenarios . . . 32
  - 2.3.1. Weighted Disjoint Hitting Set Problem . . . 33
  - 2.3.2. Minimum Weight Basis Problem for Matroids . . . 35
- 2.4. Γ-scenarios . . . 37
- 2.5. Shortest Path Problem . . . 38
  - 2.5.1. Simple (s,t)-Paths as First-Stage Solutions . . . 39
  - 2.5.2. (s,t)-Paths as First-Stage Solutions . . . 41
- 2.6. Conclusion and Open Issues . . . 48

3. Rent Recoverable Robustness 51
- 3.1. Introduction . . . 51
- 3.2. Discrete Scenarios . . . 53
- 3.3. Interval Scenarios . . . 55
- 3.4. Γ-Scenarios . . . 56
- 3.5. Approximation via Robust Solutions . . . 58
- 3.6. Conclusion and Open Issues . . . 60

4. Exact Subset Recoverable Robustness 61
- 4.1. Introduction . . . 61
- 4.2. Discrete Scenarios . . . 64
- 4.3. Interval Scenarios . . . 67
  - 4.3.1. Shortest Path Problem . . . 71
  - 4.3.2. Minimum (s,t)-Cut Problem . . . 72
- 4.4. Γ-scenarios . . . 73
  - 4.4.1. Weighted Disjoint Hitting Set Problem . . . 74
- 4.5. Matroids . . . 75
  - 4.5.1. Discrete Scenarios . . . 76
  - 4.5.2. Interval Scenarios . . . 77
  - 4.5.3. Γ-scenarios . . . 79
- 4.6. Approximation Algorithms . . . 81

4.7. Conclusion and Open Issues 83

5. A Recoverable Robust Knapsack Problem **85**
5.1. Introduction 85
5.2. Discrete Scenarios 88
5.2.1. Complexity of the (k, ℓ)-rrKP 89
5.2.2. Extended Cover Inequalities 97
5.2.3. Computational Experiments 102
5.3. Interval Scenarios 107
5.4. Γ-Scenarios 107
5.4.1. An Optimal Γ-Strategy 108
5.4.2. A Polynomial Size ILP-Formulation 111
5.4.3. Extended Cover-Inequalities 113
5.5. Conclusion and Open Issues 117

6. Recoverable Robust Train Classification **119**
6.1. Introduction 119
6.2. Encoding Classification Schedules 121
6.3. Recovery by Additional Sorting Steps 123
6.3.1. Generic Algorithm 125
6.3.2. Computational Complexity 127
6.3.3. Problem Variants 129
6.4. Limited Number of Delayed Trains 129
6.4.1. Experimental Evaluation 133
6.5. Types in Fixed Order 135
6.6. Conclusion and Open Issues 137

Bibliography **139**

A. Max-Scenario Problems **145**
A.1. Shortest Path Problem 145
A.2. Minimum (s, t)-Cut Problem 148
A.3. Weighted Disjoint Hitting Set Problem 150

B. Cardinality Constrained Minimum (s, t)-Cut Problem **153**

Zusammenfassung (German) **155**

Introduction

Combinatorial optimization is an important tool for solving optimization problems from industry like vehicle routing, network design or production scheduling. To define such an optimization problem, data concerning the cost, the constraints on the solutions or the topology of the networks are assumed to be known. However, these data can often only be estimated based on imprecise measuring methods or predictions of future events (development of the stock markets, change of weather conditions, variations in traffic volume). In several applications, average values from historical data adjusted by some anticipated changes are used to determine the problem setting.

An attractive approach for dealing with these variations in data is to include different data sets into the optimization process. Many researchers have selected a scenario approach, where each scenario represents a reasonable data set. Depending on the considered setting and the available information, such a set of data sets is equipped with a probability distribution to reflect the likelihoods of the scenarios.

There are two major trends in dealing with uncertainty given by a scenario set: stochastic programming and robust optimization. The goal in *stochastic programming* is to find some solution that is feasible in almost all considered scenarios and minimizes some stochastic function like the expected cost. This approach is only applicable if a probability distribution of the scenario set is known or can be estimated. Minimizing the expected cost is reasonable if the process is repeated several times and the same solution is chosen in each iteration. Furthermore, the approach assumes some flexibility in the realization process, since a solution may turn infeasible, and it assumes reasonable cost in the unlikely worst-case. These conditions are not satisfied when dealing with high risk situations or with basic services like planning water and power supply networks. In these settings a robust approach is more appropriate.

Robust optimization provides a high level of security but represents a rather risk-averse attitude. A solution is called *robust* if it remains feasible under all considered scenarios. The task in robust optimization is to find a robust solution that minimizes its worst-case cost. The difficulty in robustness is that feasibility in all realizations is quite demanding and may not be achieved by any solution. But even if a robust solution exists, it may generate high cost in many scenarios which is not representative for other solutions. These drawbacks have already be discussed when robustness was first applied to linear programming by Soyster [92] in 1973.

To address these concerns, Ben-Tal and Nemirovski [12], El Ghaoui et al. [40, 41], Bertsimas and Sim [15] and Bienstock [16] introduced for linear programs new types of scenario sets. They showed in particular that linear programs under their scenario sets remain tractable. Kouvelis and Yu [78] started to consider robust combinatorial optimization problems with discrete scenario sets and proved several hardness results. A famous theorem by Bertsimas and Sim [14] states that for every polynomially solvable 0-1 discrete optimization problem its robust version with so-called Γ-scenarios can also be solved in polynomial time. Note that all settings for combinatorial optimization problems assume that the values of the objective function are subject to variations, but that the set of feasible solutions remains unchanged.

Despite the progress in defining scenario sets, several researchers felt the need to relax the concept of robustness. Inspired by the idea of a recourse in two-stage stochastic programming, they defined a two-stage procedure that allows a solution chosen under uncertainty in a first stage to be modified by previously fixed means as soon as all data are known. As in the robust setting, the worst-case cost of such a solution is minimized. This idea was introduced by several groups, e.g., Ben-Tal et al. [11], Dhamdhere et al. [37] or Liebchen et al. [81], under the names of adaptive robustness, demand robustness or two-stage robustness, and recoverable robustness, respectively. In this thesis we will call this concept *recoverable robustness*.

Recoverable robust linear programs have been studied in [11, 81, 95]. One drawback of this setting is the increase in complexity, since most of these problems are strongly **NP**-hard. From a practical point of view, recoverable robustness turns out to be an important tool for modeling real world problems gaining valuable insights into the scope of applicability of optimization. For example, its application to several railway optimization problems is analyzed in [25, 26, 30, 81]. In terms of combinatorial optimization, the main focus of research activities focused on scenario sets modeling failures of the underlying topology or uncertainties on the side constraints like the demands (e.g., [37, 44, 55, 73]). The objective function in these cases is assumed to be fixed.

This thesis focuses on two aspects of recoverable robustness. The first one is motivated by a theoretical interest:

- How to define useful recoverable robust combinatorial optimization problems for scenario sets modeling uncertainties in the objective function?
- What kind of changes in respect to complexity can be observed compared to robust models?
- Which combinatorial structures or properties can be detected within the recoverable robust model?

The second aspect is a practical one and emphasizes on how a recoverable robust approach can be adapted to practical problems, e.g., what kind of scenario sets to consider and how to define the recovery actions.

Chapter 1 contains a short introduction to the concept of robustness, important results in this area and their extension to recoverable robustness. The following three Chapters 2, 3 and 4 are dedicated to theoretical issues and Chapters 5 and 6 to more practical applications. In the Appendices A and B, we investigate two combinatorial subproblems of the recoverable robust models introduced in Chapters 2, 3 and 4, in order to give a deeper understanding of the results obtained there. Apart from the definition of different scenario types in Chapter 1, all chapters are self-contained and thus can be read independently of each other. In the remainder of this introduction, we give a short overview of each chapter.

Chapter 1: Dealing with Uncertainties in a Robust Way This chapter contains the history of robustness and its extension to recoverable robustness with the main results in linear programming and combinatorial optimization. In particular we give the definition of three different types of scenario sets, which we will investigate throughout this thesis: the so-called discrete scenario sets, the interval scenario sets and the Γ-scenario sets. Note that we mainly consider uncertainties occurring in the cost function of a given optimization problem.

The considered types of scenario sets differ in the way they are modeled in the input, i.e., whether they are defined explicitly or implicitly, and by further restrictions on the values of the cost functions. In a *discrete scenario set* each scenario and its integer cost function is

explicitly given. *Interval scenario sets* consist of all scenarios that determine a cost function whose values lie in a given cost interval defined by lower and upper cost bounds. For some integer Γ, Γ-*scenario sets* are modifications of interval scenario sets. In contrast to interval scenarios, a Γ-scenario may change only Γ cost values from the lower bound, in the worst-case to the corresponding upper bound.

Chapter 2: k-Distance Recoverable Robustness This chapter covers a recoverable robust approach to combinatorial optimization which is probably the most natural: the *k-distance recoverable robustness* approach. Let us consider a linear combinatorial minimization (LCMin) problem such as the shortest path problem, and a set of scenarios, each defining a cost function, e.g., in the shortest path setting different traveling times. In the k-distance recoverable robust model, a solution is chosen in the first stage, e.g., a simple path. As soon as the scenario is revealed, i.e., once the exact data for the cost function is known, we are allowed to choose a solution that differs just a "little" from the first one. In the shortest path setting one can think of a path using small detours compared to the originally chosen path. In general the difference is measured by the number of new elements contained in the second solution. This number is bounded by some integer k, also called *recovery parameter*. The goal in the k-distance recoverable robust setting is to find a solution in the first stage with minimum total cost. This cost is composed of the so-called first-stage cost and the maximum cost occurring in a scenario for the chosen second path.

We investigate the k-distance recoverable robust version of several well-known combinatorial optimization problems, e.g., the shortest path problem, the minimum spanning tree problem and the minimum perfect matching problem, in combination with the three basic scenario sets defined above, namely discrete scenario sets, interval scenario sets and Γ-scenario sets. For discrete scenarios we start by analyzing the k-Dist-RR version of a quite simple LCMin problem, the weighted disjoint hitting set (WDHS) problem. This problem represents a special case of the deterministic shortest path problem, the minimum (s,t)-cut problem, the minimum perfect matching problem and the minimum spanning tree problem. For two scenarios we show that the k-distance recoverable robust (k-Dist-RR) version of this WDHS problem is weakly **NP**-hard. We also present a pseudo-polynomial algorithm with a run-time depending on a constant number of scenarios and the values of the scenario cost functions. If the number of scenarios is not constant, the problem turns out to be strongly **NP**-hard. In a special case, a lower bound of 1.5 (later improved independently to a bound of 2 by Kasperski and Zieliński [70]) on the best possible approximation factor is achieved, unless $\mathbf{P} = \mathbf{NP}$. The hardness results can easily be transferred to all the problems of which the WDHS problem is a subproblem.

Considering interval scenarios, where the cost function of each scenario is bounded by box-constraints, the complexity status varies between not approximable and solvable in polynomial time. On one hand, the k-Dist-RR shortest path problem is strongly **NP**-hard and cannot be approximated, unless $\mathbf{P} = \mathbf{NP}$. This is in stark contrast to the case where a not necessarily simple (s,t)-path needs to be chosen in a first stage, and the recovery parameter k is a constant, since then the problem is tractable. Also the k-Dist-RR version of the minimum weight basis problem for matroids is solvable in polynomial time for constant k. For the WDHS problem, it can be solved efficiently even if k is not constant but part of the input.

Finally, we analyze Γ-scenarios. By a reduction from the closely related max-scenario problem considered in Appendix A, we show that the k-Dist-RR shortest path problem and the k-Dist-RR minimum (s,t)-cut problem become strongly **NP**-hard.

Chapter 3: Rent Recoverable Robustness Rent recoverable robustness focuses on situations where a choice of an element in the first stage lowers the cost to actually purchase this element in the second stage. The idea is similar to option dealing and the right of first refusal. Let U be a set of elements and $\mathcal{F}$ be a set of feasible solutions, where the goal is to possess a cheap feasible solution in the second stage. In the first stage some elements can be rented. We assume that the rental cost is a fraction of the cost for the elements revealed in the second stage. If we rent an element $u \in U$, in scenario S we need to pay the rent cost of $\alpha \cdot c^S(u)$, where $c^S : U \to \mathbb{N}$ is a cost function defined by S and $0 < \alpha \leq 1$ is a previously fixed *rental factor*. To purchase the element in the second stage we just need to pay the remaining cost $(1-\alpha)c^S(u)$. On the other hand, if we did not rent the element before, we have to pay the original cost and additional inflation cost, i.e., $(1+\beta)c^S(u)$ for some *inflation factor* $\beta \geq 0$. To sum up, an element u may produce cost 0 or $\alpha c^S(u)$ or $c^S(u)$ or $(1+\beta)c^S(u)$. The task in a rent recoverable robust problem is to determine a set of elements for rent, such that the maximum cost for purchasing a solution over all scenarios is minimized. In contrast to the k-Dist-recoverable robust model, a first-stage solution does not limit the set of solutions we can choose from in the second stage.

We show that the rent recoverable robust version of all combinatorial optimization problems that contain the WDHS problem, is weakly **NP**-hard for two scenarios. If the number of scenarios is not constant, the same problems are not approximable with a factor better than 2, unless $\mathbf{P} = \mathbf{NP}$. Considering interval scenarios, a rent recoverable robust problem is solvable in polynomial time if the underlying combinatorial optimization problem is in **P**. In the case of Γ-scenarios the complexity of the rent recoverable robust problem depends on the complexity of the max-scenario problem, i.e., if the max-scenario problem is strongly **NP**-hard and some technical details are fulfilled then the rent recoverable robust version is also strongly **NP**-hard.

In the last section we provide an approximation algorithm, which depends on a robust solution. If the robust solution is a γ-approximation of the corresponding robust problem, we obtain a $\min\{\gamma + 1 + \beta, \frac{\gamma}{\alpha}\}$-approximation for the rent recoverable robust version with a given rental factor α and an inflation factor β. In general, such relations between robust and recoverable robust solutions cannot be achieved, since a robust solution may produce arbitrarily high total cost like it is the case for k-Dist-recoverable robust problems.

Chapter 4: Exact Subset Recoverable Robustness In network design the words robustness and stability are often used interchangeably. In that context a network is called robust/stable if its task is not influenced by intentional or random attacks. For example, in telecommunication networks the demand should remain routable if certain links fail. In general, a network is more robust if it contains redundant links. On the other hand, maintaining such a network is rather costly. In order to reduce the cost, one is interested in abandoning as many links as possible without losing stability. Yet, this approach does not take into account the needs of the customers. A customer is generally not only interested that his requests are routed but also that they are routed as fast as possible.

Exact subset recoverable robustness concentrates on this second requirement. Assuming uncertainties given in the cost function, e.g., in the routing times, the task is to find a subnetwork with minimum size guaranteeing that in every scenario the considered request is routed in the subnetwork as fast as in the original network according to the realized cost function. This problem can be easily extended to all combinatorial optimization problems and interpreted as finding a small set of elements such that this set always contains an optimal solution w.r.t. the original instance.

Starting again with discrete scenario sets, we show strong **NP**-hardness for the exact subset recoverable robust version of the minimum spanning tree problem, the minimum shortest path problem and the minimum perfect matching problem. The key difficulty lies in the subproblem of choosing one element out of a given set with minimum cost. This problem is a special case of the weighted disjoint hitting set problem, which was investigated already in different chapters.

For interval scenario sets the problem becomes more interesting. We develop a general criterion to decide in which case a given element needs to be part of any feasible solution. If a considered linear combinatorial optimization problem is in **P**, we can use this criterion to show that its exact subset recoverable robust version is in **coNP**. Furthermore, we use the criterion in two directions: on the one hand, we prove that the exact subset recoverable robust version of the shortest path problem and the minimum (s,t)-cut problem are not approximable within a factor of $|A|^{(1-\varepsilon)}$ on a directed graph $G=(V,A)$ for any $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$. On the other hand, we derive an algorithm for solving the exact subset recoverable robust minimum weight basis problem on matroids.

The case of Γ-scenarios is again closely related to the max-scenario problem, and thus it follows that the exact subset recoverable robust versions of the shortest path problem and the minimum (s,t)-cut problem are strongly **NP**-hard. But even for matroids the problem turns out to be more difficult. The exact subset recoverable robust minimum spanning tree problem contains as a subproblem the k-connected minimum subgraph problem and can therefore not be solved in polynomial time, unless $\mathbf{P} = \mathbf{NP}$. Finally, we introduce an approximation scheme for interval and Γ-scenarios. This chapter is based on joint work with Rico Zenklusen. In [22] results on the exact subset recoverable robust shortest path problem are published.

Chapter 5: A Recoverable Robust Knapsack Problem An important task in telecommunication is to assign bandwidth to different customers, maximizing the profit for the company. In many cases the demand of the customers varies, e.g., the source, the destination and the traffic volume. Hence, not enough capacity may be available at the point of realization, although service was granted beforehand. To obtain a trade-off between the loss of benefit due to unused resources and the loss in reputation, we allow in our model violations of up to k service promises and new service offers for up to ℓ new requests while maximizing the profit. Applying this approach to a single link of a telecommunication network leads to a recoverable robust version of the knapsack problem. Uncertainties are given in the profit function and in the weight function. As recovery action, k items of the first-stage solution may be deleted and ℓ items may be added, as soon as a scenario reveals its profits and the weights.

Our main focus in this chapter is to obtain similar results as for the classical (robust) knapsack problem. We start with an investigation of the complexity status for discrete scenario sets. As in the robust and deterministic case, the (k,ℓ)-recoverable robust knapsack ((k,ℓ)-rrKP) problem is weakly **NP**-hard and can be solved in pseudo-polynomial time via a dynamic program if the number of scenarios is constant. If this is not the case, the problem is strongly **NP**-hard and in some cases even not approximable, unless $\mathbf{P} = \mathbf{NP}$. occur to

In addition to its complexity status, we are interested in obtaining strongly polyhedral descriptions for this problem. We thus generalize the well-known concept of covers to gain valid inequalities for the recoverable robust knapsack polytope. Besides the canonical extension of covers we introduce a second kind of extension exploiting the scenario-based problem structure and producing stronger valid inequalities. Furthermore, we present two computational studies to investigate the influence of parameters k and ℓ to the objective and evaluate the effectiveness of our new class of valid inequalities.

The (k, ℓ)-rrKP problem with interval scenarios is a special case of the setting with just one discrete scenario. For Γ-scenarios this is not the case. We start by investigating the so-called maximum weight set problem and introduce a combinatorial algorithm for computing for a given set of items the scenario that induces the maximum weight on this set after the recovery is applied. Using this result we introduce an IP formulation for the (k, ℓ)-rrKP problem with Γ-scenarios and no scenario profit whose size is polynomial in the size of the knapsack instance. Note that the set of Γ-scenarios contains an exponential number of different scenarios.

As for discrete scenarios, we adapt cover inequalities and strengthen previous formulations introduced for the robust knapsack polytope by Klopfenstein and Nace [76]. Furthermore, we give an optimal pseudo-polynomial algorithm for solving the corresponding separation problem. This chapter is joint work with Arie Koster and Manuel Kutschka [23].

Chapter 6: Recoverable Robust Train Classification Train classification is an important task in railway optimization, in which a set of given freight trains is sorted to form new trains. In general these trains arrive according to a previously known order at the so-called classification yard, where the cars are decoupled such that they can be sorted. The sorting is performed by moving the cars over a hump, collecting them at some receiving track and pulling them out again if necessary, until the desired train is formed. Since trains are often delayed, the expected order of cars changes and a previously determined sorting schedule becomes infeasible. Traditional classification methods deal with this problem by assuming a worst-case scenario, i.e., that all cars arrive in reversed order, with the drawback of using more sorting steps than necessary.

In our recoverable robust model, we assume that we can interfere the sorting process after an offset of p sorting steps to add k new sorting steps. But, we expect that a recoverable robust schedule sorts the incoming trains into the desired outgoing train if no delay occurs. In the case of disturbances, feasibility of the schedule is reobtained by using the described recovery means. The parameters p and k model the trade-off between robustness and rescheduling. For large p and small k almost no changes happen to a schedule fixed in the first stage, for small p and large k many changes are possible.

We start by introducing a generic algorithm for computing a recoverable robust train classification schedule. For the special scenario set in which each scenario delays up to j trains, this algorithm can be implemented in polynomial time. Yet, in general a special **NP**-hard subproblem needs to be solved, which also induces **NP**-hardness on the recoverable robust train classification problem for $k \geq 1$. In experiments on real-world traffic data we further explore the trade-off underlining that our algorithm saves sorting steps in comparison to traditional methods even for small recovery actions. The results in this chapter are joint work with Jens Maue and partly published in [24].

Appendix A: Max-Scenario Problems The max-scenario problem is an important subproblem of several recoverable robust settings. Given a combinatorial optimization problem and a set of scenarios, the task of this problem is to find a scenario that maximizes the minimum cost of a feasible solution. For discrete scenario sets and interval scenario sets the problem is easy to solve. However, for Γ-scenarios we show that the max-scenario problem for the shortest path problem and the minimum (s, t)-cut problem become strongly **NP**-hard. Using this result, we can show in several recoverable robust settings that recoverable robust versions of these two problems are at least strongly **NP**-hard.

Appendix B: Cardinality Constrained Minimum (s,t)-Cut Problem The upper bounded cardinality constrained minimum (s,t)-cut problem asks for a minimum (s,t)-cut in a graph with bounded cardinality such that its cost is minimized. To the best of our knowledge the complexity status of this problem was open, e.g., stated in [21]. We show that the problem is strongly **NP**-hard. As a consequence, the total cost for an (s,t)-cut cannot be computed in polynomial time in the k-distance recoverable robust model. This result is joint work with Rico Zenklusen.

1. Dealing with Uncertainties in a Robust Way

The definition of an optimization problem requires the specification of several parameters, as for example, the coefficient in an objective function or constraints on the set of solutions. Often these parameters are subject to uncertainties, especially when dealing with models for real-world problems: Data may be incomplete or erroneous due to imprecise methods for measurement or may be unknown since they depend on future trends. The actual parameters which will be realized in the future can just be approximated since several such realizations are reasonable. Models integrating the idea of different future settings are normally based on a scenario approach, where a set of scenarios is given and each scenario corresponds to an assignment of plausible values to the model parameters.

In recent years, incorporating scenarios into deterministic models has generated much research activity due to their importance for practical problems. Two models are widely used: stochastic programming and robust optimization. In *stochastic programming*, one assumes to have perfect knowledge of the probability distribution of the scenarios and optimizes some stochastic variant of the original objective function, e.g., the expected value. In low risk situations the expected value or some other stochastic function is appropriate. But in high risk situations this choice is out of place, since these functions ignore the risk disposition of the decision maker. For detailed description of stochastic programming see [18].

Robust optimization reflects a natural, high risk averse attitude towards uncertainty: when in doubt, assume the worst. Thus, a solution is called *robust* if it remains feasible for any possible realization of a scenario and minimizes the worst-case cost. The performance of such a solution is guaranteed within the considered scenario set. The drawback of robustness is the unacceptably high cost of an optimal solution. The concept also ignores the fact that in most settings minor changes to the previously determined solution are possible. To overcome these drawbacks researchers focused on the development of different scenario sets and included the idea of recovery presented in stochastic programming.

In the following sections we describe and discuss the basics of robust optimization. Finally, we introduce several related methods to extend the concept of robustness to so-called *recoverable robustness*.

We will not give an introduction into the main concepts of combinatorial optimization here. We assume the reader to be familiar with basic graph theory terms such as graphs, nodes, arcs, paths, and with **NP**-hardness and definitions of well-known problems such as the minimum (s,t)-cuts problem, the minimum cover problem, the minimum spanning tree problem and the shortest path problem. As a reference, we recommend Korte and Vygen's book "Combinatorial Optimization: Theory and Algorithms" [77]. For a detailed introduction to complexity theory and the definition of several **NP**-hard problems we refer to Garey and Johnson's book "Computers and Intractability: A Guide to the Theory of **NP**-Completeness" [52].

1.1. Robustness

A robust approach is crucial when dealing with high risk events, e.g., in the scheduling of aircrafts for U.S. military operations [83], or sensor placement when planning contaminant warning system for water distribution networks [96]. In such settings, standard approaches like deterministic optimization or stochastic programming fail to hedge against severe attacks with unpredictable consequences. Furthermore, robustness is appropriate whenever a predefined goal needs to be reached under any variation of the input data [88].

We start with a historical overview of the development of robustness and different scenario sets in linear programming. In a second part, we state the main results in robust combinatorial optimization.

1.1.1. Robust Linear Programming

The Robust Formulation of Soyster In 1973, Soyster [92] was the first to investigate a robust approach for linear programming. He considers uncertainties to be given column-wise and seeks for a solution that remains feasible independent of the input data. One important advantage of this approach is that no probability distribution needs to be known. Furthermore, Soyster shows that if the set of uncertainties is given as convex sets, the problem reduces to a simple linear program and thus remains tractable.

More formally, let $c \in \mathbb{R}^n$ be a cost vector, $b \in \mathbb{R}^m$ be the right-hand side and $A^0 \in \mathbb{R}^{m \times n}$ be an $(m \times n)$-matrix. The *nominal linear program*, i.e., the linear program without uncertainties, on a set of variables $x \in \mathbb{R}^n$ is defined as

$$\begin{array}{rcl} \min c^\top x & & \\ A^0 x & \leq & b \\ x & \geq & 0. \end{array}$$

In the inexact linear program formulation of Soyster, each column of the matrix A^0 is subject to uncertainties. Note that in linear programming, variations in the cost function or in the right-hand side can easily be transferred to the case where just the coefficients of the matrix are effected. The uncertainties of A^0 are modeled via n convex sets $K_j \subseteq \mathbb{R}^m$, $j = 1, \ldots, n$, where each feasible realization chooses one column vector $a_j \in K_j$. In a scenario based formulation, each feasible scenario S defines an $(m \times n)$-matrix A^S such that we have $a_j^S \in K_j$ for each column a_j^S, $j = 1, \ldots, n$. The *inexact linear program*, or more general the *robust counterpart* of the LP, is then given by

$$\begin{array}{rcll} \min c^\top x & & & \\ A^S x & \leq & b & \forall S \in \mathcal{S} \\ x & \geq & 0, & \end{array}$$

where $\mathcal{S}$ denotes the set of feasible scenarios. As Soyster proved, an optimal solution for this problem is obtained by solving a nominal linear program for the $(m \times n)$-matrix $\overline{A}$ with $\overline{a}_{ij} = \max_{a_j \in K_j}\{a_{ij}\}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$.

Soyster's approach received little attention, since the solutions tend to be over-conservative. Only Falk (e.g. [43]) and Soyster (e.g. [93]) published short notes on the topic of inexact linear programs.

Ellipsoidal Uncertainty Sets In the 1990s Ben-Tal and Nemirovski [12] as well as El Ghaoui et al. [40, 41] independently started to approach the over-conservatism by considering row-wise uncertainties where the coefficient of each row lay within an ellipsoidal set. Row-wise uncertainties reflect the fact that usually not all coefficients of a row turn to their worst-case values simultaneously. There are several good reasons to consider ellipsoidal sets. They include a wide family of uncertainty sets, e.g., sets defined by a polytope, and they approximate more complicated uncertainty sets. In addition, they can be described parametrically by data of moderate size. Constraint-wise *ellipsoidal uncertainty* sets $\mathcal{S}_E$ are given in the following form: let $a_i^0 \in \mathbb{R}^n$ be the nominal value of the ith row, $i = 1, \ldots, m$, and let $\Delta_i \in \mathbb{R}^{n \times n}$ be some perturbation matrix of the ith row. Then a scenario $S \in \mathcal{S}_E$ defines a *feasible realization* $A^S = (a_1^S, \ldots, a_m^S)$ if there exists for $i = 1, \ldots, m$ a vector $u^i \in \mathbb{R}^n$ such that $||u^i||_2 \leq 1$ and $a_i^S = a_i^0 + \Delta_i u^i$, where $|| \cdot ||_2$ is the euclidean norm. As it turns out, the robust counterpart of a linear program with ellipsoidal uncertainty sets $\mathcal{S}_E$ is equivalent to a second-order cone program

$$\begin{array}{rcll} \min c^\top x & & & \\ a_i^0 x & \leq & b_i - ||\Delta_i x||_2 & \forall i = 1, \ldots, m \\ x & \geq & 0. & \end{array}$$

This problem can be solved in polynomial time by interior point methods.

In order to test their robust approach, Ben-Tal and Nemirovski [13] studied 90 LPs from the NETLIB collection and assumed that the given coefficients of the $(m \times n)$-matrix A are 99.9%-accurate approximations for the unknown "true" entries of the $(m \times n)$-matrix $\tilde{A}$. Since a worst-case change in the data set seemed too pessimistic, they assumed the rows of $\tilde{A}$ to be given randomly by $\tilde{a}_j = (1 + \xi_j)a_j$ with independent and uniformly distributed values of $\xi_j \in [-0.001, 0.001]$, $j = 1, \ldots, m$. In 50% of the tested cases, a nominal optimal solution became infeasible and could not be restored by moderate small correction. On the other hand, the loss in the objective of a robust solutions never exceeds 1% in their test instances. Hence, the price for robustness is relatively small compared to the gain in reliability.

Γ-Scenarios A different approach to model uncertainty sets was published in 2004 by Bertsimas and Sim [15]. Their uncertainty set controls the degree of conservatism for every constraint by hedging against all scenarios which assign to a limited number of coefficients in each row values deviating from their nominal value. Intuitively speaking, they suppose that it is very unlikely that all uncertain parameters change to their worst-case behavior as assumed by Soyster [92] and thus put a cardinality constraint on the number of changes. To this end, they introduced the concept of Γ-*scenarios*: Let A^0 be again the nominal $(m \times n)$-matrix with rows a_i^0, $i = 1, \ldots, m$. For each entry a_{ij}^0, $i = 1, \ldots, m$, $j = 1, \ldots, n$, a maximal deviation $d_{ij} \geq 0$ from this value is assumed, i.e., all values of a disturbed input lay within the interval $[a_{ij}^0 - d_{ij}, a_{ij}^0 + d_{ij}]$. In order to restrict the number of deviation in each row, parameters $\Gamma_i \geq 0$ for $i = 1, \ldots, m$ are fixed. A scenario $S \in \mathcal{S}_\Gamma$ is a feasible Γ-scenario if the realized $(n \times m)$-matrix $A^S = (a_1^S, \ldots, a_m^S)$ obeys the following three properties: First, for all $i = 1, \ldots, m$, $j = 1, \ldots, n$, the coefficients a_{ij}^S are in the interval $[a_{ij}^0 - d_{ij}, a_{ij}^0 + d_{ij}]$. Second, the number of values deviating from the nominal value in the ith row are restricted to $\lfloor \Gamma_i \rfloor + 1$, i.e., the cardinality of $J_i = \{j = 1, \ldots, n \mid |a_{ij}^S - a_{ij}^0| > 0\}$ is bounded by $\lfloor \Gamma_i \rfloor + 1$. The third condition requires to have one coefficient a_{ij}^S in each row with $a_{ij}^S = a_{ij}^0 + (\Gamma_i - \lfloor \Gamma_i \rfloor)d_{ij}$. We assume in general the values of Γ_i, $i = 1, \ldots, m$, to be integer. For $\Gamma_i = 0$, $i = 1, \ldots, m$, the robust counterpart with Γ-scenarios corresponds to the nominal problem and $\Gamma_i = n$ matches the

model of Soyster. Hence, the parameter Γ is to adjust the robustness against the level of conservatism of the solution, depending on the modeler's need.

Robustness with Γ-scenarios turned out to be quite attractive, since Bertsimas and Sim [15] showed that the robust LP problem with Γ-scenarios is not only tractable but also equivalent to a simple LP obtained from the robust counterpart by using results from duality theory. If the nominal LP consists of n variables and m constraints, the robust counterpart has maximal $(2+m)n+m$ variables and $(1+m)n+m$ constraints and is defined by

$$\begin{aligned} & \min c^\top x & & & \\ \sum_{j=1}^{n} a_{ij}^0 x_j + z_i\Gamma_i + \sum_{j=1}^{n} p_{ij} &\leq b_i & & \forall\, i = 1,\dots,m \\ z_i + p_{ij} &\geq d_{ij} y_j & & \forall\, i = 1,\dots,m,\ j = 1,\dots,n \\ x_j &\leq y_j & & \forall j = 1,\dots,n \\ y_j, z_i, x_j, p_{ij} &\geq 0 & & \forall j = 1,\dots,m,\ i = 1,\dots n. \end{aligned}$$

Furthermore, they proved that the parameters Γ_i, $i = 1,\dots,m$, control the probability that some constraint is violated by a robust solution.

The Histogram-Model In Bertsimas and Sim's model the coefficients of the matrix are bounded by a box and in most realizations they take the nominal value or one of the two box bounds. Bienstock [16] introduced in 2007 a refinement of this approach, the *Histogram-model*. Instead of one deviation value from the nominal value, several deviation bands for each parameter are defined. Each feasible scenario guarantees that a certain number of parameters fall into each band.

More formally, let A^0 be the nominal $(m \times n)$-matrix. For each coefficient a_{ij}^0, $i = 1,\dots,m$, $j = 1,\dots,n$, a sequence of K deviation values $d_{ij}^0,\dots,d_{ij}^K$ with $d_{ij}^0 = 0 \leq d_{ij}^1 < d_{ij}^2 < \dots < d_{ij}^K$ are given. Let furthermore $n_i^1,\dots,n_i^K$ and $N_i^1,\dots,N_i^K$ be natural numbers forming lower and upper bounds on the parameter distribution in the ith row, $i = 1,\dots,m$. A scenario S introduces a feasible realization $A^S \in \mathbb{R}^{m\times n}$ if in each row a_i and for all $k = 1,\dots,K$ the number of parameters a_{ij}^S with $a_{ij}^S \in [a_{ij}^0 - d_{ij}^k, a_{ij}^0 - d_{ij}^{k-1}] \cup [a_{ij}^0 + d_{ij}^{k-1}, a_{ij}^0 + d_{ij}^k]$ lies within the interval $[n_i^k, N_i^k]$, $i = 1,\dots,m$, $j = 1,\dots,n$.

The choice of the number and the size of the bands, the values n_i^k and N_i^k, $i = 1,\dots,m$, $k = 0,\dots,K$, reflect the willingness to take risks of the modeler and depends therefore highly on the problem structure. For a detailed description of this model and its application to portfolio optimization, we refer the reader to the original paper by Bienstock [16]. This model has also successfully be applied to wireless network design [17].

1.1.2. Robust Combinatorial Optimization

Robust frameworks for discrete combinatorial optimization problems where first introduced by Kouvelis and Yu in the middle of the 1990s [78]. Inspired by classical decision theory, they proposed three different classes of robust solutions: the absolute robust solution, also called min-max solution or strict robust solution, the robust deviation solution, also called min-max regret solution, and the relative robust solution, also called min-max relative regret solution. All robust solutions need to be feasible in all possible realizations. Yet, the objective function varies. In the absolute robust case, the maximum occurring cost is minimized. The regret in

a scenario $S \in \mathcal{S}$ is defined as the difference in cost by the robust solution and an optimal solution. An optimal regret robust solution minimizes the maximum regret. Finally, for the relative robust case, the regret is normalized by the value of the optimal solution. In the following we will define the framework of absolute robust solutions in detail and just call them robust solutions for the sake of simplicity.

We consider the class of combinatorial minimization problems with linear objective functions. Such problems are defined as follows:

Definition 1.1.1 (Linear Combinatorial Minimization (LCMin) problem)**.** Let U be a finite set, $\mathcal{F} \subseteq 2^U$ be a set of feasible solutions and $c : U \to \mathbb{N}$ be a cost function. The triple $(U, \mathcal{F}, c)$ defines a *linear combinatorial minimization problem* where the goal is to find a set $F^* \in \mathcal{F}$ with minimum cost $c(F^*) = \sum_{u \in F^*} c(u)$.

This class covers several classical combinatorial minimization problems like the shortest path problem, the minimum spanning tree problem, the minimum (s,t)-cut problem or the minimum perfect matching problem. In order to model uncertainties, we use sets of scenarios $\mathcal{S}$, where each scenario S defines a *scenario cost function* $c^S : U \to \mathbb{N}$ and a set of *scenario feasible solutions* $\mathcal{F}^S \subseteq \mathcal{F}$. Let now $(U, \mathcal{F}, c)$ be an LCMin instance and $\mathcal{S}$ be a set of scenarios. Then a solution $F \in \mathcal{F}$ is called *robust* if F is part of every set of scenario feasible solutions $\mathcal{F}^S$, $S \in \mathcal{S}$. The *robust linear combinatorial minimization problem* is to find for a given LCMin instance and a scenario set $\mathcal{S}$ a robust solution F with minimal robust cost $c_R(F) = \max_{S \in \mathcal{S}} c^S(F)$. In the following part we focus on scenarios which just imply uncertainties in the cost of the given LCMin instance, i.e., $\mathcal{F}^S = \mathcal{F}$ for all $S \in \mathcal{S}$.

Different Types of Scenario Sets In robust combinatorial optimization mainly three different types of scenario sets have been considered: the discrete scenario set $\mathcal{S}_D$, the interval scenario set $\mathcal{S}_I$ and the Γ-scenario set $\mathcal{S}_\Gamma$.

Let $(U, \mathcal{F}, c)$ be an LCMin problem. In the *discrete scenario set* $\mathcal{S}_D$, e.g., considered in [3, 78, 87, 101], every scenario is explicitly given with its cost function, i.e., the set $\mathcal{S}_D$ consists of r scenarios $S_1, \ldots, S_r$ where each scenario S_i, $i = 1, \ldots, r$, defines a cost function $c^{S_i} : U \to \mathbb{N}$. Also covered in several of these articles is the *interval scenario set* $\mathcal{S}_I$, an implicit description of all possible scenarios: For each element $u \in U$ a lower and an upper cost bound $\underline{c}(u), \overline{c}(u) \in \mathbb{N}$ with $0 \leq \underline{c}(u) \leq \overline{c}(u)$ is given. The cost functions of all scenarios in $\mathcal{S}_I$ obey these bounds and for any cost function $c : U \to \mathbb{N}$ with $c(u) \in [\underline{c}(u), \overline{c}(u)]$ for all $u \in U$, a scenario $S \in \mathcal{S}_I$ with this cost function exists. For the Γ-*scenario set* again upper and lower cost bounds limit the values of the scenario cost functions, as already described for linear programs [15]. But, in contrast to interval scenarios, a scenario $S \in \mathcal{S}_\Gamma$ is only allowed to have at most Γ cost values deviating from the lower bound. Note that we assume Γ to be integer.

Results for Discrete Scenarios The complexity status of robust combinatorial optimization problems has extensively been studied in recent years. Kouvelis and Yu [78] started by proving that many robust LCMin problem are **NP**-hard for a constant number of discrete scenarios by a reduction from partition, e.g., the robust version of the shortest path problem or of the minimum spanning tree problem. In [4] Aissi et al. established a close relation between the multi-objective version of an optimization problem and its robust version: They transformed any approximation algorithm of the multi-objective problem into an approximation algorithm of the robust problem. Since there is an FPTAS for the multi-objective version of the shortest path problem and the minimum spanning tree problem [85], there exists an FPTAS for their

robust versions as well and hence, they are also solvable in pseudo-polynomial time. Yet, not all robust LCMin problems with a constant number of discrete scenarios can be solved in pseudo-polynomial time. Even for two scenarios Aissi et al. showed that the robust minimum (s,t)-cut problem is strongly **NP**-hard [5].

Let us now consider the case in which the number of scenarios $\mathcal{S}_D$ is not constant. In that case most robust problems turn out to be strongly **NP**-hard by a reduction from 3-partition, which was already noted by Kouvelis and Yu [78]. Concerning the approximability, Aissi et al. [4] were the first to give lower bounds on the best possible approximation factor for robust combinatorial optimization problems where the number of scenarios is not constant. In detail, they proved a lower bound of $2-\varepsilon$ for the shortest path problem and a $1.5-\varepsilon$ bound for the minimum spanning tree problem, for any $\varepsilon > 0$. In [70] Kasperski and Zieliński improved these bounds: the robust shortest path and the robust minimum (s,t)-cut problem with discrete scenarios $\mathcal{S}_D$ are not approximable within $\log^{(1-\varepsilon)} |\mathcal{S}_D|$ for any $\varepsilon > 0$, unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\text{poly}\log n})$. Furthermore, the robust minimum spanning tree is also not approximable with a factor better than 2.

On the other hand, little has been known on approximation algorithms. A simple way to obtain an $|\mathcal{S}_D|$-approximation for any robust LCMin problem $(U, \mathcal{F}, c)$ with discrete scenarios $\mathcal{S}_D$ is by choosing an optimal solution of the corresponding $(U, \mathcal{F}, \overline{c})$ instance with $\overline{c} : U \to \mathbb{R}^+$ and $\overline{c}(u) = \frac{1}{|\mathcal{S}_D|} \sum_{S \in \mathcal{S}_D} c^S(u)$. Table 1.1 summarizes the results on the complexity status of robust combinatorial minimization problems.

Problems	$\lvert\mathcal{S}_D\rvert$ constant		$\lvert\mathcal{S}_D\rvert$ not constant	
	Complexity	Approx.	Complexity	Approx.
Shortest Path	weakly **NP**-hard [78]	FPTAS [3]	strongly **NP**-hard [78]	not $\log^{1-\varepsilon}(\lvert\mathcal{S}_D\rvert)$ [70]
Spanning Tree	weakly **NP**-hard [78]	FPTAS [3]	strongly **NP**-hard [78]	not $(2-\varepsilon)$ [70]
(s,t)-Cut	strongly **NP**-hard [5]		strongly **NP**-hard [78]	not $\log^{1-\varepsilon}(\lvert\mathcal{S}_D\rvert)$ [70]
Perfect Matching	weakly **NP**-hard*	FPTAS [4]	strongly **NP**-hard*	not $(2-\varepsilon)$*

* nowhere explicitly stated

Table 1.1.: Complexity and approximation of robust combinatorial optimization problems in the discrete scenario case.

Results for Γ-Scenarios In [14], Bertsimas and Sim applied the concept of Γ-scenarios to combinatorial optimization problems, but only assumed the cost function to vary. They considered the set of all 0-1 *discrete optimization* problems with linear cost functions, where the set of feasible solutions X is a subset of $\{0,1\}^n$, $n \in \mathbb{N}$. The shortest path problem, the minimum spanning tree problem and the minimum perfect matching belong to this problem class. The main theorem of [14] states that the robust counterpart with Γ-scenarios of these problems remains solvable in polynomial time. To this end, it suffices to solve $n+1$ special instances of the considered problem with fixed parameters. For **NP**-hard problems any α-approximation algorithm transforms in the same way to an α-approximation for the robust counterpart. The proof is limited to solutions having a 0-1 representation and cannot be adapted to general LCMin problems.

A robust approach has a number of weaknesses, as mentioned above. On further drawback is, that robust optimization corresponds to the case where all decisions have to be made before the actual realization is known. There are settings, e.g., in contaminant warning system for water networks [96], in which such an approach is appropriate. But in the majority of optimization problems of real-world origin only parts of a decision have to be taken here and

now. Small changes to a solution are acceptable as reaction to a new situation. For example, in the inventory control problem the number of produced goods needs to meet the demand of the market. Yet, this demand may change on a daily base and thus is only known after the production took place. On the other hand, in a long term process the decision on the number of products can also be adapted depending on the development of the markets [11].

Strict robustness is more or less restricted to scenarios that affect the cost function of a given problem. Yet, in general, changes may also have an influence on the set of feasible solutions as an element may not be available in a certain setting. Or the demand may change, e.g., the target vertex in a shortest path problem depends on the realized scenario or the nodes connected via a spanning graph are fixed after the edges are chosen [37]. In these cases it is rare that a robust solution exists. To address these issues, new approaches were introduced incorporating small modifications of the solution after the uncertain parameters are fixed. We will define this concept and give a short overview of the results in the next section.

1.2. Extending Robustness to Recoverable Robustness

The idea to broaden the concept of robustness and to include some changes of the solution in the revealed setting has attracted many different researchers: Mulvey et al. [83] differentiate in their new robust model between design decisions and control decisions, Ben-Tal et al. [11] introduced the concept of adjustable robustness for linear programs, Dhamdhere et al. [37] considered demand robustness, Atamtürk and Zhan [7] called their concept two-stage robustness, and Liebchen et al. [81] investigated recoverable robustness inspired by the recovery action. All these models include a two-stage process: in the first stage some decision is taken, e.g., a solution or a superset of a solution is chosen. We call it a *first-stage solution*. This solution leads to limitations of the feasible solution set in the second stage, the so-called *recovery set* of a solution. In the second stage, after the exact parameters are revealed, a solution out of the recovery set is chosen and implemented. This *second-stage solution* can be different for each scenario. Under all first-stage solutions that contain a scenario feasible solution in the second stage, one with minimal total cost is chosen. The total cost is composed of two parts: the first part does not underlie uncertainty and needs to be payed for the first-stage solution, it is called *first-stage cost*. The second part depends on the scenario cost functions. In each scenario, cost for the chosen second-stage solution and in some cases the first-stage solution have to be payed. The maximum amount of this cost over all scenarios determines the *second-stage cost*.

Beside this common structure there are small differences in the proposed models above: One feature in the robust linear program of Mulvey et al. [83] is that the first-stage solution and the recovery actions do not need to generate a feasible second-stage solution. The violation is captured in scenario-dependent error variables. Furthermore, the proposed objective function is kept in a quite general way: depending on the given information of the scenario set, it minimizes either the maximum scenario cost or, if some distribution for the scenario set is available, the expected value or some other stochastic function. In addition a penalty term for the error variables is included in the objective function. Mulvey et al. [83] applied their robust concept to the power capacity expansion problem, the matrix balancing problem, image reconstruction and an airline allocation problem for the Air Force. For these problems they compared the performance of robust solutions with the performance of stochastic linear programming solutions. In most cases the robust solutions had slightly higher expected cost than the stochastic solution but with substantially lower standard deviation. Yet, in the investigated robust models they always included some stochastic distributions.

Adjustable Robustness Ben-Tal et al. [11] extend the notion of robustness for linear programs and define an adjustable robust counterpart. They start by distinguishing between *first-stage variables* $u \in \mathbb{R}^{n_1}$ (in their terminology called non-adjustable variables), which are fixed in a first stage and do not change, and *second-stage variables* $v \in \mathbb{R}^{n_2}$ (there called adjustable variables), which may change in every scenario realization. In the same way they divide the coefficient matrix into two parts, namely, $U \in \mathbb{R}^{m \times n_1}$ and $V \in \mathbb{R}^{m \times n_2}$. In a last step, they normalize the objective function given by the vector $c \in \mathbb{R}^{n_1}$ such that it is independent of the second-stage variables. For a given right-hand side $b \in \mathbb{R}^m$, the nominal linear problem is

$$\begin{aligned} \min\, & c^\top x \\ Uu + Vv &\leq b \\ u, v &\geq 0. \end{aligned}$$

As before, uncertainties can be modeled via a scenario set $\mathcal{S}$, where each $S \in \mathcal{S}$ defines an $(m \times n_1)$-matrix $U^S \in \mathbb{R}^{m \times n_1}$ and an $(m \times n_2)$-matrix $V^S \in \mathbb{R}^{m \times n_2}$ that models the *recourse*, i.e., the allowed modifications of u by v. Note that the right-hand side $b \in \mathbb{R}^m$ and the cost vector $c \in \mathbb{R}^n$ are w.l.o.g. not effected by uncertainties. Now, the *adjustable robust counterpart* (ARC) of the nominal linear program is defined by

$$\min_u \{c^\top u \mid \text{for all } S \in \mathcal{S} \text{ there exists } v : U^S u + V^S v \leq b, u, v \geq 0\}.$$

In general, the adjustable robust counterpart is more difficult to solve than the nominal problem. Guslitser showed in [56] that even if the recourse V^S is fixed for all scenarios, i.e., $V^S = V$ for all $S \in \mathcal{S}$, and the set of realizable matrices U^S is defined by a list of linear inequalities, the ARC is an **NP**-hard problem. The same holds if we consider a discrete set of r scenarios $\mathcal{S}_D = \{S_1, \ldots, S_r\}$ where the recourse V^{S_i} may differ in each scenario, $i = 1, \ldots, r$. Yet, in the case of a fixed recourse and a set of discrete scenarios $\mathcal{S}_D$, the problem becomes solvable in polynomial time. These results motivated Ben-Tal et al. [11] to consider the case where the second-stage variables are affine functions of the corresponding data. They showed that for a fixed recourse, the ARC is tractable in many cases.

The concept of adjustable robustness has been applied in several real-world problems like the inventory management problem [11], the empty repositioning problem [42] or the air traffic control problem [27]. The overall valuation is that the price of robustness remains relatively small. For example, in the inventory management problem with variation in the demand of 20%, the average management cost for the robust adjustable policy is just $3,4\%$ worse than the corresponding ideal cost.

Demand- and Two-Stage-Robustness The development of demand robustness, also called two-stage robustness, is mainly influenced by two-stage stochastic programming and the limits of the robust model in combinatorial optimization to capture variations in the demand. For example, in the two-stage robust shortest path problem, a directed graph $G = (V, A)$ with arc cost $c : A \to \mathbb{N}$ is given. Instead of uncertainty in the arc cost, the target vertex is not known in advanced. In a first stage some arcs can already be purchased. This set must be completed to a path as soon as the target vertex is known. Arcs bought in the second stage come to an higher cost, namely $(1 + \beta^S)$ times the cost of the first stage for some scenario dependent inflation factor $\beta^S \geq 0$, $S \in \mathcal{S}$. More formally, the goal is to buy some parts of the possible path $A_1 \subseteq A$ in advance for the cost $c(A_1)$, and to complete it for the given target vertex $t^S \in V$ to a feasible (s, t^S)-path in $A_1 \cup A_2^S$ by buying the arcs A_2^S for the cost $(1 + \beta^S)c(A_2^S)$ as soon as the scenario $S \in \mathcal{S}$ has realized. The objective is to minimize this cost over all

possible scenarios, i.e., to minimize $c(A_1) + \max_{S \in \mathcal{S}}(1 + \beta^S)c(A_2^S)$. Note that in this setting a strictly robust path does not exist in general.

The investigation of this kind of uncertainties and the development of two-stage robustness has been initialized by Dhamdhere et al. [37], who considered covering problems as the minimum cut problem, the shortest path problem, Steiner trees or vertex cover with discrete scenario sets. Using LP-rounding as main technique, they achieved several approximation results for discrete scenario sets $\mathcal{S}_D$. More precisely, they achieved an $\mathcal{O}(\log |\mathcal{S}_D|)$-approximation for the robust minimum cut problem, a 30-approximation for the robust Steiner tree problem, a 16-approximation for the robust shortest path problem or a 4-approximation for the robust vertex cover problem. Furthermore, they introduced a lemma about the structure of first-stage solutions in a two-stage robust setting to derive lower bounds on the optimal cost. Golovin et al. [54] used this lemma to present a $(1 + \sqrt{2})$-approximation for the minimum cut problem and a 7.1-approximation for the robust shortest path problem.

Since considering discrete scenarios seemed rather restrictive, Feige et al. [44] suggested the k-robust model in which every subsets of the given cover problem with a cardinality of k forms a scenario, i.e., they proposed an implicit description of the scenario set. Since the number of scenarios may be exponential, there is a unique inflation factor on the cost in the second stage for all considered scenarios. Using an online algorithm for set cover they obtained an $\mathcal{O}(\log n \log m)$-approximation for the robust set cover with n elements U and a set m subsets to cover U. On the other hand, they show that the problem is not approximable within a factor better than $\Omega(\frac{\log m}{\log \log m} + \log n)$ and thus left a logarithmic gap between the lower and upper bounds. Khandekar et al. [73] adopted the model of Feige et al. [44] for the robust Steiner tree, Steiner forest and facility location problem. They noted that the techniques proposed for the set cover problem did not give good results, and developed new constant factor approximations for these problems. The last improvement in this area of robust optimization was established by Gupta et al. [55]. They introduced a general framework to design algorithms for these robust problems. Using them, they obtained an $\mathcal{O}(\log m + \log n)$-approximation for the robust set cover and improved the constant in the approximation factor for Steiner tree to 8, for Steiner forest to 10 and for min-cut to 17.

Recoverable Robustness The definition of recoverable robustness introduced by Liebchen et al. [81] is probably the most general one of the mentioned concepts. A recoverable robust problem is defined via three steps: in a first step, the original optimization problem is given. In a second step, a scenario set for this optimization problem needs to be specified, and in the last step, the limited recovery possibilities are fixed. The new input of the definition lies in the formalization of the recovery, which is represented via a class $\mathcal{A}$ of *admissible recovery algorithms*. A recoverable robust solution x in that case not only consists of the first-stage solution but also of a specific recovery algorithm $A \in \mathcal{A}$. Such a solution pair (x, A) is *feasible* for a set of scenarios $\mathcal{S}$ if for every scenario $S \in \mathcal{S}$ the algorithm A applied to x and S produces a scenario feasible solution.

In [81] several examples are given on how this setting, and especially the set of recovery algorithms $\mathcal{A}$, may be interpreted. If, for example, the set of recovery algorithms just consists of the algorithm A with $A(x, S) = x$ for all $S \in \mathcal{S}$, they obtain the standard robust setting. If the distance $d(x, y)$ from the first-stage solution x to the implemented solution x^S in the second stage shall be limited by some $\delta \geq 0$ for $S \in \mathcal{S}$, all recovery algorithms are forced to satisfy $d(A(x, S), x) \leq \delta$. Another setting to mention here covers the case where the caused cost by the second-stage solution x^S is limited by some $\lambda \geq 0$ for every $S \in \mathcal{S}$. Since usually

not only the cost for the first-stage solution but also the limit λ needs to be minimized, this last setting corresponds to finding a first-stage solution with minimal maximum scenario cost.

Especially in railway optimization the concept of recoverable robustness is widely accepted and applied to different settings. To mention a few, Cacchiani et al. [25] introduced a recoverable robust model for the rolling stock problem, Cicerone et al. [30] considered timetabling and shunting, and Caprara et al. [26] adapted this concept to approach the train platforming problem with delays. Furthermore, a simple linear program is proposed in [81] to solve linear programs with variations in the right-hand side and several different scenario sets. This setting corresponds to the network buffering problem in timetabling.

There exist further concepts of robustness beyond the scope of this thesis, such as *regret robustness* introduced by Kouvelis and Yu [78], *light robustness* investigated by Fischetti et al. [46] or *lexicographic α-robustness* considered by Kalaï et al. [65]. This introduction to robustness focuses on the settings which are closest related to the research presented here.

2. k-Distance Recoverable Robustness

This chapter presents k-distance recoverable robustness. In this setting, a solution chosen in the first stage is allowed to be modified in the second stage. Such a modified solution can contain up to k new elements compared to the solution in the first stage. These changes serve as tool for hedging against uncertainties in the cost function which are modeled via a set of scenarios. Each scenario determines a scenario cost function. The task in k-distance recoverable robustness is to find a solution in the first stage that minimizes the cost over all scenarios for a solution taken in the second stage w.r.t. the scenario cost function.

Choosing different types of scenario sets, we analyze the complexity status and combinatorial properties of the k-distance recoverable robust version of several well-known optimization problems. In the case of discrete scenarios most of these problems are already weakly **NP**-hard for two scenarios and even strongly **NP**-hard if the number of scenarios is not constant. For the weighted disjoint hitting set problem, a subproblem of the shortest path problem, the minimum spanning tree problem and the minimum perfect matching problem, we introduce a pseudo-polynomial algorithm for a constant number of scenarios.

As a second type of scenario sets, we investigate interval scenarios. In this setting the k-distance recoverable robust minimum weight basis problem for matroids with interval scenarios becomes tractable as well as the k-distance recoverable robust weighted disjoint hitting set problem. However, for the shortest path problem with simple (s,t)-paths as first-stage solution the k-distance recoverable robust version remains strongly **NP**-hard and is furthermore not approximable, unless $\mathbf{P} = \mathbf{NP}$.

Considering Γ-scenarios, in addition to the shortest path problem, also the minimum (s,t)-cut problem turns out to be strongly **NP**-hard. Finally, we analyze the special case of one scenario, a constant recovery parameter, and some not necessarily simple (s,t)-paths as first-stage solution, and present a polynomial algorithm for this k-distance recoverable robust shortest path version.

2.1. Introduction

Motivation As described in Section 1.2, recoverable robustness is a two-stage procedure. In a first stage a solution is taken without knowing the exact parameters of the optimization problem. In the second stage this solution is modified according to the realized scenario in order to obtain a feasible and potentially cheaper solution. Each scenario determines all parameters considered uncertain in the first stage. Interpreting recoverable robustness as a mediator between "optimization on the fly", where for every scenario an optimal solution is chosen in the second stage, and robustness, where no changes are allowed after the solution is chosen in the first stage, our first recoverable robust approach allows only small changes in the second stage. For that reason, we adapt the idea of neighborhoods in local search to define the set of solutions taken as recovery.

A simple concept of neighborhoods are the so-called *k-exchange neighborhoods* introduced by Kernighan and Lin [72]. The k-exchange neighborhood for a solution consists of all solutions that can be constructed by removing k elements and replacing them by k other elements. Kernighan and Lin use this concept to design heuristics for the traveling salesman problem (TSP) and the graph partitioning problem. This approach has also been applied fruitfully to the TSP with time windows and the job shop scheduling problem, see [1]. The k-exchange neighborhoods define a reasonable neighborhood if all feasible solutions of the considered optimization problem have the same cardinality, e.g., in the minimum weight basis problem for matroids or in the traveling salesman problem. For example, any TSP tour can be reached from any other TSP tour by a finite sequence of local changes within a 2-neighborhood. Yet, for problems like the shortest path problem or the minimum (s,t)-cut problem, this property is not satisfied.

We therefore broaden the definition to gain a useful description of a recovery by limiting only the number of new elements taken in the second stage. In other words, a solution is a feasible recovery for a given first-stage solution if it is constructed by removing an arbitrary number of elements and using at most k new elements. Summing up, we obtain the following recoverable robust concept called the *k-distance recoverable robustness*: Choose in a first stage a first-stage solution for the given optimization problem. As soon as all data is known, take one solution with minimum second-stage cost that includes at most k new elements w.r.t. the first-stage solution.

Model and Notation We will now formally introduce k-distance recoverable robustness for linear combinatorial minimization problems. Recall that a linear combinatorial minimization (LCMin) problem is given by a finite set U, a set of feasible solutions $\mathcal{F} \subseteq 2^U$, and a cost function $c : U \to \mathbb{N}$. The task is to find a feasible solution with minimum cost (Definition 1.1.1). We start by defining which kind of sets we consider as feasible first-stage solutions.

Definition 2.1.1 (First-Stage Solution Set)**.** Let $(U, \mathcal{F}, c)$ be an LCMin instance. A set $\mathcal{G} \subseteq 2^U$ is called a *first-stage solution set* of U if for any subset $E \subseteq U$ it can be decided in polynomial time in U whether E is in $\mathcal{G}$. A set $F \in \mathcal{G}$ is called a *first-stage solution*.

According to this definition a first-stage solution set $\mathcal{G}$ may not be given explicitly, but can be described to consist of all subsets fulfilling a certain property, e.g., containing exactly ℓ elements, $\ell \in \mathbb{N}$, or being a feasible solution of the given LCMin instance. Note that a feasible solution of the given LCMin problem does not necessarily need to be chosen in the first stage as it is also the case in two-stage robustness [37]. An important subclass of first-stage solution sets are so-called proper sets, which we define below.

Definition 2.1.2 (Proper First-Stage Solution Set)**.** A first-stage solution set $\mathcal{G}$ is *proper* if

1. every element $F \in \mathcal{G}$ contains a feasible solution of the given LCMin instance $(U, \mathcal{F}, c)$ and
2. $\mathcal{F} \subseteq \mathcal{G}$.

Combining this definition with the idea of recovery derived from local search, we obtain the following definition for k-distance recoverable robust linear combinatorial minimization problems.

Definition 2.1.3 (k-Distance Recoverable Robust Linear Combinatorial Minimization Problem (k-Dist-RR LCMin Problem))**.** Let $(U, \mathcal{F}, c)$ be a linear combinatorial minimization problem; $c^D : U \to \mathbb{N}$ be a *first-stage cost* function; $\mathcal{S}$ be a set of scenarios, where each scenario S

defines a scenario cost function $c^S : U \to \mathbb{N}$; $\mathcal{G}$ be a first-stage solution set of U, and $k \in \mathbb{N}$ be a *recovery parameter*. Then the *recovery* $\mathcal{F}_F^k$ of a first-stage solution $F \in \mathcal{G}$ consists of all sets $\bar{F} \in \mathcal{F}$ with $|\bar{F} \backslash F| \leq k$, and the *recovery cost* $c_{RR}(F)$ is determined by

$$c_{RR}(F) = \max_{S \in \mathcal{S}} \min_{\bar{F} \in \mathcal{F}_F^k} c^S(\bar{F}).$$

The *first-stage cost* of F are given by $c^D(F) = \sum_{u \in F} c^D(u)$. First-stage cost and recovery cost sum up to the total cost $c_T(F)$ of F, i.e.,

$$c_T(F) = c^D(F) + c_{RR}(F).$$

The *k-distance recoverable robust LCMin problem* is to find a feasible first-stage solution $F^* \in \mathcal{G}$ with minimum total cost $c_T(F^*)$.

In this definition the cost function c of the underlying LCMin instance $(U, \mathcal{F}, c)$ does not directly cause any cost for a first-stage solution or a solution taken in the second stage. However, we assume this function to be perturbed and hence the scenario cost functions can be seen as variations of c.

Note that for $k = 0$ and $\mathcal{G} = \mathcal{F}$ the k-Dist-RR LCMin problem is equivalent to the robust LCMin problem. Let us consider the following example to illustrate the different definitions.

Example 2.1.4. We consider a k-Dist-RR shortest path problem for $k = 1$. Let $G = (V, A)$ be the graph given in Figure 2.1. We assume that the first-stage cost is 1 for every arc in G. Furthermore, we consider Γ-scenarios defined by lower cost bounds $\underline{c}(a) = 0$, upper cost bounds $\overline{c}(a) = 10$ for all $a \in A$ and $\Gamma = 1$. Hence, a feasible Γ-scenario assigns cost 10 to one arc and cost 0 to all others. The two smaller graphs G_1 and G_2 show two of these scenarios: scenario S_1 increases the cost of arc (a, b) to 10 in the graph G_1 and scenario S_2 increases the cost for the arc (s, e) in the graph G_2. The arcs with the higher cost are marked red. Finally, we define the set of first-stage solutions $\mathcal{G}$ to consist of all simple (s, t)-paths.

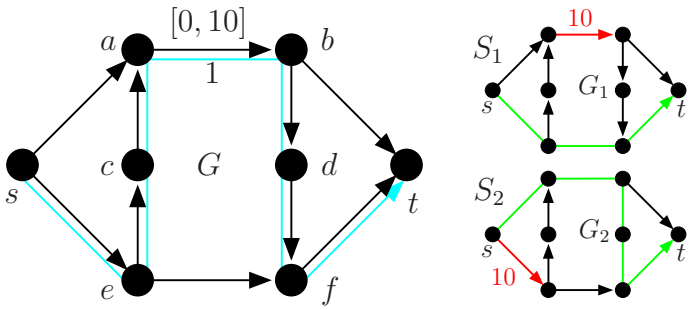


Figure 2.1.: The left graph G shows the situation in the first stage, where the different cost values of the arcs are not known. On the right side, two possible realizations are illustrated, where all arc costs are 0 except for the red arcs.

Let us consider the blue colored path $p = secabdft$ in G. The first-stage cost of p is 7, which is rather high compared to all other (s, t)-paths in G. On the other hand, its recovery set $\mathcal{F}_p^1$ consists of four paths, namely, the blue path p, $p_1 = seft$ with the new arc (e, f), $p_2 = sabdft$ with the new arc (s, a) and $p_3 = secabt$ with the arc (b, t). If, for example, the scenario S_1 is realized, the path p_1 should be chosen as recovery in the second stage inducing no extra cost. This situation is shown in graph G_1. If S_2 is realized, the path p_2 is the best recovery

(see graph G_2). Considering all scenarios in $\mathcal{S}_\Gamma$ one can easily verify that there exists always a path in the recovery with cost 0 w.r.t. the scenario cost function. Hence, the recovery cost $c_{RR}(p)$ is 0 and the total cost is 7.

For the (s,t)-path $p_1 = seft$ the first-stage cost is 3, but p_1 does not contain any other path in the recovery set. Therefore, its recovery cost is 10 induced, for example, by scenario S_2, and sum up to total cost 13. Computing the cost for all other paths shows that p is the optimal first-stage solution in the described instance. This changes if we consider a different set of first-stage solutions. If, for example, the first-stage solution set consists of all subgraphs of G that contain an (s,t)-path, an optimal solution is given by the path p_1 plus two arcs of p_2, e.g., $A' = \{(s,a),(a,b),(b,t),(s,e),(e,f)\}$. The total cost of this solution is 5, since in every scenario there exists a path in the recovery with cost 0. ⊞

A k-Dist-RR LCMin problem can be interpreted in various ways. First of all, it can be seen as a bi-criteria problem if the first-stage cost function is chosen independently from the scenario cost function for each scenario. In real world applications, the first-stage cost are usually seen as the normal cost that occur for the given problem instance. The scenario cost, on the other hand, just models slight increase on these cost. We capture this dependency in the following definitions.

Definition 2.1.5 (Worsening-Condition)**.** Let $(U,\mathcal{F},c)$ be an LCMin instance, $c^D : U \to \mathbb{N}$ be a first-stage cost function and $\mathcal{S}$ be a set of scenarios where each scenario $S \in \mathcal{S}$ defines a cost function $c^S : U \to \mathbb{N}$. Then $\mathcal{S}$ and c^D satisfy the *worsening-condition* if $c^D(u) \leq c^S(u)$ for all $u \in U$, $S \in \mathcal{S}$.

Even if a set of scenarios and a first-stage cost function obey the worsening-condition, the values of the cost functions defined by the scenarios may be arbitrarily high in comparison to the first-stage cost. To limit these variances we define the following α-deviation-condition.

Definition 2.1.6 (α-Deviation-Condition and $\alpha_{[0,1]}$-Deviation-Condition)**.** In the setting of Definition 2.1.5 a scenario set $\mathcal{S}$ and a first-stage cost function c^D fulfill the *α-deviation-condition* w.r.t. $\alpha \in \mathbb{R}^+$, if for all $u \in U$ and $S \in \mathcal{S}$ we obtain

$$c^S(u) \in [c^D(u), (1+\alpha)\cdot c^D(u)].$$

We call the factor α the *α-deviation*. If $\alpha \in [0,1]$, we say that $\mathcal{S}$ and c^D obey the *$\alpha_{[0,1]}$-deviation-condition*.

Contribution and Chapter Outline Our research focuses on the complexity status and combinatorial structure of k-Dist-RR LCMin problems. The obtained results depend on the considered LCMin problem, but are also influenced by the set of scenarios, the recovery parameter k—taken as constant or not as constant—and the first-stage solution set. We consider discrete scenario sets, interval scenario sets and Γ-scenario sets, already described in Chapter 1.

Section 2.2 covers discrete scenario sets and presents an analysis of the simple so-called weighted disjoint hitting set problem, which is a subproblem of several LCMin problems, like the shortest path problem, the minimum (s,t)-cut problem or the minimum spanning tree problem. However, its k-Dist-recoverable robust version is already weakly **NP**-hard for two scenarios, and strongly **NP**-hard if the number of discrete scenarios is not constant. For the first case we introduce a pseudo-polynomial algorithm solving this problem, whose runtime depends on the values of the scenario cost functions and is exponential in the number of scenarios.

In Section 2.3 we consider interval scenarios and show that this setting is a special case of discrete scenario sets, where the set is formed by just one single scenario. For the weighted disjoint hitting set problem, we introduce a polynomial algorithm solving this problem. If the recovery parameter k is taken as a constant, we can also solve the minimum weight basis problem for matroids efficiently. Finally, in Section 2.4 robustness w.r.t. Γ-scenarios is analyzed. Using a close relation to the max-scenario problem, we show **NP**-hardness for the k-Dist-RR minimum (s,t)-cut problem and the k-Dist-RR shortest path problem.

Section 2.5 presents results on the shortest path problem. Even for one scenario, the k-Dist-RR shortest path problem is strongly **NP**-hard and not approximable if the first-stage solution is a simple (s,t)-path. Surprisingly, an extension of the first-stage solution set to a set that consists of all (s,t)-paths, leads to a tractable k-Dist-RR shortest path problem if k is constant.

We begin by considering discrete scenario sets.

2.2. Discrete Scenarios

A discrete scenario set $\mathcal{S}_D$ consists of r scenarios $S_1,\dots,S_r$ with their scenario cost function $c^{S_i} : U \to \mathbb{N}$, $i = 1,\dots,r$, for an LCMin instance $(U,\mathcal{F},c)$. We start with an investigation of the complexity status of a k-Dist-RR LCMin problem with discrete scenario sets. To this end, we consider the *weighted disjoint hitting set problem*, which can be interpreted as a subproblem of the shortest path problem, the minimum (s,t)-cut problem, the minimum perfect matching problem and the minimum spanning tree problem, as we will later see in detail (Corollary 2.2.3). A disjoint hitting set instance contains a set $U = \{u_1,\dots,u_n\}$ of n elements, a set $\mathcal{M}$ of d pairwise disjoint subsets of U, i.e., $\mathcal{M} = \{M_1,\dots,M_d\}$ with $M_i \subseteq U$ and $M_i \cap M_j = \emptyset$ for all $i \neq j$, $i,j = 1,\dots,d$, and a cost function $c : U \to \mathbb{N}$. A set $F \subseteq U$ is a *feasible solution* if F contains exactly one element for every set $M \in \mathcal{M}$, i.e., $|F \cap M| = 1$ for all $M \in \mathcal{M}$. Then the WDHS problem is defined in the following way:

Given: A set of n elements $U = \{u_1,\dots,u_n\}$, a set of d pairwise disjoint subsets $\mathcal{M} = \{M_1,\dots,M_d\}$ and a cost function $c : U \to \mathbb{N}$.

Task: Find a feasible solution F that minimizes $c(F) = \sum_{u\in F} c(u)$.

Obviously the WDHS problem can be solved in polynomial time by choosing an element with lowest cost from every set $M_1,\dots,M_d$. Yet, this is not the case for its k-Dist-recoverable robust version with proper first-stage solutions sets and $|\mathcal{S}_D| \geq 2$.

Theorem 2.2.1. *The k-Dist-recoverable robust weighted disjoint hitting set problem is weakly* **NP***-hard for $|\mathcal{S}_D| = 2$ and proper first-stage solution sets even if the $\alpha_{[0,1]}$-deviation-condition is fulfilled and k is constant or depends on the number of elements of the WDHS instance.*

Proof. We start with the case of k being constant and show a reduction from partition. Let I be an instance of partition containing n integers $e_1,\dots,e_n$ with $\sum_{i=1}^{n} e_i = 2b$ for some $b \in \mathbb{N}$. Then the task for I is to decide whether there exists a partition $A_1 \cup A_2 = \{e_1,\dots,e_n\}$ such that $\sum_{e\in A_1} e = b$. We construct I' as the corresponding k-Dist-RR WDHS instance with a set of elements U, $n+k$ disjoint sets, two scenarios $\mathcal{S}_D = \{S_1,S_2\}$, and a proper first stage solution set $\mathcal{G}$. Suppose for a moment, that the considered proper first-stage solution set $\mathcal{G}$ consists of all feasible solutions.

The set U contains $2n+2k$ elements $u_1,\dots,u_{2n+2k}$, where the first $2n$ elements model the partition of the set $e_1,\dots,e_n$ and the last $2k$ elements capture the recovery action. We denote

with U_0 the first $2n$ element, with U_1 all elements with an odd index greater than $2n$ and with U_2 all elements with an even index greater than $2n$, i.e., $U_0 = \{u_1, \ldots, u_{2n}\}$, $U_1 = \{u_{2n+1}, u_{2n+3}, \ldots, u_{2n+2k-1}\}$ and $U_2 = \{u_{2n+2}, u_{2n+4}, \ldots, u_{2n+2k}\}$. The elements $u_1, \ldots, u_{2n+2k}$ are subdivided into $n+k$ sets $M_1, \ldots, M_{n+k}$ with $M_i = \{u_{2i-1}, u_{2i}\}$. The element u_i with i even are called the *even elements* and the element u_i with i odd the *odd elements*, $i = 1, \ldots, 2n+2k$. Since any feasible first-stage solution contains exactly one element out of the sets $M_1, \ldots, M_n$, the choice of the even elements u_i, $i = 1, \ldots, 2n$, can be interpreted as a set A_1 of a partition and the odd elements as a set A_2. More formally, let F be a feasible solution, then $A_1^F = \{e_i \mid u_{2i} \in F,\ i \in \{1, \ldots, n\}\}$ and $A_2^F = \{e_1, \ldots, e_n\} \backslash A_1^F$ are a partition induced by F. We will now define the first-stage cost function and the two scenario cost functions in a way such that they satisfy the following properties:

1. Every feasible solution F with $U_1 \subseteq F$ has lower total cost than a first-stage solution which contains one element $u_i \in U_2$. We call such a first-stage solution a *reasonable* solution.
2. Let F be a reasonable solution and $\overline{c}$ some unavoidable cost, defined later, then
$$c_T(F) = c^D(F) + \max_{i=1,2} c^{S_i}(F^{S_i}) = \overline{c} + \max\{\sum_{e \in A_1^F} e, \sum_{e \in A_2^F} e\}$$
with $F^{S_i} = \cup_{i=1}^n M_i \cap F \cup U_2$ being the best possible recovery.
3. The first-stage cost function and the scenario cost functions obey the $\alpha_{[0,1]}$-deviation-condition.

If these conditions are satisfied, I is a yes-instance if there exits a first-stage solution $F \in \mathcal{F}$ with total cost $c_T(F) = \overline{c} + b$. It remains to define the cost functions and show that we can always restrict a proper first-stage solution set $\mathcal{G}$ to the set of feasible solutions.

To this end, the first-stage cost function c^D assigns to the elements in U_2 cost $6(b+1)$ and the to the elements in U_1 cost $4(b+1)$. All other elements get first-stage cost $2(b+1)$ (see Figure 2.2). The cost function of scenario S_1 assigns cost $2(b+1)+e_i$ to the ith even element, $i = 1, \ldots, n$, and cost $6(b+1)$ to the other even elements. The odd elements of the k remaining sets $M_{n+1}, \ldots, M_{n+k}$ get cost $8(b+1)$. All other costs are set to $2(b+1)$. The scenario S_2 exchanges the cost of the first n elements, i.e., the odd elements get cost $2(b+1)+e_i$ and the even ones $2(b+1)$ (see Figure 2.2). The cost for the k sets $M_{n+1}, \ldots, M_{n+k}$ remain as for S_1.

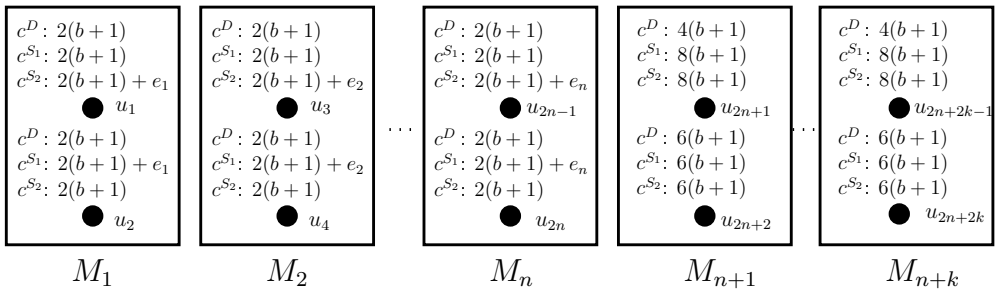


Figure 2.2.: Construction of I' for a given partition instance.

Note that all three conditions mentioned above are satisfied: If a first-stage solution contains an element $u_i \in U_2$, then its total cost are at least $(b+1)$ units higher than the total cost of the feasible solution F^1 containing all odd elements. Since the total cost of any reasonable

solution are less than $c_T(F^1)$, the first property is satisfied. The second property is easy to see for the unavoidable cost

$$\overline{c} = 2n(b+1) + 4k(b+1) + 2n(b+1) + 6k(b+1).$$

Finally, the cost functions satisfy the $\alpha_{[0,1]}$-deviation-condition.

So far, we assumed that the set of first-stage solutions consists of all feasible first-stage solutions. Due to the high first-stage cost any set $G \subseteq U$ containing more than $2n + 2k$ elements and a feasible solution $F \in \mathcal{F}$ has higher cost than a reasonable feasible solution. Thus, we can restrict any proper first-stage solution set $\mathcal{G}$ to the set $\mathcal{F}$. Furthermore, if $A_1 \cup A_2$ is a partition of $\{e_1, \ldots, e_n\}$, then we can define the reasonable solution $F_{(A_1,A_2)} \in \mathcal{F}$, which contains the even element u_{2i} of the set M_i, if $e_i \in A_1$, and the odd element u_{2i-1} of M_i, if $e_i \in A_2$, $i = 1, \ldots, n$, and the U_1. Its total cost is

$$c_T(F_{(A_1,A_2)}) = \overline{c} + \max\Big\{\sum_{b\in A_1} b, \sum_{c\in A_2} c\Big\}.$$

Hence, we obtain, that there exists a first-stage solution in I' with total cost equal to $\overline{c} + b$ if and only if I is a yes-instance.

We extend the proof to the case, when k depends on the number of elements of the WDHS instance. To this end, let $k = \frac{2(\ell-1)}{2\ell} \cdot |U|$ for any $\ell \in \mathbb{N}$. Instead of U consisting of $2n + 2k$ elements divided into $n + k$ sets, we construct a set with $2n + 2(\ell - 1) \cdot n$ elements and $\ell \cdot n$ disjoint sets. The first stage and the scenario cost for the first n sets $M_1, \ldots, M_n$ remain the same as above and the cost for the last $(\ell - 1) \cdot n$ sets are defined like the ones for the last k sets. With the same arguments as before we can restrict the set of first-stage solutions to $\mathcal{F}$. Due to the restriction of k, we are allowed to exchange

$$k = \frac{(\ell-1)}{2\ell}(2 \cdot \ell n) = (\ell - 1) \cdot n$$

elements. Thus, the recovery should be used to change the elements of the last $(\ell - 1) \cdot n$ sets. Since the unavoidable cost for any $F \in \mathcal{F}$ are $\overline{c}' = 2n \cdot 2(b + 1) + (\ell - 1) \cdot n \cdot 10(b + 1)$, any $F \in \mathcal{F}$ with total cost smaller than $\overline{c}' + 2b$ contains the $(\ell - 1) \cdot n$ odd elements of the last $(\ell - 1) \cdot n$ sets in the first stage and exchanges them in each scenario for the even elements. As in the constant case there exists a partition $A_1 \dot\cup A_2 = A$ and $\sum_{e\in A_1} e = b$ if and only if a first-stage solution with cost $\overline{c}' + b$ exists. □

At the end of this subsection we show that the WDHS can be solved in pseudo-polynomial time via dynamic programing for a constant number of scenarios. But before that, we consider the case where the number of scenarios is not constant.

Theorem 2.2.2. *The k-Dist-RR WDHS problem is strongly* **NP***-hard for discrete scenario sets and proper first-stage solution sets even if the $\alpha_{[0,1]}$-deviation-condition is fulfilled and k is constant or k depends on the number of elements of the WDHS instance.*

Proof. We start again with the case of constant k and show a reduction from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. We construct an instance of the k-Dist-RR WDHS problem I' in the following way: The set U contains for each variable x_i two elements a_i and b_i, $i = 1, \ldots, n$, where a_i represents a true assignment of x_i and b_i a false assignment of x_i. Furthermore, the sets U contains two elements c_i and d_i,

$i = n+1, \ldots, n+k$, which will capture the recovery action. The elements a_i and b_i form a set M_i for $i = 1, \ldots, n$ as the elements c_i and d_i form a set M_i for $i = n+1, \ldots, n+k$. As first-stage cost, the elements c_i get cost $6(n+k)$, the elements d_i cost $4(n+k)$ and all other elements cost $2(n+k)$. Again, these cost values are chosen in a way, that every reasonable first-stage solution contains exactly one element from each set $M_1, \ldots, M_{n+k}$.

The scenario set $\mathcal{S}$ represents the clause of the instance I. For every clause C_j, $j = 1, \ldots, m$, the scenario set $\mathcal{S}$ contains a scenario S_j with

$$c^{S_j}(a_i) = \begin{cases} 2(n+k)+2 & \text{if } \overline{x}_i \in C_j \\ 2(n+k)+1 & \text{otherwise} \end{cases} \text{ and } c^{S_j}(b_i) = \begin{cases} 2(n+k)+2 & \text{if } x_i \in C_j \\ 2(n+k)+1 & \text{otherwise} \end{cases}$$

for $i = 1, \ldots, n$ and

$$c^{S_j}(c_i) = 6(n+k) \text{ and } c^{S_j}(d_i) = 8(n+k)$$

for $i = n+1, \ldots, n+k$. As the element a_i corresponds to a true assignment and b_i to a false assignment of x_i, $i = 1, \ldots, n$, the scenario cost S_j represent a verification of a clause C_j. We suppose for a moment that a feasible first-stage solution is a feasible solution of the WDHS instance and that all recovery actions are used on the last k sets. Then the scenario cost is at most $2n(n+k) + 6 + (n-3) + 6k(n+k)$ for every scenario $S \in \mathcal{S}$. If the clause C_j, $j \in \{1, \ldots, m\}$, contains the literal x_i, $i \in \{1, \ldots, n\}$, then this scenario cost for S_j is reduced to $2n(n+k) + 5 + (n-3) + 6k(n+k)$ if the solution contains a_i. On the other hand, if the clause C_j contains the literal $\overline{x}_i$, then this cost in S_j is reduced to $2n(n+k) + 5 + (n-3) + 6k(n+k)$ if the first stage solution contains the element b_i. Hence, a clause C_j is verified if and only if the scenario cost S_j are smaller or equal to $2n(n+k) + 5 + (n-3) + 6k(n+k)$.

We will now show, that there is always an optimal first-stage solution $F \in \mathcal{F}$, if the first-stage solution set $\mathcal{G}$ is proper, and then, that any first-stage solution with small cost contains the elements d_i, $i = n+1, \ldots, n+k$.

Let $\mathcal{G}$ be a proper first-stage solution set. The first-stage cost of any first-stage solution $G \in \mathcal{G}$ are bounded by

$$c^D(G) \geq n \cdot 2(n+k) + k \cdot 4(n+k),$$

and the recovery cost of any feasible solution F of the WDHS instance by

$$2n(n+k) + 3 + (n-3) + k \cdot 6(n+k) \leq c_{RR}(F) \leq 2n(n+k) + (n-3) + 6 + k \cdot 6(n+k).$$

Hence, for every solution $F \in \mathcal{F}$, which includes the elements d_i in the first stage, $i = n+1, \ldots, n+k$, and every other solution $G \in \mathcal{G} \backslash \mathcal{F}$, we get

$$\begin{aligned} c_T(F) &= c^D(F) + c_{RR}(F) \\ &\leq 2n(n+k) + 2n + k \cdot 6(n+k) + n \cdot 2(n+k) + k \cdot 4(n+k) \\ &= 4n(n+k) + 10k(n+k) + 2n \\ &\leq 4n(n+k) + 10k(n+k) + n + 2(n+k) \leq c_T(G). \end{aligned}$$

We can therefore restrict the instance to the case where $\mathcal{F}$ is the set of feasible first-stage solutions.

Due to the definition of the cost functions, any first-stage solution with small cost should contain the elements d_i, $i = n+1, \ldots, n+k$, in the first stage and removes them in the second stage. Thus, an obvious translation from such a first-stage solution F to an assignment of the

x variables is to set $x_i =$ true if $a_i \in F$ and $x_i =$ false if $b_i \in F$, $i = 1, \ldots, n$. This assignment satisfies a clause C_j, $j = 1, \ldots, m$ if and only if the corresponding scenario cost for F are smaller than or equal to $2n(n+k) + k6(n+k) + (n-3) + 5$. Hence, I is a yes-instance if and only if there exists a solution $F \in \mathcal{F}$ with $c_T(F) \leq 4n(n+k) + 10k(n+k) + (n-3) + 5$.

Analog to the proof of Theorem 2.2.1 we can extend this construction to $k = \frac{2(\ell-1)}{2\ell} \cdot |U|$ and any $\ell \in \mathbb{N}$ with $2n + 2(\ell-1) \cdot n$ elements and $\ell \cdot n$ disjoint sets. This shows, that the problem remains **NP**-hard, if k is a fraction of the size of the considered LCMin instance. □

Note that if the $\alpha_{[0,1]}$-condition is dropped, if the first-stage solution set $\mathcal{G}$ equals the set of all feasible solutions $\mathcal{F}$, if the first-stage cost are set to 0 and if the second stage cost assign values 0 instead of $2(n+k)+1$ and 1 instead of $2(n+k)+2$, then the reduction proves that there exists no approximation algorithm with a factor better than $\rho = 1.5$. However, in this case a better lower bound on the best possible approximation factor can be achieved by adapting the reduction from 3SAT to the robust shortest path problem by Kasperski and Zieliński [70]. Their proof improves the lower bound to $\rho = 2$.

The **NP**-hardness proofs extends to the case where the first-stage solution set consists of all subsets of the given set U. For different first-stage solution sets the complexity status remains open.

The WDHS problem can be interpreted as a special case of several classical combinatorial minimization problems such as the shortest path or the minimum spanning tree problem. This is also the case for their k-Dist-recoverable robust versions, as we will show in the following corollary.

Corollary 2.2.3. *The k-Dist-RR version of the minimum spanning tree problem on series parallel graphs, the shortest path problem on series-parallel graphs, the minimum (s,t)-cut problem on series-parallel graphs and the minimum perfect matching problem on series-parallel graphs with proper first stage solution sets are weakly* **NP***-hard for $|\mathcal{S}_D| = 2$ and strongly* **NP***-hard if the number of scenarios is not constant, even if the first-stage cost function and the scenario cost functions obey the $\alpha_{[0,1]}$-deviation-condition.*

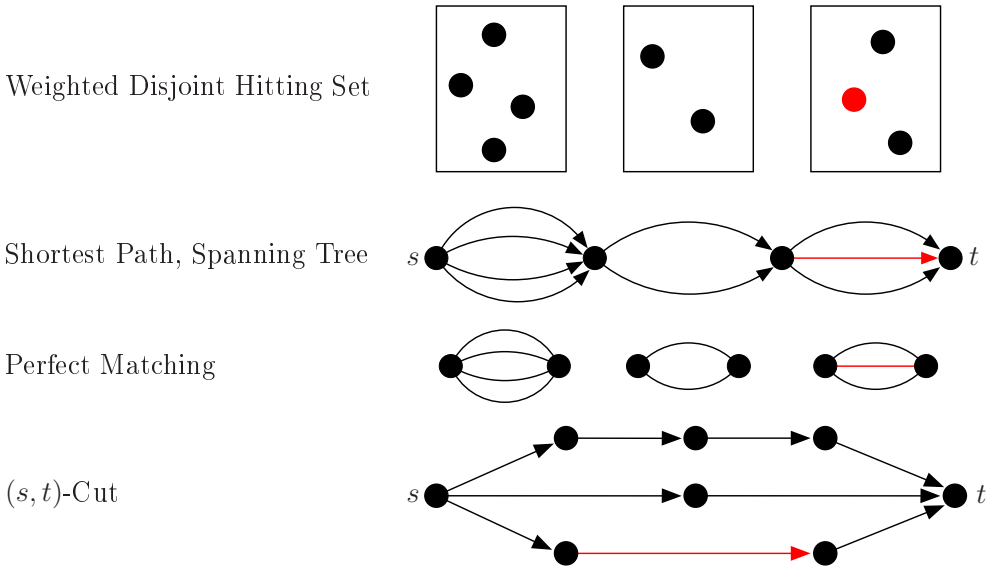


Figure 2.3.: The different constructions illustrate how to obtain a shortest path problem, a minimum spanning tree problem, a minimum perfect matching problem or a minimum (s,t)-cut problem from a WDHS instance.

Proof. Any instance of the weighted disjoint hitting set problem can be interpreted as one of the listed problems: Let I be a WDHS instance with a set U of elements and d disjoint sets $M_1, \ldots, M_d$ of U. Then the corresponding minimum spanning tree and the shortest path instance use a graph G that is an extension of a simple (s,t)-path of length d, in which the ith arc is replaced by $|M_i|$ parallel arcs, $i = 1, \ldots, d$. The length of a path is determined by its number of arcs. Obviously, there is a one-to-one correspondence between arcs and elements in the different instances (see Figure 2.3). Furthermore, any simple (s,t)-path or spanning tree equals a feasible solution of the WDHS instance, and vice verse. Also the cost functions and the first-stage solution sets can be transferred canonically. Hence, the k-Dist-RR versions are alike.

In a similar way a reduction is obtained to the minimum perfect matching problem. Instead of a path of length d, there are d arcs, where again the ith arc is replaced by $|M_i|$-parallel arcs, $i = 1, \ldots, d$. In the minimum (s,t)-cut problem, each set M_i is represented by an (s,t)-path of length $|M_i|$, $i = 1, \ldots, d$ and G is the parallel composition of these paths. All other relations remain as for the first two cases. □

An open question is to find better lower bounds on the best possible approximation factors for the k-Dist-RR WDHS problem, since these bounds transfer automatically to all other problems mentioned in Corollary 2.2.3. On the other hand, a 2-approximation algorithm for the k-Dist-RR WDHS problem would prove that the obtained lower bound is tight at least for this problem.

Before we consider an approximation algorithm, we shortly discuss the complexity of computing the total cost of a given first-stage solution. Let $(U, \mathcal{F}, c)$ be an LCMin instance solvable in polynomial time, $\mathcal{G}$ be a first-stage solution set, $c^D : U \to \mathbb{N}$ be a first-stage cost function and $\mathcal{S}_D$ be a discrete scenario set such that each scenario $S \in \mathcal{S}_D$ defines a scenario cost function $c^S : U \to \mathbb{N}$. If we can compute for every scenario $S \in \mathcal{S}_D$ a solution F^S with $|F^S \backslash F| \leq k$ and F^S minimizes $c^S(F^S)$, then the total cost of F are determined by

$$c_T(F) = c^D(F) + \max_{S \in \mathcal{S}_D} c^S(F^S).$$

The constraint $|F^S \backslash F| \leq k$ can be modeled by the cost function $\ell : U \to \mathbb{N}$ with $\ell(u) = 0$ if $u \in F$ and $\ell(u) = 1$ otherwise. Thus we obtain a constraint LCMin problem of finding a feasible solution $F' \in \mathcal{F}$ with $\ell(F') \leq k$ which minimizes the scenario cost $c^S(F')$. If k is constant, we can always solve this problem efficiently, since $\mathcal{F}_F^k$ contains a number of solutions that is bounded polynomially by the input. If k is not constant, this needs not to hold. On the one hand, this problem can be solved in polynomial time for the shortest path problem [6, 64] or the minimum spanning tree problem via weighted matroid intersection [48]. On the other hand, since the upper bounded cardinality constraint minimum (s,t)-cut problem is strongly **NP**-hard (Theorem B.0.3), also this special optimization problem cannot be solved in polynomial time, unless $\mathbf{P} = \mathbf{NP}$: We add an extra vertex s' and the arc (s', s) with scenario cost $K+1$ to the given constrained minimum (s,t)-cut instance. Then there exists a minimum (s,t)-cut C with cost $c^S(C) \leq K$ containing at most k arcs if and only if the scenario cost of the first-stage solution $C' = \{(s', s)\}$ is at most K. By adding first-stage cost 0 to (s', s), the reduction implies the **NP**-hardness of computing the total cost for a first-stage solution.

Corollary 2.2.4. *The total cost for a first-stage solution can be computed in polynomial time for the k-Dist-RR version of the minimum spanning tree or the shortest path problem. For the k-Dist-RR version of the minimum (s,t)-cut problem the computation of the total cost is strongly* **NP***-hard.*

Moving to more constructive results, there is a simple 1.5 approximation algorithm for k-Dist-RR LCMin instances obeying the $\alpha_{[0,1]}$-deviation.

Theorem 2.2.5. *Let $(U, \mathcal{F}, c)$ be an LCMin instance, $\mathcal{G}$ be a first-stage solution set, $\mathcal{S}$ be a set of scenarios, $c^D : U \to \mathbb{N}$ be a first-stage cost function and $k \in \mathbb{N}$ a recovery parameter defining a k-Dist-RR LCMin instance I'. If c^D and $\mathcal{S}$ satisfy the $\alpha_{[0,1]}$-deviation-condition and if $\mathcal{G}$ is a proper first-stage solution set, then any optimal solution of the $(U, \mathcal{F}, c^D)$ instance is a 1.5-approximation of I'.*

Proof. Let $F_{\min}$ be an optimal solution of the $(U, \mathcal{F}, c^D)$ instance. For any first-stage solution $G \in \mathcal{G}$ we get

$$\begin{aligned} c_T(G) &= c^D(G) + \max_{S \in \mathcal{S}} \min_{F \in \mathcal{F}_G^k} c^S(F) \\ &\geq c^D(F_{\min}) + \max_{S \in \mathcal{S}} \min_{F \in \mathcal{F}} c^D(F) \geq 2c^D(F_{\min}) \end{aligned}$$

and therefore

$$\mathrm{OPT} \geq 2c^D(F_{\min}),$$

where OPT denotes the total cost of an optimal solution in the given k-Dist-RR LCMin instance I'.

Using this bound we obtain

$$\begin{aligned} c_T(F_{\min}) &= c^D(F_{\min}) + \max_{S \in \mathcal{S}} \min_{F \in \mathcal{F}_{F_{\min}}^k} c^S(F) \\ &\leq c^D(F_{\min}) + \max_{S \in \mathcal{S}} c^S(F_{\min}) \\ &\leq c^D(F_{\min}) + 2 \cdot c^D(F_{\min}) \leq 1.5 \cdot \mathrm{OPT}\,. \end{aligned}$$

□

By choosing the minimum feasible solution w.r.t. to the first-stage cost function c^D, one gets, with the same argument, a $(1 + \frac{\alpha}{2})$-approximation for any first-stage cost function c^D and a scenario set $\mathcal{S}$ fulfilling the α-condition.

The arguments to prove Theorem 2.2.5 are straight forward. An open question is how to construct approximation algorithms with better approximation factors.

Before we start with an investigation of interval scenarios, we will introduce a pseudo-polynomial algorithm for solving the k-Dist-RR WDHS problem with a constant number of scenarios.

2.2.1. Weighted Disjoint Hitting Set Problem

We will now introduce a way to solve the k-Dist-RR WDHS problem in pseudo-polynomial time when the number of scenarios is constant and the set of feasible solutions is the set of feasible first-stage solutions.

Let $U = \{u_1, \ldots, u_n\}$ be a set of n elements and $\mathcal{M} = \{M_1, \ldots, M_d\}$ be a set of d pairwise disjoint subsets of U. For simplicity we denote with u_{ij} the jth element in the set M_i, $i = 1, \ldots, n$, $j = 1, \ldots, |M_i|$. We will solve the corresponding k-Dist-RR WDHS problem for a

first-stage cost function $c^D : U \to \mathbb{N}$, a given set of r scenarios $\mathcal{S}_D = \{S_1, \ldots, S_r\}$, a recovery parameter k and the set of feasible first-stage solutions

$$\mathcal{G} = \{F \subseteq U \mid |F \cap M_i| = 1 \; \forall i = 1, \ldots, d\} = \mathcal{F},$$

via a dynamic program based on a recursive formula. We therefore define the following parametrized problem for the parameter $d' \in \{1, \ldots, d\}$, $k^S \in \{0, \ldots, k\}$ and $\alpha^S, \beta^S \in \{0, \ldots, C\}$ for all $S \in \mathcal{S}_D$ with $C = \max_{S \in \mathcal{S}_D} \sum_{i=1}^{d} \max_{u \in M_i} c^S(u)$:

Given: The sets $M_1, \ldots, M_{d'}$, a scenario dependent recovery parameter $0 \leq k^S \leq k$, and an upper bound α^S on the cost occurring for scenario S in the sets $M_{d'+1}, \ldots, M_d$ and an upper bound β^S on the cost occurring for scenario S in the sets $M_1, \ldots, M_{d'}$ for every scenario $S \in \mathcal{S}_D$.

Find: A set $F \subseteq U$ and a recovery set $F^S \subseteq U$ for every scenario $S \in \mathcal{S}_D$ such that

- F and F^S are feasible solutions according to the first d' sets, i.e., $|F \cap M_i| = 1$ and $|F^S \cap M_i| = 1$ for $i = 1, \ldots, d'$ and $S \in \mathcal{S}_D$;
- F^S are feasible recovery sets according to the recovery parameter k^S, $S \in \mathcal{S}_D$, i.e, $|F^S \backslash F| \leq k^S$;
- F^S obeys the upper cost bound

$$\sum_{u \in F^S} c^S(u) \leq \beta^S;$$

- F and F^S minimize the cost function

$$c\left(d', \left\{k^S, \alpha^S, \beta^S\right\}_{S \in \mathcal{S}_D}\right) = \sum_{u \in F} c^D(u) + \max_{S \in \mathcal{S}_D} \left(\beta^S + \alpha^S\right).$$

In other words, compute a first-stage solution F and its recovery solutions according to the first d' sets using at most k^S recovery actions, such that the scenario cost induced in $M_1, \ldots, M_{d'}$ are bounded by β^S and the cost $c^D(F) + \max_{S \in \mathcal{S}_D}(\beta^S + \alpha^S)$ are minimized. An optimal solution to the k-Dist-RR WDHS problem is given by $\min_{\beta \in \{0,\ldots,C\}^r} c(d, \{k, 0, \beta^S\}_{S \in \mathcal{S}_D})$.

We will now show:

1. How we can compute $c(1, \{k^S, \alpha^S, \beta^S\}_{S \in \mathcal{S}_D})$ for any $k^S \in \{0, \ldots, k\}$ and $\alpha^S, \beta^S \in \{0, \ldots, C\}$ in polynomial time.
2. That the cost function c can be computed via a recursive formula.
3. How we can extract a feasible solution from this recursive formula.

1. Let $y = (1, \{k^S, \alpha^S, \beta^S\}_{S \in \mathcal{S}_D})$ be some parameter set. If $k^S \notin \{0, \ldots, k\}$, we set $c(y) = \infty$, since there exists no feasible solution to this problem setting. In the other case, we define $b_1^S = \arg\min_{u \in M_1} c^S(u)$ for every $S \in \mathcal{S}$. Obviously this element is chosen as recovery if we decide to use a recovery action in this set in the scenario S. Furthermore, we set $\ell \in \{0, 1\}^r$ with $\ell^{S_i} = 0$ if $k^{S_i} = 0$, and $\ell^{S_i} = 1$ otherwise, $i = 1, \ldots, r$. This vector determines whether we can use a recovery action for scenario S or not. According to ℓ, we define for every element $u \in M_1$ and scenario $S \in \mathcal{S}_D$ the function $\gamma_1^S(u, \ell)$ with

$$\gamma_1^S(u, \ell) = (1 - \ell^S)c^S(u) + \ell c^S(b_1^S),$$

which displays the induced cost in scenario S by the element u under consideration of the recovery distribution ℓ. If there exists an element $u \in M_1$ with $\gamma_1^S(u,\ell) \leq \beta^S$ for all $S \in \mathcal{S}_D$, choose such an u^* with minimal first-stage cost $c^D(u^*)$. Hence, this element is an optimal solution to $c(1, \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D})$ with

$$c(1, \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D}) = c^D(u^*) + \max_{S\in\mathcal{S}_D} \left(\beta^S + \alpha^S\right).$$

If no element $u \in M_1$ satisfies $\gamma_1^S(u,\ell) \leq \beta^S$ for all $S \in \mathcal{S}_D$, no feasible solution to the parametrized problem exists and therefore $c(1, \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D}) = \infty$.

2. In order to derive a recursive formula, we consider different distributions $\ell \in \{0,1\}^r$ over the recovery actions taken in set $M_{d'}$, i.e., $\ell^S = 1$ if a recovery action is taken in $S \in \mathcal{S}_D$ and $\ell^S = 0$ otherwise. If $\ell^S = 1$, then independent of the element $u \in M_{d'}$ being part of the first-stage solution, the element $b_{d'}^S = \arg\min_{u\in M_{d'}} c^S(u)$ is chosen as recovery. Hence, as soon as the recovery distribution is fixed, we decide for each element $u \in M_{d'}$ whether it is chosen to be part of the first-stage solution or not. Like in the case $d' = 1$, we define, for every element u and any distribution ℓ, the value $\gamma_{d'}^S(u,\ell) = (1-\ell^S)c^S(u) + \ell^S c^S(b_{d'}^S)$ and thus obtain the following recursive formula:

$$\begin{aligned} & c\left(d', \left\{k^S, \alpha^S, \beta^S\right\}_{S\in\mathcal{S}_D}\right) \\ = & \min_{\substack{u\in M_{d'} \\ \ell\in\{0,1\}^r}} \left\{c\left(d'-1, \left\{k^S - \ell^S, \alpha^S + \gamma_{d'}^S(u,\ell), \beta^S - \gamma_{d'}^S(u,\ell)\right\}_{S\in\mathcal{S}_D}\right) + c^D(u)\right\}. \end{aligned}$$

3. We can construct a feasible solution for $c(d, \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D})$ based on an optimal solution for $c(d'-1, \{k^S - \ell^S, \alpha^S + \gamma_{d'}^S(u,\ell), \beta^S - \gamma_{d'}^S(u,\ell)\}_{S\in\mathcal{S}_D})$ defined by some element $u \in M_{d'}$ and a distribution $\ell \in \{0,1\}^r$ by adding u to the first stage solution and $b_{d'}^S$ to the recovery in scenario S if $\ell^S = 1$ and the element u to the recovery in scenario S if $\ell^S = 1$, as long as $k^S - \ell^S \geq 0$.

Theorem 2.2.6. *The k-Dist-RR WDHS problem defined on a set U of n elements and d disjoint subsets $M_1, \ldots, M_d$ with r different discrete scenarios $S_1, \ldots, S_r$ defining cost functions $c^{S_i} : U \to \mathbb{N}$, $i = 1, \ldots, r$, can be solved in $\mathcal{O}\left(n^{2+r}(2C^2)^r\right)$ via dynamic programming with $C = \sum_{i=1}^{d} \max_{u\in M_i} c^S(u)$.*

Proof. It remains to show that the recursive formula is correct. Since we can derive from every feasible solution for $c(d', \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D})$ a feasible solution for $c(d'-1, \{k^S - \ell^S, \alpha^S + \gamma_{d'}^S(u,\ell), \beta^S - \gamma_{d'}^S(u,\ell)\}_{S\in\mathcal{S}_D})$ defined by some $u \in M_{d'}$ and some distribution $\ell \in \{0,1\}^r$, we obtain

$$\begin{aligned} & c\left(d', \left\{k^S, \alpha^S, \beta^S\right\}_{S\in\mathcal{S}_D}\right) \\ \leq & \min_{\substack{u\in M_{d'} \\ \ell\in\{0,1\}^r}} \left\{c\left(d'-1, \left\{k^S - \ell^S, \alpha^S + \gamma_{d'}^S(u,\ell), \beta^S - \gamma_{d'}^S(u,\ell)\right\}_{S\in\mathcal{S}_D}\right) + c^D(u)\right\}. \end{aligned}$$

Now suppose there is a solution $F_{d'}$ and $F_{d'}^S$ for all $S \in \mathcal{S}_D$ with

$$\begin{aligned} & c\left(d', \left\{k^S, \alpha^S, \beta^S\right\}_{S\in\mathcal{S}_D}\right) \\ < & \min_{\substack{u\in M_{d'} \\ \ell\in\{0,1\}^r}} \left\{c\left(d'-1, \left\{k^S - \ell^S, \alpha^S + \gamma_{d'}^S(u,\ell), \beta^S - \gamma_{d'}^S(u,\ell)\right\}_{S\in\mathcal{S}_D}\right) + c^D(u)\right\} \end{aligned}$$

and let d' be chosen minimal in this respect. Let $u_{d'j}$ be the element contained in $F_{d'}$ and $\ell^S = 0$ if $u_{d'j} \in F^S_{d'}$ and $\ell^S = 1$ otherwise. Thus

$$\begin{aligned}
&c\left(d', \{k^S, \alpha^S, \beta^S\}_{S\in\mathcal{S}_D}\right)\\
=&\sum_{u\in F_{d'}} c^D(u) + \max_{S\in\mathcal{S}_D} \beta^S + \alpha^S\\
=&\sum_{u\in F_{d'}\setminus\{u_{dj}\}} c^D(a) + \max_{S\in\mathcal{S}_D} \left\{\beta^S - \gamma^S_{d'}(u_{d'j},\ell) + \alpha^S + \gamma^S_{d'}(u_{d'j},\ell)\right\} + c^D(u_{d'j})\\
\geq& c\left(d'-1, \left\{k^S - \ell^S, \alpha^S + \gamma^S_{d'}(u_{d'j},\ell), \beta^S - \gamma^S_{d'}(u_{d'j},\ell)\right\}_{S\in\mathcal{S}_D}\right) + c^D(u_{d'j}),
\end{aligned}$$

which is a contradiction.

Since α^S and β^S can be bounded by C with $C = \sum_{i=1}^{d} \max_{u\in M_d} c^S(u)$, the dynamic program obtained through the recursive formula runs in $\mathcal{O}(n^{2+r}(2C^2)^r)$. □

For general k-Dist-RR LCMin problems there exists no pseudo-polynomial algorithm, since for the shortest path problem the k-Dist-RR version is not approximable for one discrete scenario (Corollary 2.5.3). Yet, it remains open if we can construct further pseudo-polynomial algorithms by using methods from multi-criteria optimization as it is the case for robust optimization problems. Another question is, whether an FPTAS for the k-Dist-RR WDHS problem exists.

2.3. Interval Scenarios

The interval scenario set $\mathcal{S}_I$ is defined indirectly for an LCMin instance $(U, \mathcal{F}, c)$ by lower and upper bounds $\underline{c}(u)$ and $\overline{c}(u)$ on the scenario cost functions for each element $u \in U$, $0 \leq \underline{c}(u) \leq \overline{c}(u)$. For each cost function $c : U \to \mathbb{N}$ with $c(u) \in [\underline{c}(u), \overline{c}(u)]$ there exists a scenario $S \in \mathcal{S}_I$ with $c^S = c$ and every scenario cost function obeys these bounds. The number of different scenarios in $\mathcal{S}_I$ may be exponential in the number of elements U. But due to the simple structure of the scenario set and an obvious worst-case scenario, the interval scenario case is equivalent to the discrete scenario case with just one scenario.

Theorem 2.3.1. *The k-Dist-RR LCMin problem with interval scenario sets is equivalent to the k-Dist-RR LCMin problem with just one discrete scenario whose cost function is given by the upper cost bounds on each element of the instance.*

Proof. First note that the set of first-stage solutions and the recovery of a k-Dist-RR LCMin instance are not influenced by the scenario set. Let $\underline{c}(u)$, $\overline{c}(u)$ be lower and upper cost bound on the elements $u \in U$ of the underlying LCMin instance $(U, \mathcal{F}, c)$. We define the cost function of the scenario $S_{\max}$ by $c^{S_{\max}}(u) = \overline{c}(u)$ for all $u \in U$ and show that the total cost of a first-stage solution F according to interval scenarios, denoted by $c^I_T(F)$, equals the total cost of F according to the discrete scenario $S_{\max}$, denoted by $c^{\max}_T(F)$. The cost $c^I_T(F)$ can be bounded by

$$\begin{aligned}
c^I_T(F) &= c^D(F) + \max_{S\in\mathcal{S}_I} \min_{F'\in\mathcal{F}^k_F} c^S(F')\\
&\geq c^D(F) + \min_{F'\in\mathcal{F}^k_F} \overline{c}(F') = c^{\max}_T(F)
\end{aligned}$$

and

$$\begin{aligned} c_T^I(F) &= c^D(F) + \max_{S\in\mathcal{S}} \min_{F'\in\mathcal{F}_F^k} c^S(F') \\ &\leq c^D(F) + \max_{S\in\mathcal{S}} \min_{F'\in\mathcal{F}_F^k} \overline{c}(F') = c_T^{\max}(F). \end{aligned}$$

Therefore, $c_T^I(F) = c_T^{\max}(F)$ for all feasible first-stage solutions F. □

No valid statement about the complexity status of all k-Dist-RR LCMin problems can be given since the complexity strongly depends on the given first-stage solution set $\mathcal{G}$, the considered combinatorial minimization problem, and whether k is taken as constant. In the shortest path problem, for example, the problem is strongly **NP**-hard and cannot be approximated with a constant factor when $\mathcal{G}$ is the set of all simple (s,t)-paths (Corollary 2.5.3). If $\mathcal{G}$ is the set of all (s,t)-paths, however, the k-Dist-RR shortest path problem can be solved in polynomial time for constant k (Theorem 2.5.9). For k being not constant, the problem is again strongly **NP**-hard and not approximable. On the other hand, the k-Dist-RR WDHS problem can be solved in polynomial time for an arbitrary k, whereas we only succeeded to show that the k-Dist-RR version of the minimum weight basis problem for matroids can be solved in polynomial time for constant k.

2.3.1. Weighted Disjoint Hitting Set Problem

We will consider the k-Dist-RR WDHS problem with proper first-stage solution sets for interval scenarios. Let U be a set of n elements divided into d disjoint sets $M_1,\dots,M_d$, $\mathcal{G}$ be a proper first-stage solution set, $c^D : U \to \mathbb{N}$ be a first-stage cost function and $\overline{c} : U \to \mathbb{N}$ be the upper cost bound treated as the only scenario cost function occurring in this setting. Let us start with two observations:

1. Let $F \in \mathcal{G}$ and $\bar{F} \in \mathcal{F}$ be the best solution from the recovery of F, i.e., $\bar{F} = \arg\min_{F\in\mathcal{F}_F^k} \overline{c}(F)$. With $I(F) \subseteq \{1,\dots,d\}$ we define the indices of the sets in which a recovery action is taken, i.e., $I(F) = \{i \in \{1,\dots,d\} \mid F \cap \bar{F} \cap M_i = \emptyset\}$. Then

$$c_T(F) \geq \sum_{i\in I(F)} \left(\min_{u\in M_i} c^D(u) + \min_{u\in M_i} \overline{c}(u) \right) + \sum_{\substack{i\in\{1,\dots,d\} \\ i\notin I(F)}} \min_{u\in M_i} (c^D + \overline{c})(u).$$

2. If $\mathcal{G}$ is a proper first-stage solution set, there is always an optimal solution $F \in \mathcal{G}$ with

$$|F \cap M_i| = 1 \quad \text{for all } i = 1,\dots,d.$$

Due to the first observation, we can assume w.l.o.g. that any first-stage solution F contains one element $u_{i1} = \arg\min_{u\in M_i} c^D(u)$ for $i \in I(F)$ and one element $u_{i3} = \arg\min_{u\in M_j}(c^D + \overline{c})(u)$ otherwise. Therefore, the k-Dist-RR WDHS problem reduces to deciding in which sets the recovery action should be used.

Let us consider the case of two given sets M_1 and M_2 and the decision to use one recovery action, i.e., $k = 1$. For $i = 1,2$ we define $u_{i1} = \arg\min_{u\in M_i} c^D(u)$, $u_{i2} = \arg\min_{u\in M_i} \overline{c}(u)$ and $u_{i3} = \arg\min_{u\in M_i}(c^D + \overline{c})(u)$ and finally we set $c^1(M_i) = c^D(u_{i1}) + \overline{c}(u_{i2})$ and $c^2(M_i) = (c^D + \overline{c})(u_{i3})$. The value $c^1(M_i)$ indicates the cost distributed by the set M_i to the total cost if a recovery action is taken in this set, $i = 1,2$. The value $c^2(M_i)$ indicates the

other case. Hence, if we decide to use the recovery for set M_1, we get a solution with total cost $c^1(M_1) + c^2(M_2)$. In the case we distribute the recovery to M_2, we pay $c^1(M_2) + c^2(M_1)$. Hence, the best way to use the recovery is for the set M_i, such that

$$c^1(M_i) + c^2(M_j) \leq c^1(M_j) + c^2(M_i),$$

for $i, j \in \{1, 2\}$, $i \neq j$. Since we can compare any two given sets and decide the best way to assign a recovery action, we can use this idea to derive a polynomial algorithm for computing an optimal first-stage solution.

Algorithm 2.1 k-Dist-RR WDHS with $\mathcal{S}_I$

Input : A set U, d sets $M_1, \ldots, M_d$ with $M_i \subseteq U$ and $M_i \cap M_j = \emptyset$ for all $i, j \in \{1, \ldots, d\}$ with $i \neq j$, a first-stage cost function $c^D : U \to \mathbb{N}$, scenario cost function $\overline{c} : U \to \mathbb{N}$, and a recovery parameter $k \in \mathbb{N}$.

Output : A first-stage solution F.

`/* Compute candidates for the first-stage solution */`

for $i = 1, \ldots, d$ **do**

Compute $u_{1i} = \arg\min_{u \in M_i} c^D(u)$ and $u_{2i} = \arg\min_{u \in M_i} \overline{c}(u)$ and $u_{3i} = \arg\min_{u \in M_i} (c^D(u) + \overline{c}(u))$.

Set $c^1(M_i) = c^D(u_{1i}) + \overline{c}(u_{2i})$ and $c^2(M_i) = c^D(u_{3i}) + \overline{c}(u_{3i})$

`/* Order reflects best use of recovery action */`

Sort $M_1, \ldots, M_d$ such that for all $j, \ell \in \{1, \ldots, d\}$ with $j \leq \ell$

$$c^1(M_{i_j}) + c^2(M_{i_\ell}) \leq c^1(M_{i_\ell}) + c^2(M_{i_j}).$$

return $F = \{u_{1i_1}, \ldots, u_{1i_k}, u_{3i_{k+1}}, \ldots, u_{3i_d}\}$

Theorem 2.3.2. *Algorithm 2.1 computes an optimal solution to a given k-Dist-RR WDHS problem for any proper first-stage solution set $\mathcal{G}$ and interval scenarios.*

Proof. We start by proving that the order defined by the algorithm on the sets $M_1, \ldots, M_d$ is transitive, i.e., if $M_a \leq M_b$ and $M_b \leq M_c$ then $M_a \leq M_c$. Let

$$c^1(M_a) + c^2(M_b) \leq c^1(M_b) + c^2(M_a)$$

and

$$c^1(M_b) + c^2(M_c) \leq c^1(M_c) + c^2(M_b).$$

Since $c^1(M) \leq c^2(M)$, we obtain

$$\begin{aligned} & c^1(M_a) + c^2(M_b) + c^1(M_b) + c^2(M_c) \leq c^1(M_b) + c^2(M_a) + c^1(M_c) + c^2(M_b) \\ \Leftrightarrow \quad & c^1(M_a) + c^2(M_c) \leq c^1(M_c) + c^2(M_a) \\ \Leftrightarrow \quad & M_a \leq M_c. \end{aligned}$$

For the remainder of the proof, we assume that the sets in $\mathcal{M} = \{M_1, \ldots, M_d\}$ are already ordered such that $M_i \leq M_j$ for $i \leq j$. Let us now suppose that F computed via Algorithm 2.1, is not an optimal solution, but F^* is an optimal solution with $I(F^*) \cap \{1, \ldots, k\} = \{1, \ldots, \ell\}$ and maximal ℓ. The set $I(F^*)$ consists of the indices of the sets the recovery action is used

on to obtain an optimal recovery from F^*. Let $r \in I(F^*)\backslash\{1,\dots,\ell\}$ and define the first-stage solution $F' = F^*$ but with $I(F') = I(F^*)\backslash\{r\} \cup \{\ell+1\}$. The total cost of the solution obtained by F' are

$$\begin{aligned} c_T(F') - c_T(F^*) &= \min_{u\in M_{\ell+1}} c^D(u) + \min_{u\in M_{\ell+1}} \overline{c}(u) + \min_{u\in M_r}(c^D+\overline{c})(u) \\ &\quad - \min_{u\in M_r} c^D(u) - \min_{u\in M_r} \overline{c}(u) - \min_{u\in M_{\ell+1}}(c^D+\overline{u})(u) \\ &= c^1(M_{\ell+1}) + c^2(M_r) - c^1(M_r) - c^2(M_{\ell+1}) \leq 0 \end{aligned}$$

since $r > \ell+1$. This is a contradiction to the choice of F^*. □

We showed that the k-Dist-RR WDHS problem can be solved in polynomial time if we consider proper first-stage solution sets. Similar techniques can be used to obtain the same result for first-stage solutions sets consisting of all subsets of the element set U.

2.3.2. Minimum Weight Basis Problem for Matroids

In this section we consider the k-Dist-RR version of the minimum weight basis (MWB) problem for matroids with interval scenarios and the set of bases as first-stage solutions. A *matroid* is a tuple $M = (E, \mathcal{I})$, where E is a finite set and $\mathcal{I} \subseteq 2^E$ is a non-empty collection of subsets of E, called *independent sets*, such that:

1. $\emptyset \in \mathcal{I}$;
2. if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$;
3. if $I, J \in \mathcal{I}$ with $|I| > |J|$ then there exists an element $i \in I \setminus J$ with $J + i \in \mathcal{I}$.

A set $I \subseteq E$ which is not in $\mathcal{I}$ is called a *dependent* set. A *basis* of $\mathcal{I}$ is a maximal independent set, i.e., adding one element leads to a dependent set. In a matroid all bases have the same cardinality and the *rank* r of a matroid is the cardinality of a basis. The *minimum weight basis* problem of a matroid is defined in the following way:

Given: A matroids $M = (E, \mathcal{I})$ and a cost function $c : E \to \mathbb{Z}$.

Find: A basis B of M with minimum cost $c(B)$.

A more detailed introduction to matroid theory is given in Section 4.5.

In order to define a feasible k-Dist-RR MWB instance, we add to a given matroid $M = (E, \mathcal{I})$ the set of all bases $\mathcal{B}$ as first-stage solutions, a cost function $c^D : E \to \mathbb{N}$ as first-stage cost function and a cost function $\overline{c} : E \to \mathbb{N}$ as an upper cost bound function of the interval scenarios. Then the k-Dist-RR MWB problem can be modeled by solving the following optimization problem

$$\min_{\substack{B_1,B_2\in\mathcal{B} \\ |B_2\backslash B_1|\leq k}} c^D(B_1) + \overline{c}(B_2) = \min_{\substack{B_1,B_2\in\mathcal{B} \\ |B_2\backslash B_1|\leq k}} c^D(B_1\backslash B_2) + \overline{c}(B_2\backslash B_1) + (c^D+\overline{c})(B_1 \cap B_2),$$

where B_1 is the first-stage solution and B_2 the recovery taken in the second stage. This problem can be solved efficiently, as we will show in the remaining subsection. The idea of our algorithm is to enumerate $B_1\backslash B_2$ and $B_2\backslash B_1$ and compute $B_1 \cap B_2$ instead of computing two bases simultaneously.

Let us assume that (B_1^*, B_2^*) is an optimal solution for the MWB instance I as described above and that we guessed the right set of elements $I_1 = B_1^*\backslash B_2^*$ and $I_2 = B_2^*\backslash B_1^*$ with $|I_1| =$

Algorithm 2.2 k-Dist-RR MWB problem

Input : Matroid $M = (E, \mathcal{I})$, cost functions $c^D : E \to \mathbb{N}$, $\overline{c} : E \to \mathbb{N}$, and a recovery parameter $k \in \mathbb{N}$

Output : Sets B_1 and B_2

Set r as the rank of M and $C = 1 + r \cdot \max_{e \in E}(c^D + \overline{c})(e)$.
Set $I_0^* = \emptyset$, $I_1^* = \emptyset$, $I_2^* = \emptyset$ and $D = \infty$
for $\kappa = 0, \ldots, k$ **do**
  for $I_1 \subseteq E$, $I_2 \subseteq E$ *with* $|I_1| = |I_2| = \kappa$, $I_1 \cap I_2 = \emptyset$ *and* $I_1, I_2 \in \mathcal{I}$ **do**
    Define matroids $M_i = (E, \mathcal{I}_i)$ with $\mathcal{I}_i = \{I \subseteq E \mid I \backslash I_j \in \mathcal{I}\}$ for $i, j \in \{1, 2\}$, $i \neq j$.
    Define weight function $w : E \to \mathbb{N}$ with $w(e) = C$ if $e \in I_1 \cup I_2$ and $w(e) = C - (c^D + \overline{c})(e)$ otherwise.
    Compute an optimal solution I^* to the maximum weighted matroid intersection problem (M_1, M_2, c).
    if $|I^*| = r + \kappa$ **then**
      Set $I_0' = I^* \backslash \{I_1 \cup I_2\}$
      if $c^D(I_1) + \overline{c}(I_2) + (c^D + \overline{c})(I_0') < D$ **then**
        Set $I_0^* = I_0'$, $I_1^* = I_1$ and $I_2^* = I_2$
        Compute $D = c^D(I_1) + \overline{c}(I_2) + (c^D + \overline{c})(I_0')$
return $B_1 = I_0^* \cup I_1^*$ and $B_2 = I_0^* \cup I_2^*$

$|I_2| = \kappa$, $\kappa \leq k$. Then we can compute a set $I_0 \subseteq E$ in polynomial time via weighted matroid intersection such that $I_0 \cup I_1$ and $I_0 \cup I_2$ are two bases in $(E, \mathcal{I})$ and an optimal solution to I. In general the *weighted matroid intersection problem* is given by two matroids $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ on the same set E and a weight function $w : E \to \mathbb{N}$. The objective of the weighted matroid intersection problem is to find an independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ with maximum weight $w(I)$. In our case, the corresponding weighted matroid intersection instance (M_1, M_2, w) is constructed in the following way: For $i, j \in \{1, 2\}$ and $i \neq j$ define the matroid $M_i = (E, \mathcal{I}_i)$ with $\mathcal{I}_i = \{I \subseteq E \mid I \backslash I_j \in \mathcal{I}\}$. Any set I' being part of the intersection can be transformed into two independent sets in M by setting $I_1' = I' \backslash I_1$ and $I_2' = I' \backslash I_2$. If I_1' and I_2' are bases, then I_2' is a feasible recovery for I_1', i.e., $|I_2' \backslash I_1'| = |I' \cap I_2| \leq k$. Finally we set the weight function $w : E \to \mathbb{N}$ to

$$w(e) = \begin{cases} C & \text{if } e \in I_1 \cup I_2, \\ C - (c^D + \overline{c})(e) & \text{otherwise,} \end{cases}$$

with $C = 1 + r \cdot \max_{e \in E}(c^D + \overline{c})(e)$ and r being the rank of M. Let I^* be an optimal solution to this weighted matroid intersection instance. Since $B_1^* \cup B_2^*$ is a feasible solution of matroid intersection instance, $|I^*| = r + \kappa$, as we will later show in detail. We will further prove in Theorem 2.3.3 that we obtain two bases in M by defining $\overline{B}_1 = I^* \backslash I_2$ and $\overline{B}_2 = I^* \backslash I_1$ which have the property

$$c^D(\overline{B}_1) + \overline{c}(\overline{B}_2) = c^D(B_1^*) + \overline{c}(B_2^*).$$

Thus, they are optimal solutions of the k-Dist-RR MWB instance. More formally, this procedure is described in Algorithm 2.2.

Theorem 2.3.3. *Algorithm 2.2 computes an optimal solution to a given k-Dist-RR MWB instance with interval scenarios $\mathcal{S}_I$ and the set of all bases as first-stage solution set.*

Proof. Let I^* be an optimal solution to this weighted matroid intersection instance (M_1, M_2, w) induced by two disjoint independent sets I_1 and I_2 with $|I_1| = |I_2| = \kappa$ as defined in Algorithm 2.2. Note that $|I^*| \leq r + \kappa$. If there exists a set $I_0 \subseteq E$ such that $I_0 \cup I_1$ and $I_0 \cup I_2$ are bases in M, then $I_0 \cup I_1 \cup I_2$ is a feasible solution of (M_1, M_2, w) containing exactly $\kappa + r$ elements. In that case,

$$\begin{aligned} w(I^*) \geq w(I_0 \cup I_1 \cup I_2) &= (k+\kappa) \cdot C - \sum_{e \in I_0} (c^D + \overline{c})(e) \\ &> (\kappa + r - 1) \cdot C \geq w(I'), \end{aligned}$$

for any feasible solution I' with $|I'| < (\kappa + r)$. Since $|I^* \backslash \{I_1 \cup I_2\}| = r - \kappa$ and $I_1 \cap I_2 = \emptyset$, the sets $I_0^* \cup I_1$ and $I_0^* \cup I_2$ are independent and these sets form a feasible solution for the k-Dist-RR MWB problem, where $I_0^* = I^* \backslash \{I_1 \cup I_2\}$. Furthermore, if $I_1 = B_1^* \backslash B_2^*$ and $I_2 = B_2^* \backslash B_1^*$ for an optimal first-stage solution B_1^* and its optimal recovery B_2^*, we obtain

$$\begin{aligned} c^D(B_1^*) + \overline{c}(B_2^*) &= c^D(I_1) + \overline{c}(I_2) + (c^D + \overline{c})(B_1^* \cap B_2^*) \\ &= c^D(I_1) + \overline{c}(I_2) - w(B_1 \cup B_2) + (k+\kappa) \cdot C \\ &\geq c^D(I_1) + \overline{c}(I_2) - w(I^*) + (k+\kappa) \cdot C \\ &= c^D(I_0^* \cup I_1) + \overline{c}(I_0^* \cup I_2). \end{aligned}$$

Thus, $I_0^* \cup I_1$ is an optimal solution of the k-Dist-RR MWB instance with the recovery $I_0^* \cup I_2$. Since there are at most kn^k different sets I_1 and I_2 with $I_1, I_2 \in \mathcal{I}$, $I_1 \cap I_2 = \emptyset$ and $|I_1| = |I_2| = \kappa$, $\kappa \leq k$, and since the weighted matroid intersection problem can be solved in polynomial time (Lawler [80] or Frank [48]), Algorithm 2.2 runs in polynomial time. □

Two major question remain open: If the set of first-stage solutions is proper, can the k-Dist-RR MWB problem be solved in polynomial time? In such a case, the set of all bases may not contain an optimal first-stage solution. And, if k is not constant, does the problem become **NP**-hard?

2.4. Γ-scenarios

Recall that for $\Gamma \in \mathbb{N}$ the set of Γ-scenarios of an LCMin instance $(U, \mathcal{F}, c)$ is defined as follows: Let $\underline{c}(u)$ and $\overline{c}(u)$ be lower and upper bounds on the scenario cost with $0 \leq \underline{c}(u) \leq \overline{c}(u)$ for all $u \in U$. A scenario $S \in \mathcal{S}_\Gamma$ is only allowed to have at most Γ cost values deviating from the lower bound, i.e., $|\{u \in U \mid c^S(u) > \underline{c}(u)\}| \leq \Gamma$. We focus again on the complexity status of k-Dist-RR LCMin problems.

The complexity status of a k-Dist-RR LCMin problem is closely related to the max-scenario problem introduced in Appendix A. The max-scenario problem is to find a scenario that maximizes the minimum cost of every feasible solution in the corresponding LCMin instance. We need one further definition, before we prove **NP**-hardness of several k-Dist-RR LCMin problems.

Definition 2.4.1. A class of LCMin problems satisfies the *add-condition*, if for every instance $(U, \mathcal{F}, c)$ and for every set U' with $U' \cap U = \emptyset$, there is another LCMin problem $(U \cup U', \mathcal{F}', c')$ in the same class such that $\mathcal{F}' = \mathcal{F} \cup \{U'\}$ and $c'(u) = c(u)$ for all $u \in U$.

The add-condition enables us to add a solution consisting of an arbitrary set of elements and with arbitrary cost to a given LCMin instance. The shortest path and the minimum (s,t)-cut problem satisfy the add-condition. For the minimum spanning tree problem or the minimum perfect matching problem, this is not the case.

Theorem 2.4.2. *Let $\mathcal{C}$ be a class of LCMin problems that satisfies the add-condition and such that the max-scenario problem on $\mathcal{C}$ is strongly* **NP***-hard. Then the k-Dist-RR version of the problems in $\mathcal{C}$ is strongly* **NP***-hard.*

Proof. Let $(U, \mathcal{F}, c)$ be an LCMin instance in the class $\mathcal{C}$ and let $\mathcal{S}_\Gamma$ be a set of Γ-scenarios defined by the lower and upper cost bounds $\underline{c}(u)$ and $\overline{c}(u)$ for each $u \in U$ and some $\Gamma \in \mathbb{N}$. We consider the decision version I of the corresponding max-scenario problem on $(U, \mathcal{F}, \mathcal{S}_\Gamma)$ with threshold $K \in \mathbb{N}$, which asks to decide whether there is a scenario $S \in \mathcal{S}_\Gamma$ with $\text{profit}(S) = \min_{F \in \mathcal{F}} c^S(F) \geq K$. We will now prove that the k-Dist-RR LCMin with Γ-scenarios is strongly **NP**-hard. We start by multiplying the values of $\underline{c}$, $\overline{c}$, and K by 2, such that the max-scenario instance I is transferred to the decision if there exists a scenario $S \in \mathcal{S}_\Gamma$ with $\text{profit}(S) \geq 2K$. If this is not the case, $\text{profit}(S) \leq 2K - 2$ for every $S \in \mathcal{S}_\Gamma$. We construct a k-Dist-RR instance I' of $(U, \mathcal{F}, c)$ with Γ-scenarios in the following way: We start by adding a set $U' = \{u'_1, \ldots, u'_z\}$, $z = 2 \cdot |U|$, to U such that $\mathcal{F}' = \mathcal{F} \cup \{U'\}$. All these elements get first stage, lower and upper cost 0, except $c^D(u'_1) = 2K - 1$. The upper and lower cost functions for all other elements remain as in I and the first-stage cost are set to 0. Finally, we set $k = |U|$ and the first-stage solution set to the set $\mathcal{F}'$. Note that the recovery $\mathcal{F}^k_F = \mathcal{F}$ for all $F \in \mathcal{F}$ and $\mathcal{F}^k_{U'} = \mathcal{F}'$. Hence, if I is a yes-instance, U' is the optimal first-stage solution with total cost of $c_T(U') = 2K - 1$. If I is a no-instance, any solution $F \in \mathcal{F}$ has total cost of at most $2K - 2$. Therefore, the k-Dist-RR LCMin is strongly **NP**-hard.

□

Since the shortest path problem and the minimum (s,t)-cut problem satisfy all conditions to apply Theorem 2.4.2, their k-Dist-RR versions are strongly **NP**-hard. Since all (s,t)-paths in the reduction graph for the hardness proof of the max-scenario shortest path problem have a length of at most 4 (Theorem A.1.1), the complexity result is even valid for constant $k \geq 4$.

Corollary 2.4.3. *The k-Dist-RR shortest path problem with Γ-scenarios and $k \geq 4$ and the k-Dist-RR minimum (s,t)-cut problem with Γ-scenarios are strongly* **NP***-hard.*

For other problems, like the minimum spanning tree problem or the weighted disjoint hitting set problem, the complexity status remains open. For the latter problem, we can just state that for $k \geq d$, an optimal first-stage solution is constructed by taking in each set an element of minimum first-stage cost. Furthermore, we can compute the total cost of such a solution via a constrained longest path problem in a chain graph, as described in Section A.3.

To complete the chapter about k-Dist-RR LCMin problems, we consider the shortest path problem.

2.5. Shortest Path Problem

The shortest path problem is to find a path between two designated vertices s and t in a digraph $G = (V, A)$ with minimum cost according to a cost function $c : A \to \mathbb{N}$. This problem is one of the most studied combinatorial optimization problems and can be solved efficiently

in its deterministic version with non-negative arc lengths. But in real-world applications like transportation, network design or telecommunication, some data might be subject to uncertainty.

We will consider in this section the k-Dist-RR shortest path problem and investigate its complexity status according to different first-stage solution sets and the interpretation of k as constant or not constant. Note that all results for k-Dist-RR LCMin problems are valid for the shortest path problem. Yet, we can strengthen them by taking into account its special structure. We thus obtain that if any simple (s,t)-path is a feasible first-stage path, the k-Dist-RR shortest path problem is not approximable even if we consider just one scenario, unless $\mathbf{P} = \mathbf{NP}$. This already settles the complexity for discrete scenarios, interval scenarios and Γ-scenarios.

A natural extension of simple (s,t)-paths as feasible first-stage solutions is to consider the set of all (s,t)-paths. If the parameter k is not constant, the complexity status compared to the set of simple (s,t)-paths does not change. Yet, for constant k we introduce a polynomial algorithm that solves the k-Dist-RR shortest path problem for interval scenarios.

2.5.1. Simple (s,t)-Paths as First-Stage Solutions

A natural restriction of the first-stage solution set $\mathcal{G}$ is to choose the set of all simple (s,t)-paths denoted by $\mathcal{P}$. Theorem 2.2.1 and Theorem 2.4.2 imply that the k-Dist-RR shortest path problem with $\mathcal{P}$ as first-stage solution set is weakly **NP**-hard for two scenarios and strongly **NP**-hard for Γ-scenarios. Yet, we will show that even for one scenario and thus for interval scenarios it is strongly **NP**-hard and not approximable, unless $\mathbf{P} = \mathbf{NP}$.

Theorem 2.5.1. *The k-Dist-RR shortest path problem with simple (s,t)-paths as first-stage solutions is* **NP***-hard for one scenario S, first-stage and scenario cost functions just assigning values 0 and 1 and constant $k \geq 2$ or k depending on the number of arcs.*

Proof. We show a reduction from the two vertex disjoint path problem to the k-Dist-RR shortest path problem with $\mathcal{P}$. Let I be an instance of the two vertex disjoint path problem given by a directed graph $G = (V, A)$ and two vertex pairs (v_1, u_1) and (v_2, u_2). The task in I is to decide whether two paths, a (v_1, u_1)-path p_1 and a (v_2, u_2)-path p_2, which are vertex disjoint, exist. We start with constant $k \geq 2$ and define an instance I' of the k-Dist-RR shortest path with $\mathcal{P}$ and one scenario in the following way: Let $G' = (V', A')$ be an extension of G by two vertices s and t, four arcs (s, v_1), (u_1, v_2), (v_2, u_2) and (u_2, t) and a path $p_{(v_1,u_1)}$ with length $k-1$ (see Figure 2.4). The first-stage cost function $c^D : A' \to \{0, 1\}$ adds cost 0 to all arcs except these on the path $p_{(v_1,u_1)}$ and (v_2, u_2), which get cost 1. The one scenario imposes cost 0 at the new added arcs, i.e., (s, v_1), the path $p_{(v_1,u_1)}$, (u_1, v_2), (v_2, u_2) and (u_2, t), and cost 1 to all other arcs. The size of the instance I' is polynomial in the size of the instance I.

We will now prove that there are two vertex disjoint paths p_1 and p_2 in G if and only if any optimal solution in I' has total cost 0.

Let p_1 and p_2 be two vertex disjoint paths in G. Then the path $p = (s, v_1) \cup p_1 \cup (u_1, v_2) \cup p_2 \cup (u_2, t)$ is simple and has first-stage cost 0. Furthermore, the path $\overline{p} = (s, v_1) \cup p_{(v_1,u_1)} \cup (u_1, v_2) \cup (v_2, u_2) \cup (u_2, t)$ is in the recovery set of p, since $\overline{p}$ contains only k arcs other then p (namely the ones in $p_{(v_1,u_1)}$ and (v_2, u_2)). Since the scenario cost of $\overline{p}$ is 0, path p and $\overline{p}$ form an optimal solution of I' with total cost 0.

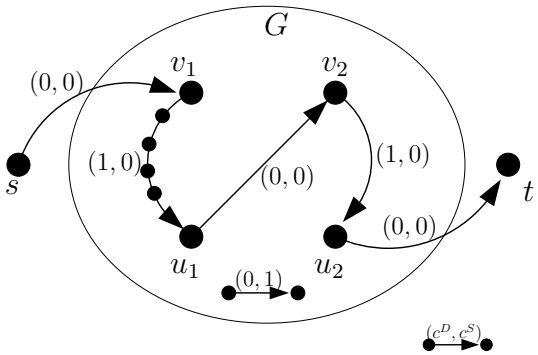


Figure 2.4.: The path $\overline{p} = (s, v_1) \cup p_{(v_1,u_1)} \cup (u_1, v_2) \cup (v_2, u_2) \cup (u_2, t)$ is the only path with scenario cost 0.

Let now p^* be an optimal first-stage path to I' with total cost 0 and let $\overline{p}^*$ be its recovery path. According to the scenario cost the only path with cost 0 is the path $\overline{p} = (s, v_1) \cup p_{(v_1,u_1)} \cup (u_1, v_2) \cup (v_2, u_2) \cup (u_2, t)$, hence $\overline{p}^* = \overline{p}$. Since any (s, t)-path crosses the arcs (s, v_1) and (u_2, t) and since $\overline{p}^*$ is in the recovery of p^*, the path p^* has to contain at least one of the arcs in $p_{(v_1,u_1)}$, the arc (u_1, v_2) or the arc (v_2, u_2). The first-stage cost of any arc in $p_{(v_1,u_1)}$ and (v_2, u_2) are set to 1. Since the first-stage cost of p^* is 0, p^* cannot contain any of these arcs and hence crosses arc (u_1, v_2). Therefore, p^* connects the vertices v_1 and u_1 and v_2 and u_2 via two paths just containing arcs of G. Since p^* is a simple path, these two paths are vertex disjoint and a feasible solution of I.

A similar construction works for the case $k = \frac{\ell}{\ell+1}|V(G')|$ for any $\ell \in \mathbb{N}$ and $k \geq 2$. In this case, the path $p_{(v_1,u_1)}$ is replaced by a path of length $\ell \cdot |V(G)| - 1$. The number of vertices in G' is

$$|V(G')| = (\ell + 1) \cdot |V(G)|$$

and at most

$$k = \ell \cdot |V(G)|$$

arcs can be recovered. Hence, any path with total cost 0 uses $\ell \cdot |V(G)| - 1$ arcs to recover the path $p_{(v_1,u_1)}$ and the last recovery action for the arc (v_2, u_2). □

Since the value of an optimal solution equals 0, any approximation algorithm with an approximation factor α computes an optimal solution. Also, no absolute performance guaranty for any approximation algorithm, i.e., $\mathrm{ALG}(I) \leq \mathrm{OPT}(I) + \beta$ for all instances I, can be given, since we can just scale the scenario cost and the first-stage cost with $(\beta + 1)$.

Corollary 2.5.2. *There is no efficient approximation algorithm for the k-Dist-RR shortest path problem with $\mathcal{P}$, unless* $\mathbf{P} = \mathbf{NP}$.

As shown in Theorem 2.3.1, the k-Dist-RR shortest path problem with $\mathcal{P}$ and $\mathcal{S}_I$ is equivalent to the problem with one discrete scenario. Hence, also in this setting there exists no approximation algorithm. The same arguments hold for Γ-scenarios.

Corollary 2.5.3. *The k-Dist-RR shortest path problem with $\mathcal{P}$ and $\mathcal{S}_I$ (or $\mathcal{S}_\Gamma$) is not approximable, unless* $\mathbf{P} = \mathbf{NP}$.

Note that in this case the complexity status of the recoverable robust version of an LCMin problem is much harder than the corresponding robust version. The robust version can always be solved efficiently for one scenario if the LCMin problem is in $\mathbf{P}$.

So far, we considered the case, in which the first-stage cost function and the scenario cost functions are chosen independently. But the proof of Theorem 2.5.1 can be extended to first-stage cost and scenario cost satisfying the worsening-condition. Instead of adding cost $(c^D, c^S) = (1, 0)$ to the arcs of the path $p_{(v_1,u_1)}$ and (v_2, u_2), we add cost $(1, 1)$ to the arc (v_2, u_2) and the first arc on $p_{(v_1,u_1)}$ and cost $(0, 0)$ to all other arcs on $p_{(v_1,u_1)}$. Furthermore, any arc in the original graph G gets cost $(0, 3)$ (see Figure 2.5, left). An optimal solution with total cost 2 exists if and only if there are two disjoint paths p_1 and p_2. Any other solution has cost of at least 3.

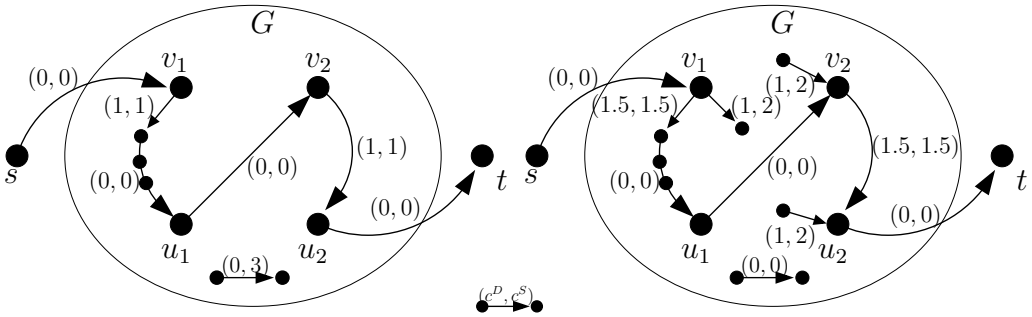


Figure 2.5.: The cost functions in the left figure obey the worsening-condition and the cost functions in the right figure satisfy the $\alpha_{[0,1]}$-deviation-condition.

Another variation of the cost structure obeying the $\alpha_{[0,1]}$-deviation can be achieved by adding cost $(1.5, 1.5)$ to the arc (v_2, u_2) and the first arc of the path (v_1, u_1), cost $(1, 2)$ to all arcs (v_1, v), (v, v_2), and (v, u_2), $v \in V$ (besides the one for the already mentioned arc cost) (see Figure 2.5, right). Hence, the cheapest recovery path has cost 3, and the cheapest first-stage path with this recovery, has first-stage cost 2. A solution with total cost 5 can only be achieved if there are two disjoint paths connecting (v_1, u_1) and (v_2, u_2). Since any other solution will have cost of at least 5.5, we get a lower bound of 1.1 on the best possible approximation factor. Thus, there remains a gap of 0.4 between the approximation algorithm introduced in Theorem 2.2.5 and this bound.

Corollary 2.5.4. *The k-Dist-RR shortest path problem with $\mathcal{P}$ cannot be approximated with a factor better than* 1.5*, if the worsening-condition is satisfied, and a factor of* 1.1*, if the $\alpha_{[0,1]}$-deviation-condition is satisfied, unless* $\mathbf{P} = \mathbf{NP}$.

Note that all results obtained from the reduction in the proof to Theorem 2.5.1 cannot be transferred to undirected graphs. The two vertex disjoint path problem is in that case solvable in polynomial time [90].

Extending the first-stage solution set from simple (s, t)-paths $\mathcal{P}$ to the set of all (s, t)-paths $\overline{\mathcal{P}}$ changes the complexity of the problem. For the interval case the k-Dist-RR shortest path with $\overline{\mathcal{P}}$ and constant k is even solvable in polynomial time, as we will show in Theorem 2.5.9.

2.5.2. (s, t)-Paths as First-Stage Solutions

The complexity status of the k-Dist-RR shortest path problem with $\mathcal{P}$ depends mainly on the fact that the first-stage solution has to be a *simple* (s, t)-path. To relax this condition we consider the k-Dist-RR shortest path problem with the set of all (s, t)-paths denoted by $\overline{\mathcal{P}}$ as first-stage solution set.

Discrete Scenarios In [70] Kasperski and Zieliński show that the robust shortest path problem with discrete scenarios $\mathcal{S}_D$ cannot be approximated with a factor better then $\log|\mathcal{S}_D|$. Since we are interested in transferring this complexity and approximability result to the k-Dist-RR shortest path problem with $\overline{\mathcal{P}}$, we introduce an L-reduction. Recall that to establish an L-reduction (e.g. [77]) from an optimization problem $\mathcal{X}$ to an optimization problem $\mathcal{X}'$ we have to define a pair of functions f and g, both computable in polynomial time, and two constants $\alpha, \beta > 0$ such that for any instance x of $\mathcal{X}$:

- $f(x)$ is an instance of $\mathcal{X}'$ with $\mathrm{OPT}(f(x)) \leq \alpha\,\mathrm{OPT}(x)$;
- for any feasible solution y' of $f(x)$, $g(x,y')$ is a feasible solution of x such that $|c_x(g(x,y')) - \mathrm{OPT}(x)| \leq \beta|c_{f(x)}(y') - \mathrm{OPT}(f(x))|$,

where c_x is the cost function of the instance x. We will now introduce an L-reduction from the robust shortest path problem to the k-Dist-RR shortest path with $\overline{\mathcal{P}}$ and constant k.

Theorem 2.5.5. *For the k-Dist-RR shortest path problem with $\overline{\mathcal{P}}$, a discrete scenario set $\mathcal{S}_D$ and constant k there exists no approximation algorithm with an approximation factor better than $\log|\mathcal{S}_D|$, unless $\mathbf{P} = \mathbf{NP}$.*

Proof. We prove Theorem 2.5.5 by an L-reduction from the robust shortest path problem. Let $G = (V, A)$ be a directed graph, $s, t \in V$ be two designated vertices and $\mathcal{S}_D$ be a scenario set, where each scenario $S \in \mathcal{S}_D$ defines a cost function. The robust shortest path problem is to find an (s,t)-path p that minimizes the maximum scenario cost, i.e., that minimize the cost function $c(p) = \max_{S\in\mathcal{S}_D} c^S(p)$. In order to define a k-Dist-RR shortest path instance with $\overline{\mathcal{P}}$ and some constant $k \in \mathbb{N}$, we add a vertex s' to V and connect s' with s via an (s', s)-path of length k. In this path, we replace every arc by two parallel arcs, a lower and an upper arc. This forms the new graph $G' = (V', A')$. The upper arcs obtain first-stage cost $C = \max_{S\in\mathcal{S}_D} c(A)$ and all other arcs first-stage cost 0. Finally, we extend for every scenario $S \in \mathcal{S}_D$ the scenario cost function $c^S : A \to \mathbb{N}$ to the set of A' by defining $c^S(a) = C$ for all lower arcs $a \in A'\backslash A$ and $c^S(a) = 0$ for all upper arcs $a \in A'\backslash A$. In this k-Dist-RR shortest path instance any first-stage path p with total cost smaller than C chooses the lower arcs to reach s and exchanges in ever scenario these arcs by the upper arcs. Such a path p can be converted into a robust path with the same robust cost as the original total cost by just considering the part $p_{[s,t]}$. □

Note that this bound on the approximation factor is stronger than the one proposed in Corollary 2.2.3, if neither the worsening nor the α-deviation-condition needs to be fulfilled. In Theorem 2.5.5 we just considered the k-Dist-RR shortest path problem with $\overline{\mathcal{P}}$ for constant k, since otherwise, the problem becomes inapproximable for one scenario, as we prove in the next theorem.

Theorem 2.5.6. *The k-Dist-RR shortest path problem with $\overline{\mathcal{P}}$ on a directed graph $G = (V, A)$ with one scenario cannot be approximated, unless $\mathbf{P} = \mathbf{NP}$.*

Proof. We show a reduction from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Each clause is formed by three literals and w.l.o.g. $n = m$. We start by constructing the graph G' of the corresponding k-Dist-RR shortest path instance with $\overline{\mathcal{P}}$.

This graph G' is composed of three parts: the variable part, the clause part and the connecting part. In the variable part we introduce, for each variable x_i, $i = 1, \ldots, n$, two vertices s_i and t_i, which are connected by two paths of length 2 (see Figure 2.6). We call the vertex in the

upper path the x_i *docking vertex* and the vertex in the lower path the $\overline{x}_i$ *docking vertex*. Furthermore, the vertices t_i and s_{i+1}, $i = 1, \ldots, n-1$, and t_n with t are connected via an arc, while s is connected with s_1 via a path of length 5. All arcs in the variable part get cost $(c^D, c^S) = (0, 1)$, where S is the only scenario considered in this setting.

The clause part of G' contains for every clause C_j, $j = 1, \ldots, n$, two vertices v_j and u_j, which are connected via three disjoint paths of length 3. We call the middle arc $a_{j\ell}$ in the ℓth path the (j, ℓ)-*literal arc*, $j = 1, \ldots, n$, $\ell = 1, 2, 3$. The vertices u_j and v_{j+1}, $j = 1, \ldots, n-1$, s and v_1 and u_n and t are again connected via an arc. The literal arcs have a cost structure of $(c^D, c^S) = (0, 0)$, while all other arcs in the clause part of G' have cost $(c^D, c^S) = (1, 0)$.

The last part, the connecting part of G', connects the clause part with the variable part. In general, we define the cycle $C[a, b]$ with $a = (u, v)$ being an arc and b being a vertex as the cycle $(b, u) \cup a \cup (v, b)$ of length 3. The connecting part of G' consists of all cycles $C[a_{j\ell}, z_{j\ell}]$ with $a_{j\ell}$ being the (j, ℓ)-literal arc and $z_{j\ell}$ the $y_{j\ell}$ docking vertex, $j = 1, \ldots, n$, $\ell \in \{1, 2, 3\}$. The arcs in the connecting part, except the literal arcs, get cost $(c^D, c^S) = (0, 1)$. Hence, the graph G' consists of $12 \cdot n + 6$ vertices and $21 \cdot n + 6$ arcs. For $k = \frac{1}{4} \cdot |V(G')|$, we are allowed to change the first-stage path p_1 by at most $3n + 1$ arcs.

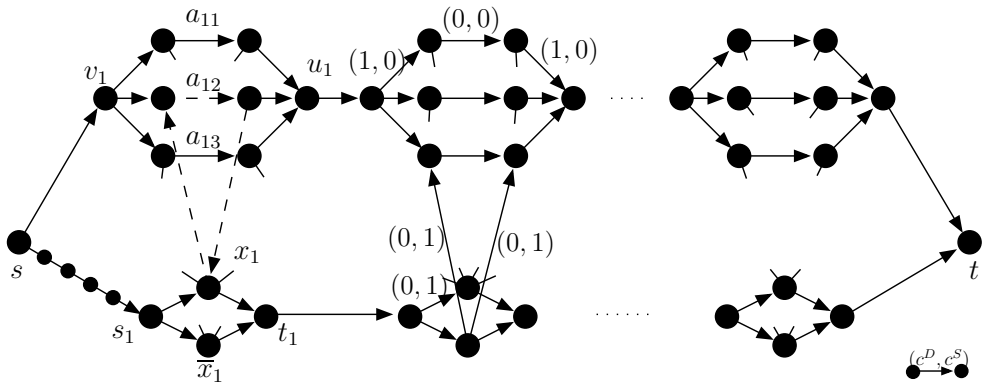


Figure 2.6.: The lower part of the graph represents the variables of I, the upper part its clauses. The dashed arcs form the cycle $C[a_{12}, z_{12}]$ for the clause $C_1 = x_2 \vee x_1 \vee x_3$, i.e., $z_{12} = x_1$, and the literal arc a_{12}.

An optimal solution to I' with total cost 0, i.e., a first-stage path p_1 and a recovery path p_2 with $|p_2 \backslash p_1| \leq 3n + 1$ and $c^D(p_1) + c^S(p_2) = 0$, exists if and only if I is a yes-instance.

Let (p_1, p_2) be an optimal solution with total cost 0. Since p_1 has first-stage cost 0, it only contains arcs of the variable part and of the $C[a_{j\ell}, z_{j\ell}]$ cycles. Therefore, for every variable x_i, $i = 1, \ldots, n$, it crosses either the x_i docking vertex or the $\overline{x}_i$ docking vertex. We define an assignment x according to p_1 by

$$x_i = \begin{cases} \text{true} & \text{if } p_1 \text{ crosses the } x_i \text{ docking vertex} \\ \text{false} & \text{if } p_1 \text{ crosses the } \overline{x}_i \text{ docking vertex} \end{cases}$$

for $i = 1, \ldots, n$. The path p_2 only crosses arcs in the clause part of G' (otherwise it induces recovery cost greater than 0). Since p_2 is a simple path, it has a length of $4n + 1$ and contains exactly n literal arcs. The only arcs in the clause part of G' with first-stage cost 0 are the literal arcs. Therefore, p_1 has to cross, for every clause C_j, $j = 1, \ldots, n$, one literal arc $a_{j\ell}$, $\ell \in \{1, 2, 3\}$. If the path p_1 contains a literal arc $a_{j\ell}$, then it also crosses the $y_{j\ell}$ docking vertex. Hence, x verifies every clause C_j if the path pair p_1, p_2 has total cost 0.

Let now be x a feasible assignment of I. We define the first-stage path p_1 in the following way: for $i = 1, \ldots, n$, the path p_1 contains the x_i docking vertex and all cycles $C[a_{j\ell}, z_{j\ell}]$ with $y_{j\ell} = x_i$ if $x_i = \text{true}$, $i \in \{1, \ldots, n\}$; and the path p_1 contains the $\overline{x}_i$ docking vertex and all cycles $C[a_{j\ell}, z_{j\ell}]$ with $y_{j\ell} = \overline{x}_i$ if $x_i = \text{false}$, $i \in \{1, \ldots, n\}$. This path is well defined, has first-stage cost 0 and crosses at least one literal arc $a_{j\ell}$ for each clause C_j, $j = 1, \ldots, n$, $\ell \in \{1, 2, 3\}$. In addition, any simple (s, t)-path p_2 in the clause part containing these literal arcs is a feasible recovery path of p_1 with recovery cost 0. Therefore, a feasible first-stage path p_1 with a recovery path p_2 and total cost 0 exists in I' if I is a yes-instance. □

The first-stage cost and the scenario cost can be scaled with a factor polynomial in the input to satisfy the worsening or the $\alpha_{[0,1]}$-deviation-condition. Note that the proof depends on the recovery action, i.e., the possibility to exchange k arcs. It remains open if the k-Dist-RR shortest path with $\overline{\mathcal{P}}$, constant k and more than one scenario is solvable in pseudo-polynomial time. For k constant and $|\mathcal{S}_D| = 1$, we introduce a polynomial algorithm in the next section.

Interval Scenarios Since the interval scenario case is equivalent to the case of just one discrete scenario (Theorem 2.3.1), the k-Dist-RR shortest path with $\overline{\mathcal{P}}$ and interval scenarios is inapproximable, unless $\mathbf{P} = \mathbf{NP}$.

Corollary 2.5.7. *The k-Dist-RR shortest path with $\overline{\mathcal{P}}$, interval scenarios and k being part of the input is strongly* **NP***-hard and inapproximable, unless* $\mathbf{P} = \mathbf{NP}$.

If k is constant, we will show that this problem can be solved in polynomial time. The main idea of the algorithm is to find, for a given recovery action, i.e., the set of arcs just taken by the recovery path, the best first-stage path. A given recovery action consists of at most k arcs and a best first-stage path can be calculated by at most $2k$ shortest path computations on the end-vertices of the detours and a set of at most $2k$ additional vertices.

Before we start with a detailed description of a polynomial algorithm, we need some general observations about the structure of feasible solutions. Let $(p, p') \in \overline{\mathcal{P}} \times \mathcal{P}$ be a first-stage path p and its recovery path p' of a k-Dist-RR shortest path instance with $\overline{\mathcal{P}}$ and constant k. W.l.o.g., we assume that p does not cross any arc twice. If this is not the case, we replace each arc used more than once by parallel arcs with the same cost structure and modify p in the canonical way to use each arc once.

According to their appearance in p and p', we divide all arcs in three sets: We call the connected components in $p' \backslash p$ the *recovery components* of (p, p'), the connected components in $p \backslash p'$ the *first-stage components* of (p, p') and the connected components in $p \cap p'$ the *shared components* (see Figure 2.7). Furthermore, we denote the end-vertices of the recovery components as *recovery break vertices* V_{rec}. Next to the recovery break vertices, we define the *inner break vertices* V_{inner} in the following way: Let $a = (u, v)$ and $a' = (v, w)$ be two consecutive arcs on p. If $a \in p \cap p'$ and $a' \in p \backslash p'$ or $a \in p \backslash p'$ and $a' \in p \cap p'$ and $v \notin V_{\text{rec}}$, than v is an *inner break vertex*. Obviously the number of recovery break vertices is bounded by $2k$, since at most k arcs may be exchanged. Furthermore, the next theorem states that also the number of inner break vertices is limited by $2k$ for at least one optimal solution.

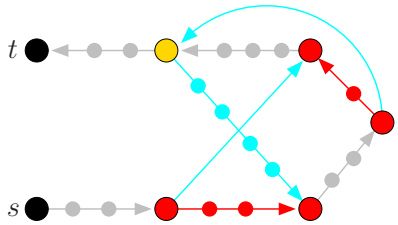


Figure 2.7.: The gray parts are the shared, the blue the first-stage and the red the recovery components. The yellow vertex is an inner break vertex and the big red ones the recovery break vertices.

Theorem 2.5.8. *For any k-Dist-RR shortest path instance with $\overline{\mathcal{P}}$, there exists an optimal solution with at most $2k-2$ inner break vertices V_{inner}.*

Proof. Let (p, p') be an optimal solution with a minimal number of inner break vertices. To simplify the remaining proof, we assume that s and t are part of a shared component. If this is not the case we add a super source connected with s and a super sink connected with t and enlarge p and p' by these two arcs. In the worst-case, we will add two inner break vertices.

We will now define a graph G_p as a simple representation of p, in which we can later bound the number of inner break vertices: Due to the definition of first-stage and shared components, the path p runs alternating through these different components. Instead of considering these subpaths, we substitute them by simple arcs and obtain the graph G_p. Furthermore, we color all arcs representing parts of a shared component gray and all arcs representing first-stage components blue (see Figure 2.8). More formally, the path p is given by the sequence of vertices $sv_1v_2 \dots v_kt$. We now define a subsequence $v_{i_0}, v_{i_1}, \dots, v_{i_\ell}$ of vertices in p obeying the following conditions: $v_{i_0} = s$, $v_{i_\ell} = t$; for odd $j \geq 1$, the vertex v_{i_j} satisfies $p_{[v_{i_{(j-1)}}, v_{i_j}]} \subseteq p \cap p'$ and $p_{[v_{i_{(j-1)}}, v_{i_j+1}]} \subsetneq p \cap p'$; and for even $j \geq 2$ the vertex v_{i_j} satisfies $p_{[v_{i_{(j-1)}}, v_{i_j}]} \subseteq p \backslash p'$ and $p_{[v_{i_{(j-1)}}, v_{i_j+1}]} \subsetneq p \backslash p'$. The graph G_p contains all vertices $\{v_{i_0}, \dots, v_{i_\ell}\}$ and all arcs $(v_{i_j}, v_{i_{(j+1)}})$ for $j = 0, 1, \dots, \ell - 1$. All arcs $(v_{i_j}, v_{i_{(j+1)}})$ with j even are colored gray and all other arcs are colored blue.

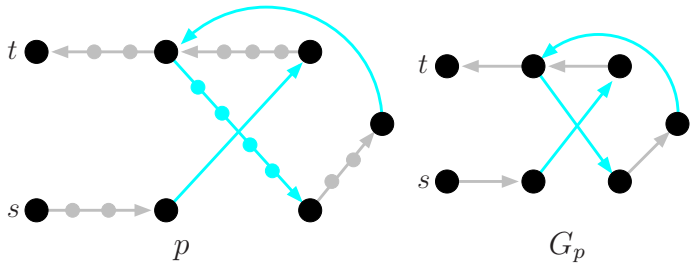


Figure 2.8.: The graph G_p shrinks the graph p maintaining the different components, the inner break vertices and the recovery break vertices.

The graph G_p has the following properties:

1. G_p contains an Eulerian path.
2. Any Eulerian path $\overline{p}$ in G_p represents an optimal first-stage path of the original instance. This path is obtained by replacing each arc (y, z) in $\overline{p}$ by the subpath $p_{[y,z]}$.
3. Every vertex in G_p besides s and t has an even degree of at most 4: Every shared component is a simple path and hence induces at most two gray arcs in G_p at one

vertex. Since p alternating by passes gray and blue arcs, there are as many blue arcs as gray arcs incident to one vertex.

4. By definition, $V_{\text{rec}} = \{v \in V(G_p) \mid \deg(v) = 2\}$ and $V_{\text{inner}} = \{v \in V(G_p) \mid \deg(v) = 4\}$, where $\deg(v)$ denotes the degree of a vertex v.

5. G_p does not contain any loop since p minimizes the number of inner break vertices.

We will now show that the number of vertices with degree 4, denoted by V_4, is bounded by the number of vertices with degree 2, denoted by V_2. We start by defining a procedure to reduce G_p to a simple (s,t)-path. In each iteration, the remaining graph contains at least two vertices of degree 2 less than the previous one and at most two vertices of degree 4 are deleted. Since the remaining path contains no vertices of degree 4, we obtain the desired estimation of inner break vertices.

Repeat the following procedure starting with $G_p^i = G_p$ and $i = 0$ until no cycle in G_p^i exists: Find an Eulerian path $\tilde{p} = su_1u_2 \dots u_kt$ in G_p^i and let w_i be the first vertex that is visited by $\tilde{p}$ for the second time. Set $K_i = \tilde{p}_{[w_i,w_i]}$ as this simple cycle starting and ending in w_i and define $\overline{G} = G_p^i \backslash K_i$. The two arcs in K_i incident to w_i have the same color as have the two arcs $e_1 = (a, w_i)$ and $e_2 = (w_i, b)$ incident to w_i in $\overline{G}$. (Otherwise, we obtain a contradiction to p minimizing the number of inner break vertices.) Now, we replace these two arcs and w_i by just one arc (a, b) of the same color as e_1 and e_2. If this new arc (a, b) is a loop, it has to be blue, since all gray components from an acyclic graph. We delete this loop from $\overline{G}$ and replace the vertex a and the two gray arcs incident to a by one gray arc. Finally, we set $G_p^{i+1} = \overline{G}$ and $i = i + 1$.

Let $i_{\max}$ be the index with which the procedure stops. In each iteration the cycle K_i has at least a length of 3 and, as we will later show, contains besides w_i just vertices of degree 2 in G_p. Thus,

$$V_2(G_p^{i+1}) = V_2(G_p^i) - V_2(K_i) \leq V_2(G_p^i) - 2$$

for $i = 0, \dots, i_{\max} - 1$. Furthermore, at most two vertices of degree 4 are deleted in each iteration. Hence,

$$V_4(G_p^{i+1}) \geq V_4(G_p^i) - 2$$

for $i = 0, \dots, i_{\max} - 1$. Since $|V_2(G_p^0)| \leq 2k$, $|V_2(G_p^{i_{\max}})| \geq 2$ and $|V_4(G_p^{i_{\max}})| = 0$, $|V_{\text{inner}}| = |V_4(G_p^0)| \leq 2k - 2$.

It remains to show that $K_i \backslash w_i$ contains just vertices of degree 2. Let us assume that $K_0 \backslash w_0$ contains one further vertex v^* with degree 4. We will show that we can construct a new first-stage path $\overline{p}$ with the same recovery path p' that has one inner break vertex less. Let $\tilde{p} = u_0u_1u_2 \dots u_ku_{k+1}$ be an Eulerian path in $\overline{G} = G_p^0 \backslash K_0$ with $u_0 = s$, and $u_{k+1} = t$. Since K_0 is a simple cycle, there exists an index r with $u_r = v^*$. Add K_0 to $\tilde{p}$ at the vertex u_r and replace all arcs by the corresponding subpaths of p to obtain a feasible first-stage path in G. The vertex w_0 is neither an inner break vertex of this new path $\overline{p}$ nor of the recovery path p'. Hence, this is a contradiction to the choice of p and p'. In any other iteration the same argument works by reverting the replacement of arcs and the deletion of cycles in the previous iterations. This concludes the proof of Theorem 2.5.8. □

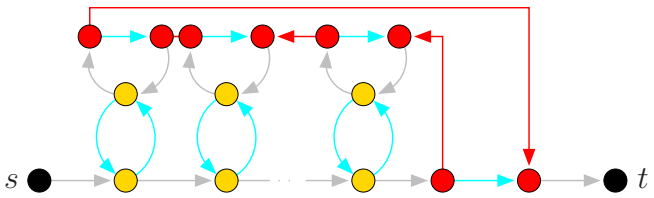


Figure 2.9.: The graph contains $2k$ recovery break vertices, colored red, and $2(k-1)$ inner break vertices, colored yellow.

The k-Dist-RR shortest path instance in Figure 2.9 shows a case in which any optimal solution has $2k-2$ inner break vertices. In this graph, the blue arcs have first-stage cost $c^D = 0$ and scenario cost $c^S = 1$, the red arcs cost $(c^D, c^S) = (1, 0)$ and the gray arcs cost $(c^D, c^S) = (0, 0)$. The blue arcs represent the first-stage components, the gray arcs the shared components and the red arcs the recovery components of an optimal solution. All recovery break vertices are marked red and the inner break vertices yellow.

Algorithm 2.3 k-Dist-RR shortest path with $\overline{\mathcal{P}}$, constant k and $\mathcal{S}_I$

Input : directed graph $G = (V, A)$, $s, t \in V$, first-stage cost $c^D : A \to \mathbb{N}$, scenario cost $c^S : A \to \mathbb{N}$, $k \in \mathbb{N}$
Output : two (s,t)-paths $\overline{p}$ and $\overline{p}'$
`/* Initialization */`
Add the vertices s' and t' and the arcs $a_1 = (s', s)$ and $a_2 = (t, t')$ to G
Set $c^D(a_i) = c^S(a_i) = 0$, $i = 1, 2$ and cost $= \infty$
`/* Computation of optimal first-stage and recovery path */`
`/* Selection of recovery */`
forall *possible recoveries* A' **do**
  `/* Selection of inner break vertices */`
  forall $V' \subseteq V$ *with* $|V'| \leq 2k$ **do**
    Set $V^+ = V_{A'}^{\text{end}} \cup V'$ and $V^- = V_{A'}^{\text{start}} \cup V'$
    `/* Fixing order of the break vertices */`
    forall $\sigma^+ : \{1, \ldots, |V^+|\} \to V^+$ *and* $\sigma^- : \{1, \ldots, |V^-|\} \to V^-$ **do**
      Set $\sigma^+(0) = s'$ and $\sigma^-(|V^-| + 1) = t'$
      `/* Computation of shared components */`
      for $i = 1, \ldots, |V^-| + 1$ **do**
        Find a shortest $(\sigma^+(i-1), \sigma^-(i))$-path $p_{[\sigma^+(i-1), \sigma^-(i)]}$ w.r.t. $c^D + c^S$
      `/* Computation of first-stage components */`
      for $i = 1, \ldots, |V^+|$ **do**
        Compute a shortest $(\sigma^-(i), \sigma^+(i))$-path $p_{[\sigma^-(i), \sigma^+(i)]}$ w.r.t. c^D
      `/* Definition of first-stage path and recovery path */`
      Set $p = p_{[\sigma^+(0), \sigma^-(1)]} \cup_{i=1}^{|V^+|} \{p_{[\sigma^-(i), \sigma^+(i)]} \cup p_{[\sigma^+(i), \sigma^-(i+1)]}\}$
      Compute in the graph $\cup_{i=1}^{|V^-|+1} p_{[\sigma^+(i-1), \sigma^-(i)]} \cup A'$ a shortest (s', t')-path p' w.r.t. $\overline{c}$
      if $c^D(p) + c^S(p') <$ cost **then**
        Set $\overline{p} = p$, $\overline{p}' = p'$ and cost $= c^D(p) + c^S(p')$
Delete (s', s) and (t, t') from $\overline{p}$ and $\overline{p}'$
return $\overline{p}, \overline{p}'$

Besides the bound on the inner break vertices, Algorithm 2.3 is based on the following observation: Let V_{rec} be a set of recovery break points, V_{inner} be a set of inner break points and $v_0, v_1, \ldots, v_\ell, v_{\ell+1}$, with $v_0 = s$ and $v_{\ell+1} = t$, be an order in which an optimal first-stage path passes these break vertices. As in the proof of Theorem 2.5.8, we assume that s and t are contained in a shared component. Then, for $i = 0, \ldots, \ell$, the subpath $p_{[v_i, v_{i+1}]}$ is a shortest path according to the cost $c^D + c^S$, if i is even, and according to c^D, if i is odd, where c^D denotes the first-stage cost and c^S the scenario cost. If an optimal first-stage path violates this property, the corresponding part could be replaced by a shortest path, a contradiction.

For the description of the algorithm, we need one more definition. An arc set $A' \subseteq A$ with $|A'| \leq k$ is called a *possible recovery* if all connected components in A' are directed paths in (V, A'). The source vertices of the connected components in A' are denoted by $V_{A'}^{\text{start}}$, the target vertices by $V_{A'}^{\text{end}}$. Note that, for any vertex $v \in V_{A'}^{\text{start}}$, there is an incoming arc of a shared component and, for every vertex $v \in V_{A'}^{\text{end}}$, there is an outgoing arc of a shared component.

Theorem 2.5.9. *Algorithm 2.3 constructs an optimal first-stage path to a given k-Dist-RR shortest path instance with $\overline{\mathcal{P}}$, constant k and $\mathcal{S}_I$ in $\mathcal{O}(2 \cdot m^k \cdot n^{2k} \cdot 2k! \cdot 2k(SP))$, where SP denotes the time to compute a shortest path.*

A canonical question is if this approach can be adapted to more scenarios, e.g., $|\mathcal{S}_D| = 2$. Yet, as Figure 2.10 shows, the number of break vertices, i.e., the separation of the first-stage path and one of the recovery paths, may not be bounded by k. We therefore think that different techniques need to be developed for that case to derive a pseudo-polynomial algorithm.

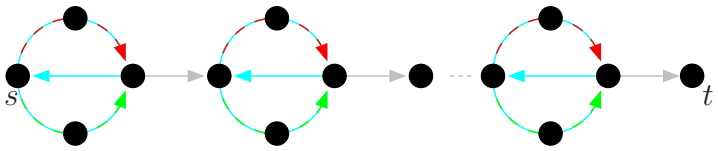


Figure 2.10.: The path p is the first-stage path and contains all arcs. In the case $k = 0$, the path p_1 is the recovery path in S_1 and crosses all lower arcs, while the path p_2 as recovery path in S_2 passes all upper arcs. Arcs contained in p, p_1, and p_2 are colored blue, red, and green, respectively. Arcs used by p, p_1 and p_2 are gray. All vertices with degree 4 or 3 represent some kind of break vertex and are not bounded by any constant.

2.6. Conclusion and Open Issues

We considered several different k-Dist-RR LCMin problems and showed that their complexity status depends on

1. the considered underlying LCMin problem;
2. whether k is constant or not;
3. the first stage decision set.

In this chapter, we investigated the weighted disjoint hitting set (WDHS) problem, the minimum weight basis (MWB) problem for matroids, e.g., the minimum spanning tree problem, the minimum (s,t)-cut problem, the minimum perfect matching problem and the shortest path problem. We showed that for discrete scenario sets the k-Dist-recoverable robust versions of these linear combinatorial optimization problems are at least weakly **NP**-hard, if the

number of scenarios is constant (Section 2.2). For the weighted disjoint hitting set problem we introduced a pseudo-polynomial time algorithm whose run-time depends on the values of the scenario cost functions and is exponential in the number of scenarios (Subsection 2.2.1).

If the number of discrete scenarios is not constant, we have seen that the problems mentioned above become strongly **NP**-hard. We also showed that for arbitrary cost functions they are not approximable with a factor better than 2. However, for the case that the cost functions, i.e., the first-stage cost function and the scenario cost functions, obey the $\alpha_{[0,1]}$-deviation-condition, we introduced a 1.5-approximation algorithm. For the shortest path problem with simple (s,t)-paths, we gave a lower bound of 1.1 on the best possible approximation factor, leaving a gap of 0.4 (Subsection 2.5).

The following Table 2.1 summarizes our results for interval scenarios and Γ-scenarios from Section 2.3, 2.4 and 2.5. The lower bound on the best possible approximation factor is given for arbitrary first-stage cost functions and scenario cost functions. Scaling these values shows the **NP**-hardness of these problems if the cost functions, i.e., the first-stage cost function and the scenario cost functions, are to obey the $\alpha_{[0,1]}$-deviation-condition. Yet, for these cases the lower bound on the approximation factor changes. In the case of Γ-scenarios, the complexity status of several k-Dist-RR LCMin problems like the WDHS, the MWB or the perfect matching problbaseem remains open.

Problems	$\mathcal{G}$	$\mathcal{S}_I$		$\mathcal{S}_\Gamma$	
		k constant	k part of input	k constant	k part of input
WDHS	proper	■	■	?	?
MWB	$\mathcal{B}$	■	?	?	?
(s,t)-cut	$\mathcal{D}$	?	?	■	■
shortest path	$\mathcal{P}$	n	n	n	n
shortest path	$\overline{\mathcal{P}}$	■	n	■	n

■ : in **P**, ■ : strongly **NP**-hard, n : not approximable

Table 2.1.: The first-stage solution set $\mathcal{G}$ is defined according to the underlying LCMin problem by $\mathcal{B}$ as the set of all bases, $\mathcal{P}$ as the set of all simple (s,t)-paths, $\overline{\mathcal{P}}$ as the set of all (s,t)-paths and $\mathcal{D}$ as the set of all (s,t)-cuts.

One direction of further research is to investigate the open problems marked by a question mark. Another is to consider special cases of the LCMin problems like the shortest path problem on series-parallel graphs or grid graphs. Furthermore, taking different neighborhoods from local search as recovery or allowing a limited number of local search steps define new interesting recoverable robust models. Besides analyzing their complexity status and combinatorial structures, research could focus on developing exact methods to solve these problems.

3. Rent Recoverable Robustness

Rent recoverable robustness deals with uncertainty in the objective function similar to the concept of option dealing: in a first stage we can rent several elements, which gives us the privilege to buy these elements for a price that will be presented in the future. If we buy an element not rented before, we generate additional inflation cost which need to be paid next to the normal price. However, renting elements costs. In our setting these cost are assumed to depend on the realized cost. As before, we model all possible realizations of the cost via sets of scenarios.

We investigate the complexity status of the rent recoverable robust version of several combinatorial optimization problems, e.g., the shortest path problem and the minimum spanning tree problem. For discrete scenario sets we show that these problems are weakly **NP**-hard for two scenarios and strongly **NP**-hard if the number of scenarios is not constant. In the case of interval scenarios the rent recoverable robust problem can be solved efficiently. However, considering Γ-scenarios the rent recoverable robust shortest path problem and the rent recoverable robust minimum (s,t)-cut problem are strongly **NP**-hard. Finally, we introduce an approximation algorithm for Γ-scenarios whose approximation factor just depends on the given so-called rental factor and inflation factor.

3.1. Introduction

Motivation A common approach for pricing elements in a two-stage setting is the following. If an element is bought in the first stage, we pay an amount of c; if we buy it in the second stage, the cost are multiplied with some inflation factor $\beta \geq 0$ and hence the element costs $(1+\beta)c$ (see Section 1.2). This approach is based on the assumption that elements bought "today" are cheaper than they are "tomorrow" [37, 55]. In our model, we extend this pricing system slightly to the case where we do not actually buy the element in the first stage, but rather buy an option it for some amount with the option to purchase it later. We gain with this action the right to buy this element but are not obligated to do so. If we actually buy it, we pay the full second-stage price. On the other hand, if we buy the element in the second stage without renting it in the first stage, we have to add some inflation cost to the price.

To avoid the drawback of a bi-criteria objective as in the k-distance recoverable robust model, we assume that the price for rental is not known in the first stage but depends on the revealed cost in the second stage. All possible realizations of cost values for the elements are captured again in a set of scenarios $\mathcal{S}$, i.e., every scenario $S \in \mathcal{S}$ defines a cost function c^S on the given set of elements. More formally, the pricing system is constructed as follows. Let $c^S(e)$ be the cost for some element e in scenario $S \in \mathcal{S}$ and $1 > \alpha > 0$ be some rental factor. If we rented this element in the first stage, we need to pay $\alpha c^S(e)$ in the second stage whether or not we actually buy it. If we actually buy the element in the second stage, we just need to pay the remaining cost, i.e., additional $(1-\alpha)c^S(e)$. On the other hand, buying this element if it is not rented before, costs us $(1+\beta)c^S(e)$ for some inflation factor $\beta \geq 0$. Thus, given a rental

factor α and an inflation factor β the *rent recoverable robust LCMin problem* is to find a set of elements that is rented in the first stage such that the maximum cost over all scenarios for purchasing a solution in the second stage is minimized.

Model and Notation As for k-distance recoverable robustness we define rent recoverable robustness for linear combinatorial minimization (LCMin) problems. Such an LCMin instance $(U, \mathcal{F}, c)$ is given by a finite set of elements U, a set of feasible solutions $\mathcal{F} \subseteq 2^U$ and a cost function $c : U \to \mathbb{N}$. Then the task is to find a feasible solution $F \in \mathcal{F}$ with minimum cost $c(F) = \sum_{u \in F} c(u)$ (Definition 1.1.1). Since recoverable robustness implicates a two-stage process, we assume that a first-stage solution set is given as in Definition 2.1.1. For a given LCMin instance $(U, \mathcal{F}, c)$ a set $\mathcal{G} \subseteq 2^U$ is called a *first-stage solution set* of U, if we can decide for any subset $E \subseteq U$ in polynomial time in U whether or not E is in $\mathcal{G}$, i.e., whether E is a feasible first-stage solution. In most cases we assume that the first-stage solution set $\mathcal{G}$ is the set of feasible solutions $\mathcal{F}$ or an extension of $\mathcal{F}$ called a *proper* first-stage solution set. Proper sets $\mathcal{G}$ contain the set $\mathcal{F}$ and any other feasible first-stage solution $G \in \mathcal{G}$ is a superset of at least one feasible solution $F \in \mathcal{F}$ (Definition 2.1.2). We will now formally define the rent recoverable robust model motivated above.

Definition 3.1.1 (Rent Recoverable Robust Linear Combinatorial Minimization Problem (Rent-RR LCMin problem))**.** Let $(U, \mathcal{F}, c)$ be an LCMin problem; $\mathcal{S}$ be a set of scenarios, where each scenario $S \in \mathcal{S}$ determines a scenario cost function $c^S : U \to \mathbb{N}$; $\mathcal{G}$ be a first-stage solution set of U; $0 < \alpha \leq 1$ be a *rental factor* and $\beta \geq 0$ be an *inflation factor*. For a set $F \in \mathcal{G}$ and a scenario $S \in \mathcal{S}$ the *rental cost* $c_R^S(F)$ in scenario S are defined as $c_R^S(F) = \alpha \cdot c^S(F)$ and the *implementation cost* $c_I^S(F)$ as

$$c_I^S(F) = \min_{F' \in \mathcal{F}} \left((1-\alpha) c^S(F') + (\alpha + \beta) c^S(F' \backslash F) \right).$$

Then the *scenario cost* of F are the sum over both cost values and the *total cost* $c_T(F)$ the maximum scenario cost, i.e.,

$$c_T(F) = \max_{S \in \mathcal{S}} \left(c_R^S(F) + c_I^S(F) \right).$$

The *rent recoverable robust linear combinatorial minimization problem* is to find a first-stage solution $F \in \mathcal{G}$ that minimizes the total cost $c_T(F)$ over all first-stage solutions $F \in \mathcal{G}$.

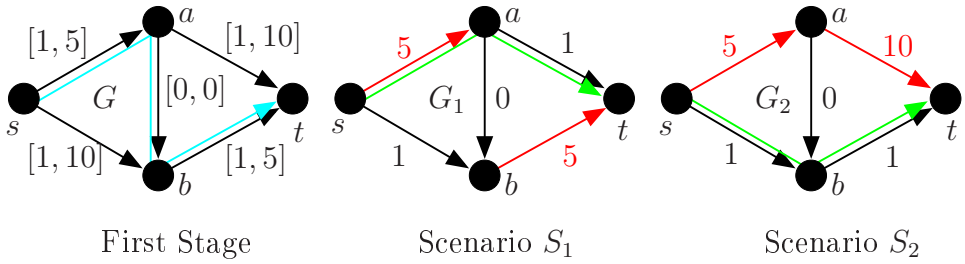
Similar to the k-Dist-recoverable robust model, the cost function c of the underlying LCMin problem $(U, \mathcal{F}, c)$ just implicitly influences the problem setting. We assume that this cost function is subject to uncertainties and hence all scenario cost functions are variations of c.

For a large inflation factor, e.g., $\beta = \max_{S \in \mathcal{S}} c^S(U)$, any optimal robust solution in the classical sense is an optimal solution for the Rent-RR LCMin problem.

In contrast to the k-Dist-recoverable robust model, the recovery set for a feasible first-stage solution is not restricted, i.e., we can choose any feasible scenario solution for the given LCMin problem. Furthermore, there are no first-stage cost. In order to illustrate the model, we consider the following example.

Example 3.1.2. We consider a rent recoverable robust shortest path problem. Let $G = (V, A)$ be the graph given in Figure 3.1 and the set of Γ-scenarios be defined by $\Gamma = 2$, the lower cost bounds and the upper cost bounds shown on the arcs. The two graphs G_1 and G_2 illustrate two feasible Γ-scenarios with their cost functions. The arcs with upper bound cost are marked red.

We now consider the rent recoverable robust problem for $\alpha = 0.2$, $\beta = 2$ and the first-stage solution set consisting of all simple (s,t)-paths.



First Stage Scenario S_1 Scenario S_2

Figure 3.1.: The left graph G illustrates the situation before any scenario is realized. The other two graphs show the cost values for each arc according to two different scenarios S_1 and S_2.

Let us investigate the blue path p in G with its cost. In the scenario S_1 shown in the graph G_1, the rental cost of p are 2 and the implementation cost 7, choosing the green colored path: for the arc (s,a) we just pay the remaining cost 4 and for the arc (a,t) we need to pay $(1+\beta)\cdot 1 = 3$. Hence, the scenario cost is 9. If the scenario S_2 is realized, the scenario cost are 5. Under all feasible Γ-scenarios, S_1 induces the maximum scenario cost on p and therefore $c_T(p) = 9$.

For the other two (s,t)-paths in G, i.e., $p_1 = sat$ and $p_2 = sbt$, we obtain the same total cost. However, if we increase the inflation factor to 3, then p is the optimal solution with total cost 10, while the other two paths have total cost 11. Note that in this example it is better to rent some arcs beforehand, since otherwise we need to pay $6(1+\beta)$. ⊞

Contribution and Chapter Outline We start again with a complexity study. For discrete scenario sets, the rent recoverable robust problem is weakly **NP**-hard for two scenarios and strongly **NP**-hard if the number of scenarios is not constant (Section 3.2). In contrast to the k-Dist-recoverable robust model, the problem is solvable in polynomial time if we consider interval scenarios (Section 3.3). Yet, for Γ-scenarios the rent recoverable robust version of the shortest path problem and the minimum (s,t)-cut problem are strongly **NP**-hard (Section 3.4).

In the last Section 3.5, we establish a close relation between robust solutions and optimal solutions for a rent recoverable robust problem. We thus are able to present a $\min\{\gamma+1+\beta, \frac{\gamma}{\alpha}\}$-approximation algorithm for a Rent-RR LCMin problem with a rental factor α and an inflation factor β if there exists a γ-approximation algorithm for the robust problem.

3.2. Discrete Scenarios

In the discrete scenario case, a set of r scenarios $S_1, \ldots, S_r$ with their corresponding cost functions is explicitly given. We start with a complexity study of the Rent-RR LCMin problem and consider again the *weighted disjoint hitting set (WDHS) problem*, since this problem can be interpreted as a special subproblem of several combinatorial optimization problems, e.g., the minimum spanning tree problem, the shortest path problem, the minimum (s,t)-cut problem or the minimum perfect matching problem (see Corollary 2.2.3). Recall that the WDHS problem is defined in the following way:

Given: A set of n elements $U = \{u_1, \dots, u_n\}$, a set of d pairwise disjoint subsets $\mathcal{M} = \{M_1, \dots, M_d\}$ and a cost function $c : U \to \mathbb{N}$.

Task: Find a set $F \subseteq U$ minimizing the cost $c(F) = \sum_{u \in F} c(u)$ such that $|F \cap M_i| = 1$ for $i = 1, \dots, d$.

We will show that already for two scenarios the rent recoverable robust version of the WDHS problem is weakly **NP**-hard.

Theorem 3.2.1. *For $\alpha > 0$ and proper first solution sets the Rent-RR WDHS problem with discrete scenarios $\mathcal{S}_D$ is weakly* **NP***-hard, even if $|\mathcal{S}_D| = 2$.*

Proof. The reduction from partition is quite similar to the one for the k-Dist-RR version (Theorem 2.2.1). Let I be an instance of partition containing n integer elements $e_1, \dots, e_n$ with $\sum_{i=1}^{n} e_i = 2b$ for some $b \in \mathbb{N}$. We construct I' as the corresponding Rent-RR WDHS instance with a set of $2n$ elements $U = \{u_1, \dots, u_{2n}\}$, n disjoint subsets $\mathcal{M} = \{M_1, \dots, M_n\}$, two scenarios $\mathcal{S}_D = \{S_1, S_2\}$ and a proper first-stage solution set $\mathcal{G}$. Each set M_i, $i = 1, \dots, n$, consists of the elements u_{2i-1} and u_{2i}, i.e., $M_i = \{u_{2i-1}, u_{2i}\}$, and the scenarios S_1, S_2 define the following cost functions

$$c^{S_1}(u) = \begin{cases} e_i & u = u_{2i-1},\ i \in \{1, \dots, n\} \\ 0 & \text{otherwise} \end{cases} \text{ and } c^{S_2}(u) = \begin{cases} e_i & u = u_{2i},\ i \in \{1, \dots, n\} \\ 0 & \text{otherwise.} \end{cases}$$

Hence, for every scenario S there exists a solution $F^S \in \mathcal{F}$ with cost $c^S(F^S) = 0$. Let F be a feasible first-stage solution, $\alpha > 0$ and $\beta \geq 0$, then

$$\begin{aligned} c_T(F) &= \max_{i=1,2} \min_{F' \in \mathcal{F}} \alpha c^{S_i}(F) + (1-\alpha) c^{S_i}(F' \cap F) + (\alpha + \beta) c^{S_i}(F' \backslash F) \\ &= \alpha \max\{c^{S_1}(F), c^{S_2}(F)\}. \end{aligned}$$

Since $\mathcal{G}$ is proper, there exists a first-stage solution F with $c_T(F) = \alpha b$ if and only if I is a yes-instance. □

We will see in the following theorem that if the number of discrete scenarios is not constant, the Rent-RR WDHS problem becomes strongly **NP**-hard. The proof is an adaption of the hardness prove for the robust shortest path problem given by Kasperski and Zieliński [70] to the Rent-RR WDHS problem.

Theorem 3.2.2. *For $\alpha > 0$ and proper first-stage solution sets the Rent-RR WDHS problem is not approximable with a factor better than 2, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. The following reduction from 3SAT shows that the problem is strongly **NP**-hard and not approximable with a factor better than 2. Let I be an instance of 3SAT with n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$ each clause consisting of three literals y_{ij}, $i = 1, 2, 3$, $j = 1, \dots, m$, with $y_{ij} \in \{x_1, \overline{x}_1, \dots, x_n, \overline{x}_n\}$. The set U' of the corresponding Rent-RR WDHS instance I' contains $3m$ different elements u_{1j}, u_{2j} and u_{3j}, $j = 1, \dots, m$. The element u_{1j} represents the first literal of the clause C_j, u_{2j} the second literal and u_{3j} the third literal, $j = 1, \dots, m$. The elements form m disjoint sets $M_j = \{u_{1j}, u_{2j}, u_{3j}\}$, $j = 1, \dots, m$. Any feasible solution for this WDHS instance chooses exactly one element out of every set. So far, two elements may be picked that represent contradictory literals, e.g., a feasible solution F' may contain u_{21} with $y_{21} = x_\ell$ and u_{32} with $y_{32} = \overline{x}_\ell$, $\ell \in \{1, \dots, n\}$. In order to punish such a choice, we construct for every pair of elements $u_{i_1 j_2}$, $u_{i_2 j_2}$ that represent such contradictory

literals, i.e., $y_{i_1j_1} = \overline{y}_{i_2j_2}$, a scenario S that assigns cost 1 to each of the two elements $u_{i_1j_1}$ and $u_{i_2j_2}$ and 0 otherwise. There are at most $3m^2$ different scenarios forming the set $\mathcal{S}_D$ that might be added to I'. Let now $\mathcal{G}$ be some proper first-stage solution set, $\alpha > 0$ be some rental factor and $\beta \geq 0$ be some inflation factor. Since for every scenario $S \in \mathcal{S}_D$ there exists a solution F^S with $c^S(F^S) = 0$, the total cost of any first-stage solution $F \in \mathcal{G}$ are given by

$$c_T(F) = \alpha \max_{S \in \mathcal{S}_D} c^S(F).$$

Hence, there exists a first-stage solution $F \in \mathcal{G}$ with total cost smaller than or equal to 1α if and only if I is a yes-instance. Since in any other case the total cost are at least 2α, no efficient approximation algorithm can be found with an approximation factor better than 2, unless $\mathbf{P} = \mathbf{NP}$. □

As in the k-Dist-RR case (Corollary 2.2.3), these results can be transferred to the shortest path problem, the minimum (s,t)-cut problem, the perfect matching problem and the minimum spanning tree problem.

Corollary 3.2.3. *The Rent-RR version of the minimum spanning tree problem, the shortest path problem, minimum (s,t)-cut problem and the minimum perfect matching problem on bipartite graphs are weakly* **NP***-hard for $|\mathcal{S}_D| = 2$ and strongly* **NP***-hard if the number of discrete scenario sets is not constant. In the later case, no approximation algorithm with an approximation factor better than 2 exists, unless* $\mathbf{P} = \mathbf{NP}$.

In contrast to the k-Dist-RR LCMin setting, the total cost for a given first-stage solution $F \in \mathcal{G}$ of a Rent-RR LCMin problem can be computed in polynomial time, when considering the scenarios as not constant and when the corresponding LCMin problem is in $\mathbf{P}$: let $(U, \mathcal{F}, c)$ be an LCMin instance, $\mathcal{G}$ be a first-stage solution set, $\mathcal{S}_D$ be a scenario set, $\alpha > 0$ be a rental factor and $\beta \geq 0$ be an inflation factor. To compute the total cost for some $F \in \mathcal{G}$ we just need to solve an LCMin instance $(U, \mathcal{F}, c_F^S)$ for every scenario $S \in \mathcal{S}_D$ with modified cost function c_F^S defined by

$$c_F^S(u) = \begin{cases} (1-\alpha)c^S(u) & \text{if } u \in F \\ (1+\beta)c^S(u) & \text{otherwise.} \end{cases}$$

If F^S is an optimal solution of $(U, \mathcal{F}, c_F^S)$, then

$$c_T(F) = \max_{S \in \mathcal{S}_D} \alpha c^S(F) + c_F^S(F^S)$$

determines the total cost of F.

In the next section we analyze interval scenarios and show that the Rent-RR LCMin problem can be solved efficiently in that case.

3.3. Interval Scenarios

Interval scenarios are given implicitly by lower and upper bounds $\underline{c}$ and $\overline{c}$ on the cost value for each element $u \in U$ of the underlying LCMin instance $(U, \mathcal{F}, c)$. In a Rent-RR version, there is one dominating scenario, namely, $S_{\max}$ with $c^{S_{\max}}(u) = \overline{c}(u)$ for every $u \in U$, as shown in the following theorem.

Theorem 3.3.1. *Let $(U, \mathcal{F}, c)$ be an LCMin instance, $\alpha > 0$ be a rental factor, $\beta \geq 0$ be an inflation factor, $\mathcal{G}$ be a proper first-stage solution set and $\underline{c}$ and $\overline{c}$ be lower and upper bounds to define interval scenarios. Then any optimal solution to the LCMin instance $(U, \mathcal{F}, \overline{c})$ is an optimal first-stage solution.*

Proof. Let $F_{\max}$ be an optimal solution of the LCMin instance $(U, \mathcal{F}, \overline{c})$. Recall that a first-stage solution set $\mathcal{G}$ is called proper if every $F \in \mathcal{G}$ contains a feasible solution $F' \in \mathcal{F}$ and $\mathcal{F} \subseteq \mathcal{G}$. Due to this property, for any $G \in \mathcal{G}$

$$\begin{aligned} c_T(G) &= \max_{S \in \mathcal{S}} \min_{F' \in \mathcal{F}} \alpha c^S(G) + (1-\alpha)c^S(F') + (\alpha+\beta) \sum_{u \in F' \setminus G} c^S(u) \\ &\geq \min_{F' \in \mathcal{F}} \alpha \overline{c}(G) + (1-\alpha)\overline{c}(F') \geq \overline{c}(F_{\max}) = c_T(F_{\max}). \end{aligned}$$

Hence, $F_{\max}$ is an optimal first-stage solution. □

For the k-Dist-RR case, the simplification for interval scenarios worked in a similar way, yet we obtained a bi-criteria optimization problem, which in general cannot be solved efficiently, in contrast to the polynomial algorithm proposed here.

3.4. Γ-Scenarios

Γ-scenarios $\mathcal{S}_\Gamma$ are a slight modification of interval scenarios. Again each scenario cost function c^S, $S \in \mathcal{S}_\Gamma$, obeys the given box constraint on the cost, yet for some integer Γ at most Γ values of c^S may deviate from their lower bounds. Although similar to interval scenarios, many recoverable robust problems tend to be strongly **NP**-hard for this scenario set. We will show that this is also the case for the Rent-RR version. To this end, we introduce the following definition.

Definition 3.4.1 (Increasing-Condition)**.** A class of LCMin problems satisfies the *increasing-condition* if for every instance $(U, \mathcal{F}, c)$ of this class there is another LCMin instance $(U', \mathcal{F}', c')$ in the same class such that $U' \setminus U = \{u\}$ and $\mathcal{F}' = \{F \subseteq U' \mid u \in F \text{ and } F \setminus \{u\} \in \mathcal{F}\}$. The instance I' is called the *increased instance* of I by the element u.

This condition enables us to uniformly increase the cost of every feasible solution in the original instance by considering the modified instance and placing arbitrary cost to the new element. Many combinatorial optimization problems, e.g., the shortest path problem, the minimum spanning tree problem or the minimum (s,t)-cut problem, obey this condition. In combination with the add-condition (Definition 2.4.1) and the **NP**-completeness of the corresponding max-scenario problem (Definition A.0.1), **NP**-hardness follows for these Rent-RR LCMin problems.

Theorem 3.4.2. *Let $\mathcal{C}$ be a class of LCMin problems that satisfy the add-condition and the increasing-condition and such that the max-scenario problem on $\mathcal{C}$ is strongly* **NP***-hard. Then the Rent-RR version of the problems in $\mathcal{C}$ is strongly* **NP***-hard.*

Proof. Let $(U, \mathcal{F}, c)$ be an LCMin instance of a class $\mathcal{C}$ described in the theorem and let I be a corresponding max-scenario version with Γ-scenarios $\mathcal{S}_\Gamma$. The task in the decision version of I is to decide for a given threshold K whether there exists a scenario $S^* \in \mathcal{S}_\Gamma$ that places more than or equal to K cost on every solution $F \in \mathcal{F}$, i.e., $\min_{F \in \mathcal{F}} c^{S^*}(F) \geq K$. We now exploit

the increasing-condition and the add-condition first to increase $(U, \mathcal{F}, c)$ by the element u and afterwards to add a solution A to built a modified LCMin instance $(U', \mathcal{F}', c')$. In order to obtain also a modified instance of the max-scenario version I, we fix the lower and upper cost bounds for A and u to a and x, respectively, while all other bounds remain as in the original max-scenario instance. To define a Rent-RR LCMin instance I', we set $\mathcal{G} = \mathcal{F}'$, the set of all feasible solutions in $(U', \mathcal{F}', c')$. Yet, as for a and x, we will set the parameters α and β later.

In the following we derive upper and lower bounds on the total cost in I' for the solutions A and and the other solutions $F \in \mathcal{F}'$ depending on I being a yes-instance or a no-instance. Using these bounds we will define α, β, a and x, such that A is the optimal solution of I' if I is a yes-instance and some other $F \in \mathcal{F}' \backslash \{A\}$ is an optimal solution of I' if I is a no-instance.

Let I be a yes-instance, i.e., there exists a scenario $S \in \mathcal{S}_\Gamma$ with $\min_{F \in \mathcal{F}} c^S(F) \geq K$. For every feasible solution $F \in \mathcal{F}' \backslash \{A\}$ the total cost can be lower bounded by

$$\begin{aligned} c_T(F) &= \max_{S \in \mathcal{S}_\Gamma} \min_{F' \in \mathcal{F}'} \alpha c^S(F) + (1-\alpha) c^S(F') + (\alpha + \beta) c^S(F' \backslash F) \\ &\geq \min\{(K+x), \alpha(K+x) + (1+\beta)a\}. \end{aligned}$$

For the solution A we get

$$c_T(A) \leq \min\{a, \max_{S \in \mathcal{S}_\Gamma}(\alpha a + (1+\beta) \min_{F \in \mathcal{F}} c^S(F))\} \leq a$$

Thus, $c_T(A) < c_T(F)$ for any $F \in \mathcal{F}' \backslash \{A\}$ if

$$a < K + x. \tag{3.1}$$

Let now I be a no-instance, i.e., for every scenario $S \in \mathcal{S}_\Gamma$ there is a solution $F \in \mathcal{F}$ with $c^S(F) \leq K - 1$. We choose a solution $\overline{F}^r \in \mathcal{F}$ and set $c^r = \max_{S \in \mathcal{S}_\Gamma} c^S(\overline{F}^r \backslash \{u\})$. For this solution we obtain an upper bound on the total cost by

$$\begin{aligned} c_T(\overline{F}^r) &= \max_{S \in \mathcal{S}_\Gamma} \min_{F' \in \mathcal{F}'} \alpha c^S(\overline{F}^r) + (1-\alpha) c^S(F') + (\alpha + \beta) c^S(F' \backslash \overline{F}^r) \\ &\leq \alpha(c^r + x) + (1-\alpha)x + (1+\beta)(K-1), \end{aligned}$$

if

$$(1+\beta)a > (1+\beta)(K-1) + (1-\alpha)x. \tag{3.2}$$

On the other hand, since $c^S(F) \geq x$ for all $S \in \mathcal{S}_\Gamma$ and all $F \in \mathcal{F}' \backslash \{A\}$, the total cost of A sum up to

$$c_T(A) = \min\{a, \max_{S \in \mathcal{S}_\Gamma} \min_{F' \in \mathcal{F}'} \alpha a + (1+\beta) c^S(F')\} = a$$

if

$$(1-\alpha)a < (1+\beta)x. \tag{3.3}$$

Comparing these two bounds, $c_T(\overline{F}^r) < c_T(A)$ if

$$\alpha(c^r + x) + (1-\alpha)x + (1+\beta)(K-1) < a. \tag{3.4}$$

As one can easily recalculate, the inequalities (3.1)-(3.4) next to $\alpha \in]0,1[$ and $\beta \geq 0$ are satisfied for $\beta = \frac{1}{2K}$, $\alpha = \frac{1}{2c^r+1}\beta$, $x = \frac{(1-\alpha)}{(\beta+\alpha)}(K - \varepsilon) + \delta$, for some $0 < \varepsilon < \frac{1}{2}$ and $\delta > 0$, and $a = K + x - \varepsilon$. Thus, A is the optimal solution of the Rent-RR LCMin instance I' with $\beta = \frac{1}{2K}$ and $\alpha = \frac{1}{2c^r+1}\beta$ if and only if the max-scenario instance I is a yes-instance. □

Since the shortest path problem and the minimum (s,t)-cut problem meet the requirements of Theorem 3.5.1, their Rent-RR versions are strongly **NP**-hard.

Corollary 3.4.3. *The Rent-RR shortest path problem and the Rent-RR minimum (s,t)-cut problem with Γ-scenarios are strongly* **NP***-hard.*

Note that the robust version of the shortest path problem as well as of the minimum (s,t)-cut problem can be solved in polynomial time [14]. It remains open, whether, for example, the Rent-RR minimum spanning tree problem with Γ-scenarios can be solved efficiently.

In the next section we show, how to get an approximation algorithm for the Rent-RR problem based on a robust solution.

3.5. Approximation via Robust Solutions

Rent-RR LCMin problems tend to be **NP**-hard and for discrete scenarios not to be approximable with a factor better than 2 (Corollary 3.2.3). We will show that an upper bound for the total cost can be deduced from an optimal robust solution. Furthermore, any approximation algorithm of the classical robust optimization problem yields an approximation algorithm of the Rent-recoverable robust version with a similar approximation guaranty. As an optimal robust solution we denote the following solution: Let $(U,\mathcal{F},c)$ be an LCMin instance and $\mathcal{S}$ be a set of scenarios, where each scenario $S\in\mathcal{S}$ defines a cost function $c^S: U\to\mathbb{N}$. Then an *optimal robust solution* is a solution $F^*\in\mathcal{F}$ which minimizes the maximum cost over all scenarios, i.e., $F^* = \arg\min_{F\in\mathcal{F}}\max_{S\in\mathcal{S}} c^S(F)$.

Theorem 3.5.1. *Let $(U,\mathcal{F},c)$ be an LCMin instance, $\mathcal{S}$ be a set of scenarios and* ALG *be an approximation algorithm for the robust LCMin problem with an approximation factor γ. For a given rental factor $\alpha\in]0,1[$ and inflation factor $\beta\geq 0$, we define algorithm* ALG′ *as:*

First Stage: *Run* ALG *and set the first-stage solution F^r to the output of* ALG.

Recovery: *For any $S\in\mathcal{S}$ calculate an optimal solution F^S to the LCMin instance $(U,\mathcal{F},c')$ with*

$$c'(u) = \begin{cases}(1-\alpha)\cdot c^S(u) & \forall u\in F^r\\ (1+\beta)\cdot c^S(u) & \forall u\notin F^r.\end{cases}$$

Then ALG′ *is an approximation algorithm with an approximation factor $\gamma' = \min\{(\gamma+1+\beta),\frac{\alpha}{\gamma}\}$ of the Rent-RR version for given α and β and proper first solution set.*

Proof. Let F^r be a solution computed by algorithm ALG for the given LCMin instance $(U,\mathcal{F},c)$ and a set of scenarios $\mathcal{S}$ and let OPT_r be the value of an optimal robust solution, i.e., $\mathrm{OPT}_r = \min_{F\in\mathcal{F}}\max_{S\in\mathcal{S}} c^S(F)$. Furthermore, let OPT be the value of an optimal solution of the Rent-RR LCMin instance with a rental factor $\alpha>0$ and an inflation factor β and some proper first-stage solution set $\mathcal{G}$. We start with two lower bounds on OPT: First

$$\begin{aligned}\mathrm{OPT} &= \min_{F\in\mathcal{F}}\max_{S\in\mathcal{S}}\min_{F'\in\mathcal{F}} \alpha c^S(F) + (1-\alpha)c^S(F') + (\alpha+\beta)c^S(F'\backslash F)\\ &\geq \alpha\min_{F\in\mathcal{F}}\max_{S\in\mathcal{S}} c^S(F) = \alpha\,\mathrm{OPT}_r \qquad (3.5)\end{aligned}$$

and second

$$\mathrm{OPT} \geq \max_{S\in\mathcal{S}} \min_{F'\in\mathcal{F}} c^S(F'). \tag{3.6}$$

We use the first bound (3.5) to obtain an estimation of the maximum rental cost of F^r, more precise,

$$c_R(F^r) = \alpha \max_{S\in\mathcal{S}} c^S(F^r) \leq \alpha \cdot \gamma \cdot \mathrm{OPT}_r \leq \gamma\,\mathrm{OPT}\,. \tag{3.7}$$

An upper bound of the implementation cost of F^r is given by

$$c_I^S(F^r) \leq (1+\beta) \max_{S\in\mathcal{S}} \min_{F'\in\mathcal{F}} c^S(F') \leq (1+\beta)\,\mathrm{OPT}, \tag{3.8}$$

using inequality (3.6). Combining estimates (3.7) and (3.8), we get

$$\begin{aligned} c_T(F^r) &= \max_{S\in\mathcal{S}} c_R^S(F^r) + c_I^S(F^r) \leq c_R(F^r) + \max_{S\in\mathcal{S}} c_I^S(F^r) \\ &\leq \gamma\,\mathrm{OPT} + (1+\beta)\,\mathrm{OPT} \leq (\gamma+1+\beta)\,\mathrm{OPT}\,. \end{aligned}$$

Thus, we have a first approximation guaranty of ALG′.

The second guaranty is based on the recovery step. In the second stage an optimal solution w.r.t. the cost function c' is chosen, hence

$$\frac{1}{\gamma} c_T(F^r) \leq \min_{F\in\mathcal{F}} \max_{S\in\mathcal{S}} c^S(F) \leq \frac{1}{\alpha}\,\mathrm{OPT}\,.$$

To sum up, algorithm ALG′ is a $\min\{\gamma+1+\beta, \frac{\gamma}{\alpha}\}$-approximation algorithm for the Rent-RR LCMin problem. □

Since the robust version of an LCMin problem with Γ-scenarios can be solved efficiently, the algorithm proposed in Theorem 3.5.1 has an approximation factor of $\min\{\frac{1}{\alpha}, 2+\beta\}$ in that case. Considering the shortest path problem, this analysis is tight for $\alpha \geq 0.5$: Let G be the given graph in Figure 3.2 with the associated lower and upper cost bounds. Then in the case of Γ-scenarios with $\Gamma = 2$, both paths p and $\overline{p}$ are robust shortest paths. If we choose $\overline{p}$, its total cost is

$$c_T(\overline{p}) = \min\{1, \alpha \cdot 1 + 0.5(1+\beta)\},$$

whereas the path p just generates total cost

$$c_T(p) = \max\{0.5, \alpha\}.$$

For $\alpha \geq 0.5$, $c_T(p)$ is the optimal solution and by choosing $\overline{p}$ we obtain a performance ratio of $\frac{1}{\alpha}$.

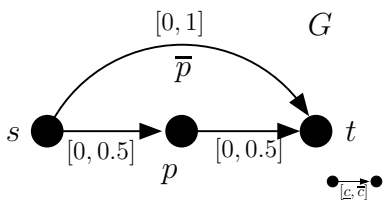


Figure 3.2.: We consider Γ-scenarios defined by the lower and upper cost bounds shown on the arcs and $\Gamma = 2$. Then both paths p and $\overline{p}$ have robust cost 1.

Note that an optimal solution of the Rent-RR shortest path problem is not necessarily a robust solution: Let us consider a graph G containing two parallel arcs a_1 and a_2, two scenarios S_1 and S_2 with $(c^{S_1}(a_1), c^{S_2}(a_1)) = (0, 50)$ and $(c^{S_1}(a_2), c^{S_2}(a_2)) = (100, 20)$, and $\alpha = 0.1$ and $\beta = 10$. In this setting a_1 is a robust shortest path but its total cost in the Rent-RR version are 50, whereas the total cost of a_2 are just 20. Yet, a_2 is not a robust shortest path.

Unfortunately, the algorithm proposed in Theorem 3.5.1 does not guarantee an approximation factor for every recoverable robust setting. Consider the following example for a k-Dist-RR shortest path problem with Γ-scenarios: Let G be the graph given in Figure 3.3 without first-stage cost c^D, i.e., $c^D \equiv 0$, and Γ-scenarios defined by upper and lower cost bounds of $[0, 1]$ and $\Gamma = 2$. Then p_1 is the only robust shortest path. Yet, if we consider the recovery parameter $k = 3$, the total cost of p_1 are 1, while any other (s, t)-path has total cost 0.

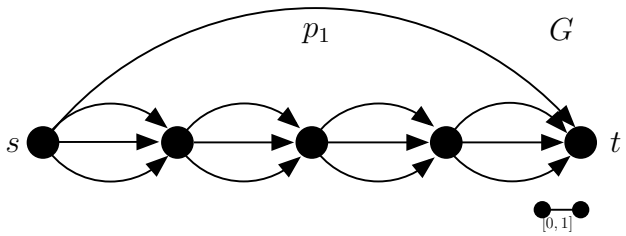


Figure 3.3.: The path p_1 is an optimal robust solution, but does not approximate the optimal solution of the k-Dist-RR shortest path version.

3.6. Conclusion and Open Issues

Rent recoverable robustness models a setting in which the choice of an element in the first stage influences the cost of purchasing the element in the second stage. For discrete scenario sets and in the case of two scenarios, we showed in Section 3.2 that the problem is at least weakly **NP**-hard. It remains open whether we can actually solve these problems in pseudo-polynomial time. The Rent-RR WDHS problem should be considered first, since this simple problem is a subproblem of several other combinatorial optimization problems (Corollary 3.2.3). If the number of discrete scenarios is not constant, the problem is even strongly **NP**-hard and not approximable with a factor better than 2, unless $\mathbf{P} = \mathbf{NP}$.

In the case of interval scenarios the Rent-RR LCMin problem with proper first-stage solution sets is tractable if the considered LCMin problem is tractable (Section 3.3). The same probably holds when the first-stage solution set consists of all subsets of the underlying set of elements. However, further restrictions on this set may change the result or make computing an optimal solution more challenging.

As last type of scenario sets, we analyzed Γ-scenarios in Section 3.4. Since the Rent-RR problem behaves like the robust version for the two other scenario types, it surprises that the Rent-RR problem is strongly **NP**-hard for the shortest path problem and the minimum (s, t)-cut problem. However, it is not known if we can solve, for example, the Rent-RR WDHS problem in polynomial time. Furthermore, no lower bound on the best possible approximation factor results from the provided hardness proof.

Finally, we proved that any γ-approximation algorithm for the robust problem induces a $\min\{\gamma + 1 + \beta, \frac{\gamma}{\alpha}\}$-approximation algorithm for the Rent-RR problem (Section 3.5). This is so far the only constructive result for discrete scenario sets and Γ-scenario sets. It would be interesting to find a different approximation algorithm, whose approximation factor is independent of the rental factor and the approximation factor.

4. Exact Subset Recoverable Robustness

In this chapter we consider exact subset recoverable robustness. For a given LCMin instance $(U, \mathcal{F}, c)$ and a scenario set $\mathcal{S}$, where each scenario defines a cost function $c^S : U \to \mathbb{N}$, this problem is to find a set $U' \subseteq U$ that contains for every scenario $S \in \mathcal{S}$ an optimal solution w.r.t. c^S of the original instance.

For discrete scenario sets the exact subset recoverable robust (exSub-RR) version of the shortest path problem, the minimum spanning tree problem or the minimum (s,t)-cut problem are strongly **NP**-hard. In the case of interval scenarios we introduce a criterion to decide for a given element, whether it is contained in every optimal solution. Using this criterion we give a lower bound of $|U|^{(1-\varepsilon)}$, for any $\varepsilon > 0$, on the approximability of the exSub-RR shortest path and the exSub-RR minimum (s,t)-cut problem, where U denotes the set of arcs in the considered graph. On the other hand, we prove that the exSub-RR minimum weight basis problem for matroids can be solved efficiently.

Considering Γ-scenarios, the exSub-RR shortest path problem and the exSub-RR (s,t)-cut problem remain **NP**-hard shown by a reduction from the corresponding max-scenario problem. But even the exact subset recoverable robust minimum spanning tree problem is strongly **NP**-hard since it is closely related to the k-connected spanning subgraph problem. Finally, we introduce an approximation algorithm with an approximation guarantee of $\frac{1}{\ell}|U|$ for any $\ell \in \mathbb{N}$ and any LCMin instance $(U, \mathcal{F}, c)$.

4.1. Introduction

Motivation Security and stability is a major concern in network design, e.g., in the design of telecommunication networks, railway networks or highway networks [19, 75]. Several different evaluation techniques have been developed like the classical concepts of connectivity or maximum distance, to measure the risk of a network break down due to random failures or intentional attacks. Often, stability of a network comes at the cost of maintaining a large system. To lower these cost, companies are interested in reducing the network without decreasing its stability. However, this criterion does not include any service promises to the customers. The obtained subnetwork may therefore be stable but the requests of the customers may be routed along very long paths.

For telecommunication networks one aspect of the trade-off between stability and fast connection is captured in the so-called *bounded disjoint path problems* [19]. In this problem we consider only the connection between two designated nodes. The task is to find a set of k disjoint paths such that the length of each path is bounded by a given value ℓ. Thus, the number of disjoint paths models the stability and the length of the path represents the service for the customers.

The *exact subset recoverable robust problem* focuses on these service guarantees. Given a network problem in which demand needs to be routed, we assume that the cost functions, e.g., the routing times, are uncertain and modeled via a set of scenarios. Then the *exact subset recoverable robust* problem asks for a minimum subnetwork such that for each scenario an optimal routing in the whole network is also contained in the subnetwork. Hence, the customer experiences no change in service, i.e., in the traveling time, although the network is reduced. Due to the modeling power of scenario sets, we will later see that the issue of connectivity can also be included in this approach.

Model and Notation Exact subset recoverable robustness was mainly motivated by network design problems and variances in the cost function. In terms of our general framework we will introduce the concept for linear combinatorial minimization problems and scenario sets that determine sets of feasible solutions. As defined before (see Definition 1.1.1), a linear combinatorial minimization (LCMin) problems is given by a set of elements U, a set of feasible solutions $\mathcal{F} \subseteq 2^U$, and a cost function $c : U \to \mathbb{N}$. The task of such a problem is to find a solution $F \in \mathcal{F}$ with minimum cost $c(F) = \sum_{u \in F} c(u)$. Typical linear combinatorial minimization problems are the shortest path problem, the minimum spanning tree problem and the minimum (s,t)-cut problem. For an LCMin we define the exact subset recoverable robust version in the following way:

Definition 4.1.1 (Exact Subset Recoverable Robust Linear Combinatorial Minimization (ex-Sub-RR LCMin) Problem)**.** Let $(U, \mathcal{F}, c)$ be an instance of an LCMin problem and $\mathcal{S}$ be a set of scenarios, such that each scenario $S \in \mathcal{S}$ defines a set of *scenario feasible solutions* $\mathcal{F}^S \subseteq \mathcal{F}$. A set $U' \subseteq U$ is a *feasible* first-stage solution if for every scenario $S \in \mathcal{S}$, there exists an $F \in \mathcal{F}^S$ with $F \subseteq U'$. The *exact subset recoverable robust linear combinatorial minimization problem* is to find a feasible first-stage solution of minimum cardinality. An instance of this problem is given by the triple $(U, \mathcal{F}, \mathcal{S})$.

The cost function c of the underlying LCMin problem $(U, \mathcal{F}, c)$ does not explicitly induce any cost for a first-stage solution. The problem setting can easily be extended to find a feasible solution U' with minimum cost $c(U')$. In order to keep it simple, we begin with a study of minimizing the cardinality of a first-stage solution.

In contrast to the other two recoverable robust models considered in Chapter 2 and 3, each scenario of the given scenario set determines a set of feasible solutions. This definition could be arbitrary, e.g., the set of scenario feasible solutions could consist of all feasible solutions with bounded cardinality. However, in this chapter we will just consider a special case of scenario sets in which every scenario $S \in \mathcal{S}$ defines a cost function $c^S : U \to \mathbb{N}$ and the set of scenario feasible solutions consists of all minimum cost solutions of the underlying LCMin problem $(U, \mathcal{F}, c)$ with respect to the cost function $c = c^S$, i.e.,

$$\mathcal{F}^S = \{F \in \mathcal{F} \mid c^S(F) = \min_{F' \in \mathcal{F}} c^S(F')\}.$$

We will still distinguish the different types of scenario sets by the way their cost functions are defined. That means that we call, for example, a set of scenarios a discrete scenario set if r scenarios $S_1, \ldots, S_r$ with their cost functions are explicitly given while the set of scenario feasible solutions may be arbitrarily large. Before we give an overview of related results, we consider a small example for the exact subset recoverable robust shortest path problem.

Example 4.1.2. Let $G = (V, A)$ be the graph shown in Figure 4.1 and $\mathcal{S}_\Gamma$ be the set of Γ-scenarios determined by $\Gamma = 1$, the lower cost bound 0 and upper cost bound 1 on every

arc. For a scenario $S \in \mathcal{S}_\Gamma$ the set of scenario feasible solutions corresponds to all (s,t)-paths in G which do not cross the arc whose cost value is increased to 1. In G_1 the scenario S_1 is shown that increases the cost of the red colored arc. The green path is a feasible recovery path in this setting and is also included in the blue subgraph, shown in the left part of the figure. For the scenario S_2 illustrated in G_2 a different path, e.g., the green colored one, needs to be chosen.

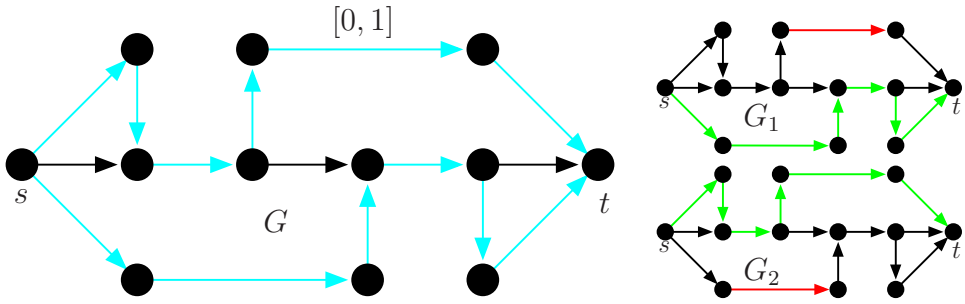


Figure 4.1.: The left graph G shows the situation in the first stage, where the different cost values of the arcs are not known. On the right side, two possible realizations are illustrated, where all arc costs are 0 except for the red arcs.

One can easily observe that any feasible solution for this exSub-RR shortest path instance contains two disjoint paths in G. Since the blue colored graph is a minimum subgraph w.r.t. the number of arcs with this property, it is an optimal solution. If we change the lower and upper bound to 1 and 2, the optimal solution consists of the middle (s,t)-path. This path is a shortest path in any feasible Γ-scenario with $\Gamma = 1$. ⊞

Related Results The classical concept of connectivity leads to the *k-spanning subgraph problem* for some integer k, where a k-connected subgraph of a given graph $G = (V, E)$ with a minimum number of edges needs to be found. A graph $G = (V, E)$ is called *k-connected* if for each vertex pair $u, v \in V$ there exist k edge disjoint paths in G. This problem has widely been studied (see [28, 51, 74] and references within). It is well-known that the problem is strongly **NP**-hard, since any Hamiltonian cycle is a minimum 2-connected spanning subgraph of a given graph. Gabow et al. [51] showed that a polynomial-time algorithm approximating the smallest k-connected subgraph within a ratio $1 + \frac{c}{k}$ for some constant c implies $\mathbf{P} = \mathbf{NP}$. On the other hand, they developed an approximation algorithm based on rounding an LP solution of the relaxation of the natural linear integer program formulation. Its approximation ratio is $1 + \frac{2}{k}$ for any directed graph and arbitrary k as well as for undirected graphs with even k. If k is odd, the performance guaranty is at least $1 + \frac{3}{k}$ for undirected graphs.

As described above, in telecommunication networks the trade-off between stability and fast connections is captured in the bounded disjoint path problem. Given a directed graph $G = (V, A)$, two vertices $s, t \in V$, a length bound $\ell \in \mathbb{N}$ and a parameter $k \in \mathbb{N}$, the problem is to find a set of k disjoint (s,t)-paths, where each path contains at most ℓ arcs. Roughly speaking, the problem of finding a maximum number of disjoint paths with a length bound $\ell \leq 3$ is solvable in polynomial time, while for $\ell \geq 5$ the problems become **APX**-complete. More details on related results can be found in [19].

In [58] Hassin and Rubinstein introduced the concept of *α-robustness* for the maximum weighted matching problem. For a given graph $G = (V, E)$ they defined a p-matching to be a subset of pairwise disjoint edges with cardinality p. An *α-robust* matching is an inclusion

wise maximal matching M, such that for a given weight function $w : E \to \mathbb{N}$ the heaviest p edges in M have a weight of at least α-times the weight of a maximum p-matching in the graph G for every $p \leq |M|$. They show that there always exists a $\frac{1}{\sqrt{2}}$-robust matching, which can be computed efficiently. Fujita et al. [50] also provided a polynomial algorithm for computing a 1-robust matching if one exists, and proved **NP**-hardness for finding an α-robust matching for $\frac{1}{\sqrt{2}} < \alpha < 1$. This concept can also be adapted to different parametrized problems, e.g., finding a maximum weighted tree spanning k vertices. Thus, Hassin and Segev [59] considered the problem of finding heavy paths and heavy trees of k edges and Fujita et al. [50] the maximum weight intersection of two matroids.

Contribution and Chapter Outline We consider again three different types of scenario sets, the discrete scenario sets, the interval scenario sets and the Γ-scenario sets. In the discrete scenario case (Section 4.2), we show that most problems are strongly **NP**-hard if the number of scenarios is not constant. For three scenarios the exSub-recoverable robust version of an artificial LCMin problem is strongly **NP**-hard by a reduction from 3-matroid intersection.

In Section 4.3 we discuss interval scenarios. If the lower cost bound is strictly smaller than the upper cost bound of an element, we introduce a necessary and sufficient criterion for this element to be contained in every feasible solution. We call such an element *necessary*. Using this concept, we show that the exSub-RR LCMin problem is in **coNP**. Furthermore, this problem cannot be approximated within a factor of $|U|^{(1-\varepsilon)}$ for any $\varepsilon > 0$ if it is **NP**-hard to decide whether an element is necessary or not. This is in particular the case for the shortest path problem or the minimum (s,t)-cut problem. On the other hand, we can use the criterion to obtain a polynomial algorithm for solving the minimum weight basis problem on matroids (Section 4.5.2).

Section 4.4 covers our results w.r.t. Γ-scenarios. For Γ-scenarios, several exSub-RR LCMin problems become again **NP**-hard, e.g., the shortest path problem, the minimum (s,t)-cut problem and the minimum spanning tree problem. The first two statements are obtained from a general construction relating the max-scenario problem to the exSub-RR LCMin problem. The last result is due to a correspondence of the k-connected subgraph problem with the exSub-recoverable robust minimum spanning tree (MST) problem with Γ-scenarios, where the set of Γ-scenarios is defined by lower cost bounds 0, upper cost bounds 1 and $\Gamma = k + 1$. A simple 2-approximation for the exSub-RR MST problem is given in Section 4.5.3. Finally we introduce in Section 4.6 approximation algorithms for the exSub-RR LCMin problem with Γ-scenarios and interval scenarios. Parts of this chapter are based on work with Rico Zenklusen. Results on the shortest path problem have been published in [22].

4.2. Discrete Scenarios

A discrete scenario set $\mathcal{S}_D$ is given by r scenarios $S_1, \ldots, S_r$, where each scenario S_i, $i = 1, \ldots, r$, defines a cost function $c^{S_i} : U \to \mathbb{N}$ on the underlying LCMin instance $(U, \mathcal{F}, c)$. We start with a complexity study of the exSub-RR LCMin problem. To this end, we analyze the exSub-RR version of the so-called minimum element problem. This problem can be interpreted as a special case of the shortest path problem, the minimum (s,t)-cut problem and the minimum perfect matching problem. If the number of scenarios is not constant, we prove its **NP**-hardness and proceed with a reduction from this problem to all exSub-RR LCMin versions mentioned above.

The *minimum element problem* is a linear combinatorial optimization problem given by a set of n elements $U = \{u_1, \dots, u_n\}$ and a cost function $c : U \to \mathbb{N}$. Every feasible solution F consists of exactly one element $u \in U$ and the task is to find an element with minimum cost $c(u)$. Although this problem is quite simple, we will show that its exact subset recoverable robust version is already strongly **NP**-hard.

Theorem 4.2.1. *The exact subset recoverable robust minimum element problem is strongly* **NP***-hard for discrete scenario sets $\mathcal{S}_D$.*

Proof. We show a reduction from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \dots, x_n$ and m clauses $C_1, \dots, C_m$. We construct an instance I' of the exact subset recoverable robust minimum element problem by defining a set U' and a set of discrete scenarios $\mathcal{S}_D$. For each variable x_i, $i = 1, \dots, n$, we introduce two elements $u_i, \overline{u}_i$ to U'. The set of scenarios $\mathcal{S}_D$ contains for every clause C_j, $j = 1, \dots, m$, a scenario S_j with the cost function $c^{S_j} : U' \to \{0, 1\}$ defined as $c^{S_j}(u) = 0$ if $u = u_i$ and $x_i \in C_j$, $c^{S_j}(u) = 0$ if $u = \overline{u}_i$ and $\overline{x}_i \in C_j$, $i = 1, \dots, n$, and $c^{S_j}(u) = 1$ otherwise. Furthermore, we add for every variable x_i, $i = 1, \dots, n$, a scenario S'_i to $\mathcal{S}_D$ with $c^{S'_i}(u) = 0$ if $u \in \{u_i, \overline{u}_i\}$, and $c^{S'_i}(u) = 1$ otherwise. Due to the latter extra scenarios each feasible solution has to contain at least one of the two elements u_i or $\overline{u}_i$, $i = 1, \dots, n$. Since $|U'| = 2 \cdot n$ and $|\mathcal{S}_D| = m + n$, the size of I' is polynomial in the size of I.

In the next two paragraphs we will prove that there exists a feasible solution U^* with $|U^*| = n$ if and only if I is a yes-instance.

Let x^* be a feasible solution of I. A solution U^* to I' is constructed by adding for each variable x_i, $i = 1, \dots, n$, the element u_i to U^* if $x^*_i = \text{true}$ and by adding the element $\overline{u}_i$ to U^* if $x^*_i = \text{false}$. Hence, $|U^*| = n$. For every scenario S'_i, $i = 1, \dots, n$, an element of cost 0, i.e., a minimum element according to the cost function $c^{S'_i}$, is contained in U^*. Since x^* is a feasible solution, U^* contains an element of cost 0 for every scenario S_j, $j = 1, \dots, m$. Therefore, U^* is a feasible first-stage solution with $|U^*| = n$.

Let U^* be a feasible first-stage solution with $|U^*| = n$. Due to the definition of the scenarios S'_i, $i = 1, \dots, n$, either u_i or $\overline{u}_i$ is part of U^*. Since $|U^*| = n$, the assignment x^* is well defined for $i = 1, \dots, n$ with

$$x^*_i = \begin{cases} \text{true} & \text{if } u_i \in U^* \\ \text{false} & \text{otherwise.} \end{cases}$$

Furthermore, for every scenario $S \in \mathcal{S}_D$ an element of cost 0 is included in U^* and thus at least one literal of any clause is verified by x^*. Therefore, x^* is a feasible solution of I. □

Obviously, the minimum element problem $(U, \mathcal{F}, c)$ can be interpreted as a shortest path, a minimum spanning tree or a perfect matching problem. The set U is represented by a graph formed by $|U|$ parallel arcs, where each arc represents an element of U. Any (s,t)-path, spanning tree or matching corresponds to a feasible solution of the original problem. Representing U as a simple (s,t)-path with length $|U|$ instead of $|U|$ parallel arcs, any (s,t)-cut presents a feasible solution. Thus, the minimum element problem can also be read as a minimum (s,t)-cut problem. Since any cost function adapts in a natural way, i.e., the cost of an element $u \in U$ are assigned to the corresponding arc in the considered graph, the complexity status of the exSub-RR version of the minimum element problem passes over to the exSub-RR versions of the shortest path problem, the minimum spanning tree problem, the minimum matching problem and the minimum (s,t)-cut problem. A similar construction

was already introduced in the proof of Corollary 2.2.3 to model a k-Dist-RR version of the weighted disjoint hitting set problem as a k-Dist-RR version of the above mentioned problems.

Corollary 4.2.2. *The exact subset recoverable robust version of the shortest path problem, the minimum spanning tree problem and the minimum (s,t)-cut problem is strongly* **NP***-hard for discrete scenario sets $\mathcal{S}_D$ even in series-parallel graphs.*

The proof of Theorem 4.2.1 depends on the assumption that the number of scenarios is not constant. But even if we restrict the set $\mathcal{S}_D$ to three scenarios, the exSub-RR LCMin problem remains strongly **NP**-hard.

Theorem 4.2.3. *The exact subset recoverable robust linear combinatorial minimization problem with a constant number of discrete scenarios is strongly* **NP***-hard.*

Proof. The reduction is from 3-matroid intersection (SP11, in [52]). We consider such an instance I given by a set E and three different independent sets $\mathcal{F}_1, \mathcal{F}_2$ and $\mathcal{F}_3$ over E, which form the matroids $(E, \mathcal{F}_1)$, $(E, \mathcal{F}_2)$ and $(E, \mathcal{F}_3)$. For some given $K \leq |E|$ the task is to decide whether there exists a subset E^* of E with $E^* \in (\mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3)$ and $|E^*| \geq K$. In order to define a linear combinatorial minimization instance $I' = (U', \mathcal{F}', c)$, we set $U' = E \cup \{u_1, u_2, u_3\}$. A subset $\overline{U} \subseteq U'$ is a feasible solution in I' if $\overline{U}$ contains exactly one of the elements u_1, u_2, u_3, namely u_i, and $E \backslash \overline{U}$ is an independent set of the corresponding matroid $(E, \mathcal{F}_i)$, i.e., $\overline{U} \cap \{u_1, u_2, u_3\} = \{u_i\}$ and $E \backslash \overline{U} \in \mathcal{F}_i$ for some $i \in \{1, 2, 3\}$. To obtain an exSub-RR version of I', we define three scenarios S_1, S_2 and S_3 with the cost function

$$c^{S_i}(u) = \begin{cases} 1 & \text{if } u = u_j,\ j \neq i \\ 0 & \text{otherwise} \end{cases}$$

for $i = 1, 2, 3$. Due to the definition of c^{S_i},

$$\begin{aligned} \mathcal{F}^{S_i} &= \{\overline{U} \subseteq U' \mid \overline{U} \cap \{u_1, u_2, u_3\} = \{u_i\} \text{ and } E \backslash \overline{U} \in \mathcal{F}_i\} \\ &= \{\overline{U} \subseteq U' \mid \overline{U} = E \backslash \tilde{U} \cup \{u_i\} \text{ and } \tilde{U} \in \mathcal{F}_i\}. \end{aligned}$$

We show that the exSub-RR version of I' with $\mathcal{S}_D = \{S_1, S_2, S_3\}$ contains a solution U^* with $|U^*| \leq |E| - K + 3$ if and only if there exists a set $E^* \in (\mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3)$ with $|E^*| \geq K$.

Let E^* be a subset of E and $E^* \in (\mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3)$ with $|E^*| \geq K$. Then $U^* = \{E \backslash E^*\} \cup \{u_1, u_2, u_3\}$ is a feasible first-stage solution since for $i = 1, 2, 3$ the set $U_i = \{E \backslash E^*\} \cup \{u_i\}$ is a subset of U^* and in $\mathcal{F}^{S_i}$. Furthermore,

$$|U^*| = |E| - |E^*| + 3 \leq |E| - K + 3.$$

On the other hand, if U^* is a feasible solution of I', it contains the elements u_1, u_2 and u_3. Furthermore, $E^* = E \backslash U^*$ is part of $\mathcal{F}_1 \cap \mathcal{F}_2 \cap \mathcal{F}_3$: the set U^* is a feasible solution. Hence, for $i = 1, 2, 3$ there exists a subset $E_i \subseteq U^*$ with $E \backslash E_i \in \mathcal{F}_i$. Since $\mathcal{F}_i$ is an independent system and $E^* = E \backslash U^* \subseteq E \backslash E_i$, E^* is also part of $\mathcal{F}_i$. Thus, E^* is a feasible solution of the intersection. If finally $|U^*| \leq |E| - K + 3$, then $|E^*| = |E| - |U^*| + 3 \geq K$. To sum up, I is a yes-instance if and only if an optimal solution of the formed exSub-RR LCMin instance contains less than or equal to $|E| - K + 3$ elements. □

The constructed linear combinatorial minimization problem in the proof is artificial, but it can be solved in polynomial time as follows. It suffices to compute for a given cost function and each matroid $(E, \mathcal{F}_i)$ a maximum weight basis B_i, $i = 1, 2, 3$, and choose the set $E \backslash B_i \cup \{u_i\}$ with minimum cost. However, no conclusion can be drawn from this result for the recoverable robust versions of the shortest path problem, the minimum spanning tree problem or some other well-known combinatorial minimization problems.

So far, we investigated optimization problems and proved their **NP**-hardness. As we will see in the following, their decision versions are in **NP**. The decision version of an exSub-RR LCMin problem is defined as:

Given: A linear combinatorial minimization problem $(U, \mathcal{F}, c)$, a discrete scenario set $\mathcal{S}_D$, such that each scenario $S \in \mathcal{S}_D$ defines a cost function $c^S : U \to \mathbb{N}$, and a size bound K.

Decide: Is there a feasible first-stage solution $U' \subseteq U$ with $|U'| \leq K$?

If the set of scenarios is part of the input, a first and simple **NP**-certificate for this decision problem is a set $U' \subseteq U$ as first-stage solution and a set $F^S \in \mathcal{F}^S$ for each scenario $S \in \mathcal{S}_D$. But since the considered LCMin problems are in **P**, the feasibility of a set $U' \subseteq U$ can be tested in polynomial time: For each scenario $S \in \mathcal{S}_D$, we define the following cost function $\overline{c}^S : U \to \mathbb{N}$ with

$$\overline{c}^S(u) = \begin{cases} c^S(u) & \text{if } u \in U' \\ M & \text{otherwise.} \end{cases}$$

Furthermore, we compute an optimal solution $F^S_{U'}$ for the instance $(U, \mathcal{F}, \overline{c}^S)$ and an optimal solution F^S_U for the instance $(U, \mathcal{F}, c^S)$. By choosing M large enough, e.g., $M = |U| \cdot \max_{u \in U} c^S(u)$, the set $F^S_{U'}$ just contains elements of U' as long as U' contains any feasible solution. If $c^S(F^S_{U'}) = c^S(F^S_U)$ for all $S \in \mathcal{S}$, the set U' is a feasible first-stage solution. Thus, a set $U' \subseteq U$ suffices as certificate for a yes-instance (in contrast to the k-Dist-RR version). Furthermore, the exSub-RR LCMin decision problem can be decided in polynomial time for K not being part of the input: We just test for every subset of U with cardinality equal to K, whether it is a feasible solution.

Discrete scenario sets are very powerful, since there is in general no dependency between the different cost functions. To restrict the possible values of these cost functions and model more realistic scenario sets, interval scenarios are often chosen.

4.3. Interval Scenarios

The interval scenario set $\mathcal{S}_I$ is defined implicitly by a set of cost functions with box constraints. More precisely, for every element $u \in U$ of an LCMin instance $(U, \mathcal{F}, c)$ we are given lower and upper bounds $\underline{c}(u)$ and $\overline{c}(u)$ on the cost with $0 \leq \underline{c}(u) \leq \overline{c}(u)$. The scenarios are then defined as follows: For each cost function $c : U \to \mathbb{N}$ with $c(u) \in [\underline{c}(u), \overline{c}(u)]$ there exists a scenario $S \in \mathcal{S}_I$ with $c^S(u) = c(u)$ for all $u \in U$ and for each $S \in \mathcal{S}_I$ the scenario cost function c^S obeys the bounds $\underline{c}$ and $\overline{c}$.

The set $\mathcal{S}_I$ contains several irrelevant scenarios and we will show that it suffices to consider a special subset, the so-called *extreme scenarios*. A scenario S is *extreme* if $c^S(u) \in \{\underline{c}(u), \overline{c}(u)\}$ for all $u \in U$. We introduce a short notation for extreme scenarios: For a subset $A \subseteq U$, we

define the extreme scenario $S(A)$ as the scenario with the cost function

$$c^{S(A)}(u) = \begin{cases} \underline{c}(u) & \text{if } u \in A \\ \overline{c}(u) & \text{otherwise.} \end{cases}$$

As a consequence of the following lemma it suffices to focus on the set of extreme scenarios instead of considering all scenarios $S \in \mathcal{S}_I$ for an exSub-RR LCMin instance.

Lemma 4.3.1. *Let $(U, \mathcal{F}, \mathcal{S}_I)$ be an exSub-RR LCMin instance with interval scenarios $\mathcal{S}_I$, $S \in \mathcal{S}_I$ be a scenario and $F \in \mathcal{F}^S$ be a scenario feasible solution. Then $\mathcal{F}^{S(F)} \subseteq \mathcal{F}^S$.*

Proof. Let S be a scenario and $F_1, F_2 \in \mathcal{F}$. Then

$$c^S(F_1) - c^S(F_2) \geq c^{S(F_1)}(F_1) - c^{S(F_1)}(F_2).$$

If furthermore $F \in \mathcal{F}^S$ and $F' \in \mathcal{F}^{S(F)}$, i.e., $c^{S(F)}(F') \leq c^{S(F)}(\bar{F})$ for all $\bar{F} \in \mathcal{F}$, we obtain $c^S(F) - c^S(F') \geq c^{S(F)}(F) - c^{S(F)}(F') \geq 0$ and thus $F' \in \mathcal{F}^S$. □

Hence, for any scenario $S \in \mathcal{S}_I$, the condition imposed by S on a solution to the exSub-RR LCMin problem is weaker than the condition imposed by the extreme scenario $S(F)$ where $F \in \mathcal{F}^S$. A set $U' \subseteq U$ is a feasible solution if and only if it contains for every extreme scenario a feasible solution. We therefore restrict our consideration to extreme scenarios in the remaining section.

A special type of elements that will show useful in the analysis of exSub-RR LCMin problems, are elements that must be part of any feasible solution.

Definition 4.3.2. Let $(U, \mathcal{F}, \mathcal{S}_I)$ be an instance of an exSub-RR LCMin problem. An element $u \in U$ is called *necessary* if there exists a scenario $S \in \mathcal{S}_I$ such that $u \in F$ for every $F \in \mathcal{F}^S$. We call S a *witness* of u.

For a given scenario S, we can test if S is a witness for u by increasing $c^S(u)$ by one unit and computing an optimal solution with respect to this new cost function. Then the scenario S is a witness for u if and only if the value of an optimal solution w.r.t. the modified cost is higher than the value of an optimal solution w.r.t. the cost c^S.

In general, an optimal solution of an exSub-RR LCMin problem also contains non-necessary elements. Yet, in a large class of such problems with interval scenarios, the optimal solution is unique and just consists of the set of necessary elements. More precisely, we will show that this is the case if the cost value of every element $u \in U$ may suffer deviation, i.e., $\underline{c}(u) < \overline{c}(u)$ for all $u \in U$. Interval scenarios that satisfy this condition are called *strictly deviating*. We start by focusing on a special subset of scenarios and then derive a close relation between necessary elements and these scenarios.

Definition 4.3.3. Let $F \in \mathcal{F}$. We call $S(F)$ *dominant* if $\mathcal{F}^{S(F)} = \{F\}$.

Karaşan et al. [68] already use this concept to show that any optimal path in a robust regret shortest path problem with interval scenarios is a dominant path. The following lemma provides a key result for the role of dominant scenarios in a strictly deviating interval scenario set for exSub-RR LCMin problems.

Lemma 4.3.4. *Let $(U, \mathcal{F}, \mathcal{S}_I)$ be an exSub-RR LCMin instance with strictly deviating interval scenarios and let U^* be an optimal solution. Then $u \in U^*$ if and only if there exists a dominant witness of u.*

Proof. Let $U^* \subseteq U$ be a feasible solution with a minimum number of elements to a given exSub-RR LCMin instance $(U, \mathcal{F}, \mathcal{S}_I)$. We assume that there is an element $u' \in U^*$, but there exists no dominant witness $S(F')$ of u' with $F' \in \mathcal{F}$. Let us consider the subset $U' = U^* \backslash \{u'\}$. Since U' is no feasible solution, there exists a scenario $S \in \mathcal{S}_I$ in which all solutions $F \in \mathcal{F}^S$ with $F \subseteq U^*$ contain u'. Let $F^i \in \mathcal{F}^S$ and $i = 1$. Due to our assumption $S(F^i)$ is not dominant. Hence, there exists a solution $F^{i+1} \in \mathcal{F}^{S(F^i)} \backslash \{F^i\}$. If $S(F^{i+1})$ is not dominant, we set $i = i + 1$ and choose $F^{i+1} \in \mathcal{F}^{S(F^i)} \backslash \{F^i\}$. We repeat this scheme until $\{F^i\} = \mathcal{F}^{S(F^i)}$ for some $i_{\max} \in \mathbb{N}$. Since $\underline{c}(u) + 1 \leq \overline{c}(u)$ and $0 \leq c^{S(F^i)}(F^i) + 1 \leq c^{S(F^{i-1})}(F^{i-1})$ for $1 \leq i \leq i_{\max}$, the routine stops. Due to the construction of the solutions F^i, $i = 1, \ldots, i_{\max}$, $\mathcal{F}^{S(F^i)} \subseteq \mathcal{F}^{S(F^{i-1})}$ with $S(F^0) = S$ as shown in Lemma 4.3.1. Thus, $F^{i_{\max}} \in \mathcal{F}^S$, $F^{i_{\max}} \subseteq U^*$, and therefore $u' \in F^{i_{\max}}$. This is a contradiction to the assumption that there is no dominant witness of u'. □

As a direct consequence of Lemma 4.3.4 we get the following results.

Theorem 4.3.5. *An exSub-RR LCMin problem with strictly deviating interval scenarios has a unique optimal solution consisting of all necessary elements.*

The above characterization allows us to derive conclusions about the complexity status of the decision version of an exSub-RR LCMin problem.

Corollary 4.3.6. *Consider an LCMin problem which lies in* **P**. *Then the decision version of the corresponding exSub-RR LCMin problem with strictly deviating interval scenarios is in* **coNP**.

Proof. In the decision version of an exSub-RR LCMin problem $(U, \mathcal{F}, \mathcal{S}_I)$ we are given an integer K, and the task to decide whether there is a solution with at most K elements. Hence, for this problem to be in **coNP**, we have to guarantee that there is a polynomial certificate to verify that for a given no-instance, every feasible solution needs indeed $K + 1$ or more elements.

We assume that we are dealing with a no-instance. Let U^* be an optimal solution to the exSub-RR LCMin problem $(U, \mathcal{F}, \mathcal{S}_I)$, which needs at least $K + 1$ elements. By Lemma 4.3.4 for every element $u \in U^*$ there is a dominant witness $S(F)$ with $F \in \mathcal{F}$ and $u \in F$. Hence, if I is a no-instance, there exists a set of dominant scenarios $\{S(F_1), \ldots, S(F_\ell)\}$ with $\ell \leq K + 1$ and $|\cup_{i=1}^{\ell} F_i| \geq K + 1$. If the dominance of a scenario $S(F)$ can be tested in polynomial time this can be done in polynomial time for all scenarios $S(F_1), \ldots, S(F_\ell)$. In that case, we obtain a polynomial certificate for a no-instance.

A scenario $S(F)$ is dominant if F is the unique minimizer of $\min_{F' \in \mathcal{F}} c^{S(F)}(F')$. This can be tested as follows: for each $f \in F$ we define a cost function $c^f : U \to \mathbb{N}$ with

$$c^f(u) = \begin{cases} c^{S(F)}(u) & \text{if } u \neq f \\ c^{S(F)}(u) + 1 & \text{if } u = f. \end{cases}$$

One can easily check that F is the unique minimizer of $\min_{F' \in \mathcal{F}} c^{S(F')}(F)$ if and only if $c^{S(F)}(F) < \min_{F' \in \mathcal{F}} c^f(F')$ for each $f \in F$. Since the LCMin problem is by assumption in **P**, this can be done in polynomial time. In conclusion, a set of at most $K+1$ dominant scenarios $S(F_1), \ldots, S(F_\ell)$ with $|\cup_{i=1}^{\ell} F_i| \geq K + 1$ is a polynomial certificate for a no-instance. □

Corollary 4.3.7. *The exSub-RR shortest path problem, the exSub-RR minimum spanning tree problem and exSub-RR the minimum (s,t)-cut problem with strictly deviating interval scenarios are in* **coNP**.

Before we show that certain problems are solvable in polynomial time, we focus on a statement about the approximability of exSub-RR LCMin problems and show that a large class of these problems are hard to approximate. We use the following notion of *duplication-condition* to characterize this problem class.

Definition 4.3.8 (Duplication-Condition)**.** A class of LCMin problems satisfies the *duplication-condition* if for every instance $(U,\mathcal{F},c)$ of this class and for every element $u \in U$, there is another LCMin instance $(U' = U \cup \{u'\}, \mathcal{F}', c')$ in the same class such that $\mathcal{F}' = \mathcal{F} \cup \{F' \subseteq U' \mid u' \in F',\, u \notin F',\, F' \backslash \{u'\} \cup \{u\} \in \mathcal{F}\}$. The new element u' is called a *duplication* of u.

The shortest path problem, the minimum spanning tree problem and the minimum (s,t)-cut problem fulfill the duplication-condition by adding a parallel arc for any arc in the first two cases and dividing an arc into two consecutive arcs in the last case. The vertex cover problem, on the other hand, does not satisfy the duplication-condition.

Theorem 4.3.9. *Let $\mathcal{C}$ be a class of LCMin problems $(U,\mathcal{F},c)$ that satisfy the duplication-condition and such that it is in general* **NP***-hard to decide for a given element $u \in U$ whether u is necessary in the exSub-RR version of a problem in $\mathcal{C}$ with strictly deviating interval scenarios. Then there is no $|U|^{(1-\varepsilon)}$-approximation algorithm, $\varepsilon > 0$, for the exSub-RR version with strictly deviating interval scenarios of the problems in $\mathcal{C}$, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. Let us consider a class $\mathcal{C}$ of LCMin problems as described in the theorem. By contradiction assume that there exists an approximation algorithm ALG for the exSub-RR version of a problem in $\mathcal{C}$ with strictly deviating interval scenarios with an approximation factor of $|U|^{(1-\frac{1}{\ell})}$ for some $\ell \in \mathbb{N}$. Let $I = (U,\mathcal{F},c)$ be an instance of $\mathcal{C}$, let $\overline{u} \in U$ and $\mathcal{S}_I$ be an interval scenario set. We consider an instance $I' = (U',\mathcal{F}',c')$ which corresponds to I with $|U|^\ell - |U|$ duplications of $\overline{u}$ and the scenario set $\mathcal{S}'_I$ which extends the scenario $\mathcal{S}_I$ by assigning the same upper and lower bounds of $\overline{u}$ to any duplication of $\overline{u}$.

One can easily observe that because of the properties of element duplication, $\overline{u}$ is part of the (unique) optimal solution of the instance I if and only if $\overline{u}$ or any duplication of $\overline{u}$ is part of the optimal solution of I'. Furthermore, if any duplication of $\overline{u}$ or the element $\overline{u}$ itself is contained in the optimal solution I', then all of these elements are in the optimal solution.

Hence, if the element $\overline{u}$ is not part of the optimal solution for instance I, the optimal solution for the instance I' contains at most $|U|-1$ elements. In that case, the cardinality of U_{ALG} returned by the algorithm is bounded by

$$|U'|^{\frac{\ell-1}{\ell}}(|U|-1) = |U|^{\ell-1}(|U|-1) = |U|^\ell - |U|^{\ell-1}.$$

Since the number of duplications of $\overline{u}$ exceeds this bound for $\ell \geq 3$ and $|U| > 1$, not all of these elements can be contained in the solution U_{ALG}. On the other hand, if all of the duplications of $\overline{u}$ and $\overline{u}$ itself are in U_{ALG}, then the optimal solution of I must contain $\overline{u}$, since otherwise ALG would not satisfy the assumed approximation guaranty. Hence, the existence of the algorithm ALG would allow to find an optimal solution of every problem in $\mathcal{C}$ in polynomial time. This is not possible, unless $\mathbf{P} = \mathbf{NP}$. □

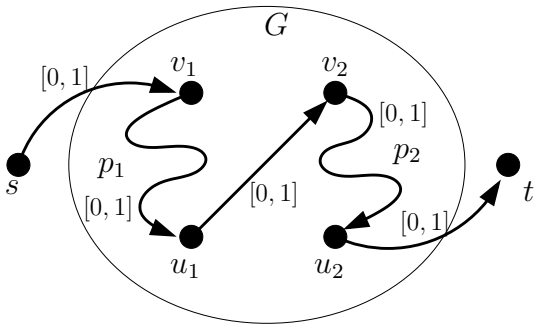


Figure 4.2.: Every dominant path is a simple path.

Any algorithm just returning the given set U is already an approximation algorithm with a factor $|U|$. We will now consider the exSub-RR shortest path and the exSub-RR minimum (s,t)-cut problem and show that we can apply Theorem 4.3.9.

4.3.1. Shortest Path Problem

Let $G = (V, A)$ be a directed graph with a set of vertices V and a set of directed arcs A. Let $s, t \in V$ and $\underline{c}(a) \in \mathbb{N}$ and $\overline{c}(a) \in \mathbb{N}$ be lower and upper bounds on the cost for every arc $a \in A$. The exSub-RR shortest path problem asks for a subgraph G' of G with a minimum number of arcs such that G' contains for every scenario in $\mathcal{S}_I$ a shortest path of G.

Corollary 4.3.10. *There exists no efficient approximation algorithm with an approximation factor $|A|^{(1-\varepsilon)}$ for any $\varepsilon > 0$ for the exSub-RR shortest path problem with strictly deviating interval scenario sets, unless $\mathbf{P} = \mathbf{NP}$, where A denotes the set of arcs in the given shortest path instance.*

For the proof it suffices to show that it is already **NP**-hard to decide whether an arc is necessary and thus has a dominant witness (Lemma 4.3.4 and Theorem 4.3.9). This was stated in [68] without proof. For completeness, we prove the **NP**-hardness in Theorem 4.3.11 by a reduction from two vertex disjoint path.

Theorem 4.3.11. *The decision problem whether an arc is necessary for the class of exSub-RR shortest path problems with strictly deviating interval scenarios is strongly* **NP***-hard.*

Proof. We show a reduction from the two vertex disjoint path problem. Let I be a two vertex disjoint path instance composed of a directed graph $G = (V, A)$ and four vertices $v_1, u_1, v_2, u_2 \in V$. The task is to decide whether two vertex disjoint paths p_1 and p_2 exist such that p_1 connects v_1 and u_1 and p_2 connects v_2 and u_2. We transform this instance to an instance I' of the exSub-RR shortest path problem by adding two vertex s and t and the arcs (s, v_1), (u_1, v_2) and (u_2, t) to G. The lower and upper bounds on each arc are set to 0 and 1 (see Figure 4.2). The size of this new graph G' and the value of the cost bounds are polynomial in the size of the instance I.

We will show that there exists a (v_1, u_1)-path p_1 and a disjoint (v_2, u_2)-path p_2 in G if and only if the arc (u_1, v_2) is necessary.

If the arc (u_1, v_2) is necessary, there exists a dominant scenario $S(p)$ with $(u_1, v_2) \in p$ (Lemma 4.3.4). Since $S(p)$ is dominant, p is a simple path. Hence, the subpaths of p connecting v_1 with u_1 and v_2 with u_2 are vertex disjoint and form a feasible solution to I.

On the other hand, let p_1 and p_2 be two vertex disjoint paths in G from v_1 to u_1 and from v_2 to u_2, respectively. The path $p = (s, v_1) \cup p_1 \cup (u_1, v_2) \cup p_2 \cup (u_2, t)$ in G' is a simple path with $c^{S(p)}(p) = 0$. The cost of any other (s,t)-path in G' w.r.t. the cost function $c^{S(p)}$ is at least 1. Thus, $S(p)$ is dominant by definition and since p traverses (u_1, v_2), the arc (u_1, v_2) is necessary. □

For undirected graphs the two vertex disjoint path problem can be solved in polynomial time [90]. Hence, we do not know if the exSub-RR shortest path problem remains **NP**-hard on this graph class.

4.3.2. Minimum (s,t)-Cut Problem

Let $G = (V, A)$ be a directed graph with a vertex set V and a directed arc set A and $s, t \in V$. A subset of arcs $\delta^+(X)$ with $X \subseteq V \backslash \{t\}$, $s \in X$, $\delta^+(X) = \{(u, v) \in A \mid u \in X, v \in V \backslash X\}$ is called an (s,t)-*cut*. For a given cost function $c : A \to \mathbb{N}$ the minimum (s,t)-cut problem is to find an (s,t)-cut $\delta^+(X)$ with minimum cost $c(\delta^+(X))$. For general graph classes the exSub-RR version of the minimum (s,t)-cut problem with interval scenarios is strongly **NP**-hard.

Theorem 4.3.12. *The exSub-RR minimum (s,t)-cut problem with strictly deviating interval scenarios is not approximable within a factor of $|A|^{(1-\varepsilon)}$ for any $\varepsilon > 0$, unless $\mathbf{P} = \mathbf{NP}$, where A denotes the set of arcs in the given instance.*

Proof. We show again a reduction from the two vertex disjoint path problem to prove that it is **NP**-hard to decide whether a given arc is necessary. In combination with Theorem 4.3.9, this results in the statement of the theorem.

Let I be an instance of the two vertex disjoint path problem given by a directed graph $G = (V, A)$ and two vertex pairs (v_1, u_1) and (v_2, u_2). In order to construct an instance I' of the exSub-RR minimum (s,t)-cut problem, we add the arc $\overline{a} = (u_1, v_2)$ to the graph and set $s = v_1$ and $t = u_2$. Furthermore, we assign 0 as the lower bound on the cost of all arcs $a \in A$, 2 as the upper bound of all arcs $a \in A$ and set the bounds for $\overline{a}$ to $\underline{c}(\overline{a}) = 1$ and $\overline{c}(\overline{a}) = 2$. The reduction is the following: Arc $\overline{a}$ is necessary in I' if and only if there exist two disjoint paths connecting v_1 with u_1 and v_2 with u_2.

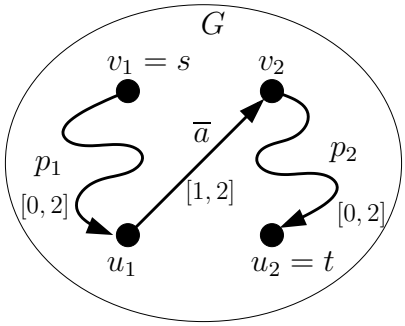


Figure 4.3.: All arcs except $\overline{a}$ have lower cost bounds 0 and upper cost bounds 2.

Let $\overline{a}$ be a necessary arc. Since $\underline{c}(a) < \overline{c}(a)$ for all $a \in A \cup \{\overline{a}\}$, there exists a dominant scenario S in which the unique minimum (s,t)-cut $\delta^+(X)$ contains the arc $\overline{a}$ (Lemma 4.5.6). Therefore, $v_1, u_1 \in X$ and $u_2, v_2 \in V \backslash X$. Let f be a maximum (s,t)-flow in G' where the upper capacities on the arcs are given by the cost of the scenario S. Hence, $f(\overline{a}) \geq 1$ and

since $\delta^+(X)$ is a minimum (s,t)-cut, no backward arc is allowed to carry any flow. Therefore, there is a path from v_1 to u_1 just using arcs of $A(X)$ and a path connecting v_2 and u_2 just using arcs of $A(V\backslash X)$.

Conversely, let there be two simple paths p_1 and p_2 where p_1 connects v_1 with u_1 and p_2 connects v_2 with u_2. We combine these paths via the arc $\overline{a}$ to an (s,t)-path and define a cost function for $S \in \mathcal{S}_I$ in the following way:

$$c^S(a) = \begin{cases} 2 & \text{if } a \in p_1 \cup p_2, \\ 1 & \text{if } a = \overline{a}, \\ 0 & \text{otherwise.} \end{cases}$$

Every minimum (s,t)-cut according to this scenario has cost 1 and therefore contains the arc $\overline{a}$. Hence, $\overline{a}$ is by definition necessary. Applying Theorem 4.3.9 we obtain the lower bound on the best possible approximation factor. □

In the case of outer-planar graphs the exSub-RR minimum (s,t)-cut problem is solvable in polynomial time, since there are at most n^2 simple (s,t)-cuts in a given graph. The dominance of each scenario induced by such a cut can be tested efficiently, as described in the proof of Corollary 4.3.6.

4.4. Γ-scenarios

Recall that the Γ-scenario set is defined as follows: Let $\underline{c}(u)$ and $\overline{c}(u)$ be lower and upper bounds on the scenario cost for each element $u \in U$ of the given LCMin instance $(U, \mathcal{F}, c)$ and let Γ be some integer. The cost function c^S of each scenario $S \in \mathcal{S}_\Gamma$ obeys these cost bounds and there are at most Γ values deviating from the lower bound, i.e., $|\{u \in U \mid c^S(u) > \underline{c}(u)\}| \leq \Gamma$. As in the interval case it suffices to consider cost functions only taking values equal to $\underline{c}(u)$ or $\overline{c}(u)$ for all $u \in U$.

The complexity of an exSub-RR LCMin problem is, as in the k-Dist-RR version, closely related to the max-scenario problem (Definition A.0.1) and the add-condition (Definition 2.4.1). The max-scenario problem asks for a scenario which induces the maximum cost on a minimum solution of the LCMin instance w.r.t the scenario cost function. The add-condition allows to add a feasible solution to a given instance by extending the set of feasible solutions by just this one solution.

Theorem 4.4.1. *Let $\mathcal{C}$ be a class of LCMin problems $(U, \mathcal{F}, c)$ that satisfy the add-condition and such that the max-scenario problem on $\mathcal{C}$ is strongly* **NP***-hard. Then the exSub-RR LCMin version of the problems in $\mathcal{C}$ with Γ-scenarios is strongly* **NP***-hard and not approximable within a factor $|U|^{(1-\varepsilon)}$ for any $\varepsilon > 0$, unless* $\mathbf{P} = \mathbf{NP}$.

Proof. Let $I = (U, \mathcal{F}, c)$ be an instance of the class $\mathcal{C}$ with the properties stated in the theorem. Let furthermore $\mathcal{S}_\Gamma$ be a set of Γ-scenarios defined by the lower and upper bounds $\underline{c}(u)$ and $\overline{c}(u)$ for each $u \in U$, such that the corresponding max-scenario version of I is strongly **NP**-hard. I.e., it is **NP**-hard to decide for a given threshold K whether there is a scenario $S \in \mathcal{S}_\Gamma$ with $\text{profit}(S) = \min_{F \in \mathcal{F}} c^S(F) \geq K$. Finally, we assume by contradiction that there exists an algorithm ALG allowing to approximate the exSub-LCMin problem with an approximation factor $|U|^{(1-\frac{1}{\ell})}$ for some $\ell \in \mathbb{N}$.

We define the following exSub-LCMin instance $I' = (U', \mathcal{F}', c')$ as a modification of I by adding a set $W = \{w_1, \dots, w_p\}$ of $p = (|U|+1)^\ell - |U|$ elements to U and the set of feasible solutions $\mathcal{F}$ according to the add-condition. Furthermore, we define the lower and upper bounds for the Γ-scenario set $\mathcal{S}'_\Gamma$ on I' by

$$\underline{c}'(u) = \begin{cases} \underline{c}(u) & \text{for all } u \in U, \\ K-1 & u = w_1, \\ 0 & \text{otherwise,} \end{cases} \quad \text{and } \overline{c}'(u) = \begin{cases} \overline{c}(u) & \text{for all } u \in U, \\ K-1 & u = w_1, \\ 0 & \text{otherwise.} \end{cases}$$

The new instance I' contains $(|U|+1)^\ell$ elements. If I is a no-instance, then for every scenario $S \in \mathcal{S}_\Gamma$ there exists a solution $F \in \mathcal{F}$ with $c^S(F) \leq K-1$. Hence, the size of an optimal solution in I' is bounded by $|U|$. Thus, ALG returns at most $(|U|+1)^{\ell-1}$ elements. Since $p > (|U|+1)^{\ell-1}$, not all elements of W are part of the solution computed by the algorithm. On the other hand, if I is a yes-instance, there exists a scenario $S' \in \mathcal{S}_\Gamma$ with $c^{S'}(F) \geq K$ for all $F \in \mathcal{F}$. Thus, $\mathcal{F}^{S'} = \{W\}$ and any feasible solution contains W. Hence, we can decide the max-scenario instance I with ALG. □

Since the maximum scenario version of the shortest path and the minimum (s,t)-cut problem is strongly **NP**-hard for Γ-scenarios (Theorem A.1.1 and Theorem A.2.1) and both problem classes satisfy the add-condition, we can apply Theorem 4.4.1.

Corollary 4.4.2. *The exSub-RR version of the shortest path and the minimum (s,t)-cut problem with Γ-scenarios is strongly* **NP***-hard and not approximable with a factor better than $|A|^{(1-\varepsilon)}$ for any $\varepsilon > 0$, unless* $\mathbf{P} = \mathbf{NP}$*, where A denotes the set of arcs of the corresponding graph.*

Finally, we consider the weighted disjoint hitting set problem, whose exSub-RR version with Γ-scenarios can be solved efficiently.

4.4.1. Weighted Disjoint Hitting Set Problem

The weighted disjoint hitting set (WDHS) problem is a simple extension of the minimum element problem. In this problem a set $U = \{u_1, \dots, u_n\}$, a set of d pairwise disjoint subsets $\mathcal{M} = \{M_1, \dots, M_d\}$ of U and a cost function $c : U \to \mathbb{N}$ are given. A set $F \subseteq U$ is a feasible solution if F contains from every set $M \in \mathcal{M}$ exactly one element. The WDHS problem is to find a feasible solution with minimum cost. We will prove that the exact subset recoverable robust disjoint hitting set problem with Γ-scenarios is in **P**.

Theorem 4.4.3. *The exSub-RR WDHS problem with Γ-scenarios can be solved in polynomial time.*

Proof. Let U be a set of n elements, $M_1, \dots, M_d$ be d pairwise disjoint subsets of U, $\underline{c}(u)$ and $\overline{c}(u)$ be lower and upper bounds on all elements $u \in U$ and $\Gamma \in \mathbb{N}$. First note that U^* is an optimal solution of this given exSub-RR WDHS instance if and only if for $i = 1, \dots, d$ the sets $U_i^* = U^* \cap M_i$ form optimal solutions for the exSub-RR WDHS instance restricted to U_i. Hence, we can confine ourselves to the case of $d = 1$.

We start by ordering the elements of U, such that $\underline{c}(u_i) \leq \underline{c}(u_j)$ for $i < j$ and $\overline{c}(u_i) \leq \overline{c}(u_j)$ if $\underline{c}(u_i) = \underline{c}(u_j)$ and $i < j$. Let ℓ be the first index with $\min_{i=1,\dots,\ell-1} \overline{c}(u_i) \leq \underline{c}(u_\ell)$ if such an

index exists and $\ell = \infty$ otherwise. Then $U^* = \{u_1, \ldots, u_k\}$ with $k = \min\{\ell - 1, \Gamma + 1\}$ is an optimal solution: Let us first assume $|U^*| = \Gamma + 1$. In that case the feasibility is obvious, since any scenario can increase at most Γ values. Thus, one of the elements in U^* is an optimal solution. Assume that u' can be deleted from U^*. Increasing all other elements in U^*, leads to u' being an optimal solution. Due to the definition of ℓ, we have $\overline{c}(u) > \underline{c}(u')$ for all $u \in U^*$. Therefore, none of the other elements in U^* can be an optimal solution, a contradiction.

A similar argumentation works for the case $|U^*| = \ell - 1$: If a scenario increases the cost of all elements $u_1, \ldots, u_{\ell-1}$ there is still an optimal solution within this set. If the cost of one of these elements is not increased, than this element is an optimal solution. □

4.5. Matroids

Matroids form the special class of combinatorial optimization problems, where the set of feasible solutions is described by an independence system. They have several nice properties: given two independent sets there always exist two elements that can be exchanged to gain a new independent set; maximal independent sets of a matroid, so-called *bases*, have the same length; and the greedy-algorithm always computes a basis of minimum cost. We start with some basic definitions and results on matroids provided, for example, in the book "Combinatorial Optimization: Theory and Algorithms" by Korte and Vygen [77].

Definition 4.5.1. A *matroid* is a tuple $M = (E, \mathcal{I})$, where E is a finite set and $\mathcal{I} \subseteq 2^E$ is a non-empty collection of subsets of E, called *independent sets*, such that:

1. $\emptyset \in \mathcal{I}$;
2. if $I \in \mathcal{I}$ and $J \subseteq I$, then $J \in \mathcal{I}$;
3. if $I, J \in \mathcal{I}$ with $|I| > |J|$ then there exists an element $i \in I \setminus J$ with $J + i \in \mathcal{I}$.

Any subset of E that is not in $\mathcal{I}$ is called *dependent*, and minimal dependent sets are called *circuits*. A *basis* in $\mathcal{I}$, on the other hand, is a maximal independent set, i.e., adding any element leads to a dependent set. The cardinality of a basis in a matroid M is called the *rank* of the matroid. Special circuits are the so-called fundamental circuits constructed by a basis B and an element not contained in the basis.

Definition 4.5.2. Let $M = (E, \mathcal{I})$ be a matroid and B be a basis in M. For any element $e \in E \backslash B$ the unique circuit in $B \cup \{e\}$ is called the *fundamental circuit* induced by e. We denote this circuit with $C_e(B)$.

Another basic concept in matroid theory is duality.

Definition 4.5.3. Let $M = (E, \mathcal{I})$ be a matroid. We define the *dual matroid* of M by $M^* = (E, \mathcal{I}^*)$, where

$$\mathcal{I}^* = \{I \subseteq E \mid \text{there is a base } B \text{ of } (E, \mathcal{F}) \text{ such that } I \cap B = \emptyset\}.$$

The circuits of the dual matroid M^* are called *cocircuits* of M. Similar to fundamental circuits we can define fundamental cocircuits.

Definition 4.5.4. Let $M = (E, \mathcal{I})$ be a matroid. For a given basis B of M and an element $e \in B$, we define the unique cocircuit in $E \backslash B \cup \{e\}$ as the *fundamental cocircuit* induced by e. We denote this cocircuit with $K_e(B)$.

The probably most basic optimization problem concerning matroids is the *minimum weight basis (MWB) problem*:

Given: A matroid $M = (E, \mathcal{I})$ and a weight function $c : E \to \mathbb{Z}$.

Task: Find a basis B of M with minimum cost $c(B)$.

Since in the context of matroids, the notion of *weight* is more commonly used than *cost*, we use these two terms interchangeably in this section. If not further specified, we assume that the independent sets of a given matroid are implicitly given by an independence oracle, i.e., for a given set $I \subseteq E$, we can call an oracle to decide whether $I \in \mathcal{I}$.

Before we investigate the exSub-recoverable robust minimum weight basis problem of a matroids, we need two more properties of matroids:

1. Let $M = (E, \mathcal{I})$ be a matroid, C be a circuit of M and D be a cocircuit of M, then $|C \cap M| \neq 1$.
2. Let $M = (E, \mathcal{I})$ be a matroid and $c : E \to \mathbb{Z}$ be a cost function. Then the following statements are equivalent:
   a) The basis B^* is a minimum weight basis.
   b) For every element $e \in B^*$ and any other element $e' \in K_e(B^*)$ it holds: $c(e) \leq c(e')$.
   c) For every element $e \in E \backslash B^*$ and any other element $e' \in C_e(B^*)$ it holds: $c(e) \geq c(e')$.

We will now consider the exact subset recoverable robust minimum weight basis problem of a matroids with different types of scenario sets.

4.5.1. Discrete Scenarios

We start investigating again discrete scenario sets, where each scenario defines a cost function. Since any basis in a matroid has the same cardinality, the exSub-RR MWB problem with one discrete scenario S corresponds to the MWB problem with c^S as cost function. As we will show in the following theorem, for two discrete scenarios the problem becomes more difficult but can still be solved in polynomial time by using matroid intersection. The *matroid intersection problem* is defined in the following way:

Given: Two matroids $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$ on the same set E and a cost function $c : E \to \mathbb{Z}$.

Task: Find a set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ with maximum cost $c(I)$.

The matroid intersection problem can be solved in polynomial time with the algorithm of Lawler [80] or Frank [48].

Theorem 4.5.5. *The exSub-RR minimum weight basis problem of matroids can be solved in polynomial time via matroid intersection for two discrete scenarios.*

Proof. Let $(E, \mathcal{I})$ be a matroid as described above, S_1 and S_2 be two scenarios defining cost functions $c^{S_1} : E \to \mathbb{N}$ and $c^{S_2} : E \to \mathbb{N}$. According to these cost functions, we define the following two matroids on E: Let $\mathcal{B}^{S_i}$ be the set of minimum bases of the MWB instance $(E, \mathcal{I}, c^{S_i})$, $i = 1, 2$. We define an independent system

$$\mathcal{I}(S_i) = \{F \subseteq E \mid \text{there exists a } B \in \mathcal{B}^{S_i} \text{ with } F \subseteq B\},$$

which determines the matroid $M^{S_i} = (E, \mathcal{I}(S_i))$, as shown in [77]. For a given set $F \subseteq E$ we can test in polynomial time, whether $F \in \mathcal{I}(S_i)$, by the following method: Compute a minimum basis B_i as reference basis for $(E, \mathcal{I}, c^{S_i})$. Furthermore, define the cost function $c_F^{S_i} : E \to \mathbb{Z}$ as

$$c_F^{S_i}(e) = \begin{cases} -1 & \text{if } e \in F \\ c_F^{S_i}(e) & \text{otherwise} \end{cases}$$

and compute a minimum basis B for the problem $(E, \mathcal{I}, c_F^{S_i})$. Since

$$c^{S_i}(B) = c_F^{S_i}(B) + |F \cap B| + c^{S_i}(F \cap B)$$

and $c_F^{S_i}(e) \leq c_F^{S_i}(e')$ for all $e \in F$ and $e' \in E \backslash F$, $F \in \mathcal{I}(S_i)$ if and only if $F \subseteq B$ and $c^{S_i}(B) = c^{S_i}(B_i)$.

Using this independence oracle we can compute an independent set E' in M^{S_1} and M^{S_2} of maximum size in polynomial time. Extending E' to a basis in M^{S_1} and M^{S_2} via elements E_1 and E_2 respectively, $E' \cup E_1 \cup E_2$ is an optimal solution of the given exSub-RR minimum weight basis instance. □

For more scenarios it is rather unlikely that the exSub-RR MWB problem is solvable in polynomial time, since it seems similar to the intersection problem with three matroids. But, we did not succeed to show its **NP**-hardness.

4.5.2. Interval Scenarios

We will show that the exSub-RR MWB problem with interval scenarios is in **P** and start with a characterization of necessary elements. Recall that we denote with $C_e(B)$ the *fundamental circuit* induced by e for a basis B and some element $e \in E \backslash B$ and with $K_e(B)$ the *fundamental cocircuit* induced by e for some element $e \in B$. The next lemma states how to decide whether an element is necessary for a given exSub-RR MWB instance.

Lemma 4.5.6. *Let $(E, \mathcal{I})$ be a matroid and $(E, \mathcal{I}, \mathcal{S}_I)$ be the corresponding exSub-RR version with interval scenarios defined by lower and upper cost bounds $\underline{c}(e)$ and $\overline{c}(e)$ for each element $e \in E$. For an element $e \in E$ we define with $S(e)$ the scenario that assigns cost $\overline{c}(e')$ to all $e' \in E \backslash \{e\}$ and $\underline{c}(e)$ to e. Then $e \in E$ with $\underline{c}(e) < \overline{c}(e)$ is necessary if and only if $S(e)$ is a witness of e.*

Proof. Let $(E, \mathcal{I}, \mathcal{S}_I)$ be an exSub-RR MWB instance with interval scenarios as described in the lemma. Let further $\overline{e} \in E$ be an element with $\underline{c}(\overline{e}) < \overline{c}(\overline{e})$. If $S(\overline{e})$ is a witness of $\overline{e}$, $\overline{e}$ is a necessary element by Definition 4.3.2. Conversely, let $\overline{e}$ be a necessary element and $S^* \in \mathcal{S}_I$ be a witness of $\overline{e}$. Let us further assume that there exists a basis $B' \in \mathcal{B}^{S(\overline{e})}$ with $\overline{e} \notin B'$. For every scenario $S \in \mathcal{S}_I$ and every element $e \in C_{\overline{e}}(B')$,

$$c^S(\overline{e}) \geq c^{S(\overline{e})}(\overline{e}) = \underline{c}(\overline{e}) \geq \overline{c}(e) = c^{S(\overline{e})}(e) \geq c^S(e),$$

since B' is a minimum weight basis according to $c^{S(\overline{e})}$. Let now $B \in \mathcal{B}^{S^*}$ and $e^* \in K_{\overline{e}}(B) \cap C_{\overline{e}}(B') \backslash \{\overline{e}\}$. Since B is a minimum basis according to the cost function c^{S^*}, $c^{S^*}(\overline{e}) \leq c^{S^*}(e^*)$ and due to our assumption $c^{S^*}(\overline{e}) = c^{S^*}(e^*)$. Therefore, $\bar{B} = B \backslash \{\overline{e}\} \cup \{e^*\}$ is also a minimum basis according to c^{S^*}. This contradicts the assumption that S^* is a witness of $\overline{e}$. □

As a consequence, we can decide for every element e with deviating cost whether this element is necessary by solving two MWB instances: first we compute a minimum weight basis for $c^{S(e)}$ and in a second step a minimum weight basis for a modification of $c^{S(e)}$ by increasing the cost of e by one unit. If both bases have the same cost w.r.t. $c^{S(e)}$, e is not necessary. Yet, in general these elements do not suffice to form an optimal solution. The next lemma indicates which elements to add.

Lemma 4.5.7. *Let $(E,\mathcal{I})$ be a matroid and $(E,\mathcal{I},\mathcal{S}_I)$ a corresponding exSub-RR version with interval scenarios. Let $E_> = \{e \in E \mid \underline{c}(e) < \overline{c}(e)\}$ and $N_> \subseteq E_>$ be the set of all necessary elements in $E_>$. If $\bar{B} \in \mathcal{B}^{S(E)}$, then $\bar{B} \cup N_>$ is a feasible solution of $(E,\mathcal{I},\mathcal{S}_I)$.*

Proof. Let us assume by contradiction that there exists a scenario $S' \in \mathcal{S}_I$, such that every basis $B \in \mathcal{B}^{S'}$ contains at least one element e which is not in $\bar{B} \cup N_>$. Let $B_1 \in \mathcal{B}^{S'}$ such that $|B_1 \backslash \{\bar{B} \cup N_>\}|$ is minimal, and let $e_1 \in B_1 \backslash \{\bar{B} \cup N_>\}$. Furthermore, we denote with E_0 the set of all elements for which their lower bound equals their upper bound, i.e., $E_0 = \{e \in E \mid \underline{c}(e) = \overline{c}(e)\}$.

Case 1: $e_1 \in E_0 \backslash \bar{B}$. Since $e_1 \notin \bar{B}$, for every element $e' \in C_{e_1}(\bar{B})$,

$$c^{S'}(e_1) = c^{S(E)}(e_1) \geq \overline{c}(e') \geq c^{S'}(e').$$

Furthermore, there exists an element $\overline{e} \in C_{e_1}(\bar{B}) \cap K_{e_1}(B_1)$ with $\overline{e} \neq e_1$. Hence, $B' = B_1 \cup \{\overline{e}\} \backslash \{e_1\}$ is a minimum basis. This is a contradiction to the choice of B_1.

Case 2: $e_1 \in E_>$. Since $e_1 \notin N_>$, there exists a basis $B_2 \in \mathcal{B}^{S(e_2)}$ with $e_1 \notin B_2$. For every element $e' \in C_{e_1}(B_2)$ and every scenario $S \in \mathcal{S}_I$,

$$c^S(e_1) \geq \underline{c}(e_1) \geq \overline{c}(e') \geq c^S(e').$$

Let $e_3 \in C_{e_1}(B_2) \cap K_{e_1}(B_1)$ and $e_3 \neq e_1$. Then $B_3 = B_1 \cup \{e_3\} \backslash \{e_1\}$ is a minimum basis according to S' and $c^{S'}(e_3) = \overline{c}(e_3)$. If $e_3 \in N_> \cup \bar{B}$, this is a contradiction to the choice of B_1. Thus, B_3 also minimizes the number of elements not in $\bar{B} \cup N_>$. If $e_3 \in E_0 \backslash \bar{B}$, we can use the same arguments as in Case 1 this time considering B_3 to derive a contradiction. Therefore, it remains the case $e_3 \in E_> \backslash N_>$. We will use the same exchange argument as for e_1: Since $e_3 \notin N_>$, there exists a basis $B_4 \in \mathcal{B}^{S(e_3)}$ with $e_3 \notin B_4$. For every element $e' \in C_{e_3}(B_4)$ and every scenario $S \in \mathcal{S}_I$,

$$c^S(e_3) \geq \underline{c}(e_3) \geq \overline{c}(e') \geq c^S(e').$$

Let $e_5 \in C_{e_3}(B_4) \cap K_{e_3}(B_3)$ and $e_5 \neq e_3$. Since $c^{S'}(e_3) = \overline{c}(e_3) > \underline{c}(e_3) \geq c^{S'}(e_5)$ and $B_5 = B_3 \cup \{e_5\} \backslash \{e_3\}$ is a basis, we have a contradiction to the minimality of B_1. Thus, for every scenario $S \in \mathcal{S}_I$ there exists a basis $B \in \mathcal{F}^S$ with $B \subseteq N_> \cup \overline{B}$. □

These two lemmas lead to the following algorithm.

Algorithm 4.1 Exact Subset for MWB with interval scenarios.

Input : Matroid $(E,\mathcal{I})$ via an independence oracle, lower and upper bounds $\underline{c}(e)$ and $\overline{c}(e)$ for all $e \in E$.
Output : Set $\overline{E} \subseteq E$.

Set $E_> = \{e \in E \mid \underline{c}(e) < \overline{c}(e)\}$.
Determine the set $N_>$ of all necessary elements in $E_>$.
Compute a minimum weight basis B w.r.t. the cost function of $S(E)$ such that $|B \cap N_>|$ is maximum.
return $N_> \cup B$.

Theorem 4.5.8. *Algorithm 4.1 computes for any exSub-RR MWB problem with interval scenarios an optimal solution in polynomial time.*

Proof. For a given exSub-RR MWB instance $I = (E, \mathcal{I}, \mathcal{S}_I)$ with interval scenarios $\mathcal{S}_I$ the set $N_> \cup B$ defined as in the Algorithm 4.1 is a feasible solution according to Lemma 4.5.7. Since $S(E)$ is a feasible scenario, any feasible solution for I has to contain at least $|N_> \cup B|$ elements.

It remains to show that every step can be accomplished in polynomial time. Determining all necessary elements is achieved by solving at most $2|E|$ minimum weight basis problems for matroids. Constructing a minimum weight basis B with maximum elements from $N_>$ is solved by matroid intersection: Let $\mathcal{B}$ be the set of minimum bases of the MWB instance $(E, \mathcal{I}, c^{S(E)})$. We define $M_1 = (E, \mathcal{I}_1)$ with

$$\mathcal{I}_1 = \{F \subseteq E \mid \text{there exists } B \in \mathcal{B} \text{ with } F \subseteq B\}$$

as in the proof of Theorem 4.5.5. The second matroid $M_2 = (E, \mathcal{I}_2)$ on E is defined by the independent system

$$\mathcal{I}_2 = \{F \subseteq E \mid F \subseteq N_>\}.$$

Computing a maximum intersection B_0, this set can be extended to a minimum basis B in $\mathcal{B}$. This basis B satisfies the required properties. □

In the next section we focus on Γ-scenarios and will show that these problems are in general strongly **NP**-hard.

4.5.3. Γ-scenarios

The exSub-RR MWB with Γ-scenarios is more difficult to solve than its counterpart with interval scenarios. To this end, we consider the exSub-RR version of the minimum spanning tree (MST) problem with lower and upper bounds 0 and 1 and some $\Gamma \in \mathbb{N}$. An optimal solution of this instance with Γ-scenarios corresponds in most cases to a $(\Gamma+1)$-edge connected spanning subgraph of minimum size, as we will see in the following theorem.

Theorem 4.5.9. *For $\Gamma \in \mathbb{N}$, let $G = (V, E)$ be an undirected $(\Gamma+1)$-edge connected graph and let $\underline{c}(e) = 0$ and $\overline{c}(e) = 1$ be lower and upper cost bounds for each edge $e \in E$. Then a set $U \subseteq E$ is a solution to this exSub-RR minimum spanning tree problem with Γ-scenarios if and only if the graph (V, U) is $(\Gamma+1)$-edge connected.*

Proof. First note that since G is $(\Gamma+1)$-edge connected, the minimum weight basis for any scenario is 0. Hence, a set of edges $U \subseteq E$ is a feasible solution to the exSub-RR MST problem with Γ-scenarios if and only if for every scenario, there is a spanning tree with cost 0 in U. If U is not $(\Gamma+1)$-edge connected, then there exists a cut $\delta^+(X) \subseteq U$ in the graph (V, U) with at most Γ edges. Consider the scenario given by the cost function $c(e) = 1$ for $e \in \delta^+(X)$, and $c(e) = 0$ for $e \in E \setminus \delta^+(X)$. Since every spanning tree in U must contain an edge of $\delta^+(X)$, every spanning tree in U has cost at least 1, and hence, U is not a solution to the exSub-RR MST problem.

Conversely, if U is $(\Gamma+1)$-edge connected, then for every Γ-scenario $S \in \mathcal{S}_\Gamma$ every cut in (V, U) contains an edge e with $c^S(e) = 0$. Hence, the edges in U of cost 0 with respect to c^S, span all vertices. Thus, U contains a spanning tree of cost 0. □

In the setting described by Theorem 4.5.9, the exSub-RR MST problem with Γ-scenarios equals the minimum $(\Gamma+1)$-edge connected subgraph problem, which is a problem that is well-known to be **NP**-hard. In particular, for a given graph G one can easily observe that there is a Hamiltonian cycle in G if and only if the minimum 2-edge connected subgraph of G is a Hamiltonian cycle. Furthermore, also results about the approximation hardness of the minimum k-edge connected subgraph problem carry over to the exSub-RR MST problem with Γ-scenarios. The following result is due to Gabow et al. [51]: There exists a constant $c > 0$ such that for any fixed integer $k \geq 2$, no polynomial time algorithm approximates the minimum k-edge connected spanning subgraph problem on undirected multigraphs within a ratio $1 + \frac{c}{k}$, unless $\mathbf{P} = \mathbf{NP}$. Hence by Theorem 4.5.9 we can restate this result as well for the exSub-RR MST problem with Γ-scenarios as follows.

Corollary 4.5.10. *There exists a constant $c > 0$ such that for any fixed integer $\Gamma \geq 1$, no polynomial time algorithm approximates the exSub-RR MST problem with Γ-scenarios on undirected multigraphs within a ratio $1 + c/(\Gamma+1)$, unless $\mathbf{P} = \mathbf{NP}$.*

In the reduction the lower and upper bounds are fixed to 0 and 1. For this special case the following algorithm derives a $\Gamma+1$-approximation for general matroids and a 2-approximation for the minimum spanning tree problem.

Algorithm 4.2 Approximation for matroids

Input : Matroid $(E, \mathcal{I})$, lower and upper bounds $\underline{c}(e) = 0$ and $\overline{c}(e) = 1$ for all $e \in E$, $\Gamma \in \mathbb{N}$.
Output : Feasible solution $\overline{E}$.

Set $M_0 = (E, \mathcal{F})$ and $i = 0$.
while $i \leq \Gamma$ **do**
  Compute a basis B_i in M_i.
  Set $M_{i+1} = (E_{i+1}, \mathcal{I}_{i+1})$ with $E_{i+1} = E_i \backslash B_i$ and $\mathcal{F}_{i+1} = \{F \subseteq \mathcal{I}_i \mid F \subseteq E_{i+1}\}$ and $i = i+1$.
return $\overline{E} = \cup_{i=0}^{\Gamma} B_i$

Theorem 4.5.11. *Algorithm 4.2 computes a $(\Gamma+1)$-approximation for the exSub-RR MWB problem and a 2-approximation for the exSub-RR MST problem.*

Proof. First note that for $i = 0, \ldots, \Gamma$ the system $(E_i, \mathcal{I}_i)$ is a matroid. Hence, computing a basis B_i is well defined.

We now need to show that $\overline{E}$ is a feasible solution. Let us assume that there exists a scenario $S \in \mathcal{S}_\Gamma$, such that $\overline{E}$ does not contain an optimal solution. Let us choose B^* as a minimum weight basis of the instance $(E, \mathcal{I}, c^S)$ and $\bar{B}$ as a minimum weight basis of the instance $(\overline{E}, \overline{\mathcal{I}} = \{I \subseteq \overline{E} \mid I \in \mathcal{I}\}, c^S)$ such that B^* and $\bar{B}$ have a maximum number of common elements. W.l.o.g., $c^S(B^*) = 0$. Let $y \in B^*$ but $y \notin \bar{B}$. Such an element exists, since $c^S(\bar{B}) > c^S(B^*)$. If $y \in \overline{E}$, then $\bar{B} \cup \{y\} \backslash \{z\}$ is a minimum basis in $\overline{E}$ according to c^S with $z \in C_y(\bar{B}) \cap K_y(B^*)$, $z \neq y$. This is a contradiction to the choice of $\bar{B}$ and B^*. Hence, $y \notin \overline{E}$. Then $y \in E_i$ for $i = 0, \ldots, \Gamma$ and the fundamental cycle induced by y and a basis B_i for $(E_i, \mathcal{F}_i)$ is well defined. Thus, there exists an element $e_i \in C_y(B_i) \cap K_y(B^*)$ for $i = 0, \ldots, \Gamma$. Since at most Γ elements have cost 1, there is an element $e_i \in \overline{E}$ with $c^S(e_i) = 0$. Then the basis $B' = B^* \backslash \{y\} \cup \{e_i\}$ is an optimal solution in $\mathcal{F}^S$. Choosing e_i and B' instead of y and B^* in the above argumentation, we obtain a contradiction. It remains to show the approximation guaranty of the algorithm.

Let r be the rank of the considered matroid. Since any optimal solution contains at least r elements and the computed solution contains maximal $(\Gamma + 1)r$ elements, the proposed algorithm is in a $(\Gamma + 1)$-approximation.

We finally strengthen the estimation on the number of elements contained in an optimal solution for the MST problem. Let $G = (V, E)$ be a graph and $\underline{c}(e) = 0$ and $\overline{c}(e) = 1$ for all $e \in E$. We partition V into two sets V_1 and V_2 with $V_1 = \{v \in V \mid d(v) \leq \Gamma\}$ and $V_2 = V \backslash V_1$. Let now $G^* = (V, E^*)$ be an optimal solution to this exSub-RR MST instance. Then,

$$|E^*| \geq \frac{\Gamma + 1}{2}|V_2| + \sum_{v \in V_1} \frac{d(v)}{2}.$$

On the other hand, the number of edges E_{ALG} chosen by Algorithm 4.2 is bounded by

$$E_{\text{ALG}} \leq (\Gamma + 1) \cdot |V_2| + \sum_{v \in V_1} d(v).$$

Hence, this is a 2-approximation. □

In the last section we introduce an approximation scheme to solve any exSub-RR LCMin problem.

4.6. Approximation Algorithms

For any exSub-RR LCMin problem independent of the type of scenario sets, an algorithm returning the whole set U is an $|U|$-approximation. We improve this factor to $\frac{|U|}{\ell+1}$ for any $\ell \in \mathbb{N}$ and a run-time of $\mathcal{O}(2^\ell \ell |U|^\ell \text{X})$, where X denotes the run-time to compute an optimal solution of a corresponding LCMin instance. We start by introducing a criterion to test the feasibility of a given subset in the case of interval scenarios. The test runs in polynomial time if the size of the given subset is constant.

Lemma 4.6.1. *Let $(U, \mathcal{F}, \mathcal{S}_I)$ be an instance of an exSub-RR LCMin problem with interval scenarios given by lower and upper cost bounds $\underline{c}(u)$ and $\overline{c}(u)$ for all $u \in U$. Let $U' \subseteq U$ and*

$$\mathcal{S}_I(U') = \{S \in \mathcal{S}_I \mid c^S(u) = \underline{c}(u)\ \forall u \notin U' \text{ and } c^S(u) \in \{\underline{c}(u), \overline{c}(u)\}\ \forall u \in U'\}.$$

If there exists a solution $F \in \mathcal{F}^S$ with $F \subseteq U'$ for every $S \in \mathcal{S}_I(U')$, then U' is a feasible solution of the given exSub-RR LCMin instance.

Proof. Let $S \in \mathcal{S}_I$ and $S' \in \mathcal{S}_I(U')$ with $c^{S'}(u) = c^S(u)$ for all $u \in U'$ and $c^{S'}(u) = \underline{c}(u)$ otherwise. For any $F \in \mathcal{F}^S$ and $F' \in \mathcal{F}^{S'}$ with $F' \subseteq U'$,

$$c^S(F) - c^S(F') \geq c^{S'}(F) - c^{S'}(F') \geq 0,$$

since $F' \subseteq U'$. Thus, U' is a feasible solution of the given instance. □

A similar result can be achieved for Γ-scenarios.

Algorithm 4.3 An $\frac{|U|}{\ell+1}$-approximation for $\mathcal{S}_\Gamma$ and $\mathcal{S}_I$.

Input : A set U, a set of feasible solutions $\mathcal{F}$, lower and upper cost bounds $\underline{c}(u) \in \mathbb{N}$ and $\overline{c}(u) \in \mathbb{N}$ for all $u \in U$, a scenario parameter $\Gamma \in \mathbb{N}$, and an approximation factor $\ell \in \mathbb{N}$.

Output : Feasible solution U'.

for $i = \ell, \dots, 1$ **do**
  forall $\overline{U} \subseteq U$ *with* $|\overline{U}| = i$ **do**
    Set $\mathcal{S}_\Gamma = \{S \in \mathcal{S}_\Gamma \mid c^S(u) = \underline{c}(u)\ \forall u \notin U', c^S(u) \in \{\underline{c}(u), \overline{c}(u)\}\ \forall u \in U'\}$.
    if $\overline{U}$ *is feasible according to* $\mathcal{S}_\Gamma(\overline{U})$ **then**
      Set $U' = \overline{U}$
if $U' = \emptyset$ **then**
  return U
else
  return U'

Lemma 4.6.2. *Let $(U, \mathcal{F}, \mathcal{S}_\Gamma)$ be an instance of an exSub-RR LCMin problem with Γ-scenarios given by lower and upper bounds $\underline{c}(u)$ and $\overline{c}(u)$ for all $u \in U$ and a parameter $\Gamma \in \mathbb{N}$. Let $U' \subseteq U$ and*

$$\mathcal{S}_\Gamma(U') = \{S \in \mathcal{S}_\Gamma \mid c^S(u) = \underline{c}(u)\ \forall u \notin U' \text{ and } c^S(u) \in \{\underline{c}(u), \overline{c}(u)\}\ \forall u \in U'\}.$$

If there exists a solution $F \in \mathcal{F}^S$ with $F \subseteq U'$ for every $S \in \mathcal{S}_\Gamma(U')$, then U' is a feasible solution of the given exSub-RR LCMin instance.

Both sets $\mathcal{S}_I(U')$ and $\mathcal{S}_\Gamma(U')$ contain exponentially many scenarios in the cardinality of U'. If this cardinality is constant, the feasibility of a subset U' can be tested in $\mathcal{O}(2^{|U'|}\mathrm{X})$, where X denotes the time to solve a corresponding LCMin instance. This idea is captured in Algorithm 4.3, which test for all subsets with less than or equal to ℓ elements their feasibility, $\ell \in \mathbb{N}$. Choosing a feasible set with a minimum number or the whole set, if no feasible subset is found, leads to a $\frac{|U|}{\ell+1}$-approximation.

Theorem 4.6.3. *Let $(U, \mathcal{F}, \mathcal{S})$ be an exSub-RR LCMin instance with Γ-scenarios or interval scenarios and let U_{OPT} be an optimal solution. Then Algorithm 4.3 calculates a feasible solution U_{ALG} with*

$$\frac{|U_{\mathrm{ALG}}|}{|U_{\mathrm{OPT}}|} \leq \frac{|U|}{\ell+1}$$

for any $\ell \in \mathbb{N}$ and $\Gamma = |U|$ in the interval case. The run-time of the algorithm is $\mathcal{O}(2^\ell \ell |U|^\ell X)$.

Proof. If $U_{\mathrm{OPT}} \leq \ell$, the set $\overline{U} = U_{\mathrm{OPT}}$ is tested and U' is set to U_{OPT}. Hence, $|U'| = |U_{\mathrm{ALG}}| = |U_{\mathrm{OPT}}|$. If the optimal solution contains more than $\ell + 1$ elements,

$$\frac{|U_{\mathrm{ALG}}|}{|U_{\mathrm{OPT}}|} \leq \frac{|U|}{\ell+1}.$$

□

4.7. Conclusion and Open Issues

We considered the exact subset recoverable robust problem and investigated its complexity status and combinatorial properties. For discrete scenario sets and the case that the number of scenarios is not constant, we showed in Section 4.2 that the problem is strongly **NP**-hard. The hardness result is based on a simple subproblem, the minimum element problem. In order to construct approximation algorithms, one should start investigating this problem in more detail.

If the number of scenarios is constant, we only succeeded to prove that the exSub-RR LCMin problem is strongly **NP**-hard for a rather artificial LCMin problem. Thus, analyzing the exSub-RR shortest path problem or exSub-RR minimum spanning tree problem would be interesting. For a special type of scenarios, where the cost functions take only the values 0 or 1, the exSub-RR minimum spanning tree problem with three scenarios seems to be similar to finding a maximum independent set in the intersection of three graphical matroids. Whether this problem can be solved efficiently is—to the best of my knowledge—open. Note that for a constant number of scenarios the minimum element problem is solvable in polynomial time, since any optimal solution contains at most one element for any scenario.

For interval scenarios, the gap concerning complexity seems rather large: the exSub-RR minimum weight basis problem for matroids is solvable in polynomial time while the exSub-RR shortest path problem or exSub-RR minimum (s,t)-cut problem on a graph $G = (V, A)$ cannot be approximated with a factor better than $|A|^{(1-\varepsilon)}$ for any $\varepsilon > 0$. It would be interesting to find a problem that is strongly **NP**-hard but can be approximated with a constant factor. Considering strictly deviating cost bounds for such a problem, the decision whether an element is necessary needs to be strongly **NP**-hard. Furthermore, this problem is not allowed to satisfy the duplication-condition. Hence, the minimum perfect matching problem is no candidate, since it satisfies this condition.

In case of Γ-scenarios, most problems become again strongly **NP**-hard. This is the case for the exSub-RR shortest path problem and the exSub-RR minimum (s,t)-cut problem and the exSub-RR minimum spanning tree problem. For the first two problems we also provided a lower bound of $|A|^{(1-\varepsilon)}$ for any $\varepsilon > 0$ on the approximability. In the case of the minimum spanning tree problem, the complexity is based on a reduction from the k-connected spanning subgraph problem. The considered Γ-scenarios define cost functions with values 0 or 1. For this case, we give a 2-approximation algorithm. However, algorithms with a better approximation factor can probably be designed using more of the techniques proposed by Gabow et al. [51]. It remains open how to approximate the exSub-RR minimum spanning tree problem if the cost bounds are chosen arbitrarily.

We finally proposed approximation algorithms for the exSub-RR problem with interval scenarios and Γ-scenarios. It remains open if better run-times or approximation guarantees can be achieved.

5. A Recoverable Robust Knapsack Problem

Admission control problems have been studied extensively in the past. In a typical setting, resources like bandwidth have to be distributed to different customers according to their demands maximizing the profit for the company. Yet, in real-world applications these demands deviate over time. In order to satisfy the service promises of the company, often a robust approach is chosen wasting benefits for the company. Our model overcomes this problem by allowing a limited recovery of a previously fixed assignment as soon as the data are known by violating at most k service promises and serving up to ℓ new customers. Applying this approach to the call admission problem on a single link of a telecommunication network leads to a recoverable robust version of the knapsack problem.

In this chapter we consider on the one side the complexity status for this recoverable robust knapsack problem and on the other side its polyhedral structure. For discrete scenario sets, we prove weakly **NP**-hardness for a constant number of scenarios and introduce a pseudo-polynomial algorithm. If the number of scenarios is not constant, the problem is strongly **NP**-hard and in some cases not approximable. In order to obtain valid inequalities for the recoverable robust knapsack polytope, we generalized the well-known concept of covers and extended covers. Finally, we present two computational studies to investigate the influence of parameters k and ℓ on the objective and evaluate the effectiveness of our new class of valid inequalities.

For Γ-scenarios we introduce an IP-formulation which is polynomial in the size of the input if the scenario profit is 0. Furthermore, we also adapt the concept of covers and extended covers and gain valid inequalities for the knapsack polytope strengthening the results obtained for the robust knapsack version by Klopfenstein and Nace [76]. This chapter is based on work with Arie M. C. A. Koster and Manuel Kutschka [23].

5.1. Introduction

Motivation We consider a telecommunication network in which an operator has to decide which demand is granted admission at which time. In many cases these requests made by the customers are handed in before their actual realization. Each request specifies the amount of traffic, the source and destination. The task of the operator is to give service promises according to these data maximizing the total profit for the company. Yet, in many applications the requests change at the time of realization. In case of slight deviations the customer still expects the promised quality of service.

A straight-forward approach dealing with such demand uncertainties estimates the unknown demand with its deviations, and bases the resource distribution on the worst-case occurrence: a robust approach. In most cases this leads to over-conservative decisions where the largest

part of the available bandwidth capacity is unused. This is clearly neither cost-efficient nor resource exploiting.

There are at least two means of increasing the economical benefits for the company: first by relaxing the deviation assumption and second by satisfying all but k service promises. The first idea is reflected in the consideration of different types of scenario sets. The second idea is captured in the two stage concept of recoverable robustness. Here over-conservatism in robust optimization is avoided by allowing a limited recovery after the full data is revealed. This concept has been introduced, for example, by Liebchen et al. [81] (see also Section 1.2), and applied to railway optimization problems as delay-management, platforming and shunting.

We adopted these approaches for the call admission problem on a single link of a telecommunication network. This problem can be seen as a classical knapsack problem, where the demands are represented by items and the amount of traffic is equivalent to the weights. The profit of an item is divided into two parts: the first-stage profit captures the basic fee for a customer to use the network and the scenario profit the profit obtained by routing the demand. Finally, the bandwidth of the link is modeled by the knapsack capacity. As an adaption of the recoverable robust approach, a first-stage solution is in our case a subset of items such that the sum of weights does not exceed the link capacity. In the second stage, when the scenario is revealed and the scenario profits and weights are known, k service promises, i.e., k items, may be removed from the first-stage solution. The new subset must satisfy the link capacity constraint of this scenario. The objective is to find a first-stage solution with maximum total profit. The total profit of a given subset is the sum of the first-stage profit representing the basic fee of the customers to use the network and the minimal scenario profit representing the actual gain through the routing rates.

Another motivation for the investigation of this recoverable robust knapsack problem can be found in wireless (cellular) networks (e.g., WLAN, 3G). Each antenna has a limited bandwidth capacity to be partitioned among the users in its cell. Users, however, do not generate a constant traffic rate. Depending on their needs (e.g., data traffic, web browsing), the requested data rate fluctuates (e.g., 64 Kbit/s, 384 Kbit/s, 2 Mbit/s). In the network capacity planning phase usually averages are considered. During operation, individual users with their actual data rates are admitted to the cell. In the context of recoverable robust knapsack, the planning problem can be enhanced by considering the capacity planning as first-stage knapsack, whereas snapshots of the operation can be taken as scenarios in the second stage. Compared to the planning phase, up to k users can be refused a connection, whereas up to ℓ new users can be admitted. Since this admission problem has to be considered by each cell, the recoverable robust knapsack problem is a subproblem of such a recoverable robust wireless network planning problem.

Model and Notation We introduce a recoverable robust version of the knapsack problem, in which the weights as well as the profits are subject to uncertainties. These uncertainties are given via a set of scenarios. In this recoverable robust counterpart a first-stage solution is a subset of items such that its first-stage weight does not exceed the first-stage capacity. In the second stage, i.e., when the scenario is revealed and the profits and weights are known, k items may be removed from the first-stage solution and ℓ items may be added. This new subset must satisfy the scenario capacity restriction according to the weights of the scenario. The objective is to find a first-stage solution with maximum total profit. The total profit is the sum of the first-stage profit and the minimum scenario profit. More formally, the problem is defined in the following way.

Definition 5.1.1 ((k,ℓ)-Recoverable Robust Knapsack ((k,ℓ)-rrKP) Problem)**.** Let $N = \{1,\dots,n\}$ be a set of n items, $c^0 \in \mathbb{N}$ be the first-stage capacity, p_j^0 be the first-stage profit and w_j^0 be the first-stage weight of each item $j \in N$. Each scenario S of a given set of scenarios $\mathcal{S}$ defines a profit function $p^S : N \to \mathbb{N}$, a weight function $w^S : N \to \mathbb{N}$ and a capacity $c^S \in \mathbb{N}$. Furthermore, the parameter $k \in \mathbb{N}$ limits the number of deletable items from a first-stage solution and the parameter $\ell \in \mathbb{N}$ the number of new items contained in a recovered solution. For a given subset $X \subseteq N$ the *recovery set* $\mathcal{X}_X^{(k,\ell)}$ includes all subsets of N that contain at most ℓ additional items and fail to contain at most k items of X, i.e.,

$$\mathcal{X}_X^{(k,\ell)} = \{X' \subseteq N \mid |X \backslash X'| \leq k \text{ and } |X' \backslash X| \leq \ell\}.$$

With $\mathcal{X}^S$ for $S \in \mathcal{S}$ we denote all subsets of $X' \subseteq N$ that satisfy the *scenario weight constraint* $\sum_{i \in X'} w_i^S \leq c^S$. A *feasible first-stage solution* is a subset $X \subseteq N$ which satisfies the *first-stage weight constraint* $\sum_{i \in X} w_i^0 \leq c^0$ and $\mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S \neq \emptyset$ for all $S \in \mathcal{S}$. The *total profit* $p(X)$ of a feasible solution X is defined as

$$p(X) = \sum_{i \in X} p_i^0 + \min_{S \in \mathcal{S}} \max_{X^S \in \mathcal{X}_X^{(k,\ell)} \cap \mathcal{X}^S} \sum_{i \in X^S} p_i^S.$$

The *recoverable robust knapsack problem* is to find a feasible first-stage solution with maximum total profit.

Related Work The knapsack problem is one of the basic problems in combinatorial optimization. An instance of the knapsack problem (KP) consists of n items collected in an item set $N = \{1,\dots,n\}$ with integer profits p_j and integer weights w_j for all items $j \in N$, and a capacity $c \in \mathbb{N}$. The objective is to select a subset X of these items such that the profit of X is maximized and the total weight does not exceed c. The knapsack problem provides a relaxation of several combinatorial optimization problems (e.g., generalized assignment, capacitated facility location, capacitated vehicle routing, airline scheduling). Despite its simple structure the problem is known to be weakly **NP**-hard [69] but solvable via dynamic programming in pseudo-polynomial time [10, 35]. Different branch-and-cut algorithms are used to solve this problem in practice. A detailed introduction to the knapsack problem and its variations can be found in Martello and Toth [82] and Kellerer, Pferschy and Pisinger [71].

In Yu [100] a robust version of the knapsack problem is defined by introducing uncertainty in the profit values: Let $\mathcal{S}$ be a set of scenarios, each scenario S determining a profit function $p^S : N \to \mathbb{N}$ on the given set of items N. The robust knapsack problem is to find a set of items maximizing the minimum profit over all scenarios such that the total weight does not exceed the capacity. By a reduction from the set covering problem, Yu showed that for discrete scenario sets the decision version of the problem is strongly **NP**-hard if the number of scenarios is not constant. As mentioned by Aissi et al. [3] this proof includes the result that the optimization version is not f-approximable for any function $f : N \to (1,\infty)$. For a discrete scenario set with a constant number of scenarios the problem is weakly **NP**-hard, solvable in pseudo-polynomial time [65, 100] and there exists an FPTAS [3]. Iida [61] provided a computational study on the robust knapsack problem with discrete scenarios comparing several methods to derive upper and lower bounds for the profit. A different type of scenarios, so-called Γ-scenarios, were introduced by Bertsimas and Sim [14]. Their main result for the robust counterpart of 0-1 combinatorial optimization problems with cost uncertainties shows that robust knapsack problems with Γ-scenarios can be solved in pseudo-polynomial time.

A different robust version of the knapsack problem assumes the weights to be subject to uncertainties. The objective is to find a set of items which maximizes the profit and satisfies

the weight restriction for any weight function $w^S : N \to \mathbb{N}$ defined by a scenario $S \in \mathcal{S}$. The main focus in Klopfenstein and Nace [76] is to find strong polyhedral descriptions for this robust problem with Γ-scenarios. For this model Bertsimas and Sim [14] investigated the trade-off between robustness and optimality in a short computational study.

Contribution and Chapter Outline Our research focuses on adapting classical results of the (robust) knapsack problem for this recoverable robust version. We start with a complexity study. For a constant number of discrete scenarios, we show that the recoverable robust knapsack problem is weakly **NP**-hard. Any such instance can be solved in pseudo-polynomial time by a dynamic program. If the number of discrete scenarios is not constant, the problem is strongly **NP**-hard and in special cases not approximable in polynomial time, unless $\mathbf{P} = \mathbf{NP}$ (Section 5.2.1). For Γ-scenarios the complexity status is still unknown. So far we just established a polynomial size ILP-formulation for the case without any scenario profits (Section 5.4.2).

Next to its complexity we are interested in obtaining strong polyhedral descriptions of all feasible first-stage solutions. For discrete scenarios and Γ-scenarios we adapt the well-known (minimal/extended) cover inequalities for the knapsack problem to gain valid inequalities for the recoverable robust knapsack polytope (Section 5.2.2 and 5.4.3). Any 0-1-point satisfying all these inequalities is a feasible solution to the (k, ℓ)-rrKP instance. For Γ-scenarios, we manage to strengthen the extended robust cover inequalities introduced by Klopfenstein and Nace in [76]. Finally, we present computational studies to investigate the impact of these new inequalities for solving (k, ℓ)-rrKP instances as well as the gain of recovery (Section 5.2.3).

5.2. Discrete Scenarios

In a discrete scenario set $\mathcal{S}_D$ every scenario $S_1, \ldots, S_r$ is explicitly given with its weight function $w^{S_i} : N \to \mathbb{N}$ and its profit function $p^{S_i} : N \to \mathbb{N}$ on the item set N and its capacity c^{S_i}, $i = 1, \ldots, r$. The following integer program models the (k, ℓ)-recoverable robust knapsack problem

$$
\begin{aligned}
\text{(drrKP-IP)} \qquad \max \sum_{i \in N} p_i^0 x_i + \omega & & & \\
\sum_{i \in N} w_i^0 x_i &\leq c^0 & & (5.1)\\
\sum_{i \in N} w_i^S x_i^S &\leq c^S & \forall\, S \in \mathcal{S}_D & (5.2)\\
x_i^S - x_i &\leq y_i^S & \forall\, S \in \mathcal{S}_D,\, i \in N & (5.3)\\
x_i - x_i^S &\leq z_i^S & \forall\, S \in \mathcal{S}_D,\, i \in N & (5.4)\\
\sum_{i \in N} y_i^S &\leq \ell & \forall\, S \in \mathcal{S}_D & (5.5)\\
\sum_{i \in N} z_i^S &\leq k & \forall\, S \in \mathcal{S}_D & (5.6)\\
\omega - \sum_{i \in N} p_i^S x_i^S &\leq 0 & \forall\, S \in \mathcal{S}_D & (5.7)\\
x_i, x_i^S, y_i^S, z_i^S &\in \{0,1\} & \forall\, S \in \mathcal{S}_D,\, i \in N & \\
\omega &\geq 0 & &
\end{aligned}
$$

The variable x represents the first-stage solution, x^S the solution taken in scenario $S \in \mathcal{S}_D$, y_i^S determines if item i is added in S and z_i^S if item i is removed from x in S, $i \in N$. Inequalities (5.3)-(5.6) guarantee that x^S is a feasible recovery for the first-stage solution x and (5.1) and (5.2) that the weight constraints are obeyed. The last inequality (5.7) in combination with the objective function models the total profit

$$\max \sum_{i \in N} p_i^0 x_i + \min_{S \in \mathcal{S}_D} \max \sum_{i \in N} p_i^S x_i.$$

The minimum scenario profit, i.e., $\min_{S \in \mathcal{S}_D} \max \sum_{i \in N} p_i^S x_i$, is captured in the variable ω. The size of the program depends on the number of scenarios and the number of items.

5.2.1. Complexity of the (k, ℓ)-rrKP

We start with an analysis of the complexity status of the (k, ℓ)-rrKP for discrete scenarios. Since the nominal knapsack problem is part of the second stage, it is already weakly **NP**-hard to compute the total profit for a given set of items even for one scenario.

Lemma 5.2.1. *The decision if the total profit of a feasible first-stage solution X is greater than or equal to a constant K for just one scenario is weakly* **NP***-hard, even if $k = 0$ or $\ell = 0$.*

Proof. We start with the general case of $k \neq 0$ and $\ell \neq 0$ and show a reduction from the nominal knapsack problem. Let I be a knapsack instance with an item set $N = \{1, \ldots, n\}$, a weight function $w : N \to \mathbb{N}$, a profit function $p : N \to \mathbb{N}$ and a capacity value $c \in \mathbb{N}$. We construct a (k, ℓ)-recoverable robust knapsack instance I' with one scenario S in the following way: the set of items remains the same, $w^0 = p^0 = c^0 = 0$, $k = n$, $\ell = n$, $w^S = w$, $p^S = p$ and $c^S = c$. Any subset $X \subseteq \{1, \ldots, n\}$ is a feasible solution of the instance I'. The total profit of any set X is greater than or equal to K if and only if there exists a feasible solution to I with a profit greater than or equal to K. Replacing $k = n$ by $k = 0$ in I', the total profit of the feasible first-stage solution $X = \emptyset$ in I' indicates whether I is a yes-instance. For $\ell = 0$, the total profit of N is greater than or equal to K if and only if I is a yes-instance. □

If k and ℓ are constant, the recovery set $\mathcal{X}_X^{(k,\ell)}$ contains a constant number of solutions for any $X \subseteq N$. Hence, the total profit can be computed in polynomial time by enumeration.

Since the knapsack problem is a special case of the (k, ℓ)-rrKP problem, the (k, ℓ)-rrKP problem remains at least weakly **NP**-hard for one scenario. We will later show by introducing an optimal pseudo-polynomial algorithm that the problem is weakly **NP**-hard if the number of scenarios is constant. We now prove that even in the case $p^0 = 0$ the (k, ℓ)-rrKP is strongly **NP**-hard if the number of scenarios in $\mathcal{S}_D$ is not constant.

Theorem 5.2.2. *The (k, ℓ)-rrKP problem is strongly* **NP***-hard for discrete scenario sets even if there is no first-stage profit.*

Proof. We show a reduction from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Each clause is formed by three literals. The corresponding instance I' of the (k, ℓ)-rrKP problem contains a set of $2n + k + \ell$ items $N = N_1 \cup N_2 \cup N_3$ with $N_1 = \{1, \ldots, 2n\}$, $N_2 = \{2n + 1, \ldots, 2n + k\}$ and $N_3 = \{2n + k + 1, \ldots, 2n + k + \ell\}$. The items in N_1 represent the true or false assignment of the variables x. The last $k + \ell$

items are auxiliary items to fix the recovery action for any reasonable first-stage solution. The parameters of the first stage are set to

$$c^0 = n + k,\ w^0(i) = \begin{cases} 1 & i \in N_1 \cup N_2 \\ c^0 + 1 & \text{otherwise} \end{cases},\ p^0(i) = 0\ \forall i \in N,\ \text{and}\ K = 1 + \ell.$$

Hence, no feasible first-stage solution contains any of the items in N_3. But the profit functions of each scenario will be constructed in a way such that all these items have to be added to the recovery solution for each scenario to obtain a total profit of at least K.

It remains to define these scenarios $\mathcal{S}$. This set consists of three different types: The first type forces any first-stage solution with a total profit greater than or equal to K to incorporate the items in N_2. Furthermore, these items need to be removed in any other recovery solution and thus no item of N_1 can be removed. The second type guarantees that for $i = \{1, \dots, n\}$ either the item $2i-1$ or $2i$ is part of the first-stage solution. The choice models the assignment of the variables x_i to true or false, $i = 1, \dots, n$. The last type assures that all clauses are satisfied.

First type of scenarios: For the items $j \in N_2$ we add a scenario S_j^1 to I' with $c^{S_j^1} = 0$,

$$p^{S_j^1}(i) = \begin{cases} 1 & i = j \text{ or } i \in N_3 \\ 0 & \text{otherwise} \end{cases} \text{ and } w^{S_j^1}(i) = 0 \text{ for } i \in N.$$

Hence, a first-stage solution must contain these k items, to get a value greater than or equal to K. Second type of scenarios: For each variable x_j, $j \in \{1, \dots, n\}$, we add S_j^2 to I' defining $c^{S_j^2} = 0$,

$$p^{S_j^2}(i) = \begin{cases} 1 & i \in \{2j-1, 2j\}, i \in N_3 \\ 0 & \text{otherwise} \end{cases} \text{ and } w^{S_j^2}(i) = \begin{cases} 1 & i \in N_2 \\ 0 & \text{otherwise} \end{cases} \text{ for } i \in N.$$

Third type of scenarios: For each clause $j = 1, \dots, m$ we construct one scenario S_j^3 which puts a profit of 1 to all items representing a verification of the clause, i.e., $p^{S_j^3}(i) = 1$ for an even item $i \in N_1$ if and only if $\overline{x}_i \in C_j$ and $p^{S_j^3}(i) = 1$ for an odd item $i \in N_1$ if and only if $x_i \in C_j$. Furthermore, all items $i \in N_3$ are assigned a profit of 1. The weights of all items $i \in N_1 \cup N_3$ in all these scenarios and the scenario capacity are set to 0. The weights of items $i \in N_2$ are set to 1. The size of the constructed instance is polynomial in the size of I. A true assignment of the 3SAT instance I exists if and only if there exists a solution of I' with a total profit greater than K. □

Note that the optimal value of the (k, ℓ)-rrKP instance in the reduction determines a lower bound of $\frac{\ell+1}{\ell}$ for $\ell \in \mathbb{N}$ on the best possible approximation factor. This implies that any $(k, 0)$-rrKP is inapproximable, unless $\mathbf{P} = \mathbf{NP}$.

Finally, we consider the special case, in which all scenario profits are set to 0.

Theorem 5.2.3. *The (k, ℓ)-rrKP is strongly* **NP***-hard for discrete scenario sets even when there is no scenario profit.*

Proof. The reduction is a modification of the proof of Theorem 5.2.2. In that case the first-stage profit is set to 1 for all items $i \in N_1$, to $2n$ for $i \in N_2$, to 0 otherwise and $K = n + 2nk$. Hence, any first-stage solution X with $p(X) \geq K$ needs to contain all items of N_2. The first-stage weight equals 1 for all items and the first-stage capacity $n + k$. A scenario

S_j^1 representing a clause C_j, $j = 1, \ldots, m$, sets $w^{S_j^1}(i) = 1$ if and only if the corresponding variable falsifies the clause for $i \in N_1$, $w^{S_j^1}(i) = 3$ for $i \in N_2$ and 0 otherwise. The capacity for each clause equals 2. Furthermore, for each variable x_j, $j = 1, \ldots, n$, we add a scenario S_j^2 with $w^{S_j^2}(2j-1) = 1$, $w^{S_j^2}(2j) = 1$, $w^{S_j^2}(i) = 2$ for $i \in N_2$, $w^{S_j^2}(i) = 0$ otherwise and $c^{S_j^2} = 1$. These scenarios guarantee that either the item $2j-1$ or $2j$ is in a feasible first-stage solution. □

This reduction does not imply a lower bound on the approximation factor.

Constant Number of Scenarios We will now demonstrate that the recoverable robust knapsack problem can be solved in pseudo-polynomial time for a constant number of scenarios. The presented algorithm is based on dynamic programming and extends the idea of Yu [100] for the robust case. The concept of the algorithm is to compute recursively the value of the optimization problem when the optimal selection is made among the first t items under the first-stage capacity d^0 with $c \geq d^0 \geq 0$, the scenario dependent recovery parameters (j^S, i^S), $0 \leq j^S \leq k$, $0 \leq i^S \leq \ell$, the scenario capacities d^S with $c^S \geq d^S \geq 0$ and a guaranteed extra profit in each scenario S of α^S induced by the items $\{t+1, \ldots, n\}$. This problem is modeled by an integer program quite similar to the drrKP-IP-formulation (page 88): As in that setting the variable x represents the first-stage solution, x^S the solution taken in scenario $S \in \mathcal{S}_D$, y_i^S indicates if item i is added in S and z_i^S if item i is removed from x in S, $i \in N$.

$$
\begin{aligned}
g(t, d^0, v^{S_1}, \ldots, v^{S_r}) = \max \sum_{i \in N_t} p_i^0 x_i + \omega & \\
\sum_{i \in N_t} w_i^0 x_i \leq d^0 & \\
\sum_{i \in N_t} w_i^S x_i^S \leq d^S & \qquad \forall\, S \in \mathcal{S}_D \\
x_i^S - x_i \leq y_i^S & \qquad \forall\, i \in N_t,\, \forall\, S \in \mathcal{S}_D \\
x_i - x_i^S \leq z_i^S & \qquad \forall\, i \in N_t,\, \forall\, S \in \mathcal{S}_D \\
\sum_{i \in N_t} y_i^S \leq i^S & \qquad \forall\, S \in \mathcal{S}_D \\
\sum_{i \in N_t} z_i^S \leq j^S & \qquad \forall\, S \in \mathcal{S}_D \\
\omega - \sum_{i \in N_t} p_i^S x_i^S + \alpha^S \leq 0 & \qquad \forall\, S \in \mathcal{S}_D \qquad (5.8) \\
x_i, x_i^S, y_i^S, z_i^S \in \{0,1\} & \qquad \forall\, i \in N_t,\, S \in \mathcal{S}_D \\
\omega \geq 0 &
\end{aligned}
$$

to the given parameters t, d^0, v^S with $v^S = (d^S, \alpha^S, j^S, i^S)$ and $\alpha^S \geq 0$ for all $S \in \mathcal{S}_D$ and $N_t := \{1, \ldots, t\}$. The inequality (5.8) differs from the inequality (5.7) in the drrKP-IP-formulation by the addition of the parameter α^S to the maximum profit achieved by any recovered solution in scenario S. This parameter α^S models a guaranteed profit in scenario S obtained by items $t+1, \ldots, n$ under consideration of the remaining capacity $c^S - d^S$ and recovery means $(k - j^S, \ell - i^S)$.

In order to simplify the initialization, we add an item $i = 0$ with $p_0^0 = 0$, $p_0^S = 0$ for all scenarios $S \in \mathcal{S}_D$, $w_0^0 = c^0 + 1$ and $w_0^S = c^S + 1$ for all $S \in \mathcal{S}$. Hence, for $|\mathcal{S}_D| = r$ any feasible solution sets $x_0 = 0$ and $g(0, d^0, v^{S_1}, \dots, v^{S_r}) = 0$ if $d^0 \geq 0$, $d^S \geq 0$, $j^S \geq 0$ and $i^S \geq 0$ for all $S \in \mathcal{S}_D$. We denote this set of all feasible parameter sets with $\mathcal{Z}$, i.e.,

$$\mathcal{Z} = \{0, \dots, n\} \times \{0, \dots, c^0\} \times \big\{\{0, \dots, c^S\}, \{0, \dots, P_{\max}\}, \{0, \dots, k\}, \{0, \dots, \ell\}\big\}^r$$

with $P_{\max} = \max_{S \in \mathcal{S}_D} \sum_{i=1}^n p_i^S$. For $\gamma = (t, d^0, d^{S_1}, \alpha^{S_1}, j^{S_1}, i^{S_1}, \dots, i^{S_r})$ we denote γ_1 with $t(\gamma)$, γ_2 with $d^0(\gamma)$ and so on. If there is a parameter set $\gamma \notin \mathcal{Z}$ with $\gamma(t) = 0$, there exists no feasible solution for $g(\gamma)$ and thus, we set $g(\gamma) = -\infty$.

In the next step, we define a recursion formula computing the value of $g(\gamma)$ out of the values of $g(\gamma')$ with $t(\gamma') = t(\gamma) - 1$. Loosely speaking, we decide if item $t(\gamma)$ is part of the first-stage solution and in which scenarios the recovery action is taking place. More formally, for a given parameter set γ we define $g^+(\gamma)$ as the maximum value of $g(\gamma)$ if the item $t(\gamma)$ is added to the first-stage solution. As we will show $g^+(\gamma)$ can recursively be computed by

$$g^+(t, d^0, v^{S_1}, \dots, v^{S_r}) = \max_{\beta \in \{-,\sim\}^r} g(t-1, d^0 - w_t^0, v_{\beta_1}^{S_1}, \dots, v_{\beta_r}^{S_r}) + p_t^0$$

with $v_-^S = (d^S, \alpha^S, j^S - 1, i^S)$ and $v_\sim^S = (d^S - w_t^S, \alpha^S + p_t^S, j^S, i^S)$. Here, the vector v_-^S represents the decision to delete this item in the recovered solution of scenario S and $v_\sim^S$ the decision to keep the item. In the same way we define $g^-(\gamma)$ as the maximum value of $g(\gamma)$ if the item $t(\gamma)$ is not added to the first-stage solution. This can also be recursively be computed by

$$g^-(t, d^0, v^{S_1}, \dots, v^{S_r}) = \max_{\beta \in \{+,0\}^r} g(t-1, d^0, v_{\beta_1}^{S_1}, \dots, v_{\beta_r}^{S_r})$$

with $v_+^S = (d^S - w_t^S, \alpha^S + p_t^S, j^S, i^S - 1)$ and $v_0^S = v^S$. The vector v_+^S represents the case to add this item in the recovered solution of scenario S and v_0^S to refrain from doing so. The combination of these two values determines $g(\gamma)$ as

$$g(\gamma) = \max\{g^+(\gamma), g^-(\gamma)\}. \tag{5.9}$$

The value of an optimal solution is given by $g(n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$ with $\overline{v}^{S_i} = (c^{S_i}, 0, k, \ell)$. To obtain an optimal solution $x^*, x^{S_1}, \dots, x^{S_r}$ to $g(n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$ we consider a feasible sequence γ^t, $t = 0, \dots, n$, in $\mathcal{Z}$ with $\gamma^n = (n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$. A sequence γ^t, $t = 0, \dots, \ell$, is called *feasible*, if $\gamma^t \in \mathcal{Z}$ for all $t = 0, \dots, \ell$, $t(\gamma^i) = i$, and $\gamma^{t-1} \in \mathcal{Z}(\gamma^t)$ with

$$g(\gamma^t) = \begin{cases} g(\gamma^{t-1}) & \text{if } \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \\ g(\gamma^{t-1}) + p_i^0 & \text{if } \gamma^{t-1} \in \mathcal{Z}^+(\gamma^t) \end{cases}$$

for $t = 1, \dots, \ell$. The set $\mathcal{Z}(\gamma)$ contains all possible predecessors of a given parameter γ. An element γ' is called a *predecessor* of γ, if $t(\gamma') = t(\gamma) - 1$, $d^0(\gamma') = d^0(v) - w_{t(\gamma)}^0$ or $d^0(\gamma') = d^0(\gamma)$ and

$$v^S(\gamma') \in \begin{cases} \{v_+^S(\gamma), v_0^S(\gamma)\} & \text{if } d^0(\gamma') = d^0(\gamma) \\ \{v_-^S(\gamma), v_\sim^S(\gamma)\} & \text{if } d^0(\gamma') = d^0(\gamma) - w_{t(\gamma)}^0. \end{cases}$$

A predecessor of γ is element of $\mathcal{Z}^+(\gamma)$, if $d^0(\gamma') = d^0(\gamma) - w_{i(\gamma)}^0$ and otherwise it is an element of $\mathcal{Z}^-(\gamma)$. According to such a feasible sequence γ^t, $t = 0, \dots, n$, with $\gamma^n = (n, c^0, \overline{v}^{S_1}, \dots, \overline{v}^{S_r})$, also called a *solution sequence*, a solution $x^*, x^{S_1}, \dots, x^{S_r}$ is defined by $x_0 = 0$, $x_0^S = 0$ for all $S \in \mathcal{S}_D$,

$$x_t^* = \begin{cases} 0 & \text{if } \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \quad x_t^S = \begin{cases} 0 & \text{if } v^S(\gamma^{t-1}) = \{v_0^S(\gamma^t), v_\sim^S(\gamma^t)\} \\ 1 & \text{otherwise} \end{cases}$$

for all $S \in \mathcal{S}_D$ and $t \geq 1$. Based on the initial condition and the recursion a dynamic program can easily be achieved with a run-time of $\mathcal{O}(n \cdot c_{\max}(c_{\max} \cdot P_{\max} \cdot k \cdot \ell)^{|\mathcal{S}_D|} \cdot 2^{|\mathcal{S}_D|})$ with $c_{\max} = \max\{c^0, \max_{S \in \mathcal{S}_D} c^S\}$ and $P_{\max} = \max\{\sum_{i=1}^{n} p_i^0, \max_{S \in \mathcal{S}_D} \sum_{i=1}^{n} p_i^S\}$.

Theorem 5.2.4. *Let $x^*, x^{S_1}, \ldots, x^{S_r}$ be some 0-1 points computed according to the recursive formula (5.9) and a solution sequence γ^t, $t = 0, \ldots, n$. Then this is an optimal solution to the given (k, ℓ)-rrKP instance with $p(x^*) = g(\gamma^n)$.*

Proof. Let $\gamma^t \in \mathcal{Z}$, $t = 0, \ldots, \bar{t}$ be any feasible sequence and the corresponding 0-1 points $\overline{x} \in \{0,1\}^{\bar{t}+1}$ and $\overline{x}^S \in \{0,1\}^{\bar{t}+1}$ with $x_0 = 0$, $\overline{x}_0^S = 0$ for all $S \in \mathcal{S}_D$,

$$\overline{x}_t = \begin{cases} 0 & \text{if } \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \\ 1 & \text{otherwise} \end{cases} \quad \text{and} \quad \overline{x}_t^S = \begin{cases} 0 & \text{if } v^S(\gamma^{t-1}) = \{v_0^S(\gamma^t), v_\sim^S(\gamma^t)\} \\ 1 & \text{otherwise} \end{cases}$$

for $t \geq 1$. Furthermore, we define the optimization problem $\phi(\gamma)$ for $\gamma \in \mathcal{Z}$ by

$$\begin{aligned}
\Phi(\gamma) = \max \sum_{i=0}^{t(\gamma)} p_i^0 x_i + \omega & \\
\sum_{i=0}^{t(\gamma)} w_i^0 x_i \leq d^0(\gamma) & \\
\sum_{i=0}^{t(\gamma)} w_i^S x_i^S \leq d^S(\gamma) & \quad \forall\, S \in \mathcal{S}_D \\
x_i^S - x_i \leq y_i^S & \quad \forall\, i = 0, \ldots, t(\gamma),\ \forall S \in \mathcal{S}_D \\
x_i - x_i^S \leq z_i^S & \quad \forall\, i = 0, \ldots, t(\gamma),\ \forall S \in \mathcal{S}_D \\
\sum_{i=0}^{t(\gamma)} y_i^S \leq i^S(\gamma) & \quad \forall\, S \in \mathcal{S}_D \\
\sum_{i=0}^{t(\gamma)} z_i^S \leq j^S(\gamma) & \quad \forall\, S \in \mathcal{S}_D \\
\omega - \min_{S \in \mathcal{S}_D} \sum_{i=0}^{t(\gamma)} p_i^S x_i^S + \alpha^S(\gamma) \leq 0 & \quad \forall\, S \in \mathcal{S}_D \\
x_i, x_i^S, y_i^S, z_i^S \in \{0,1\} & \quad i = 0, \ldots, t(\gamma), S \in \mathcal{S}_D, \\
\omega \geq 0 &
\end{aligned}$$

with $p_0^0 = 0$, $w_0^0 = c^0 + 1$, $w_0^S = 0$ and $p_0^S = 0$ for all $S \in \mathcal{S}_D$. The value of an optimal solution of $\phi(\gamma)$ is denoted by $\Phi(\gamma)$. We will show that $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is an optimal solution to the optimization problem $\phi(\gamma^{\bar{t}})$.

Let us assume for contradiction that this is not the case and γ^t, $t = 0, \ldots, t'$, is a feasible sequence with minimum t' such that either $\Phi(\gamma^{t'}) \neq g(\gamma^{t'})$ or $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is not a feasible solution. For any $\gamma \in \mathcal{Z}$ with $t(y) = 0$ and $\overline{x}_0 = 0 = \overline{x}_0^S$ we obtain $g(\gamma) = \Phi(\gamma)$, and since $\overline{x}_0 = 0 = \overline{x}_0^S$ is a feasible solution for $\phi(\gamma)$, t' is greater than or equal to 1. We will show in the next three claims that

1. $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is a feasible solution of $\phi(\gamma^{t'})$;
2. $$g(\gamma^{t'}) = \sum_{i=0}^{t'} p_i^0 \overline{x}_i + \min_{S \in \mathcal{S}_D} \sum_{i=1}^{t'} p_i^S \overline{x}_i^S + \alpha^S(\gamma^{t'});$$
3. and $g(\gamma^{t'}) \geq \Phi(\gamma^{t'})$.

This leads to a contradiction to our assumption.

Claim. The solution $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is a feasible solution of $\phi(\gamma^{t'})$.

Proof. Due to definition of $\overline{x}, \ldots, \overline{x}^{S_r}$ and our assumption, $\{\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}\}_{|[0,\ldots,t'-1]}$ is a feasible solution of $\phi(\gamma^{t'-1})$. We will now consider all four cases to construct $\gamma^{t'}$ out of $\gamma^{t'-1}$ and show that in any case the solution $\overline{x}, \ldots, \overline{x}^{S_r}$ remains feasible.

If $\gamma^{t'-1} \in \mathcal{Z}^-(\gamma^{t'})$, then $\overline{x}_{t'} = 0$ and $\overline{x}_{t'} \leq \overline{x}_{t'}^S$ for all $S \in \mathcal{S}_D$. Therefore,

$$\left|\left\{j \in \{0, \ldots, t'\} \mid \overline{x}_j - \overline{x}_j^S > 0\right\}\right| = j^S(\gamma^{t'-1}) = j^S(\gamma^{t'})$$

for all $S \in \mathcal{S}_D$ and

$$\sum_{j=0}^{t'} w_j^0 \overline{x}_j = \sum_{j=0}^{t'-1} w_j^0 \overline{x}_j \leq d^0(\gamma^{t'-1}) = d^0(\gamma^{t'}).$$

For $S \in \mathcal{S}_D$ with $v^S(\gamma^{t'-1}) = v_0^{S_i}(\gamma^{t'})$ the solution $\{\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}\}_{|[0,\ldots,t'-1]}$ obeys the inequalities

$$\sum_{j=0}^{t'-1} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'-1}),\ \left|\left\{j \in \{0, \ldots, t'-1\} \mid \overline{x}_j^S - \overline{x}_j > 0\right\}\right| \leq i^S(\gamma^{t'})$$

and $\overline{x}_{t'}^S = 0$, and thus

$$\sum_{j=0}^{t'} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'-1}) = d^S(\gamma^{t'}) \text{ and } \left|\left\{j \in \{0, \ldots, t'\} \mid \overline{x}_j^S - \overline{x}_j > 0\right\}\right| \leq i^S(\gamma^{t'}).$$

If $v^S(\gamma^{t'-1}) = v_+^{S_i}(\gamma^{t'})$ the inequalities

$$\sum_{j=0}^{t'-1} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'-1}) = d^S(\gamma^{t'}) - w_{t'}^S \text{ and}$$

$$\left|\left\{j \in \{0, \ldots, t'-1\} \mid \overline{x}_j^S - \overline{x}_j > 0\right\}\right| \leq i^S(\gamma^{t'-1}) = i^S(\gamma) - 1$$

are also satisfied and we obtain with $\overline{x}_{t'}^S = 1$,

$$\sum_{j=0}^{t'} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'}) \text{ and } \left|\left\{j \in \{0, \ldots, t'-1\} \mid \overline{x}_j^S - \overline{x}_j > 0\right\}\right| \leq i^S(\gamma^{t'}).$$

If $\gamma^{t'-1} \in \mathcal{Z}^+(\gamma^{t'})$, then $\overline{x}_{t'} = 1$, and $\overline{x}_{t'} \geq \overline{x}_{t'}^S$ for all $S \in \mathcal{S}_D$. Hence,

$$\left|\left\{j \in \{0, \ldots, t'\} \mid \overline{x}_{t'}^S - \overline{x}_{t'} > 0\right\}\right| \leq i^S(\gamma^{t'-1}) = i^S(\gamma^{t'}).$$

Since $d^0(\gamma^{t'-1}) = d^0(\gamma^{t'}) - w_{t'}^0$, we get

$$\sum_{j=0}^{t'} w_j^S \overline{x}_j = \sum_{j=0}^{t'-1} w_j^S \overline{x}_j^S + w_j^S \leq d^0(\gamma^{t'}).$$

For $S \in \mathcal{S}_D$ with $v^S(\gamma^{t'-1}) = v^S_-(\gamma^{t'})$ we obtain

$$\sum_{j=0}^{t'-1} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'-1}) = d^S(\gamma^{t'}) \text{ and}$$

$$\left|\left\{j \in \{0,\ldots,t'-1\} \mid \overline{x}_j - \overline{x}_j^S > 0\right\}\right| \leq j^S(\gamma^{t'-1})$$

and $\overline{x}_j^S = 0$, and thus

$$\sum_{j=0}^{t'} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'}) \text{ and } \left|\left\{j \in \{0,\ldots,t'\} \mid \overline{x}_j - \overline{x}_j^S > 0\right\}\right| \leq j^S(\gamma^{t'-1}) + 1 = j^S(\gamma^{t'}).$$

In the case of $v^S(\gamma^{t'-1}) = v^S_\sim(\gamma^{t'})$, the following inequalities follow from the construction:

$$\sum_{j=0}^{t'-1} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'-1}) = d^S(\gamma^{t'}) - w_{t'}^S,$$

$$\left|\left\{j \in \{0,\ldots t'-1\} \mid \overline{x}_j - \overline{x}_j^S > 0\right\}\right| \leq j^S(\gamma^{t'-1})$$

and $\overline{x}_j^S = 1$,

$$\sum_{j=0}^{t'} w_j^S \overline{x}_j^S \leq d^S(\gamma^{t'}) \text{ and}$$

$$\left|\left\{j \in \{0,\ldots,t'\} \mid \overline{x}_j - \overline{x}_j^S > 0\right\}\right| \leq j^S(\gamma^{t'-1}) = j^S(\gamma^{t'}).$$

Therefore, $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is a feasible solution of $\phi(\gamma^{t'})$. △

Since $\overline{x}, \overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ is a feasible solution, it remains to show within the next two claims that $g(\gamma^{\bar{t}}) = \Phi(\gamma^{\bar{t}})$.

Claim. The value $g(\gamma^{t'})$ equals the total profit of $\overline{x}$, if $\overline{x}^{S_1}, \ldots, \overline{x}^{S_r}$ are taken as recovery and $\alpha^S(\gamma^{t'})$ is added to each scenario profit $S \in \mathcal{S}$, i.e.,

$$g(\gamma^{t'}) = \sum_{i=0}^{t'} p_i^0 \overline{x}_i + \min_{S \in \mathcal{S}_D} \sum_{i=1}^{t'} p_i^S \overline{x}_i^S + \alpha^S(\gamma^{t'}).$$

Proof. Let $I_-^S := \{t \in \{1,\ldots,t'\} \mid \gamma^{t-1} \in \mathcal{Z}^-(\gamma^t) \text{ and } i^S(\gamma^{t-1}) = i^S(\gamma^t) - 1\}$ and $I_+^S := \{t \in \{1,\ldots,t'\} \mid \gamma^{t-1} \in \mathcal{Z}^+(\gamma^t) \text{ and } j^S(\gamma^{t-1}) = j^S(\gamma^t)\}$ for each $S \in \mathcal{S}$. Thus

$$\alpha^S(\gamma^0) = \alpha^S(\gamma^{t'}) + \sum_{i \in I_-^S} p_i^S + \sum_{i \in I_+^S} p_i^S.$$

Since

$$I_-^S = \left\{t \in \{1,\ldots,t'\} \mid \overline{x}_t = 0 \text{ and } \overline{x}_t^S = 1\right\} \text{ and}$$
$$I_+^S = \left\{t \in \{1,\ldots,t'\} \mid \overline{x}_t = 1 \text{ and } \overline{x}_t^S = 1\right\},$$

we obtain

$$\alpha^S(\gamma^0) = \alpha^S(\gamma^{t'}) + \sum_{i \in I_-^S} p_i^S + \sum_{i \in I_+^S} p_i^S = \alpha^S(\gamma^{t'}) + \sum_{i=1}^{t'} p_i^S \overline{x}_i^S.$$

Furthermore,

$$g(\gamma^t) = g(\gamma^{t-1}) + p_i^0 \cdot \overline{x}_i$$

due to the definition of $\overline{x}$ and thus

$$g(\gamma^{t'}) = g(\gamma^0) + \sum_{i=0}^{t'} p_i^0 \cdot \overline{x}_i.$$

Since

$$g(\gamma^0) = \min_{S \in \mathcal{S}_D} \alpha^S(\gamma^0) = \min_{S \in \mathcal{S}_D} \alpha^S(\gamma^{t'}) + \sum_{i=0}^{t'} p_i^S \overline{x}_i^S$$

we get

$$g(\gamma^{t'}) = \sum_{i=0}^{t'} p_i^0 \cdot \overline{x}_i + \min_{S \in \mathcal{S}_D} \alpha^S(\gamma^{t'}) + \sum_{i=0}^{t'} p_i^S \overline{x}_i^S.$$

△

As a consequence of these two claims $g(\gamma^{t'}) \leq \Phi(\gamma^{t'})$. Thus, it remains to show that $g(\gamma^{t'}) \geq \Phi(\gamma^{t'})$ to obtain a contradiction.

Claim. It holds $g(\gamma^{t'}) \geq \Phi(\gamma^{t'})$.

Proof. Let $\tilde{x}, \tilde{x}^{S_1}, \ldots, \tilde{x}^{S_r}$ be an optimal solution of $\phi(\gamma^{t'})$. According to this solution we define a feasible sequence $\overline{\gamma}^0, \ldots, \overline{\gamma}^{t'}$ with $\overline{\gamma}^{t'} = \gamma^{t'}$ in the following way:

$$d^0(\overline{\gamma}^{t-1}) = \begin{cases} d^0(\overline{\gamma}^t) & \text{if } \tilde{x}_t = 0 \\ d^0(\overline{\gamma}^t) - w_t^0 & \text{if } \tilde{x}_t = 1 \end{cases} \text{ and } v^S(\overline{\gamma}^{t-1}) = \begin{cases} v_0^S(\overline{\gamma}^t) & \text{if } \tilde{x}_t = 0,\ \tilde{x}_t^S = 0 \\ v_+^S(\overline{\gamma}^t) & \text{if } \tilde{x}_t = 0,\ \tilde{x}_t^S = 1 \\ v_-^S(\overline{\gamma}^t) & \text{if } \tilde{x}_t = 1,\ \tilde{x}_t^S = 0 \\ v_\sim^S(\overline{\gamma}^t) & \text{if } \tilde{x}_t = 1,\ \tilde{x}_t^S = 1. \end{cases}$$

Due to our assumption and the construction of $\overline{\gamma}^t$, $g(\overline{\gamma}^t) = \Phi(\overline{\gamma}^t)$ for $t = 0, \ldots, t' - 1$ and $\overline{\gamma}^{t'-1} \in \mathcal{Z}(\gamma^{t'})$. If $\overline{\gamma}^{t'-1} \in \mathcal{Z}^+(\gamma^{t'})$, then

$$\Phi(\gamma^{t'}) = \Phi(\overline{\gamma}^{t'-1}) + p_{t'}^0 = g(\overline{\gamma}^{t'-1}) + p_{t'}^0 \leq g(\gamma^{t'})$$

and if $\overline{\gamma}^{t'-1} \in \mathcal{Z}^-(\gamma^{t'})$, then

$$\Phi(\gamma^{t'}) = \Phi(\overline{\gamma}^{t'-1}) = g(\overline{\gamma}^{t'-1}) \leq g(\gamma^{t'}).$$

This is a contradiction. △

To sum up, any 0-1 point corresponding to a feasible sequence $\gamma^0, \ldots, \gamma^t$ determines an optimal solution of $\phi(\gamma^t)$. □

In the following subsection we will focus on the polyhedral structure of a (k, ℓ)-rrKP instance.

5.2.2. Extended Cover Inequalities

As pointed out by Crowder et al. [32] one motivation for studying the polytope of the knapsack problem is that valid inequalities for the knapsack polytope can be used as cutting planes for general 0-1 linear integer programs (ILP). The idea is to consider each individual constraint of a 0-1 ILP as a 0-1 knapsack constraint. Several classes of valid inequalities are known, such as the lifted cover inequalities (LCIs) of Balas [8] and Wolsey [98], the weight inequalities (WIs) of Weismantel [97], or the $(1,k)$-configurations introduced by Padberg [84]. For the class of (k,ℓ)-rrKP problems we will focus on extended cover inequalities which have been shown to be quite efficient in solving knapsack instances by Kaparis and Letchford [67].

The knapsack polytope is the convex hull of the set of solution vectors for a knapsack problem, i.e.,

$$\mathcal{K} = \operatorname{conv}\Big\{x \in \{0,1\}^n \mid \sum_{i\in N} w_i x_i \leq c\Big\}.$$

An important class of valid inequalities for the knapsack problem are the so-called minimal cover inequalities and their extensions. A subset $C \subseteq N$ is called a *cover* for $\mathcal{K}$ if $\sum_{j\in C} w_j > c$. A cover is *minimal* if $\sum_{j\in C\setminus\{i\}} w_j \leq c$ for every $i \in C$. For a (minimal) cover C the *(minimal) cover inequality*

$$\sum_{j\in C} x_j \leq |C| - 1$$

provides a valid inequality for $\mathcal{K}$. The minimal cover inequalities are facet-defining for $C = N$, but in general they are not. Yet, they can be lifted to define a facet of the knapsack polytope. One way to lift cover inequalities is based on an *extension* $E(C)$ of a cover C

$$E(C) = C \cup \{j \in N\setminus C \mid w_j \geq w_i, \text{ for all } i \in C\}.$$

The resulting *extended cover inequality* for $\mathcal{K}$ reads

$$\sum_{j\in E(C)} x_j \leq |C| - 1.$$

In general, it is not reasonable to generate all extended cover inequalities, but solve the LP-relaxation and find for a non-integer point a violated inequality. This problem is called the *separation problem*. Thus, in the case of (extended) cover inequalities we are interested in finding for a given fractional solution such an inequality that is violated or state that no such inequality exists. More formally, let $x^* \in [0,1]^n$ be a fractional solution with $\sum_{i\in N} w_i x_i \leq c$. The separation problem is to find a cover $C \subseteq N$, such that

$$\sum_{j\in C} x_j^* \geq |C|.$$

This task can be transformed into

$$\begin{aligned} & \sum_{j\in C} x_j^* \leq |C| - 1 \quad \forall\, C \text{ cover} \\ \Leftrightarrow \quad & \max_{C \text{ cover}} \sum_{j\in C} x_j^* - |C| \leq -1 \end{aligned}$$

Introducing 0-1 variables y_j taking the value 1 if and only if item j is part of the cover, $j \in N$, the last problem is equivalent to solving the knapsack problem

$$\begin{aligned} z^* = \max \sum_{j \in N} (x_j^* - 1) y_j \qquad (5.10) \\ \sum_{j \in N} w_j y_j \geq c + 1 \\ y_j \in \{0,1\} \ \forall\, j \in N. \end{aligned}$$

There is a violated cover inequality to the point x^* if and only if $z^* > -1$, as Crowder et al. noted in [32]. Yet, solving the problem is weakly **NP**-hard [45].

In the following part we show how the classical concept of cover inequalities can be generalized for the (k, ℓ)-rrKP polytope in a quite natural fashion.

RRKP Extended Cover Inequalities Before we start with the investigation of robust covers, we introduce the following notation: for a set $N = \{1, \ldots, n\}$, a function $f : N \to \mathbb{N}$, a set $X \subseteq N$ and an integer r we define $f(X) = \sum_{i \in X} f(i)$, $f(\max, X, r) = \max_{\substack{K \subseteq X \\ |K| \leq r}} f(K)$ and $f(\min, X, r) = \min_{\substack{K \subseteq X \\ |K| \geq r}} f(K)$.

For a given (k, ℓ)-rrKP instance the (k, ℓ)-recoverable robust knapsack polytope $\mathcal{K}_D(k)$ of a discrete scenario set $\mathcal{S}_D$ is the convex hull over all valid first-stage solutions, i.e.,

$$\mathcal{K}_D(k) := \operatorname{conv}\Big\{ x \in \{0,1\}^n \mid \sum_{i \in N} w_i^0 x_i \leq c^0 \text{ and } \min_{\substack{T \subseteq N \\ |T| \leq k}} \Big(\sum_{i \in N \setminus T} w_i^S x_i \Big) \leq c^S \Big\}.$$

Note that ℓ does not play a role in the feasibility of a first-stage solution. The polytope $\mathcal{K}_D(k)$ has full dimension if and only if $w_i^0 \leq c^0$ for all $i \in N$ and $(1-k) w_i^S \leq c^S$ for all $S \in \mathcal{S}$.

Following the concept of covers we call a set $C \subseteq N$ an *rrKP cover* if one of the next conditions holds

1. $w^0(C) \geq c^0 + 1$
2. there is a scenario $S \in \mathcal{S}_D$ with

$$w^S(C) - w^S(\max, C, k) \geq c^S + 1.$$

An rrKP cover C is *minimal* if

1. $w^0(C) - w^0(\min, C, 1) \leq c^0$ and
2. $w^S(C) - w^S(\max, C, k) - w^S(\min, C, 1) \leq c^S$ for all $S \in \mathcal{S}_D$.

and an rrKP cover defines the following *cover inequality*:

$$\sum_{i \in C} x_i \leq |C| - 1.$$

Theorem 5.2.5. *Let x be a 0-1 point. Then $x \in \mathcal{K}_D(k)$ if and only if x satisfies all minimal cover inequalities.*

Proof. ($\Rightarrow$): Let $x \in \mathcal{K}_D(k)$ and let us assume that there exists a minimal rrKP cover C such that

$$\sum_{i \in C} x_i = |C|.$$

Since C is a cover, there exists either a scenario $S \in \mathcal{S}_D$ such that

$$w^S(C) - w^S(\max, C, k) \geq c^S + 1$$

or

$$w^0(C) \geq c^0 + 1.$$

This is a contradiction to the feasibility of x.

($\Leftarrow$): Let $T_x = \{i \in N \mid x_i = 1\}$ be the support of x. Let us further assume that x is infeasible, i.e., either

$$w^0(T_x) \geq c^0 + 1$$

or there exists a scenario $S \in \mathcal{S}_D$ with

$$w^S(T_x) - w^S(\max, T_x, k) \geq c^S + 1.$$

Hence, T_x is an rrKP cover by definition. In a last step we modify T_x to become a minimal rrKP cover. For this reason we repeatedly remove an item $i \in T_x$ with $i = \arg\min_{i \in T_x} w_i^0$ if $w^0(T_x) \geq c^0 + 1$ or $i = \arg\min_{i \in T_x} w_i^S$ if $w^S(T_x) - w^S(\max, T_x, k) \geq c^S + 1$ for a scenario $S \in \mathcal{S}$, until the remaining set T_x' is a minimal rrKP cover. Thus, x violates the (minimum) cover inequality defined by T_x'. □

As in the case of the nominal knapsack polytope, a cover inequality is facet defining if a minimal rrKP cover C consists of N: let us consider the sets $N_i := N \backslash \{i\}$ for $i = 1, \ldots, n$. All these sets are feasible solutions to the knapsack instance, since $C = N$ is a minimal rrKP cover. Thus, N_i, $i = 1, \ldots, n$, are n affine independent solutions and C is therefore facet defining.

The concept of rrKP covers can be similarly extended to strengthen the cover inequalities as for the deterministic knapsack polytope. A canonical way to define an extension is by adding all items whose weight is greater than or equal to the highest not recovered item in an rrKP cover C. More formally, let C be an rrKP cover and S be a scenario such that the scenario weight inequality is violated by C. A *canonical extension* $\overline{E}^S(C)$ is given by

$$\overline{E}^S(C) = \{i \in N \mid w_i^S \geq w^S(\max, C, k+1) - w^S(\max, C, k)\} \cup C. \tag{5.11}$$

Yet, it even suffices for an item to be added to C if its weight exceeds the residual capacity according to the weight of the first $|C| - k - 1$ items with lowest weight and the weight of the item with the second highest not recovered item in C.

Definition 5.2.6. Let C be an rrKP cover and S be a scenario with $w^S(C) - w^S(\max, C, k) \geq c^S + 1$. An *extension* $E^S(C)$ of C according to S is defined by

$$\begin{aligned} E^S(C) = C \cup \{i \in N \mid\ & w_i^S \geq c^S - w^S(C) + w^S(\max, C, k+1) + 1, \\ & w_i^S \geq w^S(\max, C, k+2) - w^S(\max, C, k+1)\} \end{aligned}$$

and determines the *rrKP extended cover inequality*

$$\sum_{i \in E^S(C)} x_i \leq |C| - 1.$$

For the nominal knapsack problem the set of inequalities obtained by extending minimal covers is independent of the extension method, as we will show in the following: Let C be a minimal cover and $j_{\max} = \arg\max_{j \in C} w(C)$ and $i_{\min} = \arg\min_{i \in E(C) \setminus \overline{E}(C)} w(i)$. Hence, $C' = C \cup \{i_{\min}\} \setminus \{j_{\max}\}$ is a minimal cover with $|C'| = |C|$ and $\overline{E}(C') = E(C)$.

For the recoverable robust knapsack problem the canonical extensions define weaker inequalities than the rrKP extended cover inequalities as the following example indicates: In the instance described in Table 5.2 with 6 items $\{1, \ldots, 6\}$, no first-stage weights and two scenarios S_a and S_b and $k = 1$, the set $\overline{C} = \{1, 2, 4, 5\}$ is a minimal cover. If we extend $\overline{C}$ by the second method according to scenario S_a, $E^{S_a}(\overline{C}) = \{1, 2, 3, 4, 5, 6\}$ although $w^{S_a}(3) < w^{S_a}(4)$. But the set $C' = \{1, 2, 3, 5\}$ is not a minimal cover, since $w^{S_b}(C') - w^{S_b}(\max, C', 1) - w^{S_b}(\min, C', 1) > 3$. Also no other minimal cover induces a canonical extended cover inequality as strong as the one derived from $E^{S_a}(\overline{C})$.

$\mathcal{S} \setminus N$	1	2	3	4	5	6	c^S
S_a	2	2	3	4	8	9	6
S_b	10	1	5	2	1	0	3

Table 5.2.: The table shows the knapsack capacity and the weights for each item according to the different scenarios.

In the following lemma we prove that the rrKP extended cover inequalities are feasible.

Lemma 5.2.7. *Let C be an rrKP cover and $E^S(C)$ be an extension according to scenario S. Then*

$$\sum_{i \in E^S(C)} x_i \leq |C| - 1 \tag{5.12}$$

is a valid inequality for $\mathcal{K}_D(k)$.

Proof. Let us assume that there exists an integer point $x \in \mathcal{K}_D(k)$, a minimal cover C and a scenario S such that

$$\sum_{i \in E^S(C)} x_i = |C|.$$

Let $T_x = \{i \in N \mid x_i = 1\}$ be the support of x and i_a be an item in $T_x \cap E^S(C) \setminus C$ with minimum weight according to S. We furthermore order the items in C increasingly according to the weights of S, i.e., $w^S(i_1) \leq w^S(i_2) \leq \ldots \leq w^S(i_{|C|})$ and define $X = \{i_1, \ldots, i_{|C|-k-1}\}$. Since $w^S(i_a) \geq c^S - w^S(C) + w^S(\max, C, k+1) + 1$, $w^S(i_a) \geq w^S(\max, C, k+2) - w^S(\max, C, k+1)$ and $w^S(X \cup \{i_{|C|-k}\}) \geq c^S + 1$,

$$\begin{aligned} w^S(T_x) - w^S(\max, T_x, k) &= \min_{\substack{K \subseteq T_x \\ |K| \leq k}} w^S(T_x \setminus K) \\ &\geq w^S(X) + \min\{w^S(i_{|C|-k}), w^S(i_a)\} \\ &\geq c^S + 1. \end{aligned}$$

This is a contradiction to the feasibility of x. □

Separation of the RRKP Extended Cover Inequalities for $\mathcal{S}_D$ The separation problem is to find an inequality that is violated by a given non-integer point, or to prove that none exists. In our case, we are interested in finding for a given fractional solution $x^* \in [0,1]^n$ an rrKP extended cover inequality. Since the definition of a cover is based on the consideration of a single scenario or to the first-stage weight set, we can separately consider each scenario and hence are interested in finding an rrKP extended cover inequality according to one scenario $S \in \mathcal{S}_D$ or the first-stage weight constraint. For simplicity we drop the S on the variables, the weights and the knapsack capacity and set $k = 0$ when considering the first-stage weight constraint.

Our first approach uses integer programming and is based on the following observation: In order to find a violated extended cover inequality, it suffices to determine the items which are part of the cover and not recovered (i.e., not deleted to satisfy the capacity constraint). We call these items the *core of the cover*. In a second step all other items which exceed the weight of every item in the core of the cover, can be added to form possibly an extended rrKP cover. This is the case if more than k items are added to the core. Although these extra items are fixed as soon as the core of a cover is known, we introduce an integer program to determine both sets for two reasons: first, the number of additional items is crucial for the detection of an rrKP cover and secondly, even for nominal knapsack instances a fractional point x^* may not violate a cover inequality but an extended cover inequality.

For the IP formulation we introduce two different binary variables y_i and z_i for each item $i \in N$. The variable y_i determines whether item i is part of the core of the cover and z_i whether item i is added in the second step. In order to define a cover the inequality $\sum_{i \in N} w_i y_i \geq c + 1$ has to be obeyed by y. An item i is added as extension of the core ($z_i = 1$, $i \in N$) if its weight exceeds the weight of every item in the core. Note that if an item with a sufficient large weight is added to the recovery or the extension, all items with larger weights may also be added to the extension as they are exchangeable with the first one. In order to efficiently implement this condition, we group the items according to their weights: Let $0 \leq w_{i_1} < w_{i_2} < \ldots < w_{i_\rho} \leq c$ be an ordering of all different item weights occurring in the scenario and $R := \{1, \ldots, \rho\}$. We define $N(r) := \{j \in N \mid w_j = w_{i_r}\}$ for all $r \in R$. Then $z_i \leq z_j$ is valid for all $i \in N(r)$, $j \in N(r+1)$, $r = 1, \ldots, \rho - 1$. Hence, we obtain the following ILP

$$
\begin{aligned}
\text{(rrEC-IP)} \qquad \max \sum_{j \in N} (x_j^* - 1) y_j + \sum_{j \in N} x_j^* z_j - k & \\
\sum_{j \in N} w_j y_j &\geq c + 1 \\
y_j + z_j &\leq 1 \qquad \forall\, j \in N \\
z_i &\leq z_j \qquad \forall\, i \in N(r),\ j \in N(r+1),\ r \in R \\
y_j, z_j &\in \{0,1\} \ \forall\, j \in N.
\end{aligned}
$$

An optimal solution defines a violated rrKP extended cover inequality if the value of the objective function is greater than -1. Note that in this case more than k items are added to the core of the cover.

Based on this integer formulation and dynamic programming we will finally introduce a pseudo-polynomial algorithm solving the separation problem for the extended cover inequalities. We define $U = \cup_{i=1}^n \{w_i\}$ and $D = \sum_{i=1}^n w_i$. For all $t = 1, \ldots, n$, $d = 0, \ldots, D$ and $\omega \in U$ we solve the problem to find a part of the core of a cover and added items within the set $\{1, \ldots, t\}$ such that the violation of x^* is maximized, the weight of the core equals d, their weights are

below ω and the weight of all items added in the extension is greater than or equal to ω. More formally, we consider the following function

$$\begin{aligned} f_\omega(t,d) = \max \sum_{i=1}^{t}(x_i^*-1)y_i + \sum_{i=1}^{t} x_i^* z_i \\ \sum_{i=1}^{t} w_i y_i &= d \\ y_i + z_i &\leq 1 \qquad \forall\, i = 1,\dots,t \\ z_i &= 0 \qquad \forall\, i : w_i < \omega \\ y_i &= 0 \qquad \forall\, i : w_i > \omega \\ y_i, z_i &\in \{0,1\} \; \forall\, i = 1,\dots,t. \end{aligned}$$

The optimal solution to the separation problem is given by

$$\max_{\substack{\omega \in U \\ d \geq c+1}} f_\omega(n,d).$$

The function $f_\omega(1,d)$ can easily be solved via three case distinctions:

Case 1: if $w_1 = d$ and $\omega > d$, then $f_\omega(1,d) = (x_1^* - 1)$

Case 2: if $d = 0$ and $w_1 \geq \omega$, then $f_\omega(1,d) = x_1^*$

Case 3: otherwise, $f_\omega(1,d) = -\infty$.

For all other combinations of $t \geq 2$, $d = 0,\dots,D$ and $\omega \in U$, the general recursive formula referring to the three cases above holds

$$\begin{aligned} f_\omega(t,d) = \max\{ & f_\omega(t-1,d), \\ & f_\omega(t-1,d-w_t) + (x_t^* - 1) && \text{if } w_t \leq \omega, \\ & f_\omega(t-1,d) + x_t^* && \text{if } w_t \geq \omega\}. \end{aligned}$$

In other words we decide, whether t is not in the core and not added to the extension; whether t is part of the core of the cover; or whether t is added to the extension taking into account the weight bound imposed by ω. Obviously we can construct out of an optimal solution of $f_\omega(t-1,d)$, $f_\omega(t-1,d-w_t)$, and $f_\omega(t-1,d)$ three feasible solutions for $f_\omega(t,d)$. Note that the recursion formula is valid, since otherwise we obtain a contradiction to the optimality of $f_\omega(1,d')$. The run-time of this approach is in $\mathcal{O}(D \cdot n^2)$, since $|U| \leq n$.

5.2.3. Computational Experiments

In this section we present computational results on the recoverable robust knapsack problem with discrete scenarios. We start with an investigation of the gain of recovery, i.e., the increase in the profit obtained by allowing recovery compared with the profit obtained in the robust case. Afterwards we study the impact of rrKP extended cover inequalities on the relaxed recoverable robust knapsack problem.

The considered (k,ℓ)-rrKP instances are modifications of multi-dimensional knapsack instances from the OR-library [9] created by Chu and Beasley. The original nine problem sets

are characterized by the number of items $n \in \{100, 250, 500\}$, and the number of knapsack constraints $m \in \{5, 10, 30\}$. For each n-m combination Chu and Beasley generated 30 instances with different tightness ratios $\alpha \in \{0.25, 0.5, 0.75\}$. This ratio is given by $c_j = \alpha \sum_{i=1}^{n} w_{ji}$, where w_{ji} denotes the weight of item i in the jth constraint and the c_j the capacity of the jth constraint, $i \in \{1, \ldots, n\}$, $j \in \{1, \ldots, m\}$. In order to obtain (k, ℓ)-rrKP instances, we kept the set of items, treated the first knapsack constraint as first-stage weight constraint and used each further constraint as scenario weight constraint for an individual discrete scenario. Finally, we divided the profit of each item according to the multi-dimensional knapsack instance into first-stage profit and scenario profit in the following way: 70% of the profit is assigned to the items as first-stage profit and between 20% and 40% of the profit is chosen uniformly at random in each scenario as profit. Thus, the scenario profits range from 29% to 57% of the corresponding first-stage profits. We just considered the 10 instances of each n-m combination with a tightness ratio of $\alpha = 0.5$. The parameter k and ℓ are chosen as fractions of the number of items $k, \ell \in \{0.00, 0.01, 0.05, 0.10, 0.25, 0.05, 1\}$, i.e., if we consider 100 items and $k = 0.01$, then we are allowed to delete up to one item.

We implemented the drrKP-IP formulation (page 88) of the recoverable robust knapsack problem with discrete scenarios in C++ using SCIP 1.2.0 [2] as branch-and-cut framework with IBM ILOG CPLEX 12.1 as underlying LP solver. Applying the callback functionality of SCIP, we added a separator for rrKP extended cover inequalities which exactly generates violated cuts by solving the rrEC-IP formulation (page 101). In our first study concerning the gain of recovery the separator is turned off.

The experiments were carried out on a Linux machine with 2.93 GHz Intel Xeon W3540 CPU and 12 GB RAM. A time limit of one hour was set for solving each problem instance. All other solver settings were left at their defaults.

Gain of Recovery The gain of recovery is a measure for the impact of recovery on the objective function in a robust setting. Applied to the recoverable robust knapsack problem we obtain the following definition: let $X_{(k,\ell)}$ be an optimal first-stage solution of a given (k, ℓ)-rrKP instance according to the recovery parameters $k \geq 0$ and $\ell \geq 0$, where k denotes the number of items that may be removed from a first-stage solution and ℓ denotes the number of items that may be added to a first-stage solution. Then, $X_{(0,0)}$ is an optimal robust solution in the classical sense which is equivalent to an optimal solution of the multi-dimensional knapsack problem. The *gain of recovery* $\text{gain}(k, \ell)$ is defined as

$$\text{gain}(k, \ell) = \frac{p(X_{(k,\ell)})}{p(X_{(0,0)})}.$$

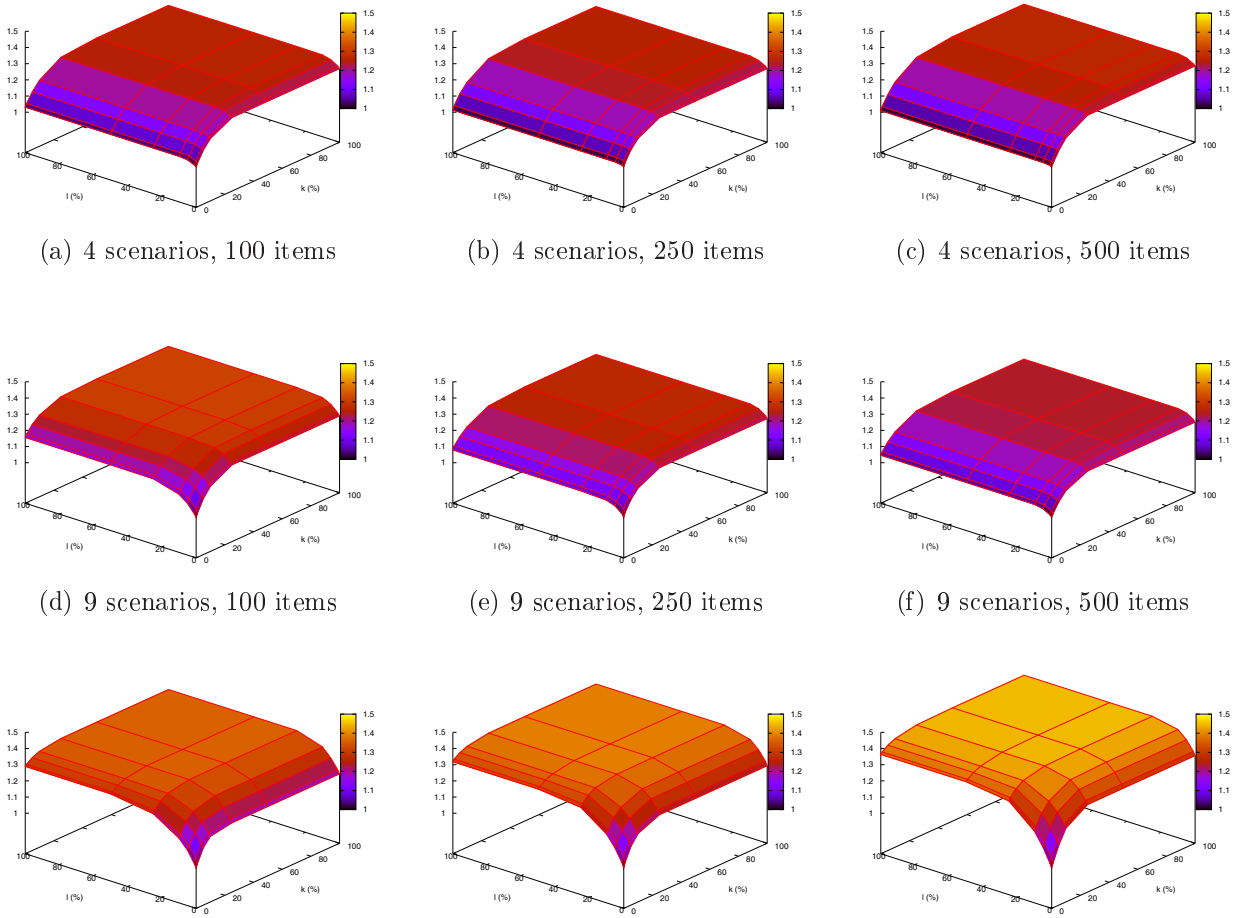
Note that $\text{gain}(k, \ell) \geq \text{gain}(k', \ell')$ with $k \geq k'$ and $\ell \geq \ell'$, since $X_{(k',\ell')}$ is a feasible solution of the instance with recovery parameters of k and ℓ, and thus $\text{gain}(k, \ell) \geq 1$.

In our computational study we considered the nine combinations of n and m as described above with all parameter combinations of k and ℓ given as fractions of the number of items, $k, \ell \in \{0.00, 0.01, 0.05, 0.10, 0.25, 0.05, 1\}$. For each of these 63 settings we computed the objective value of an optimal solution, respectively the best known solution within the time limit of one hour, for 10 different instances. Since the achieved optimality gaps are very small, we did not distinguish these cases. Table 5.3 shows the average values of the gain of recovery w.r.t. the selected values of k and ℓ and the combination of n and m. A graphical representation is given in Figure 5.1.

$\lvert\mathcal{S}\rvert$		4			9			29		
$\lvert N\rvert$		100	250	500	100	250	500	100	250	500
k	ℓ									
0.00	0.00	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	0.01	1.0093	1.0076	1.0059	1.0185	1.0195	1.0133	1.0233	1.0295	1.0266
	0.05	1.0268	1.0135	1.0104	1.0699	1.0542	1.0370	1.0933	1.1057	1.1112
	0.10	1.0319	1.0137	1.0104	1.1099	1.0726	1.0463	1.1504	1.1752	1.1929
	0.25	1.0320	1.0137	1.0104	1.1545	1.0798	1.0474	1.2508	1.2899	1.3295
	0.50	1.0320	1.0137	1.0104	1.1563	1.0797	1.0474	1.2888	1.3183	1.3635
	1.00	1.0320	1.0137	1.0104	1.1568	1.0799	1.0470	1.2901	1.3187	1.3637
0.01	0.00	1.0210	1.0248	1.0206	1.0244	1.0286	1.0212	1.0267	1.0323	1.0285
	0.01	1.0297	1.0314	1.0260	1.0421	1.0455	1.0329	1.0496	1.0593	1.0539
	0.05	1.0454	1.0363	1.0298	1.0907	1.0759	1.0540	1.1134	1.1332	1.1360
	0.10	1.0495	1.0365	1.0298	1.1287	1.0922	1.0625	1.1718	1.1998	1.2157
	0.25	1.0496	1.0365	1.0298	1.1700	1.0983	1.0632	1.2670	1.3068	1.3446
	0.50	1.0496	1.0365	1.0298	1.1723	1.0982	1.0631	1.2969	1.3299	1.3743
	1.00	1.0496	1.0365	1.0299	1.1720	1.0980	1.0632	1.2983	1.3304	1.3745
0.05	0.00	1.0840	1.0856	1.0826	1.0962	1.0905	1.0773	1.1054	1.1135	1.1224
	0.01	1.0906	1.0908	1.0867	1.1103	1.1037	1.0868	1.1230	1.1368	1.1456
	0.05	1.1034	1.0943	1.0888	1.1509	1.1273	1.1034	1.1805	1.1993	1.2177
	0.10	1.1056	1.0943	1.0888	1.1840	1.1390	1.1082	1.2317	1.2554	1.2829
	0.25	1.1056	1.0943	1.0888	1.2163	1.1421	1.1083	1.3073	1.3447	1.3848
	0.50	1.1056	1.0943	1.0888	1.2165	1.1421	1.1083	1.3216	1.3541	1.4002
	1.00	1.1056	1.0943	1.0888	1.2168	1.1422	1.1083	1.3231	1.3545	1.4004
0.10	0.00	1.1359	1.1367	1.1404	1.1533	1.1417	1.1267	1.1557	1.1725	1.2064
	0.01	1.1421	1.1415	1.1440	1.1663	1.1531	1.1347	1.1724	1.1945	1.2264
	0.05	1.1550	1.1450	1.1458	1.2024	1.1726	1.1481	1.2279	1.2503	1.2858
	0.10	1.1570	1.1450	1.1458	1.2303	1.1805	1.1503	1.2749	1.2998	1.3383
	0.25	1.1570	1.1450	1.1457	1.2542	1.1823	1.1502	1.3361	1.3702	1.4153
	0.50	1.1570	1.1450	1.1457	1.2544	1.1821	1.1501	1.3419	1.3725	1.4201
	1.00	1.1570	1.1450	1.1458	1.2542	1.1821	1.1502	1.3420	1.3731	1.4203
0.25	0.00	1.2047	1.2049	1.2191	1.2246	1.2074	1.1863	1.1783	1.2288	1.2909
	0.01	1.2146	1.2150	1.2271	1.2387	1.2214	1.1976	1.1983	1.2519	1.3112
	0.05	1.2327	1.2284	1.2375	1.2747	1.2424	1.2160	1.2566	1.3090	1.3664
	0.10	1.2384	1.2299	1.2379	1.2963	1.2494	1.2193	1.3111	1.3543	1.4100
	0.25	1.2386	1.2299	1.2378	1.3067	1.2501	1.2194	1.3610	1.3940	1.4450
	0.50	1.2386	1.2299	1.2379	1.3067	1.2502	1.2194	1.3611	1.3940	1.4450
	1.00	1.2386	1.2299	1.2379	1.3068	1.2502	1.2194	1.3613	1.3941	1.4451
0.50	0.00	1.2056	1.2054	1.2217	1.2250	1.2088	1.1869	1.1787	1.2287	1.2917
	0.01	1.2170	1.2168	1.2317	1.2401	1.2232	1.1992	1.1986	1.2522	1.3127
	0.05	1.2427	1.2393	1.2539	1.2788	1.2521	1.2258	1.2568	1.3099	1.3718
	0.10	1.2555	1.2497	1.2641	1.3043	1.2651	1.2370	1.3110	1.3573	1.4187
	0.25	1.2575	1.2508	1.2646	1.3164	1.2666	1.2375	1.3617	1.3958	1.4501
	0.50	1.2575	1.2508	1.2647	1.3164	1.2667	1.2375	1.3618	1.3958	1.4501
	1.00	1.2575	1.2508	1.2647	1.3164	1.2664	1.2374	1.3618	1.3959	1.4501
1.00	0.00	1.2056	1.2054	1.2218	1.2254	1.2087	1.1871	1.1793	1.2293	1.2918
	0.01	1.2170	1.2169	1.2317	1.2399	1.2235	1.1995	1.1994	1.2529	1.3131
	0.05	1.2427	1.2393	1.2540	1.2787	1.2525	1.2259	1.2582	1.3103	1.3720
	0.10	1.2555	1.2497	1.2642	1.3042	1.2652	1.2370	1.3115	1.3573	1.4188
	0.25	1.2575	1.2508	1.2648	1.3164	1.2668	1.2376	1.3617	1.3959	1.4501
	0.50	1.2575	1.2508	1.2648	1.3163	1.2667	1.2375	1.3617	1.3959	1.4501
	1.00	1.2575	1.2508	1.2648	1.3164	1.2668	1.2375	1.3619	1.3959	1.4502

Table 5.3.: Gain of recovery gain(k, ℓ) for selected values of k and ℓ. Averages of 10 instances are shown for 100, 250, 500 items and 4, 9, 29 scenarios.

Fixing the number of items we observe a rise in the gain of recovery when the number of scenarios increases (e.g., compare Figures 5.1(a), 5.1(d), and 5.1(g)). This can be explained as follows: the profit achieved by a robust solution decreases depending on to the number of scenarios. On the other hand, the scenario profit remains more or less the stable.



(a) 4 scenarios, 100 items (b) 4 scenarios, 250 items (c) 4 scenarios, 500 items

(d) 9 scenarios, 100 items (e) 9 scenarios, 250 items (f) 9 scenarios, 500 items

(g) 29 scenarios, 100 items (h) 29 scenarios, 250 items (i) 29 scenarios, 500 items

Figure 5.1.: Gain of recovery gain(k, ℓ) for selected values of k and ℓ. Averages of 10 instances are shown for 100, 250, 500 items and 4, 9, 29 scenarios.

Next, we compare the influence of the parameters k and ℓ on the gain of recovery. According to our computational results an increase of k leads to a strong increase in the scenario profits. This is reasonable since the first-stage profit exceeds the scenario profit and items with heavy weights but high profits can be removed in the second stage. The influence of ℓ on the gain of recovery seems to vary with the number of scenarios. In the case of four scenarios almost no improvement is achieved. On the other hand, the additional profit of $\ell = 1$ (and $k = 0$) exceeds the additional profit of $k = 1$ (and $\ell = 0$) for 29 scenarios independently of the number of items.

In summary, allowing recovery produces a gain of up to 45% (e.g., for $k = \ell = 0.5$). But even a rather limited recovery of $k = \ell = 0.1$ achieves an additional profit ranging from 15% to 33% compared to the robust case.

RRKP Extended Cover Inequalities In the following we present the results of our second study where we investigate the effectiveness of rrKP extended cover inequalities using the canonical extension (5.11). In order to separate violated rrKP extended cover inequalities exactly (but still within a time frame of one hour time per instance), we implemented the rrEC-IP formulation (page 101) of the corresponding separation problem. Whenever our separator is called the first-stage weight constraint is tested. If no violated extended cover is found, all scenarios are tested beginning with the last scenario which provided a violated cut. As soon as a violated rrKP extended cover inequality has been determined, it is added to the LP and the separation round is aborted. Hence, we separate at most one cut per call. Furthermore, only the root node of the recoverable robust knapsack problem is solved in this study.

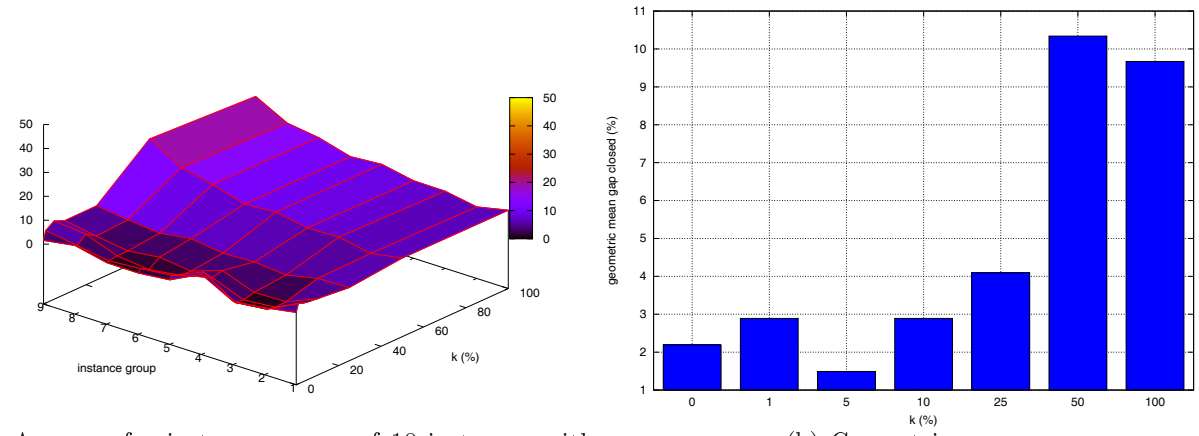
For each generated (k, ℓ)-rrKP instance we investigated two different methods based on LP-relaxations to obtain upper bounds on the total profit: We started by solving the canonical LP-relaxation without any (external or internal) separators to determine the integrality gap of the instance. In a second step we solved the same LP-relaxation this time with an exact separation of violated rrKP extended cover inequalities and again evaluated the integrality gap. The impact of the separation routine can be seen by comparing these two values.

m	4			9			29			
n	100	250	500	100	250	500	100	250	500	geom.
k										mean
0.00	5,27	3,02	1,53	8,54	1,98	0,63	0,76	3,69	1,61	2,19
0.01	8,60	2,00	0,62	9,09	3,46	2,15	1,11	3,28	5,37	2,89
0.05	7,47	1,25	0,07	5,22	1,87	0,08	1,57	6,42	7,77	1,49
0.10	6,99	1,52	1,20	6,09	1,64	3,24	1,62	3,65	5,89	2,89
0.25	5,60	5,21	2,42	3,71	3,11	4,95	3,04	4,36	6,03	4,09
0.50	7,87	5,61	8,20	7,78	10,16	9,29	13,28	16,04	23,90	10,34
1.00	7,74	4,91	7,25	7,40	10,00	9,05	12,70	14,60	21,56	9,67
geom. mean	6,98	2,88	1,40	6,57	3,52	2,03	2,72	6,03	7,42	

Table 5.4.: Gap closed for selected values of k. Averages are shown for the number of scenarios $m \in \{4, 9, 29\}$ and number of items $n \in \{100, 250, 500\}$, in the case of an exact separation of rrKP extended cover inequalities.

Table 5.4 reports to which percentage an additional improvement in the integrality gap is achieved by rrKP extended cover inequalities for selected values of k. Note that violated rrKP extended cover inequalities do not depend on the parameter ℓ. As in the study of the gain of recovery, all values are given as the average over 10 instances with the same number of items and scenarios. In addition, the geometric means over all instances is shown for fixing either the value of k or the number of items and scenarios.

Finally, in Figure 5.2(a) the average percental gap closed is described for the selected values of k and each combination of n and m. For a better illustration, we divide the instances according to the number of items and scenarios into 9 different groups corresponding to the columns in Table 5.4, e.g., instances in group 1 have 4 scenarios and 100 items, instances in group 2 have 4 scenarios and 250 items. In Figure 5.2(b) the geometric mean of the group averages are shown depending on k.



(a) Averages for instance groups of 10 instances with same number of scenarios and number of items

(b) Geometric means

Figure 5.2.: Integrality gap closed by separating violated rrKP extended cover inequalities.

An evaluation of the data shows that the geometric mean of the integrality gap closed lies in the range from 1.49% ($k = 0.05$) to 10.34% ($k = 0.50$). Considering all instances with 4 (9, 29) scenarios it varies from 0.07% (0.08%, 0.76%) to 8.60% (10.16%, 23.90%). This suggests that an increase in the number of scenarios results in better bounds. Unfortunately, there is no clear dependency. For example, the average gap closed for instances with 100 items and $k = 0.10$ is 6.99%, 6.09%, and 1.62% for 4, 9, and 29 scenarios, respectively. But considering the instances with 250 items an improvement of 1.52%, 1.64%, and 3.65% is achieved. In other cases the changes are neither monotonic increasing nor decreasing.

In summary, this study shows that the lower bound obtained via LP-relaxation improves by adding all violated rrKP extended cover inequalities. In the best case the gap was further closed by 23.90%. In more than 50% of all considered settings at least 5% improvement was obtained. In addition, the overall computational time spend for one run of this study was less than half an hour. Hence, the separation of violated rrKP extended cover cuts has a high potential to tighten the linear relaxation of the recoverable robust knapsack problem and to speed-up the solving process significantly.

5.3. Interval Scenarios

In the interval scenario case the scenario set is implicitly described. For given lower and upper bounds $\underline{p}, \overline{p}, \underline{w}, \overline{w} \in \mathbb{N}$ each scenario $S \in \mathcal{S}_I$ has to fulfill $p^S \in [\underline{p}, \overline{p}]$ and $w^S \in [\underline{w}, \overline{w}]$. Any (k, ℓ)-rrKP instance with interval scenarios can be transformed to a (k, ℓ)-rrKP instance with one discrete scenario $\overline{S}$. The profit of this function is defined by $p_i^{\overline{S}} = \underline{p}_i$ and the weight function by $w_i^{\overline{S}} = \overline{w}_i$ for every item i of the item set $N = \{1, \ldots, n\}$. Hence, the interval scenario problem is weakly **NP**-hard, but can be solved in pseudo-polynomial time (Theorem 5.2.4).

5.4. Γ-Scenarios

The Γ-scenario set $\mathcal{S}_\Gamma$ limits the power of interval scenarios by allowing at most Γ values of the scenario weight- and profit function to change from the lower weight bound and the upper

profit bound, $\Gamma \in \mathbb{N}$. More formally, for each item $i \in N$, $N = \{1, \dots, n\}$, two intervals $[\overline{p}_i - \hat{p}_i, \overline{p}_i]$ and $[\underline{w}_i, \underline{w}_i + \hat{w}_i]$ are given with $\underline{w}_i + \hat{w}_i \leq c$. All values $\underline{w}_i, \hat{w}_i, \overline{p}_i$ and $\hat{p}_i$ are assumed to be positive integers, $\underline{w}$ is the nominal weight and $\hat{w}$ the maximum deviation for each item, as $\overline{p}$ is the nominal profit and $\hat{p}$ its maximum deviation. In any scenario $S \in \mathcal{S}_\Gamma$ the weight and the profit of an item i lies within the intervals $[\underline{w}_i, \underline{w}_i + \hat{w}_i]$ and $[\overline{p}_i - \hat{p}_i, \overline{p}_i]$. As a slight variation of Bertsimas and Sims definition of Γ-scenarios [15], there are at most Γ values w_i^S and p_i^S, $i \in N$, which vary from the lower bound or the upper bound respectively, i.e., $|J| \leq \Gamma$ with $J = \{i \in N \mid w_i^S - \underline{w}_i > 0 \text{ or } \overline{p}_i - p_i^S > 0\}$. This definition forces all scenarios to focus their disturbances on Γ items. In contrary to the discrete scenario set we assume the knapsack capacity c to be constant for every scenario but independent of the first-stage capacity c^0.

Obviously, the (k, ℓ)-rrKP is weakly **NP**-hard. Yet, we did not succeed in either constructing a pseudo-polynomial algorithm for this problem or providing a proof for strong **NP**-hardness. Since the number of scenarios in $\mathcal{S}_\Gamma$ has exponential size in the number of items, we start by investigating the problem of testing a given set of items for its feasibility, i.e., finding a scenario $S \in \mathcal{S}_\Gamma$, such that the induced weight for a given first-stage solution is maximized where the k heaviest items are already deleted. This problem can be solved efficiently (Subsection 5.4.1) and can be used to obtain an ILP-formulation of polynomial size for the (k, ℓ)-rrKP with Γ-scenarios and $\overline{p}_i = \hat{p}_i = 0$. In the last subsection we focus again on (extended) cover inequalities and their separation.

5.4.1. An Optimal Γ-Strategy

In this subsection we consider the problem of finding a scenario $S \in \mathcal{S}_\Gamma$ that imposes the maximum weight on a given first-stage solution. More formally, let $X = \{1, \dots, n'\} \subseteq N$ be a set of items. For given parameters $\Gamma \in \mathbb{N}$ and $k \in \mathbb{N}$ we define the *weight* of a subset $\overline{X} \subseteq X$ as

$$\text{weight}(\overline{X}) = \sum_{i \in \overline{X}} \hat{w}_i - \max_{\substack{Y \subseteq X \\ |Y| \leq k}} \Big(\sum_{i \in Y} \underline{w}_i + \sum_{i \in Y \cap \overline{X}} \hat{w}_i \Big).$$

A *maximum weight set* X_Γ^k is a subset of X with $|X_\Gamma^k| \leq \Gamma$ and with maximum weight. The *maximum weight set problem* is to find for a given set X, and parameters Γ and k, a maximum weight set X_Γ^k.

As the following example indicates, there is no inclusion relation between optimal solutions of a maximum weight set problem for different Γ values, i.e., in general $X_\Gamma^k \subsetneq X_{\Gamma+1}^k$.

Example 5.4.1. Consider the set $X = \{1, \dots, 4\}$ with weights and deviation given in Table 5.5 and $k = 1$.

i	1	2	3	4
$\underline{w}_i$	3	3	10	10
$\hat{w}_i$	2	2	5	5

Table 5.5.: A maximum weight set instance.

For $\Gamma = 1$, the sets $X' = \{1\}$ and $X'' = \{2\}$ are the maximum weight sets for this instance with $\text{weight}(X') = -8$. On the other hand, $\bar{X} = \{3, 4\}$ is the maximum weight set with $\text{weight}(\bar{X}) = -5$ for $\Gamma = 2$, whereas the sets $\{1, 3\}$, $\{1, 4\}$, $\{2, 3\}$ and $\{2, 4\}$ have a weight of -8. ⊞

Algorithm 5.1 Maximum weight set

Input : item set $X = \{1, \dots, n'\}$, $\underline{w}_i \in \mathbb{N}$ and $\hat{w}_i \in \mathbb{N}$ for all $i \in X$, $\Gamma \in \mathbb{N}$, $k \in \mathbb{N}$
Output : maximum weight set X_Γ^k

Set $U = \{\underline{w}_i + \hat{w}_i \mid i \in X\} \cup \{\underline{w}_i \mid i \in X\} \cup \{0\}$
forall $u \in U$ **do**
  Set $w_i(u) = \min\{\hat{w}_i, -\underline{w}_i + u\}$ for all $i \in X$.
  Set $X^-(u) = \{i \in X \mid w_i(u) < 0\}$.
  Select in $X(u)$ the (at most) Γ items $u \in X$ with largest $w_i(u) \geq 0$.
  Compute

$$c(u) = \sum_{i \in X(u)} w_i(u) + \sum_{i \in X^-(u)} w_i(u) - k \cdot u.$$

return $X_\Gamma^k = X(u_{\max})$ with $u_{\max} = \arg\max_{u \in U} c(u)$.

Yet, the problem can be solved in polynomial time by Algorithm 5.1. This algorithm exploits the structure of an ILP-formulation of the maximum weight set problem. Details are provided in the proof of Theorem 5.4.2.

Theorem 5.4.2. *Let X be a set of items, $\underline{w}_i$ be the integer nominal weight and $\hat{w}_i$ be the maximal deviation for all $i \in X$ and Γ, $k \in \mathbb{N}$. Then Algorithm 5.1 computes a maximum weight set X_k^Γ in $\mathcal{O}(|X|^2)$.*

Proof. Let $X = \{1, \dots, n'\}$ be some item set with nominal weights $\underline{w}_i$ and maximum deviation values $\hat{w}_i$ for all $i \in X$. Let Γ and k be some positive integers. We will prove that Algorithm 5.1 computes an optimal solution of an IP-formulation of this maximum weighted set instance. We start with the following IP

$$\max \Big(\sum_{i=1}^{n'} \hat{w}_i y_i \quad - \quad \max \sum_{i=1}^{n'} (\underline{w}_i + \hat{w}_i y_i) z_i \Big)$$
$$\sum_{i=1}^{n'} y_i \leq \Gamma$$
$$y_i \in \{0,1\} \; \forall \, i \in X$$
$$\sum_{i=1}^{n'} z_i \leq k \tag{5.13}$$
$$z_i \in \{0,1\} \; \forall \, i \in X \tag{5.14}$$

The variables y_i represent the choice, whether an item i is in the maximum weight set, and z_i, whether the item i is recovered. Given a vector y, the remaining ILP can be solved as LP, relaxing the integrality constraint. Since the matrix is totally unimodular the optimal value of the ILP and the LP are the same. By dualizing it, we obtain

$$\begin{aligned} \min ku + \sum_{i=1}^{n'} v_i & \\ u + v_i \geq \underline{w}_i + \hat{w}_i y_i & \;\; \forall i \in X \\ u \geq 0, \, v_i \geq 0 & \;\; \forall i \in X. \end{aligned}$$

The dual variable u corresponds to constraint (5.13) and v_i to the upper bound on z_i for $i = 1, \dots, n'$. Replacing the primal version by the dual, we get a new ILP-formulation of the maximum weight set problem

$$
\begin{aligned}
(\text{ILP}^*) \qquad \max \sum_{i=1}^{n'} \hat{w}_i y_i - k \cdot u - \sum_{i=1}^{n'} v_i & \\
\sum_{i=1}^{n'} y_i &\leq \Gamma \\
\hat{w}_i \cdot y_i - u - v_i &\leq -\underline{w}_i \quad \forall i \in X \\
v_i &\geq 0 \quad \forall i \in X \\
u &\geq 0 \\
y_i &\in \{0,1\} \ \forall i \in X.
\end{aligned}
$$

We parametrize (ILP*) by the value of u and denote with $z(u)$ the optimal value. In the remaining proof we will first show that $z(u)$ can be combinatorial computed for a given u, as described in the subroutine of Algorithm 5.1. Secondly we prove that the optimal value of the (ILP*) is given by $z(u^*) = \max_{u \in U} z(u)$ with $U = \{\underline{w}_i + \hat{w}_i \mid i \in X\} \cup \{\underline{w}_i \mid i \in X\} \cup \{0\}$.

Let $u' \geq 0$ be some value. As in Algorithm 5.1 we define $w_i(u') = \min\{\hat{w}_i, -\underline{w}_i + u'\}$ for all $i \in \{1, \dots, n'\}$, $X^-(u') = \{i \in X \mid w_i(u') < 0\}$, $X(u') \subseteq X \backslash X^-(u')$ with $|X(u')| \leq \Gamma$ and $\sum_{i \in X(u')} w_i(u')$ maximal, and

$$
c(u') = \sum_{i \in X(u')} w_i(u') + \sum_{i \in X^-(u')} w_i(u') - k \cdot u'.
$$

We show $z(u') = c(u')$. Let $y_i^* \in \{0,1\}$ and $v_i^* \geq 0$, for all $i \in X$, be an optimal solution of the (ILP*) with $u = u'$. If $-\underline{w}_i + u' < 0$ for some $i \in X$, $\hat{w}_i \cdot y_i^* - v_i^* = -\underline{w}_i + u'$ due to the coefficients of y_i^* and v_i^* in the objective. If $y_x^* = 1$ for an item x with $-\underline{w}_x + u' < 0$, the solution

$$
\overline{y}_i = \begin{cases} 0 & i = x \\ y_i^* & \text{otherwise} \end{cases} \text{ and } \overline{v}_i = \begin{cases} -\underline{w}_x + u' & i = x \\ v_i^* & \text{otherwise} \end{cases}
$$

is also an optimal solution. Therefore, we can assume that w.l.o.g. $y_i^* = 0$ for all $i \in X^-(u')$.

Let $X(y^*) = \{i \in X \mid y_i^* = 1\}$. Then $X(y^*) \subseteq X \backslash X^-(u)$ with $|X(y^*)| \leq \Gamma$. If $y_i^* = 0$ for $i \in X \backslash X^-(u)$, then $v_i^* = 0$. If $y_i^* = 1$, $\hat{w}_i - v_i \leq -\underline{w}_i + u'$. Hence, if $\hat{w}_i > -\underline{w}_i + u'$ for some $i \in X$, $v_i^* = \hat{w}_i + \underline{w}_i - u'$ and $v_i^* = 0$ otherwise. Therefore,

$$
\begin{aligned}
z(u') &= \sum_{i=1}^{n'} \hat{w}_i y_i^* - \sum_{i=1}^{n'} v_i^* - k \cdot u' \\
&= \sum_{i \in X^-(u')} (-\underline{w}_i + u') + \sum_{i \in X(y^*)} \min\{-\underline{w}_i + u', \hat{w}_i\} - k \cdot u' \\
&= \sum_{i \in X^-(u')} w_i(u') + \sum_{i \in X(y^*)} w_i(u') - k \cdot u' \\
&\leq \sum_{i \in X^-(u')} w_i(u') + \sum_{i \in X(u')} w_i(u') - k \cdot u' = c(u').
\end{aligned}
$$

Thus, to solve the (ILP*) we only need to find the optimal value of u.

We will show that there always exists an optimal solution (u^*, y^*, v^*) of the (ILP*) with $u^* \in U$, where the set U consists of the values $0, \underline{w}_i, \underline{w}_i + \hat{w}_i$, $i \in X$.

Let (u^*, y^*, v^*) be an optimal solution to the (ILP*) with $u^* \notin U$. Then, we consider $\underline{u} \in U$ with $\underline{u} = \arg\min\{u^* - u \mid u \in U, u < u^*\}$ and $\overline{u} \in U$ with $\overline{u} = \arg\min\{u - u^* \mid u \in U, u > u^*\}$. Define $\tilde{X} = \{i \in \{1, \dots, n'\} \mid w_i(u^*) < \hat{w}_i\}$ and $r = |\tilde{X}|$. Since $u^* \notin U$, $-\underline{w}_i + u^* \neq \{\hat{w}_i, 0\}$ for $i = 1, \dots, n'$, and for $i \in \tilde{X}$

$$w_i(\underline{u}) + (u^* - \underline{u}) = w_i(u^*) = w_i(\overline{u}) + (\overline{u} - u^*).$$

If $r < k$,

$$\begin{aligned} c(u^*) &= \sum_{i \in X(u^*)} w_i(u^*) + \sum_{i \in X^-(u^*)} w_i(u^*) - k \cdot u^* \\ &= \sum_{i \in X(\underline{u})} w_i(\underline{u}) + \sum_{i \in X^-(\underline{u})} w_i(\underline{u}) + r \cdot (u^* - \underline{u}) - k \cdot u^* \\ &= c(\underline{u}) + (r - k)(u^* - \underline{u}) \leq c(\underline{u}). \end{aligned}$$

Otherwise,

$$\begin{aligned} c(u^*) &= \sum_{i \in X(u^*)} w_i(u^*) + \sum_{i \in X^-(u^*)} w_i(u^*) - k \cdot u^* \\ &= \sum_{i \in X(\overline{u})} w_i(\overline{u}) + \sum_{i \in X^-(\overline{u})} w_i(\overline{u}) - r \cdot (\overline{u} - u^*) - k \cdot u^* \\ &= c(\overline{u}) + (k - r) \cdot (\overline{u} - u^*) \leq c(\overline{u}). \end{aligned}$$

Hence, there exists always an optimal solution (u^*, y^*, v^*) with $u^* \in U$. Since U contains at most $(2|X| + 2)$ values, Algorithm 5.1 runs in $\mathcal{O}(|X|^2)$. □

5.4.2. A Polynomial Size ILP-Formulation

In this subsection we present an ILP-formulation for the (k, ℓ)-rrKP problem with Γ-scenarios and $\overline{p} = \hat{p} = 0$. Note that in this case it is not beneficial to add any items to the recovered solution. As before the binary variables $x_i \in \{0, 1\}$, $i = 1, \dots, n$, model the first-stage solution X as subset of the n items in $N = \{1, \dots, n\}$, i.e., $i \in X$ if and only if $x_i = 1$. Due to the restriction of $\overline{p} = \hat{p} = 0$, the profit for any feasible solution $X \subseteq N$ equals $\sum_{i \in X} p_i^0$ and is expressed by the linear objective function $\sum_{i=1}^n p_i^0 x_i$. Considering the feasibility, any 0-1 point x satisfying the following inequalities represents a feasible first-stage solution:

$$\sum_{i=1}^{n} w_i^0 x_i \leq c^0 \tag{5.15}$$

$$\sum_{i=1}^{n} \underline{w}_i x_i + \max_{\substack{\bar{X} \subseteq N \\ |\bar{X}| \leq \Gamma}} \Big(\sum_{i \in \overline{X}} \hat{w}_i x_i - \max_{\substack{Y \subseteq N \\ |Y| \leq k}} \big(\sum_{i \in Y} \underline{w}_i x_i + \sum_{i \in \bar{X} \cap Y} \hat{w}_i \big) \Big) \leq c. \tag{5.16}$$

We focus on the second inequality and show in the remaining part how to linearize it by using the results of the previous Section 5.4.1.

Let us define $U = \{\underline{w}_i + \hat{w}_i \mid i \in N\} \cup \{\underline{w}_i \mid i \in N\} \cup \{0\}$, $w_i(u) = \min\{-\underline{w}_i + u, \hat{w}_i\}$ for $i = 1, \ldots, n$ and $u \in U$, $X^-(u) = \{i \in N \mid w_i(u) < 0\}$ and $u \in U$ as in Algorithm 5.1. Hence, inequality (5.16) is equivalent to

$$\sum_{i=1}^{n} \underline{w}_i x_i + \max_{u \in U} \Big(\sum_{i \in X^-(u)} w_i(u) \cdot x_i - k \cdot u + \max_{\substack{X' \subseteq N \\ |X'| \leq \Gamma}} \sum_{i \in X} w_i(u) \cdot x_i \Big) \leq c,$$

as shown in the proof of Theorem 5.4.2. This inequality can be transformed into the following set of constraints

$$\begin{aligned} \sum_{i=1}^{n} \underline{w}_i x_i + \sum_{i \in X^-(u)} w_i(u) \cdot x_i + \max \sum_{i=1}^{n} w_i(u) \cdot x_i \cdot y_i^u \leq c + ku \; \forall\, u \in U \\ \sum_{i=1}^{n} y_i^u \leq \Gamma \qquad \forall\, u \in U \\ y_i^u \in \{0,1\} \quad \forall\, i \in N, \forall\, u \in U. \end{aligned}$$

We dualize the last part, which is totally unimodular, and obtain the following linear ILP

$$\begin{aligned} \sum_{i=1}^{n} \underline{w}_i x_i + \sum_{i \in X^-(u)} w_i(u) \cdot x_i + \min \Big(\Gamma \cdot \xi^u + \sum_{i=1}^{n} \theta_i^u \Big) \leq c + k \cdot u \quad \forall\, u \in U \\ \xi^u + \theta_i^u \geq w_i(u) \cdot x_i \quad \forall\, i \in N, \forall\, u \in U \\ \xi^u, \theta_i^u \geq 0 \qquad \forall\, i \in N, \forall\, u \in U. \end{aligned}$$

with new (dual) variables ξ^u and θ_i^u, $u \in U$, $i \in N$. Combining this set of inequalities with the objective function, we obtain

$$\begin{aligned} \max \sum_{i=1}^{n} p_i^0 x_i \\ \sum_{i=1}^{n} w_i^0 x_i \leq c^0 \\ \sum_{i=1}^{n} \underline{w}_i x_i + \sum_{i \in X^-(u)} w_i(u) \cdot x_i + \Gamma \cdot \xi^u + \sum_{i=1}^{n} \theta_i^u - k \cdot u \leq c \qquad \forall\, u \in U \\ \xi^u + \theta_i^u - w_i(u) \cdot x_i \geq 0 \qquad \forall\, i \in N, \forall\, u \in U \\ \xi^u, \theta_i^u \geq 0 \qquad \forall\, i \in N, \forall\, u \in U \\ x_i \in \{0,1\} \quad \forall\, i \in N. \end{aligned}$$

The size of this ILP depends on the number of different values of $\underline{w}_i, \hat{w}_i$, $i = 1, \ldots, n$, but does not contain more than $(1+n) \cdot (2n+1) + n$ variables and $4n + 3$ inequalities.

5.4.3. Extended Cover-Inequalities

As already described at the beginning of Section 5.2.2, the concept of extended cover inequalities is quite efficient to solve the knapsack problem. We will introduce a version of recoverable robust extended cover inequalities for Γ-scenarios, which dominate the inequalities introduced by Klopfenstein and Nace [76] for the robust case. Furthermore, we give an optimal pseudo-polynomial algorithm computing these inequalities instead of an ILP-formulation as in [76].

Let $N = \{1, \dots, n\}$ be a set of n items, c^0 be integer first-stage capacity, w_i^0 be integer first-stage weights, $\underline{w}_i$ be nominal weights, $\hat{w}_i$ maximal integer deviations, $i \in N$, and $k, \ell \in \mathbb{N}$ the recovery parameters. In the following we denote with $\overline{w}_i = \underline{w}_i + \hat{w}_i$ the highest weight of item $i \in N$. The (k,ℓ)-recoverable robust knapsack polytope of a Γ-scenario set $\mathcal{S}_\Gamma$ is the convex hull over all valid first-stage solutions for $\Gamma \in \mathbb{N}$ i.e.,

$$\mathcal{K}_\Gamma(k) := \operatorname{conv}\Big\{x \in \{0,1\}^n \mid \sum_{i \in N} w_i^0 x_i \leq c^0 \text{ and } \min_{\substack{T \subseteq N \\ |T| \leq k}} \sum_{i \in N \setminus T} w^S x_i \leq c,\ \forall S \in \mathcal{S}_\Gamma\Big\}$$

Similar to [76] a quadruple (C, K_1, J, K_2) is an rrKP *cover* with $C \subseteq N$, $K_1 \subseteq C$, $J \subseteq N$ and $K_2 \subseteq J$ if $C \cup J$ is a cover according to the first-stage weight constraint, i.e.,

$$\sum_{i \in C \cup J} w_i^0 \geq c^0 + 1,$$

or if

1. $J \cap C = \emptyset$
2. $\underline{w}_i \geq \underline{w}_j$ for all $i \in K_1$ and $j \in C \setminus K_1$
3. $\underline{w}_i \geq \overline{w}_j$ for all $i \in K_1$ and $j \in J \setminus K_2$
4. $\overline{w}_i \geq \overline{w}_j$ for all $i \in K_2$ and $j \in J \setminus K_2$
5. $\overline{w}_i \geq \underline{w}_j$ for all $i \in K_2$ and $j \in C \setminus K_1$
6. $|K_1| + |K_2| = k$
7. $|J| \leq \Gamma$
8. $$\sum_{i \in C \setminus K_1} \underline{w}_i + \sum_{i \in J \setminus K_2} \overline{w}_i \geq c + 1.$$

with $\overline{w}_i = \underline{w}_i + \hat{w}_i$, $i = 1, \dots, n$. As in the discrete case, we already fix the recovery action in the definition of an rrKP cover. Due to the conditions (2)-(5) the items in K_1 and K_2 are recovered. Condition (8) guarantees that the remaining items in the cover exceed the knapsack capacity.

An rrKP cover (C, K_1, J, K_2) defines an rrKP *cover inequality*

$$\sum_{i \in (C,J)} x_i \leq |C \cup J| - 1.$$

The following theorem shows that these inequalities are feasible and describe all feasible first-stage solutions.

Theorem 5.4.3. *Let x be a 0-1 point. Then $x \in \mathcal{K}_\Gamma(k)$ if and only if x obeys all cover inequalities.*

Proof. ($\Rightarrow$:) Obviously any $x \in \mathcal{K}_\Gamma(k)$ obeys all rrKP cover inequalities derived from an rrKP cover based on the first-stage weight constraint. Let us therefore assume that there exists an rrKP cover (C, K_1, J, K_2) with

$$\sum_{j \in C \setminus K_1} \underline{w}_j + \sum_{j \in J \setminus K_2} \overline{w}_j \geq c + 1,$$

such that

$$\sum_{i \in C \cup J} x_i \geq |C \cup J| - 1.$$

According to this rrKP cover, we define a scenario S with its weight function w^S in the following way:

$$w^S(i) = \begin{cases} \overline{w}_i & \text{if } i \in J \\ \underline{w}_i & \text{otherwise.} \end{cases}$$

Since $|J| \leq \Gamma$, this weight function is feasible in $\mathcal{S}_\Gamma$. But

$$\begin{aligned} \sum_{i \in N} w_i^S x_i - \max_{\substack{K \subseteq N \\ |K| \leq k}} w_i^S x_i &\geq \sum_{i \in C \cup J} w_i^S x_i - \max_{\substack{K \subseteq C \cup J \\ |K| \leq k}} w_i^S x_i \\ &= \sum_{i \in C \setminus K_1} w_i^S + \sum_{i \in J \setminus K_2} w_i^S \geq c + 1, \end{aligned}$$

a contradiction to $x \in \mathcal{K}_\Gamma(k)$.

($\Leftarrow$:) Since x satisfies all rrKP cover inequalities derived from an rrKP cover based on the first-stage weight constraint, x obeys the first-stage weight constraint.

We now assume that there exists a vector $x \in \{0,1\}^n$ which satisfies all rrKP cover inequalities and yet there exists a scenario $\overline{S} \in \mathcal{S}_\Gamma$ such that $w^{\overline{S}}(T_x) - w^{\overline{S}}(\max, T_x, k) \geq c + 1$ with $T_x = \{i \in N \mid x_i = 1\}$. Let $\overline{K} = \arg\max_{\substack{K \subseteq T_x \\ |K| \leq k}} w^{\overline{S}}(K)$. We define the sets $\overline{K}_1 = \{i \in \overline{K} \mid w_i^{\overline{S}} = \underline{w}_i\}$, $\overline{K}_2 = \{i \in \overline{K} \mid w_i^{\overline{S}} = \overline{w}_i\}$, $\overline{J} = \{i \in N \mid w_i^{\overline{S}} = \overline{w}_i \text{ and } x_i = 1\}$ and $\overline{C} = \{i \in N \mid w_i^{\overline{S}} = \underline{w}_i \text{ and } x_i = 1\}$. Obviously $(\overline{C}, \overline{K}_1, \overline{J}, \overline{K}_2)$ is an rrKP cover with $\underline{w}(\overline{C} \setminus \overline{K}_1) - \overline{w}(\overline{J} \setminus \overline{K}_2) \geq c + 1$. Its rrKP cover inequality demands $\sum_{j \in T_x} x_j \leq |T_x| - 1$, a contradiction. □

In [76] Klopfenstein and Nace propose the following method to extend an rrKP cover $D = (C, K_1, J, K_2)$:

$$\overline{E}(D) = \begin{cases} C \cup J \cup \{i \in N \mid \overline{w}_i \geq \max\limits_{\substack{j \in (C \cup J) \\ j \notin K_1 \cup K_2}} \overline{w}_j\} & \text{if } |C| \leq \Gamma \\ C \cup J \cup \{i \in N \mid \underline{w}_i \geq \max\limits_{\substack{j \in (C \cup J) \\ j \notin K_1 \cup K_2}} \underline{w}_j,\ \overline{w}_i \geq \max\limits_{\substack{j \in (C \cup J) \\ j \notin K_1 \cup K_2}} \overline{w}_j\} & \text{if } |C| \geq \Gamma + 1. \end{cases}$$

This set of extensions can be strengthened in the following way: Add all items, whose lower weight is greater than or equal to the maximum lower weight in $C \setminus K_1$ and whose higher

weight is greater than or equal to the maximum weight in $J \backslash K_2$. Formally, we define an rrKP extension $E(C, K_1, J, K_2)$ by

$$E(C, K_1, J, K_2) := \left\{ i \in N \mid \underline{w}_i \geq \max_{j \in C \backslash K_1} \underline{w}_j \text{ and } \overline{w}_i \geq \max_{j \in J \backslash K_2} \overline{w}_j \right\} \cup \{C \cup J\}.$$

This rrKP extension determines the *rrKP extended cover inequality*

$$\sum_{i \in E(C,K_1,J,K_2)} x_i \leq |C \cup J| - 1.$$

The following example shows a case in which the rrKP extended inequalities based on $E(C, K_1, J, K_2)$ dominate the rrKP extended inequalities defined by $\overline{E}(C, K_1, J, K_2)$ even for the robust case $k = 0$.

Example 5.4.4. Consider the robust KP instance with Γ-scenarios for $N = \{1, 2, 3\}$, $\underline{w} = (5, 6, 5)$, $\overline{w} = (6, 7, 8)$, $\Gamma = 1$, and $c = 11$ without any first-stage weights. The following table contains all rrKP covers for $\Gamma = 1$, their extensions according to Klopfenstein and Nace [76] and inequalities. Since no recovery action is allowed, we drop the sets K_1 and K_2 in our notation.

$C \cup J$	$\overline{E}(C, J)$	$E(C, J)$-inequality
$\{1, 2\}$	$\{1, 2\}$	$x_1 + x_2 \leq 1$
$\{1, 3\}$	$\{1, 3\}$	$x_1 + x_3 \leq 1$
$\{2, 3\}$	$\{2, 3\}$	$x_2 + x_3 \leq 1$

Table 5.6.: rrKP covers and their extensions $\overline{E}(C, J)$.

Yet, for the cover $\overline{C} = \{1\}$ and $\overline{J} = \{2\}$, the rrKP extension is $E(\overline{C}, \overline{J}) = \{1, 2, 3\}$ and yields the inequality $x_1 + x_2 + x_3 \leq 1$. ⊞

In the following part we introduce a method to compute violated rrKP extended cover inequalities via an ILP-formulation or a dynamic program in pseudo-polynomial time.

Separation of RRKP Extended Cover Inequalities The objective in the separation problem is to decide for a given non-integer point $x^* \in [0, 1]^n$, whether there exists a violated rrKP extended cover inequality. In order to find such a violated inequality according to the first-stage constraint, we can either use the methods proposed in literature (e.g. [66, 82]) or the methods described for the discrete case in Section 5.2.2. We will therefore focus on the problem to find a violated rrKP extended cover for one of the scenarios using again integer programming.

Let N be a set of n items. For every item $i \in N$ we introduce five different binary variables to represent a cover and its extension, namely

$\underline{y}_i$ with $\underline{y}_i = 1$ if $i \in C \backslash K_1$,

$\overline{y}_i$ with $\overline{y}_i = 1$ if $i \in J \backslash K_2$,

$\underline{z}_i$ with $\underline{z}_i = 1$ if $i \in K_1$,

$\overline{z}_i$ with $\overline{z}_i = 1$ if $i \in K_2$ and

α_i with $\alpha_i = 1$ if i is part of the extension but not in the cover.

Since any item can take at most one of these five states, $\underline{y}_i + \overline{y}_i + \underline{z}_i + \overline{z}_i + \alpha_i \leq 1$ for every $i \in N$. Furthermore, the associated weight of the not recovered items needs to exceed the knapsack capacity, i.e., $\sum_{i\in N} \underline{w}_i\underline{y}_i + \sum_{i\in N} \overline{w}_i\overline{y}_i \geq c+1$. Since the recovery action has to be fixed, $\sum_{i\in N} \underline{z}_i + \overline{z}_i = k$. Also the set of variables taking the weight $\overline{w}_i$ is bounded by Γ, reflected in $\sum_{i\in N} \overline{z}_i + \overline{w}_i \leq \Gamma$. Finally, the lower weight of any item in K_1 has to be greater than or equal to the associated weight of any item in $C\backslash K_1$ or $J\backslash K_2$. Similar conditions need to be fulfilled by any variable with $\overline{z}_i = 1$ being part of K_2, $i \in N$. In summary we obtain the following ILP

$$
\begin{aligned}
\max \sum_{i\in N}(x_i^*-1)(\underline{y}_i + \overline{y}_i + \underline{z}_i + \overline{z}_i) + \sum_{i\in N} x_i^*\alpha_i & \\
\sum_{i\in N}\underline{w}_i\underline{y}_i + \sum_{i\in N}\overline{w}_i\overline{y}_i &\geq c+1 \\
\sum_{i\in N}(\underline{z}_i + \overline{z}_i) &= k \\
\sum_{i\in N}(\overline{y}_i + \overline{z}_i) &\leq \Gamma \\
\alpha_i + \underline{y}_i + \overline{y}_i + \underline{z}_i + \overline{z}_i &\leq 1 \qquad \forall\, i \in N \\
\underline{y}_j + \underline{z}_i + \alpha_i &\leq 1 \qquad \forall\, i,j \in N : \underline{w}_j > \underline{w}_i \\
\overline{y}_j + \underline{z}_i + \alpha_i &\leq 1 \qquad \forall\, i,j \in N : \overline{w}_j > \underline{w}_i \\
\overline{y}_j + \overline{z}_i + \alpha_i &\leq 1 \qquad \forall\, i,j \in N : \overline{w}_j > \overline{w}_i \\
\underline{y}_j + \overline{z}_i + \alpha_i &\leq 1 \qquad \forall\, i,j \in N : \underline{w}_j > \overline{w}_i \\
\alpha_i, \underline{y}_i, \overline{y}_i, \underline{z}_i, \overline{z}_i &\in \{0,1\} \ \forall\, i \in N.
\end{aligned}
$$

The computed rrKP extended cover with $C = \{i \in N \mid \underline{y}_i + \underline{z}_i = 1\}$, $K_1 = \{i \in N \mid \underline{z}_i = 1\}$, $J = \{i \in N \mid \overline{y}_i + \overline{z}_i = 1\}$ and $K_2 = \{i \in N \mid \overline{z}_i = 1\}$ determines an rrKP-extended cover inequality which is violated by x^* if and only if the value of the ILP is greater than -1.

We finally present a pseudo-polynomial algorithm for solving this separation problem. Let $\underline{U} = \cup_{i=1}^n \{\underline{w}_i\}$, $\overline{U} = \cup_{i=1}^n \{\overline{w}_i\}$ and $D = \sum_{i\in N} \overline{w}_i$. For every $t = 0,\ldots,n$, $d = 0,\ldots,D$, $\gamma = 0,\ldots,\Gamma$, $\kappa = 0,\ldots,k$, $\underline{\omega} \in \underline{U}$, and $\overline{\omega} \in \overline{U}$ we solve the problem of finding an rrKP extended cover within the first t items, such that at most γ items are in J, κ items are removed and the remaining items have a weight of d. Furthermore, the items in $C\backslash K_1$ have to obey $\underline{w}_i \leq \underline{\omega}$ whereas any item in K_1 satisfies $\underline{w}_i \geq \underline{\omega}$. Similar, $\overline{w}_i \leq \overline{\omega}$ for all items in $J\backslash K_2$ and $\overline{w}_i \geq \overline{\omega}$ for all items in K_2. We thus consider the following function with $N(t) = \{1,\ldots,t\}$

$$
\begin{aligned}
f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d) = \max \sum_{i\in N(t)}(x_i^*-1)(\underline{y}_i + \overline{y}_i + \underline{z}_i + \overline{z}_i) + \sum_{i\in N(t)} x_i^*\alpha_i & \\
\sum_{i\in N(t)}\underline{w}_i\underline{y}_i + \sum_{i\in N(t)}\overline{w}_i\overline{y}_i &= d \\
\sum_{i\in N(t)}\underline{z}_i + \overline{z}_i &= \kappa \\
\sum_{i\in N(t)}\overline{y}_i + \overline{z}_i &= \gamma \\
\underline{y}_i + \overline{y}_i + \underline{z}_i + \overline{z}_i + \alpha_i \leq 1 &\quad \forall\, i = 1,\ldots t \\
\underline{y}_i = 0 &\quad \forall\, i : \underline{w}_i > \underline{\omega} \\
\overline{y}_i = 0 &\quad \forall\, i : \overline{w}_i > \overline{\omega} \\
\underline{z}_i = \alpha_i = 0 &\quad \forall\, i : \underline{w}_i < \underline{\omega} \\
\overline{z}_i = \alpha_i = 0 &\quad \forall\, i : \overline{w}_i < \overline{\omega} \\
\alpha_i, \underline{y}_i, \overline{y}_i, \underline{z}_i, \overline{z}_i \in \{0,1\} &\quad \forall\, i \in N(t).
\end{aligned}
$$

The optimal solution to the separation problem is given by

$$\max_{\substack{\underline{\omega}\in\underline{U},\overline{\omega}\in\overline{U}\\ d=c+1,\dots,D}} f_{\underline{\omega}}^{\overline{\omega}}(n,k,\Gamma,d)$$

and for $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)$ with $t=1$ we can obtain the optimal value by considering the following cases:

Case 1: if $\underline{w}_1 \leq \underline{\omega}$, $d=\underline{w}_1$ and $\kappa=0$, then $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=(x_1^*-1)$ with $\underline{y}_1=1$

Case 2: if $\underline{w}_1 \geq \underline{\omega}$, $d=0$ and $\kappa=1$, then $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=(x_1^*-1)$ with $\underline{z}_1=1$

Case 3: if $\overline{w}_1 \leq \overline{\omega}$, $d=\overline{w}_1$, $\kappa=0$ and $\gamma\geq 1$, then $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=(x_1^*-1)$ with $\overline{y}_1=1$

Case 4: if $\overline{w}_1 \geq \overline{\omega}$, $d=0$, $\kappa=1$ and $\gamma\geq 1$, then $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=(x_1^*-1)$ with $\overline{z}_1=1$

Case 5: if $\underline{w}_1 \geq \underline{\omega}$, $\overline{w}_1 \geq \overline{\omega}$, $d=0$, $\kappa=0$, then $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=x_1^*$ with $\alpha_1=1$

Case 6: otherwise, $f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d)=-\infty$.

Adapting these case distinctions for general values of t,κ,γ and d, we receive the following recursive formula

$$\begin{aligned} f_{\underline{\omega}}^{\overline{\omega}}(t,\kappa,\gamma,d) = \max\{ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa,\gamma,d), \\ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa,\gamma,d-\underline{w}_t)+(x_t^*-1) && \text{if } \underline{w}_t \leq \underline{\omega}, \\ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa-1,\gamma,d)+(x_t^*-1) && \text{if } \underline{w}_t \geq \underline{\omega}, \\ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa,\gamma-1,d-\overline{w}_t)+(x_t^*-1) && \text{if } \overline{w}_t \leq \overline{\omega}, \\ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa-1,\gamma-1,d)+(x_t^*-1) && \text{if } \overline{w}_t \geq \overline{\omega}, \\ & f_{\underline{\omega}}^{\overline{\omega}}(t-1,\kappa,\gamma,d)+x_t^* && \text{if } \underline{w}_t \geq \underline{\omega} \text{ and } \overline{w}_t \geq \overline{\omega}\}. \end{aligned}$$

The validation of this formula can easily be achieved via a proof by contradiction. The runtime of the pseudo-polynomial algorithm obtained by the recursive formula lies in $\mathcal{O}(D\cdot n^5)$.

5.5. Conclusion and Open Issues

In this chapter we studied the (k,ℓ)-recoverable robust knapsack problem with different types of scenario sets. For discrete scenarios, we obtained in Subsection 5.2.1 similar complexity results as for the robust knapsack problem. If the number of scenarios is constant, the problem is weakly **NP**-hard and can be solved optimally in pseudo-polynomial time. However, the total cost for a given first-stage solution cannot be computed efficiently. If the number of scenarios is not constant, the problem becomes strongly **NP**-hard. The lower bound on the best possible approximation factor depends on ℓ, the number of items that might be added in the second stage. One question is whether there is a lower bound that is independent of ℓ. Another challenge is to find an approximation algorithm with a constant approximation factor for $\ell \geq 1$.

Next to the complexity status, we considered the polyhedral structure of the (k,ℓ)-rrKP problem. To this end, we adapted the well-known cover inequalities and extended cover inequalities to the (k,ℓ)-rrKP problem. In our computational study we showed that these inequalities improve the bound obtained by the naive linear program relaxation of the integer program. One topic of future research is to adjust further inequalities to the (k,ℓ)-rrKP problem.

There are several open problems concerning the (k, ℓ)-rrKP problem with Γ-scenarios. A first one is to find an IP-formulation that is polynomial in the size of the input, i.e., the number of items. So far, we just introduced a formulation for the special case where no scenario profit functions are given (Section 5.4.2). Furthermore, the complexity status is still open. The problem is obviously weakly **NP**-hard, but it might even be strongly **NP**-hard. One could also try construct a pseudo-polynomial algorithm. A different task is to test the impact of our extended cover inequalities on the upper bounds for the total profit via an LP-relaxation, in a similar way as in the discrete scenario case. Furthermore, it could be interesting to extend different inequalities to strengthen the polyhedral formulation of the (k, ℓ)-rrKP polytope.

So far, we considered rather theoretical problems. However, our investigation was motivated by the admission control problem from telecommunication. Therefore, we are on the one hand interested in modeling Γ-scenarios, such that they reflect the actual data setting and, on the other hand, in testing how our solutions perform compared to a robust approach. Furthermore, to extend the modeling power of Γ-scenarios, we want to adapt the Histogram model of Bienstock [16]. For a special nested setting of bands, we think that the robust problem is solvable in polynomial time. It remains open how the bounds can be defined to represent a real data set. Another open question is whether we can extend our results to the (k, ℓ)-rrKP problem with Histogram scenarios.

6. Recoverable Robust Train Classification

An essential task in railway optimization is train classification: according to a classification schedule incoming trains are split up into their single cars and are reassembled to form new outgoing trains. Often such a prepared sorting schedule becomes infeasible when the incoming trains are subject to delay and arrive in an unexpected order. Classification methods applied today deal with this issue by completely disregarding the input order of cars, which presents robustness against any amount of disturbance but also wastes the potential contained in an a priory knowledge of the input.

We introduce a new method that provides a feasible sorting schedule for the expected input and allows to flexibly insert additional sorting steps if the schedule has become infeasible after revealing the disturbed input. By excluding disruptions that almost never occur from our consideration, we obtain a classification process that is quicker than the current railway practice but still provides robustness against realistic delays. In fact, our algorithm allows a trade-off of fast classification and high degrees of robustness depending on the respective need. We further explore this flexibility in experiments on real-world traffic data underlining that our algorithm improves on the methods currently applied in practice. This chapter is based on joint work with Jens Maue and is partly published in [24].

6.1. Introduction

In freight railway, cars are carried from different origins to different destinations in a given network. According to their origin and destination a route for each car is fixed and trains composed of several cars and a locomotive are formed along these routes depending on the demand. In general a car is part of several different trains until it reaches its final destination. At certain stops, so-called *classification yards*, incoming trains need to be sorted to form new outgoing trains. With increasing world-wide freight traffic, operating freight trains efficiently becomes more and more important, and reducing the dwell time of cars in railway yards is one of the important factors to improve freight service profitability.

Train classification problems have been studied under different assumptions on the physical layout of the shunting yard, elementary sorting operations, constraints on the classification schedules and objective functions (e.g. [34, 38, 39, 49, 57]). We focus on shunting over a hump of ℓ incoming trains to form a single outgoing train as described in Jacob et al. [63]. In that case the shunting yard consists of incoming tracks, a hump track, a hump, several switches and classification tracks (see Figure 6.1 (a)). The incoming trains are collected at one of the *incoming tracks* in their arrival order (see Figure 6.1 (b)). From there all trains are brought to the *hump track*, where they are disconnected and slowly pushed over the *hump* to roll to their destination track, the *classification tracks*. This operation is called a *roll-in*. A hump is a small hill such that the cars are accelerated by gravity. The guiding of each car is done by

a series of switches. Normally it does not suffice to roll-in the cars just one time to form the desired outgoing train. Hence, all cars of one track are coupled and pulled back to the hump track by a shunting engine (*pull-out*) where a further roll-in is performed. A pair of pull-out and roll-in operations is called a *(sorting) step* and an initial roll-in followed by a sequence of h sorting steps is called a *classification schedule* of *length* h. The number of steps h essentially determines the time required to conclude the sorting procedure. Hence, train classification aims at transforming the incoming trains into the desired outgoing train with a small number of sorting steps (see Figure 6.1 (c)).

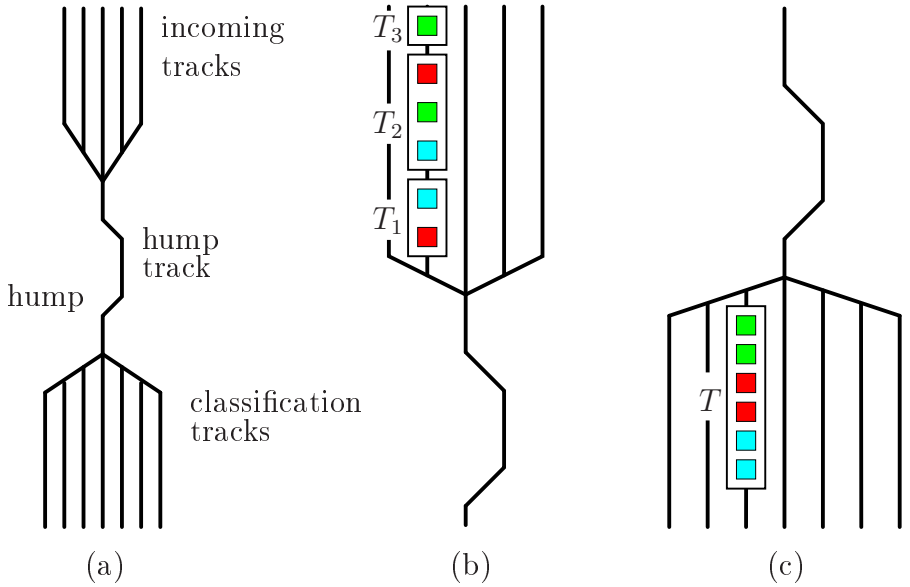


Figure 6.1.: (a) A classification yard consists of incoming tracks, a hump track, a hump and classification tracks. (b) The incoming trains T_1, T_2 and T_3 are collected at one incoming track. (c) After several sorting steps the trains are reordered to form the outgoing train T at one classification track.

The methods for train classification most-commonly applied today and widely studied are triangular and geometric sorting [33, 47, 79, 86, 91]. Classification schedules constructed in these ways do not depend on the order of railway cars entering the classification process. In the case of an unknown input geometric sorting minimizes the number of sorting steps and triangular sorting is optimal for restricting the number of roll-ins per car to three [63]. However, neither method exploits the situation of a partially ordered input sequence and thus generates more sorting steps than necessary.

To take advantage of information about the input order a new classification algorithm was developed by Jacob in [62] and Hansmann and Zimmermann in [57] independently. This sorting method derives optimal schedules w.r.t. the number of sorting steps in case of unrestricted track length and an unbounded number of classification tracks. A round-robin variation solves the classification problem if the number of tracks is restricted. Yet, if the length of the tracks is bounded the problem is strongly **NP**-complete [62] but can be 2-approximated [63]. Recent overviews of train classification can be found in [53] and [57].

Often inbound trains are subject to delay, so we might be faced with an unexpected inbound order of trains and an optimal classification schedule according to the expected order does not achieve the desired result. Since changes during the process of scheduling are time consuming, a certain amount of robustness is crucial for classification methods to work in practice. Providing

strict robustness, however, wastes a lot of potential to disruption scenarios that almost never occur in practice. We deal with this dilemma by regarding realistic scenarios of delay and present optimal robust solutions w.r.t. a limited amount of recovery in case of disturbance. We call such solutions recoverable robust.

The general concept of recoverable robustness was introduced, for example, by Liebchen et al. [81]. This method is applied to several railway-related optimization problems such as rolling stock scheduling by Cacchiani et al. [25] or timetabling by Cicerone et al. [30, 31]. A first attempt to study recoverable robustness for train classification is made by Cicerone et al. [29, 30] for a single inbound and outbound train. Besides the situations of strict robustness and complete recomputation from scratch, which are more of theoretical interest, they consider a recovery action that allows completely changing the classification instruction for one set of cars that was planned to run through the same series of tracks. The most relevant disturbances in their setting are one additional car in the input and one car occurring at a different position than expected. Furthermore, they deal with the problem of a single classification track becoming unavailable before the classification starts. This may, for example, be caused by a faulty switch.

In our model we focus on the most relevant reason for disruptions, which is delayed trains. All possible settings of disturbed inbound orders are given by a *set of scenarios* $\mathcal{S}$. Each scenario $S \in \mathcal{S}$ defines a *modified instance*, which is a permutation of the original inbound train sequence. In a first stage we fix a feasible schedule for the original instance, which is called a *first-stage solution*. This schedule may be infeasible for a modified instance corresponding to some arising scenario. In response to the modified input, we are prepared to insert up to k additional sorting steps after the pth step of the first-stage solution, providing a *recovered schedule*. A first-stage solution is called *recoverable robust* if there exists a recovered schedule for every scenario $S \in \mathcal{S}$. Given a sequence of ℓ inbound trains, a specification of the outbound train, and a set of scenarios $\mathcal{S}$, the *recoverable robust train classification problem* is to find a recoverable robust, feasible first-stage solution of minimum length.

In the next section we will introduce the train classification problem with its terminology and notation as proposed in [63]. Section 6.3 contains a more detailed description of the recoverable robust train classification problem, first properties of an optimal schedule and a generic algorithm for computing such a schedule. Depending on the scenario set, the generic algorithm may have exponential run-time. We further prove that, for every constant $k \geq 1$, finding a recoverable robust schedule of minimum length is an **NP**-hard problem. As a generalization of the scenario set of one car in unexpected position introduced in [29], we investigate the practically relevant scenario set of delaying up to j arbitrary trains. For this setting the problem can be solved in polynomial time as shown in Section 6.4. Furthermore, we evaluate our new algorithm on real-world traffic data for various parameter values k, p, and j. It turns out that our algorithm yields very short schedules while providing a fair degree of robustness. In the last Section 6.5, we consider the setting in which cars are not unique. This allows a more flexible order of cars in their outbound trains. Nevertheless, the problem can be solved in polynomial time if, for example, the set of scenarios is constant.

6.2. Encoding Classification Schedules

We start with an introduction of the terminology and notation to describe a feasible train classification schedule based on the notation of [63]. In this classification setting every car is represented by a positive integer $\tau \in \mathbb{N}$ and a train T by a sequence of cars $T = (\tau_1, \ldots, \tau_t)$,

where t is called the length of T. In our model, a set of ℓ inbound trains needs to be sorted into one specified outbound train $(1, \ldots, n)$. We assume that the concatenation of the inbound trains is a permutation of $(1, \ldots, n)$ and in all sections except the last one that all cars are distinct. Furthermore, the number of available classification tracks and their lengths are unrestricted. Note that the problem setting for several outbound trains remains the same, since we can compute for every single outbound train a classification schedule and due to the property of the classification tracks just apply them simultaneously.

A sorting procedure consists of a sequence of alternating roll-in and pull-out steps. In order to specify a single pull-out step, it suffices to specify the classification track from which to pull out all cars. Since the number of tracks is unrestricted, the number of actually used tracks is identical with the number of sorting steps h. The track pulled in the kth step is referred to by θ_k, $k = 0, \ldots, h-1$. Note that w.l.o.g. always all cars on a track are pulled out. The journey, a car τ takes through the classification yard, can be encoded by bitstrings $b^\tau = b^\tau_{h-1} \ldots b^\tau_0$. The car is pulled out in the kth step if $b^\tau_k = 1$. After such a pull-out, the car is sent to track θ_ℓ with $\ell = \min\{i | k < i < h,\ b^\tau_i = 1\}$; if $b^\tau_i = 0$ for all $i > k$, it is moved to the destination track of its outbound train. A classification schedule of length h can now be represented by a $(n \times h)$ binary matrix, where each row b^τ describes the journey of car $\tau \in \{1, \ldots, n\}$, and each column a sorting step. The classification schedule is read from right to left (see Figure 6.2).

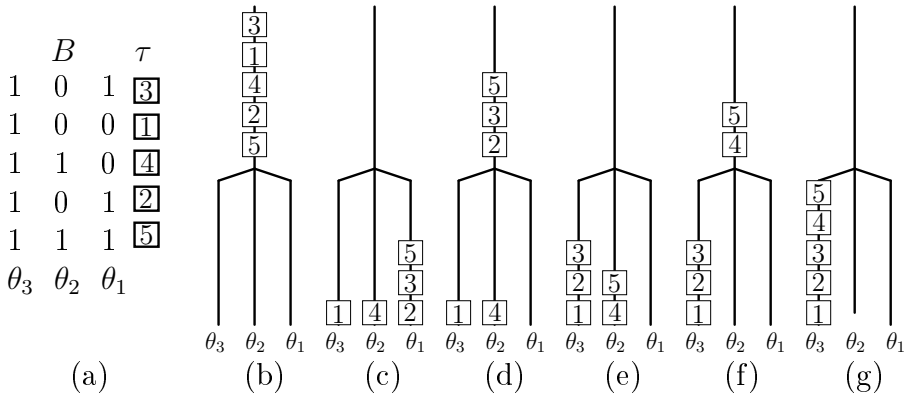


Figure 6.2.: The schedule B represents a sorting procedure to order the incoming train $T = (5, 2, 4, 1, 3)$ with two sorting steps. Note that due to the 3th column b_3 all trains are collected at track θ_3. In general this last sorting step is left out in the description of a schedule.

Before we explain when such a $(n \times h)$-matrix is a feasible classification schedule, we set some notations for bitstrings. We will refer to the binary representation of a decimal integer $j \geq 0$ by $[j]_2$. Given any bitstring $b = b_{h-1} \ldots b_0$ of length h, let $\mathrm{num}(b)$ denote the integer number represented by b, i.e., $\mathrm{num}(b) = \sum_{i=0}^{h-1} 2^i b_i$. For two bitstrings b_1, b_2 we define $b_1 < b_2$ if and only if $\mathrm{num}(b_1) < \mathrm{num}(b_2)$.

In [63] Jacobs et al. showed the following property of a schedule B for sorting n cars to be feasible: two cars τ and $\tau + 1$ must be assigned bitstrings $b^\tau \leq b^{\tau+1}$, $\tau \in \{1, \ldots, n-1\}$. If these cars occur in reversed order in the inbound sequence, $b^\tau < b^{\tau+1}$ is required to swap their relation during the sorting process; in that case the tuple $\beta = (\tau, \tau + 1)$ is called a *break*. According to this observation, they present the following procedure to obtain an optimal classification schedule: let $\beta_1 = (\tau_1, \tau_1 + 1), \ldots, \beta_\kappa = (\tau_\kappa, \tau_{\kappa+1})$ be the set of breaks with $\tau_i < \tau_j$ for $i < j$, $i, j \in \{1, \ldots, \kappa\}$, according to the concatenation of the train sequence $T_1, \ldots, T_\ell$ and the specifications of the outbound train. Add $\beta_0 = (0, 1)$ and $\beta_{\kappa+1} = (n, n+1)$ as dummy breaks and define the jth *chain* $[\tau_j, \tau_{j+1}]$ to consist of all cars τ with $\tau_j + 1 \leq \tau \leq \tau_{j+1}$ for

$j \in \{0, \ldots, \kappa\}$. The schedule B with $b^\tau = [j]_2$ for all cars τ in the jth chain, $j \in \{0, \ldots, \kappa\}$, is a feasible train classification schedule with minimal length $h = \lceil \log_2(\kappa + 1) \rceil$. Hence, the length of a classification schedule depends only on the number of breaks.

Figure 6.2 shows a classification process and the corresponding encoding: cars 1 and 2 arrive in reversed order as cars 3 and 4 and cars 4 and 5. These three breaks $\beta_1 = (1, 2)$, $\beta_2 = (3, 4)$ and $\beta_3 = (4, 5)$ define the chains [1], [2, 3], [4] and [5] and induce the classification schedule $b^1 = (0, 0)$, $b^2 = (0, 1)$, $b^3 = (0, 1)$, $b^4 = (1, 0)$ and $b^5 = (1, 1)$ (see Figure 6.2 (a) and (b)). In (c) the cars are the first time pushed over the hump and according to the schedule car 1 is guided to the final track θ_3, whereas cars 2, 3 and 5 visit first track θ_1 and car 4 track θ_2. In the following pull-out step (see Figure 6.2 (d)), cars 2, 3, and 5 are coupled and pulled to the hump track again. From there cars 2 and 3 are led to track θ_3 and car 5 is led to track θ_2 and finally in the second sorting step car 4 and 5 are pulled out and guided to track θ_3.

6.3. Recovery by Additional Sorting Steps

In this section we introduce the recovery strategy of inserting a limited number of additional sorting steps to a classification schedule as a response to delay and start with some further observations on considered disturbances.

As already mentioned, delay of trains is the most common source for variations in the inbound train sequence. We model these uncertainties by a set of scenarios $\mathcal{S}$, where each scenario defines a permutation of the original inbound train sequence. A break β is *induced by* S if β is a break in the modified instance corresponding to S. In order to distinguish between breaks for the expected order of inbound trains and a modified order, we call the first ones *original breaks* and all other breaks *potential breaks*. For any scenario S, X^S denotes the set of potential breaks induced by S. Note that this set is uniquely defined for every scenario $S \in \mathcal{S}$ and thus we will repeatedly regard these sets of breaks without considering the actual underlying scenario. In particular, we will describe sets of scenarios implicitly by providing the set of induced breaks of every scenario.

In the train classification process the planed schedule B becomes infeasible if and only if the revealed scenario S induces a break $\beta = (\tau, \tau + 1)$ which is unresolved by B. A break β is called *unresolved* w.r.t. S if β is induced by S and $b^\tau = b^{\tau+1}$. With the recovery action of inserting up to k additional sorting steps to the first-stage solution, $k \in \mathbb{N}$, we seek to obtain a feasible schedule for the modified instance. The classification process according to the originally planned schedule may have started when the delay of a train is realized. Furthermore, distributing the recovered solution, i.e., the changed schedule, to all people involved in the operation takes some time depending on the available communication channels. For these reasons, inserting additional sorting steps is only allowed after an offset of p steps, $p \in \mathbb{N}$. Since every car may be affected by the additional sorting steps, we assume that all cars are collected at the $(p+1)$th track. From that position the k additional sorting steps are performed before the remaining schedule is executed.

In terms of classification schedules expressed as binary matrix this means the following: given two parameters $p \geq 0$ and $k \geq 0$ and a classification schedule B of length h with $b_p = 1$, B is to be recovered by inserting up to k additional columns with indices greater than $p+1$. Since every car visits the $(p+1)$th track, the $(p+1)$th column of every feasible schedule equals the 1-vector. For the sake of simplicity we will omit this sorting step in the remaining part and formalize the concept in the following definition.

Definition 6.3.1. Let $B = (b_{h-1}, \ldots, b_0)$ and $B' = (b'_{h-1+j}, \ldots, b'_0)$ be two classification schedules for n cars of length h and $h+j$, $j \geq 0$, respectively. Let further $p \geq 0$ and $k \geq 0$. The schedule B' is called a (p,k)-*extension* of B if $j \leq k$, $b_i = b'_i$ for all $0 \leq i < p$ and $b_{i-j} = b'_i$ for all $p-1+j \leq i \leq h-1+j$.

The notion of (p,k)-extensions yields a natural concept of recoverable robustness as stated in the following definition.

Definition 6.3.2 (Recoverable Robust Train Classification Problem)**.** Let $T_1, \ldots, T_\ell$ be a sequence of ℓ inbound trains and $\mathcal{S}$ be a set of scenarios. Each scenario defines a permutation of the original train sequence. A feasible classification schedule B is called *recoverable robust* if, for every scenario $S \in \mathcal{S}$ there is a (p,k)-extension of B that is feasible w.r.t. S.

Our objective is to find recoverable robust classification schedule of minimum length. In order to specify when a given schedule is recoverable robust for a given set of scenarios, we use the notion of a *block* of a schedule. A block basically is a maximal set of bitstrings representing integers between two powers of two.

Definition 6.3.3. Let B be a schedule of length h for an inbound train sequence of n cars, and $p \geq 1$. For any bitstring b^j of B, $b^j_{h-1}...b^j_p$ is called the *leading part* of b^j, denoted by $b^j_{>p}$, and $b^j_{p-1} \ldots b^j_0$ the *trailing part* of B, denoted by $b^j_{<p}$. A subset of λ consecutive bitstrings $b^j, \ldots, b^{j+\lambda-1}$ of B is called a *block* of B if their leading parts satisfy $b^{j-1}_{>p} < b^j_{>p}$, $b^j_{>p} = b^{j+x}_{>p}$ for all $1 \leq x \leq \lambda - 1$, and $b^{j+\lambda-1}_{>p} < b^{j+\lambda}_{>p}$, while λ is called the *size* of the block. Furthermore, the jth car of the inbound train sequence is called the *head* of the block.

The following lemma states necessary and sufficient conditions for the existence of (p,k)-extensions. The second part of its proof presents a method to derive a (p,k)-extension for a given scenario S making use of the block structure of the original schedule. We will refer to this construction by *canonical* recovery and call the resulting schedule a *canonical* (p,k)-extension.

Lemma 6.3.4. *Let $T_1, \ldots, T_\ell$ be a sequence of ℓ inbound trains, B be a feasible classification schedule, S be a scenario, and $p, k \geq 0$. Then, there exists a (p,k)-extension of B that is feasible for S if and only if the number of unresolved breaks w.r.t. S does not exceed $2^k - 1$ for any block of B.*

Proof. ($\Rightarrow$): Let $X' = \{(\tau_1, \tau_1+1), \ldots, (\tau_t, \tau_t+1)\}$ be the set of unresolved breaks w.r.t. S within a block of B. Assume $t > 2^k - 1$. For every $(\tau_i, \tau_i+1) \in X'$, schedule B satisfies $b^{\tau_i} = b^{\tau_i+1}$, so, for every (p,k)-extension $\bar{B} = \bar{b}_{h-1+k} \ldots \bar{b}_0$ of B, the leading part and the trailing part of $\bar{b}^{\tau_i}$ and $\bar{b}^{\tau_i+1}$ are equal. For $\bar{B}$ to be a valid schedule w.r.t. scenario S, $\text{num}(\bar{b}^{\tau_i}_{p-1+k} \ldots \bar{b}^{\tau_i}_p) < \text{num}(\bar{b}^{\tau_i+1}_{p-1+k} \ldots \bar{b}^{\tau_i+1}_p)$ for every unresolved break (τ_i, τ_i+1), $i = 1, \ldots, t$. Since also $\text{num}(\bar{b}^{\tau_1}_{p-1+k} \ldots \bar{b}^{\tau_1}_p) \geq 0$ and further $\bar{b}^{\tau_{i+1}}_{p-1+k} \ldots \bar{b}^{\tau_{i+1}}_p \geq \bar{b}^{\tau_i}_{p-1+k} \ldots \bar{b}^{\tau_i}_p$ must hold for every $i = 1, \ldots, t-1$, we obtain $\bar{b}^{\tau_{t+1}}_{p-1+k} \ldots \bar{b}^{\tau_{t+1}}_p \geq t > 2^k - 1$, which is a contradiction. Hence, there is no (p,k)-extension of B that is valid w.r.t. S if $t > 2^k - 1$.

($\Leftarrow$): Let κ denote the number of blocks of B, $\bar{\tau}_i$ the head of the ith block, and λ_i the size of the ith block, $i = 1, \ldots, \kappa$. Let further $X'_i = \{(\tau^i_1, \tau^i_1+1), \ldots, (\tau^i_{t_i}, \tau^i_{t_i}+1)\} \subseteq X'$ be the subset of unresolved breaks of the ith block with t_i denoting their number.

We extend B in the following way to a schedule B': for the ith block, we put $b^\tau_{p-1+k} \ldots b^\tau_p = [j]_2$ for every car τ with $\tau^i_j + 1 \leq \tau \leq \tau^i_{j+1}$, where $\tau^i_0 + 1 := \bar{\tau}_i$ is the head and $\tau^i_{t_i+1} := \bar{\tau}_i + \lambda_i - 1$ the last car of the ith block. If $|[j]_2| < k$, we simply add leading zeros to obtain a bitstring of length k. Since $j < t_i < 2^k - 1$, also $|[j]_2| \leq k$ and the construction results in a (p,k)-extension of B.

It remains to show that this schedule B' is feasible w.r.t. S. First, if $(\tau_j^i, \tau_j^i + 1) \in X_i'$, then $b^{\tau_j^i} = b^{\tau_j^i+1}$ in the original schedule B. The construction yields

$$b_{p-1+k}^{\tau_j^i} \dots b_p^{\tau_i^j} = [j-1]_2 < [j]_2 = b_{p-1+k}^{\tau_j^i+1} \dots b_p^{\tau_i^j},$$

so $b^{\tau_j^i} < b^{\tau_j^i+1}$ holds for B'. Second, if $(\tau, \tau+1) \in X \setminus X'$, where X denotes the set of all breaks induced by S, then $b^\tau < b^\tau + 1$ in B. If, in this case, b^τ and $b^\tau + 1$ are in the same block, then $b_{p-1+k}^\tau \dots b_p^\tau = b_{p-1+k}^{\tau+1} \dots b_p^{\tau+1}$, so $b^\tau < b^{\tau+1}$ holds for B'. Otherwise, let b^τ be contained in the ith block and $b^\tau + 1$ in the $(i+1)$th; then, they satisfy $b_{>p}^\tau < b_{>p}^{\tau+1}$, so $b^\tau < b^{\tau+1}$ for B'. Finally, if $(\tau, \tau+1) \notin X$, then $b^\tau \leq b^{\tau+1}$. If b^τ and $b^{\tau+1}$ are in the same block, $b_{p-1+k}^\tau \dots b_p^\tau = b_{p-1+k}^{\tau+1} \dots b_p^{\tau+1}$, so $b^\tau \leq b^{\tau+1}$ still holds for the extension. Otherwise, let b^τ be contained in the ith and $b^{\tau+1}$ in the $(i+1)$th block, $i \in \{1, \dots, i_{\max} - 1\}$; then, $b_{>p}^\tau < b_{>p}^{\tau+1}$ already for B, so $b^\tau < b^{\tau+1}$ also holds for B'. Therefore, the extension B' presents a feasible schedule w.r.t. S. □

6.3.1. Generic Algorithm

Exploring this last observation we introduce a generic algorithm for computing recoverable robust train classification schedules. Basically, the algorithms successively grows the size of a block to its maximum size. This size is determined by two factors: First, a schedule B assigns at most 2^p different bitstrings to the trailing part of cars in the same block, i.e., at most $2^p - 1$ breaks can be resolved. Secondly, the number of unresolved breaks in a block is limited by $2^k - 1$ induced by one scenario. In order to formalize these conditions we denote with $X_{(\tau_1,\tau_2)}$ the set of all original and potential breaks occurring between two cars τ_1 and τ_2, $1 \leq \tau_1 \leq \tau_2 \leq n$ and consider so-called k-recoverable break sets.

Definition 6.3.5. Let $T_1, \dots, T_\ell$ be a sequence of ℓ inbound trains with a total of n cars and $k \geq 0$. Let further $\mathcal{S}$ be a set of scenarios and $X_{(\tau_1,\tau_2)}$ be the set of all original and potential breaks between the cars τ_1 and τ_2. A set of breaks $X' \subseteq X_{(\tau_1,\tau_2)}$ is called *k-recoverable* if

$$|X' \cap X^S| \leq 2^k - 1 \quad \forall S \in \mathcal{S}.$$

Algorithm 6.1 (p,k)-recoverable robust train classification

Input : number of cars n, set of original breaks X_{org}, set of scenarios $\mathcal{S}$, $k, p \geq 0$
Output : k-recoverable robust classification schedule B

Set $i = 0$, $\tau_i = 1$, $\tau_{\max} = 0$, $X = X' = \emptyset$
while $\tau_i \leq n$ **do**
  while $\tau_{\max} < \tau_i + 2^p + |X'|$ *and* $\tau_i + 2^p + |X'| \leq n$ **do**
    Set $\tau_{\max} = \tau_i + 2^p + |X'|$
    Set $X = X_{(\tau_i,\tau_{\max})} \cap (\cup_{S \in \mathcal{S}} X^S)$
    Compute a maximum k-recoverable set of breaks $X' \subseteq X$
  Set $\tau_{\max} = \tau_{i+1} = \min(\tau_i + 2^p + |X'|, n+1)$
  Compute a subschedule of length p for the cars $\tau_i, \dots, \tau_{i+1} - 1$ feasible w.r.t. the breaks $X_{(\tau_i,\tau_{i+1}-1)} \setminus X'$
  Set $i = i + 1$ and $X' = \emptyset$
Set $h' = \lceil \log_2 i - 1 \rceil$
for $j = 0, \dots, i-1$ **do**
  Set $b_{p+h'-1}^\tau \dots b_p^\tau = [j]_2$ for all $\tau_j \leq \tau \leq \tau_{j+1} - 1$
return B

Algorithm 6.1 now tests for a given range of cars τ' to $\bar{\tau}$ whether there exists a k-recoverable break set in $X_{(\tau',\bar{\tau})}$, such that the number of remaining breaks is smaller than 2^p. If this is the case, we consider in a next step the cars from τ' to $\bar{\tau}+1$. Otherwise τ' to $\bar{\tau}$ is fixed to form a block. In Theorem 6.3.6 we show that this strategy constructs an optimal recoverable robust schedule.

Theorem 6.3.6. *For any $p \geq 0$ and $k \geq 0$, Algorithm 6.1 computes an optimal recoverable robust train classification schedule.*

Proof. Let $\bar{B}$ denote a schedule of κ blocks computed by Algorithm 6.1 and $\bar{\tau}_1, \ldots, \bar{\tau}_\kappa$ be the heads of the blocks. The unresolved breaks of any block of $\bar{B}$ present a k-recoverable set of breaks by construction and all original breaks are considered. It follows that $\bar{B}$ is a feasible recoverable robust algorithm.

Assume for contradiction that $\bar{B}$ is not optimal. Let B^* be an optimal recoverable robust schedule, κ^* its number of blocks, and $\tau_1^*, \ldots, \tau_{\kappa^*}^*$ their heads. Let $\bar{\tau}_i \neq \tau_i^*$, $i \in \{0, \ldots, \kappa\}$, with $\bar{\tau}_j = \tau_j^*$ for all $j < i$, and w.l.o.g. let B^* be an optimal schedule that *maximizes* the value of i. We distinguish two cases:

Case 1: $\bar{\tau}_i > \tau_i^*$. Replacing the bitstrings $b^{*\tau_{i-1}^*}, \ldots, b^{*\bar{\tau}_i}$ in B^* by the bitstrings $\bar{b}^{\tau_{i-1}^*}, \ldots, \bar{b}^{\bar{\tau}_i}$ of $\bar{B}$ produces an optimal schedule in which the head of the ith block is given by $\bar{\tau}_i$. This contradicts the choice of B^*.

Case 2: $\bar{\tau}_i < \tau_i^*$. Let Y_{i-1}^* be the set of unresolved breaks of the $(i-1)$th block of B^*, and $\overline{Y}_{i-1}$ be the set of unresolved breaks of the $(i-1)$th block of $\bar{B}$. Since $\overline{Y}_{i-1}$ is a maximum k-recoverable break set, its size satisfies $|\overline{Y}_{i-1}| \geq |Y_{i-1}^*|$. Consider the number of resolved breaks of B^* in its $(i-1)$th block:

$$|X_{(\bar{\tau}_{i-1},\tau_i^*)}| - |Y_{i-1}^*| \geq |X_{(\bar{\tau}_{i-1},\tau_i^*)}| - |\overline{Y}_{i-1}| > |X_{(\bar{\tau}_{i-1},\bar{\tau}_i)}| - |\overline{Y}_{i-1}| = 2^p - 1.$$

This is a contradiction since there are at most 2^p different bitstrings in a block. As a consequence $\bar{B}$ is optimal. □

In Algorithm 6.1 the step of computing a maximum k-recoverable break set is not specified. One way of solving this problem is by integer programming. For any break $\beta \in X = \cup_{S \in \mathcal{S}} X^S \cap X_{(\tau_1,\tau_2)}$ the binary variable x_β indicates if β is part of the k-recoverable break set ($x_\beta = 1$) or not ($x_\beta = 0$). Any optimal solution of the following ILP represents a maximum k-recoverable break set

$$\begin{aligned} \max \sum_{\beta \in X} x_\beta & \\ \sum_{\beta \in X^S} x_\beta &\leq 2^k - 1 \quad \forall S \in \mathcal{S} \\ x_\beta &\in \{0,1\} \quad \forall \beta \in X. \end{aligned}$$

As we will show in the following section there is no polynomial time algorithm for solving this problem for general scenario sets, unless $\mathbf{P} = \mathbf{NP}$. On the other hand, for the class of scenarios, in which each scenario delays up to j trains, the maximum k-recoverable break set can be computed efficiently and thus the recoverable robust train classification problem can be solved in polynomial time.

6.3.2. Computational Complexity

In this subsection we assume w.l.o.g. that we are looking for a maximum k-recoverable break set between the cars 1 and n, i.e., let $\mathcal{S}$ be a set of scenarios, find a maximum k-recoverable break set X' of $X = \cup_{S\in\mathcal{S}} X^S$. By a reduction from the independent set problem, this problem is strongly **NP**-hard for $k = 1$.

Lemma 6.3.7. *Let $T_1, \ldots, T_\ell$ be a sequence of ℓ inbound trains, $\mathcal{S}$ be a set of scenarios, and $K \geq 0$. Deciding whether there exists a feasible 1-recoverable break set $X' \subseteq \cup_{S\in\mathcal{S}} X^S$ of size greater than or equal to K is strongly* **NP***-hard.*

Proof. We show a reduction from the independent set problem, which is well known to be strongly **NP**-hard [52]. Let $G = (V, E)$, $V = \{1, \ldots, n\}$, be a graph in which we look for an independent set of size a. W.l.o.g., we assume that G contains no isolated vertex and $|V| - a = 2^p - 1$. In the corresponding 1-recoverable break set instance there are $n + 1$ cars. Furthermore, for every edge $(i, j) \in E$ the scenario set contains a scenario S that induces the set of breaks $X^S = \{(i, i+1), (j, j+1)\}$. Note that the size of the break set $X = \cup_{S\in\mathcal{S}} X^S$ and the number of scenarios $\mathcal{S}$ is polynomial in the size of G. We show that there is a 1-recoverable break set $X' \subseteq X$ of size a if and only if there is an independent set of size a in G.

Let A be an independent set in G with $|A| = a$. Define the break set $X_A = \{(i, i+1) | i \in A\}$. For any scenario $S \in \mathcal{S}$ with $X^S = \{(i, i+1), (j, j+1)\}$, there is an edge $\{i, j\} \in E$ and either $i \notin A$ or $j \notin A$. For this reason at most one of the breaks of X^S is an element of X_A and thus X_A is a 1-recoverable break set of size a.

Conversely, let X' be a 1-recoverable break set of size a. Define $A = \{i \mid (i, i+1) \in X'\}$. For any edge $(i, j) \in E$ there is a scenario $S \in \mathcal{S}$ with $X^S = \{(i, i+1), (j, j+1)\}$. Since either $(i, i+1) \notin X'$ or $(j, j+1) \notin X'$, there is at most one endpoint of the edge (i, j) in A. Therefore, A is an independent set of size a. □

A quite different reduction from 2^kSAT leads to the **NP**-hardness of the maximum k-recoverable break set problem for any constant $k \geq 2$.

Lemma 6.3.8. *It is strongly* **NP***-hard to decide whether there exists a k-recoverable break set of size K for any constant $k \geq 2$.*

Proof. We show a reduction from 2^kSAT, which is strongly **NP**-hard [52]. Let I be an instance of 2^kSAT composed of n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Each clause consists of 2^k different literals. For each variable x_i we add the auxiliary variables $x_{1,i}, \ldots, x_{2^{k-1}-1,i}$ and the clause

$$\overline{C}_i = x_i \vee \bar{x}_i \vee x_{1,i} \vee \bar{x}_{1,i} \vee \ldots \vee x_{2^{k-1}-1,i} \vee \bar{x}_{2^{k-1}-1,i}$$

to the problem instance. The new instance is a yes-instance if and only if the old instance is one.

In the maximum k-recoverable break set instance there are $2(n + n(2^{k-1} - 1)) + 1$ cars. For each clause C_j we construct a scenario S_j with $\beta_i = (2i - 1, 2i) \in X^{S_j}$ if and only if $x_i \in C_j$ and $\bar{\beta}_i = (2i, 2i + 1) \in X^{S_j}$ if and only if $\bar{x}_i \in C_j$. Furthermore, $\mathcal{S}$ contains for every clause $\overline{C}_i$, $i = 1, \ldots, n$, the scenario $\overline{S}_i$ which induces the breaks β_i, $\bar{\beta}_i$, $\beta_{j,i} = (2n + 2^k(i - 1) + 2j - 1, 2n + 2^k(i - 1) + 2j)$ and $\bar{\beta}_{j,i} = (2n + 2^k(i - 1) + 2j, 2n + 2^k(i - 1) + 2j + 1)$ for $j = 1, \ldots, 2^{k-1} - 1$. The maximum k-recoverable break set instance contains $m + n$ scenarios and $2n + n(2^k - 2)$ breaks. Since k

is constant, the size of the instance is polynomial in the size of the input data. We will show that there exists a k-recoverable break set $X' \subseteq \cup_{S\in\mathcal{S}} X^S$ with exactly $n + n(2^k - 2)$ breaks if and only if I is a yes-instance.

Let X' be a maximum k-recoverable break set with $|X'| = n + n \cdot (2^k - 2)$. W.l.o.g., all breaks $\beta_{i,j}$ and $\bar{\beta}_{i,j}$ are contained in X'. Otherwise replace β_i by one missing $\beta_{i,j}$ and $\bar{\beta}_i$ by one missing $\bar{\beta}_{i,j}$ to obtain again a maximum k-recoverable break set. We will show that X' defines a feasible assignment to I.

Due to $|\cup_{S\in\mathcal{S}} X^S| - |X'| = n$ and the definition of $\overline{C}_i$, $i = 1, \ldots, n$, the set X' contains either β_i or $\bar{\beta}_i$. Thus, the following assignment is well defined:

$$x_i^* = \begin{cases} \text{true} & \text{if } \beta_i \notin X' \\ \text{false} & \text{if } \bar{\beta}_i \notin X' \end{cases}.$$

Since every scenario S_j contains 2^k breaks, at least one break in X^{S_j} is resolved, i.e., there exists an $\ell \in \{1, \ldots, n\}$ such that $\beta_\ell \in X^{S_j} \backslash X'$ or $\bar{\beta}_\ell \in X^{S_j} \backslash X'$. If $\beta_\ell \in X^{S_j} \backslash X'$, the clause C_j contains the literal x_ℓ and $x_\ell^* = \text{true}$ verifies C_j. If $\bar{\beta}_\ell \in X^{S_j} \backslash X'$, the clause C_j contains the literal $\bar{x}_\ell$ and $x_\ell^* = \text{false}$. Hence, x^* verifies every clause.

Conversely, let x^* be an assignment, which satisfies the 2^kSAT instance I. If $x_i^* = \text{true}$, we delete β_i from $\cup_{S\in\mathcal{S}} X^S$ and if $x_i^* = \text{false}$, we delete $\bar{\beta}_i$ to construct a maximum k-recoverable break set X'. The set X' consists of $n + n \cdot (2^k - 2)$ elements. Furthermore, in every scenario S_j there are at most $2^k - 1$ breaks unresolved. Therefore, X' is a k-recoverable break set. □

Note that this reduction does not imply **NP**-hardness for $k = 1$, since 2SAT is solvable in polynomial time.

Both lemmas not only state that Algorithm 6.1 can just be implemented in polynomial time, if $\mathbf{P} = \mathbf{NP}$, but also enable us to prove the **NP**-hardness of the recoverable robust classification problem.

Theorem 6.3.9. *Let $T_1, \ldots, T_\ell$ be a sequence of ℓ inbound trains, $\mathcal{S}$ be a set of scenarios, $h, p \geq 0$, and $k \geq 1$ constant. Deciding whether there exists a feasible recoverable robust classification schedule of length of at most h is strongly* **NP***-hard.*

Proof. We show a reduction from the decision version of the maximum k-recoverable break set problem. Let $\mathcal{S}$ be a set of scenarios, in which each scenario induces breaks between the cars $1, \ldots, n$, $X = \cup_{S\in\mathcal{S}} X^S$ and $K \in \mathbb{N}$ form a maximum k-recoverable break set instance. W.l.o.g., $K < |X|$. In order to schedule all cars within one block, we suppose for a moment $|X| - K = 2^\nu - 1$ for some integer $\nu \in \mathbb{N}$. In that case there exists a feasible recoverable robust schedule with $p = \nu$ of length at most ν for this instance if and only if there is a k-recoverable break set X' of size K in X.

If $2^\nu - 1 > |X| - K > 2^{\nu-1} - 1$, we define $\gamma = 2^\nu - 1 - |X| + K$ and add the cars $n+1, \ldots, n+\gamma+1$ to the instance in reversed order. Thus, there are γ original breaks that need to be recovered using $\gamma + 1$ different bitstrings. A maximum k-recoverable break set X' for the old instance is also a maximum k-recoverable break set for this new instance. Furthermore, there exists a feasible recoverable robust schedule with $p = \nu$ of length at most ν for this instance if and only if there is a k-recoverable break set X' of size K in X. □

Note that if the number of scenarios is polynomially bounded by the number of cars, the recoverable robust train classification problem is in **NP**, since we can test the feasibility of a schedule for every single scenario exploiting Lemma 6.3.4. In general this may not be the case.

6.3.3. Problem Variants

Infeasible Initial Solutions In our model the first-stage solution is a feasible classification schedule for the original order of trains. A special case of this setting is, to allow recovery even in the case of no disturbance. We can then model the original breaks by a scenario S_{org} which induces all original breaks and no original breaks are considered, i.e., one assumes that the cars arrive in perfect order.

Extremal Values of p So far, we were given some values $p, k \geq 0$ and looked for the optimal length h of a recoverable robust schedule. Conversely, given a length value $h \geq 0$ and some $k \geq 0$, we can ask for the maximum value of p for which there exists a recoverable robust schedule of length at most h.

An obvious way to calculate this optimal threshold p is the following: successively increasing p from 0 to h, calculate the minimum value $h_{p,k}$ for which there is a feasible recoverable robust schedule of length $h_{p,k}$; then, simply take the maximum value p for which $h_{p,k} \leq h$. Note that this value can also be estimated with binary search between $p = 0$ and $p = h$. For a general set of scenarios $\mathcal{S}$, calculating this value of p is an **NP**-hard problem since deciding whether there exists a feasible recoverable robust classification schedule of length at most h with $k = 1$ is an **NP**-hard problem by Corollary 6.3.9. A corresponding statement holds for fixing h and p and minimizing the value of k.

Several Outgoing Trains As shown in [63], the classification task involving m outgoing trains with their order specification is equivalent of finding an optimal schedule for one outgoing train as long as the length of the classification tracks is unbounded. In that case, one computes for every single outgoing train an optimal classification schedule and applies these schedules simultaneously. Since the recovery action can be carried out independently for every schedule, this approach can also be used to deal with the recoverable robust train classification problem with several outgoing trains.

6.4. Limited Number of Delayed Trains

As mentioned before, providing strict robustness wastes a lot of potential to extreme scenarios which rarely occur. For this reason we introduce a simple yet general class of scenarios, in which the number of delayed trains is restricted.

Given some parameter j, the scenario set $\mathcal{S}_j$ consists of all scenarios which delay up to j trains. More formally, let $T_1, \ldots, T_\ell$ be a set of trains. A scenario S defines an order $\sigma_S : \{1, \ldots, \ell\} \to \{1, \ldots, \ell\}$ in which the trains actually arrive at the yard, i.e., $\sigma_S(i)$ indicates the position of train i and thus they form the inbound train $\theta^{\sigma_S} = T_{\sigma_S^{-1}(1)} \ldots T_{\sigma_S^{-1}(\ell)}$. Then, a sequence $T_{\bar{\sigma}^{-1}(1)}, \ldots, T_{\bar{\sigma}^{-1}(\ell)}$, where $\bar{\sigma}$ is some permutation, is called an (α, k)-*delayed train sequence* of θ^{σ_S} if $\sigma_S(\alpha) < k$ and the following conditions hold: $\bar{\sigma}(x) = \sigma_S(x)$ if $\sigma_S(x) < \sigma_S(\alpha)$ or $\sigma_S(x) > k$, $\bar{\sigma}(x) = \sigma_S(x) - 1$ for $\sigma_S(\alpha) < \sigma_S(x) < k$, and $\bar{\sigma}(\alpha) = k$ (see Figure 6.3). In other words, train T_α is delayed from the $\sigma_S(\alpha)$th to the kth position. The set of scenarios $\mathcal{S}_j$, $0 \leq j \leq \ell$, is now defined to contain a scenario S (inducing some sequence θ^S) if and only if there is a sequence $\theta^0, \ldots, \theta^j$ of train sequences θ^i such that $\theta^0 = \theta^{\text{id}}$ with $\text{id}(j) = j$, $j = 1, \ldots, \ell$, θ^i is an (α_i, k_i)-delayed sequence of θ^{i-1} for all $i = 1, \ldots, j$, and $\theta^i = \theta^S$. Every train T_{α_i} will be called to be delayed by S.

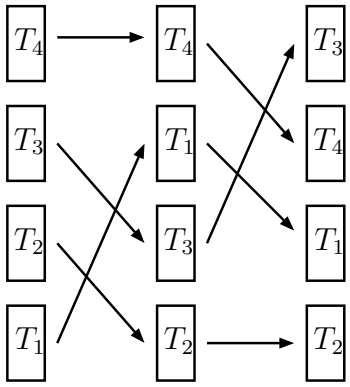


Figure 6.3.: In a first step, train T_1 is delayed to position 3 and in a second step train T_3 to position 4. The final sequence belongs to a scenario in $\mathcal{S}_2$.

As we will see in Theorem 6.4.1 all our considerations can be restricted to a dominant subset $\overline{\mathcal{S}}_j$ of the scenario set $\mathcal{S}_j$. A scenario S is a member of $\overline{\mathcal{S}}_j$ if there is a sequence $\theta^0, \ldots, \theta^j$ of train sequences θ^i such that $\theta^0 = \theta^{\mathrm{id}}$, θ^i is an (α_i, ℓ)-delayed sequence of θ^{i-1} for all $i = 1, \ldots, j$, $\alpha_i < \alpha_{i-1}$ for all $i = 1, \ldots, j$ and $\theta^j = \theta^S$. In other words, if two trains are delayed by a scenario $S \in \mathcal{S}_j$, they swap their relative order and arrive later than all punctual trains. Note that for uniquely defining a scenario $S \in \mathcal{S}_j$ it suffices to list the j delayed trains since the order and amount of their delay is determined by the definition of $\overline{\mathcal{S}}_j$.

Theorem 6.4.1. *If B is a recoverable robust classification schedule for $\overline{\mathcal{S}}_j$ then B is a recoverable robust classification schedule for $\mathcal{S}_j$.*

Proof. Let $S \in \mathcal{S}_j$ and $\{T_{i_1}, \ldots, T_{i_j}\}$ be the set of trains delayed by S with $i_1 < i_2 < \ldots < i_j$. Consider the scenario $S' \in \overline{\mathcal{S}}_j$ delaying these trains and let $\sigma_{S'}$ be their corresponding order. Let $(\tau, \tau+1)$ be a break induced by S. If $(\tau, \tau+1)$ is an original break, B resolves it. If $(\tau, \tau+1)$ is a potential break, then there are trains $T_x, T_y \in \{T_1, \ldots, T_\ell\}$, $x < y$ with $\tau \in T_x$ and $\tau+1 \in T_y$. If $y \notin \{i_1, \ldots, i_j\}$, $\sigma_{S'}(x) \geq \ell - j + 1 \geq \sigma_{S'}(y)$. If $y \in \{i_1, \ldots, i_j\}$, $\sigma_{S'}(x) > \sigma_{S'}(y)$. In both cases, S' induces the break $(\tau, \tau+1)$. Therefore, a (p,k)-extension of B resolving all breaks of $\overline{S}$ also resolves all breaks of S. Thus, B is a recoverable robust classification schedule for $\mathcal{S}_j$. □

Due to Theorem 6.4.1 we restrict all further observations to the scenario set $\overline{\mathcal{S}}_j$. For this scenario set not only a scenario S is already defined by the set of delayed trains, but also the set of potential breaks induced by S.

Lemma 6.4.2. *Let $\beta = (\tau, \tau+1)$ be a potential break with $\tau \in T_x$, $x \in \{1, \ldots, \ell\}$. Then the scenario $S \in \overline{\mathcal{S}}_j$ induces β if and only if T_x is a train delayed by S.*

Proof. Let $\tau+1 \in T_y$, $y \in \{1, \ldots, \ell\}$. Since β is a potential break, $x < y$. If train T_x is delayed by S, $\sigma_S(x) > \sigma_S(y)$ and therefore $\beta \in X^S$. If $\beta \in X^S$, $\sigma_S(x) > \sigma_S(y)$. Since $\sigma_S(x) < \sigma_S(y)$ for all $y > x$ if T_x is not delayed by scenario S, the train T_x is delayed by S. □

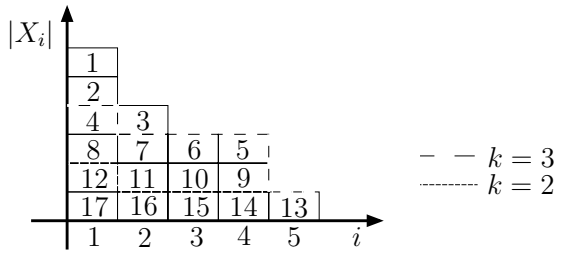


Figure 6.4.: The numbers indicate the order of removal of a break in X_i by Algorithm 6.2. For $j = 2$, $k = 3$, the algorithm stops after the 3rd iteration. In the case of $k = 2$ it stops after the 11th removal of a break.

Since any car belongs to exactly one train, the set of potential breaks X^S of any scenario $S \in \overline{\mathcal{S}}_j$ can be partitioned into disjoint subsets w.r.t. the respective delayed train causing the break. We denote this set of breaks induced by train T_i with X_i, i.e., $X_i = \{(\tau, \tau+1) \mid \tau \in T_i, \exists y > i : \tau + 1 \in T_y\}$. The observation that these sets are disjoint provides the key observation for the following algorithm, which computes a maximum k-recoverable break set for $\overline{\mathcal{S}}_j$ in the set $X_{(\tau',\bar{\tau})}$ for two given cars τ', $\bar{\tau}$. The algorithm successively removes one break from the break set $X_i \cap X_{(\tau',\bar{\tau})}$ with the highest number of potential breaks until there are less than 2^k potential breaks in the worst-case scenario, i.e., in the union of the j break sets induced by trains with maximum remaining size (see Figure 6.4).

Algorithm 6.2 Maximum k-Recoverable Break Set for $\mathcal{S}_j$

Input : Parameters $j, k \in \mathbb{N}$ and sets of induced breaks $X_1, \ldots, X_\ell$, with $X_i \subseteq \{(\tau_1, \tau_1+1), \ldots, (\tau_2, \tau_2+1)\}$, $j, k \in \mathbb{N}$
Output : Maximum recoverable break set

Sort $X_1, \ldots, X_\ell$ such that $|X_{i_1}| \geq |X_{i_2}| \geq \ldots \geq |X_{i_\ell}|$
Set $\alpha := \max\{i_t \mid |X_{i_t}| = |X_{i_1}|\}$
while $\sum_{t=1}^{j} |X_{i_t}| \geq 2^k$ **do**
  Remove an arbitrary break from X_{i_α}
  Update α
return X'

Lemma 6.4.3. *Given a set of potential breaks X_i induced by the delay of train T_i according to the expected inbound order for $i = 1, \ldots, \ell$, Algorithm 6.2 computes a maximum k-recoverable break set $X' \subseteq \cup_{i=1}^{\ell} X_i$ for $\overline{\mathcal{S}}_j$ in $\mathcal{O}(x \log \ell)$ with $x = |\cup_{i=1}^{\ell} X_i|$.*

Proof. For the proof we assume that the numbering of the trains is monotonically decreasing according to the input size of X_i, i.e., $|X_i| \geq |X_j|$ for $i < j$. Note that this order is invariant for the while-loop of Algorithm 6.2. Let X' be the set of breaks returned by the algorithm and $Y_i = X' \cap X_i$ for $i = 1, \ldots, \ell$.

We start by proving that Algorithm 6.2 computes a feasible recoverable break set. Let $S \in \overline{\mathcal{S}}_j$ be any scenario delaying trains $T_{i_1}, \ldots, T_{i_j}$. Due to the construction of X' we get

$$|X' \cap X^S| = \sum_{r=1}^{j} |X' \cap X_{i_r}| = \sum_{r=1}^{j} |Y_{i_r}| \leq \sum_{r=1}^{j} |Y_r| \leq 2^k - 1.$$

Therefore, all unresolved breaks of X' induced by S can be recovered (Lemma 6.3.4), and thus X' is a k-recoverable break set.

We continue with the optimality of X'. Let us consider the following ILP-formulation to compute a maximum k-recoverable break set, where the variables $z_i \in \mathbb{N}$ represent the number of remaining breaks of the set X_i, $i = 1, \ldots, \ell$:

$$\begin{aligned} \text{(ILP1)} \qquad & \max \sum_{i=1}^{\ell} z_i \\ & \sum_{i \in J} z_i \leq L' \qquad \forall J \subseteq \{1, \ldots, \ell\}, |J| = j \\ & z_i \leq |X_i| \quad \forall i = 1, \ldots, \ell \\ & z_i \in \mathbb{N}. \end{aligned}$$

With $L' = 2^k - 1$ any solution z^* can be transformed into a k-recoverable break set by removing $|X_i| - z_i^*$ potential breaks from the set X_i, $i = 1, \ldots, \ell$. The ILP1 can be solved by repeatedly solving ILP2 with L taken as variable and constant c by starting with $c_0 := \sum_{i=1}^{\ell} |X_i|$ and decreasing in every iteration c by one unit until the optimal value of ILP2 is not greater than L':

$$\begin{aligned} \text{(ILP2)} \qquad & \min L \\ & \sum_{i=1}^{\ell} z_i = c \\ & \sum_{i \in J} z_i \leq L \qquad \forall J \subseteq \{1, \ldots, \ell\}, |J| = j \\ & z_i \leq |X_i| \quad \forall\, i = 1, \ldots, \ell \\ & z_i \in \mathbb{N} \end{aligned}$$

We will show that in the tth iteration of the while loop the reduced size of X_i forms an optimal solution to ILP2 for $c = c_0 - t$. We therefore denote with X_i^t the current size of X_i after the tth iteration took place and define $z_i^t = |X_i^t|$ and $L^t = \sum_{i=1}^{j} z_i^t$. Any solution z_i^t is monotonically decreasing, i.e., $z_i^t \geq z_j^t$ for $i \geq j$. Furthermore, let κ^t be the maximum index with $|X_{\kappa^t}| - z_{\kappa^t}^t > 0$. Then due to the construction in the algorithm, $z_1^t \leq z_{\kappa^t}^t - 1$.

Let us now assume that there exists a t', such that the values of $z_i^{t'}$ and $L^{t'}$ are not an optimal solution for ILP2 with $c = \sum_{i=1}^{\ell} x_i - t'$ and $L^{t'} \geq 2^k - 1$ but z^* and L^*. The solution z_i^* is w.l.o.g. monotonically decreasing, since for $a, b \in \{1, \ldots, \ell\}$ with $z_a^* < z_b^*$ but $a > b$, $z_b' := z_b^* - 1$ and $z_i' = z_i^*$ is an optimal solution. Furthermore, $L^t \geq L^* + 1$ and thus we obtain

$$\sum_{i=1}^{j} z_j^{t'} = L^{t'} \geq L^* + 1 \geq \sum_{i=1}^{j} z_j^* + 1.$$

Hence, there exists an index $i_1 \in \{1, \ldots, j\}$ with $z_{i_1}^{t'} > z_{i_1}^*$. For all indices $i \in \{j+1, \ldots, \kappa^{t'}\}$ we get $z_i^{t'} \geq z_{i_1}^{t'} - 1 \geq z_{i_1}^* \geq z_i^*$. Since $z_i^{t'} = |X_i|$ for all $i \geq \kappa^{t'} + 1$,

$$\sum_{i=1}^{\ell} z_j^* \leq \sum_{j=1}^{j} z_j^{t'} - 1 + \sum_{i=j+1}^{\ell} z_j^{t'} = c - 1,$$

a contradiction.

The loop in Algorithm 6.2 is executed at most x times with $x = \sum_{t=1}^{j} |X_{i_t}|$, and one execution takes $\mathcal{O}(\log \ell)$. Outside the loop, the sorting takes $\mathcal{O}(\ell \log \ell)$, so the total running time is $\mathcal{O}(x \log \ell)$ and thus polynomial. □

Theorem 6.4.4. *The recoverable robust train classification problem with $\mathcal{S}_j$ is solvable in polynomial time.*

Theorem 6.4.4 is an immediate consequence of Lemma 6.4.3 and an optimal recoverable robust train classification schedule can be computed efficiently by combining Algorithm 6.2 and Algorithm 6.1. The resulting algorithm is implemented in the following section and tested for a number of real-world classification instances.

6.4.1. Experimental Evaluation

For the evaluation of the algorithm just described, we took the five real-world instances used in [60], which correspond to five days of traffic in the Swiss classification yard Lausanne Triage. Their volumes range from 310 to 486 with numbers of inbound trains between 44 and 49, outbound trains between 24 and 27, and numbers of breaks between 24 and 28 (see Table 6.1). In order to obtain unique types of cars, we converted all cars of the same type between two consecutive original breaks to distinct types ascending in the order the cars appear between the breaks, as shown in the following example: if $\tau_3 = \tau_6 = 3$, $\tau_7 = \tau_9 = \tau_{10} = 4$, and (τ_{12}, τ_3) as well as (τ_{10}, τ_2) formed original breaks, we set $\tau_3 = 3$, $\tau_6 = 4$, $\tau_7 = 5$, $\tau_9 = 6$, $\tau_{10} = 7$, and further $\tau_2 = 8$.

instance	n	ℓ	m	$\beta_{\max}$	β
inst-1	486	49	24	3	28
inst-2	329	44	24	4	24
inst-3	310	47	24	3	25
inst-4	364	44	24	3	25
inst-5	368	44	27	3	25

Table 6.1.: The five problem instances correspond to five traffic days: n denotes the number of cars, ℓ denotes the number of inbound trains, m denotes the number of outbound trains, $\beta_{\max}$ denotes the maximum number of breaks of outbound trains, and β denotes the total number of breaks.

Essentially, through adjusting the parameters p, k, and j, the algorithm allows flexibly trading off shortest schedules against the other extreme of strict robustness. Given some train classification instance, let $\underline{h}$ denote the length of an optimal non-robust schedule and $\bar{h}$ the length of an optimal strictly robust schedule. The values $\underline{h}$ and $\bar{h}$ present the lower and upper bounds for the length resulting from any combination of j, k, and p.

Table 6.2 summarizes the computed length of an optimal recoverable robust schedule according to the different parameters p, k and j. As lower and upper bounds for these lengths inst-2 requires $\underline{h} = 3$, while all other instances yield $\underline{h} = 2$ and $\bar{h} = 5$ for all instances.

If only small amounts of recovery action ($k = 1$) are allowed, for $j = 1$ the schedule length does not exceed $\underline{h}$ for inst-1 and inst-5 with $p = 0$, for inst-3 and inst-4 with $p \leq 1$, and for inst-2 even for $p \leq 3$, so yet for lowest degrees of recovery we obtain some robustness without increasing the length beyond that of an optimal non-robust schedule. Raising the degree of disturbance to $j \geq 2$, we still obtain a length $h = 4 < \bar{h}$ if the value of p is increased to $p = 1$ for inst-1, to $p = 2$ for inst-2, inst-4, and inst-5, and even to $p = 3$ for inst-3. These values are significantly smaller than these for the strictly robust methods.

	$k=0$		$k=1$										$k=2$												
	$p=0$		$p=0$		$p=1$		$p=2$		$p=3$		$p=4$		$p=0$		$p=1$				$p=2$				$p=3$		
j	0	1	1	2	1	2	1	2	1	2	1	2	3	4	1	2	3	4	1	2	3	4	2	3	4
inst-1	2	5	2	4	3	4	3	5	4	5	4	5	2	3	2	3	3	4	2	3	3	4	3	4	5
inst-2	3	5	3	4	3	4	3	4	3	5		5	3	3	3	3	3	3	3	3	3	4	3	4	4
inst-3	2	5	2	4	2	4	3	4	3	4		5	2	3	2	2	2	3	2	3	3	4	3	3	4
inst-4	2	5	2	4	2	4	3	4	3	5		5	2	3	2	2	2	3	2	3	3	4	3	3	4
inst-5	2	5	2	4	3	4	3	4	3	5		5	2	3	2	3	3	3	2	3	3	4	3	3	4

	$k=2$		$k=3$																$k=4$		
	$p=4$		$p=0$		$p=1$				$p=2$						$p=3$				$p=1$	$p=2$	
j	3	4	7	8	5	6	7	8	3	4	5	6	7	8	4	5	6	7	8	≥ 1	13	≥ 14
inst-1	4	5	2	3	2	3	3	3	2	3	3	3	3	4	3	3	3	4	4	2	2	2
inst-2	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	4	3	3	3
inst-3		4	2	3	2	2	2	3	2	2	2	3	3	3			3	3	4	2	2	2
inst-4		5	2	2	2	2	2	3	2	2	2	3	3	3			3	3	4	2	2	3
inst-5		4	2	2	2	2	3	3	2	2	3	3	3	3		3	3	3	4	2	2	2

Table 6.2.: Optimal length of a recoverable robust classification schedule for different values of recovery parameters p and k and values for S_j. Omitted entries represent no meaningful choice of p.

The degree of robustness grows rapidly with increasing degrees of recovery, and for $k = 4$ with $p \leq \underline{h}$ — except for inst-4 with $p = 2$ — we can allow any number of delayed trains and still achieve the length $\underline{h}$ of an optimal non robust schedule. Between these extremes, Table 6.2 shows how far the value of p can be raised for $k = 2$ and arbitrarily high amounts of delay $j \geq 4$: most instances allow $p = 1$ to obtain $h = 3$, and $h = 4$ can be achieved even for $p = 4$ for three out of five instances. For $k = 3$, Figure 6.5 (left) summarizes the values of inst-1: a schedule of length 3 with a recovery action starting after the third sorting step suffices to cope with a delay of up to six trains and $p = 1$ allows $h \leq 3$ even for any disturbance value j. Similarly, for a fixed value of $p = 2$, Figure 6.5 (center) shows the rapid growth of robustness: except for inst-2, k must be raised rather quickly between $j = 1$ and $j = 3$ to achieve a length of $\underline{h}$, whereas the required value of k does not exceed 4 for higher disturbances $j \geq 6$. Conversely, Figure 6.5 (right) fixes $k = 2$ and shows the maximum value of p that allows a length of $\underline{h}$: $j = 1$ still allows $p = \underline{h}$ for all instances, but, except for inst-2, this length $\underline{h}$ cannot be achieved for any choice of p for high amounts of delay $j \geq 4$. Hence, higher values of k contribute much more to the potential of recovery than low values of p. Summarizing, through adjusting the recovery parameters k and p, our algorithm presents a tool to flexibly trade off between fast classification and robust schedules.

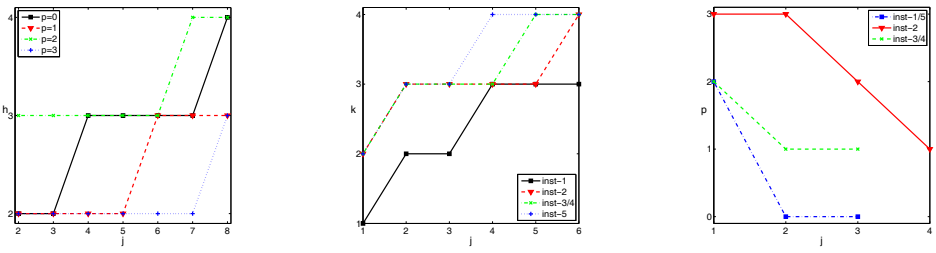


Figure 6.5.: left: optimal schedule lengths h of inst-1 for $k = 3$; center: smallest possible values of k to achieve a length of $\underline{h}$ for $p = 2$; right: highest possible values of p to achieve a length of $\underline{h}$ for $k = 2$;

6.5. Types in Fixed Order

So far we investigated the case in which all cars were distinct and had a fixed position in the outgoing train. Often cars are grouped in the classification process according to common characteristics like destination, design and capacity. In general just an order of groups is given as specification for the outbound train, where cars of the same group must appear consecutively but the relative order of cars in a group is free. In order to adapt the model described in Section 6.2 we represent the different cars $1, \ldots, n$ by a positive integer $\tau_i \in \{1, \ldots, \theta\}$, $\theta \leq n$, called the *type* of the car i, $i \in \{1, \ldots, n\}$. The method to derive a train classification schedule and properties of the schedules can also be easily adjusted as shown in [63]: A classification schedule is represented by an $(n \times h)$-matrix and it is feasible if both of the following conditions are satisfied for every pair $\tau_x < \tau_y$: if $x \leq y$, then $b^x \leq b^y$ and if $x > y$, then $b^x < b^y$ for all $x, y \in \{1, \ldots, n\}$. The algorithms to compute optimal schedules for distinct cars are based on determining maximal ascending sequences of consecutive cars that can be assigned the same bitstring. A partition of the inbound sequence of cars into such subsequences is called a *chain decomposition*. We redefine this notion for the generalized model where the inbound cars are not necessarily distinct in the following definition.

Definition 6.5.1. Let $T_1, \ldots, T_\ell$ be a sequence of inbound trains, $T_{\text{org}} = (\tau_1, \ldots, \tau_n)$ be their concatenation with $\tau_x \in \{1, \ldots, \theta\}$, $x = 1, \ldots, n$, and let $\tau_{n+1} = \theta + 1$. For any pair of cars x, y with $1 \leq \tau_x < \tau_y \leq \theta + 1$, let $[x, y]$ contain the car z, $1 \leq z \leq n$, if and only if one of the following three conditions holds:

1. $\tau_z = \tau_x$ and $z \leq x$,
2. $\tau_x < \tau_z < \tau_y$, or
3. $\tau_z = \tau_y$ and $z > y$.

We call $[x, y]$ a *chain* if $w < z$ for every pair $w, z \in [x, y]$ with $\tau_w < \tau_z$. A sequence $i_1, \ldots, i_{q+1}$ with $\tau_{i_1} < \ldots < \tau_{i_{q+1}}$ defines a *chain decomposition* (of *length* q) if $i_1 = \max\{z | \tau_z = 1\}$, $i_{q+1} = n + 1$, and $[i_x, i_{x+1}]$ is a chain for every $x = 1, \ldots, q$.

Note that the introduced dummy car $n + 1$ is the only car not contained in the chain decomposition. For a given chain decomposition $i_1, \ldots, i_{q+1}$, the schedule B with $b^z = [x]_2$ for $z \in [i_x, i_{x+1}]$ is a feasible schedule. According to these observations, we define predecessors $y \in \{1, \ldots, n + 1\}$ for some $x \in \{1, \ldots, n + 1\}$ if they can occur consecutively in a feasible chain decomposition.

Definition 6.5.2. Let $T_1, \ldots, T_\ell$ be a sequence of inbound trains with cars of types $\tau_i \in \{1, \ldots, \theta\}$, $i = 1, \ldots, n$, and let $\tau_{n+1} = \theta + 1$. For any car $y \in \{1, \ldots, n + 1\}$, we define the subset of cars

$$\text{pred}(y) = \{x \mid \tau_x < \tau_y \text{and } [x, y] \text{is a chain}\}.$$

Using this concept of predecessors, we set α as the largest index of any car of type 1, i.e., $\alpha = \max\{1 \leq i \leq n + 1 \mid \tau_i = 1\}$, and define the *chain representation graph* $G = (V, A)$ of a train classification instance as follows:

$$\begin{aligned} V &= \{1 \leq i \leq n + 1 \mid \tau_i > 1\} \cup \{\alpha\} \quad \text{and} \\ A &= \{(x, y) \in V \times V \mid x \in \text{pred}(y)\}. \end{aligned}$$

Any path from α to $n + 1$ in $G = (V, A)$ defines a chain decomposition, and a shortest path according to the number of arcs presents an optimal solution to the nominal train classification

problem. In the following we will extend this approach of finding a shortest path in the chain representation graph to the robust problem, i.e., to an uncertain order of the inbound sequence.

In order to embed the set of scenarios $\mathcal{S}$ into the chain representation model, we add for every scenario $S \in \mathcal{S}$ a cost function $c^S : A \to \{1, \ldots, n\}$ to the graph G. The cost function c^S assigns to every arc $(x, y) \in A$ the number of breaks induced by S in the chain $[x, y]$. A scenario S defining the new order of cars σ^S induces a *break* $(\tau, \tau + 1)$ in $[x, y]$ if there exist two cars $w, z \in [x, y]$ with $\tau_w = \tau$, $\tau_w = \tau_z + 1$ and $\sigma^S(w) > \sigma^S(z)$.

To derive a feasible recoverable robust train classification schedule, we use the same process as in the generic Algorithm 6.1: instead of a whole schedule we compute step by step a block of maximum length. W.l.o.g., we assume to start with the first block. Any block assigns at most 2^p different bitstrings to the trailing parts of its cars and in every scenario S there are at most $2^k - 1$ unresolved breaks. Thus, in the chain representation graph any block corresponds to a path p which starts in the vertex α, has a length of at most $2^p + 1$, and obeys $c^S(p) \leq 2^k - 1$ for all $S \in \mathcal{S}$. In order to find a block of maximum size, we add for every vertex $x \in V$ an arc $a_x = (x, n + 1)$ to G and set $c^S(a_x) = 0$ for all $S \in \mathcal{S}$ and $x \in V$. Furthermore, we define a new cost function $d : A \to \mathbb{N}$ rating the number of cars covered by a chain. More formally, for $a = (x, y) \in A$ we define

$$d(a) = \begin{cases} (\theta + 1 - \tau_x)n + x & \text{if } a = (x, \theta + 1) \\ 0 & \text{otherwise.} \end{cases}$$

Hence, a block of maximum length corresponds to a $(\alpha, \theta + 1)$-path p with length at most $2^p + 1$ and $c^S(p) \leq 2^k - 1$ for every $S \in \mathcal{S}$ that minimizes $d(p)$. This constrained shortest path problem can be solved in polynomial time if the number of constraints is constant and the values of the cost functions are bounded by the input. A dynamic program for it was introduced by Joksch [64] and a labeling Dijkstra algorithm by Aneja et al. [6]. We sum up this analysis in the following theorem.

Theorem 6.5.3. *The recoverable robust train classification problem can be solved in polynomial time via a constrained shortest path problem if the number of scenarios is bounded by a constant.*

Note that the run-time of the proposed procedure depends on the constant number of scenarios.

Single Cars and $\mathcal{S}_j$ In this part we consider a slight modification of the general setting in which each train of the incoming sequence consists of exactly one car. In this case the delay of one train, respectively one car, results in at most one break. Exploding this fact leads to a polynomial algorithm for $\mathcal{S}_j$. For $j \leq 2^k - 1$ this is obvious since the number of unresolved breaks induced in a block by any scenario in $\mathcal{S}_j$ is bounded by j. In any scenario these breaks can be recovered and hence any feasible schedule for the case of no delay is a feasible (p, k)-extension for $\mathcal{S}_j$. For $j > 2^k - 1$ the set $\mathcal{S}_j$ can be reduced to a single scenario as shown below.

Lemma 6.5.4. *Let $i_1, \ldots, i_q$, $\tau_{i_1} < \ldots < \tau_{i_q}$, with $q \leq 2^p + 1$ be a chain decomposition for a single block. For $x = 1, \ldots, q - 1$, define $f(x) = \tau_{i_{x+1}} - \tau_{i_x}$ if there is no car $z > i_{x+1}$ with $\tau_z = \tau_{i_{x+1}}$, and $f(x) = \tau_{i_{x+1}} - \tau_{i_x} - 1$ otherwise. Then, putting $b^z = [x]_2$ for every car $z \in [i_x, i_{x+1}]$ yields a recoverable robust subschedule for the cars of this block if and only if $\sum_{x=1}^{q-1} f(x) \leq 2^p + 2^k - 1$.*

Proof. Assume that $\sum_{x=1}^{q-1} f(x) > 2^p + 2^k - 1$, and define a set of cars X in the following way: for every chain $[i_x, i_{x+1}]$, $x = 1, \ldots, q-1$, choose $f(x) - 1$ different cars $z \in [i_x, i_{x+1}]$, such that each car is of a different type and $\tau_z > \tau_{i_x}$. Since $\sum_{x=1}^{q-1} f(x) > 2^p + 2^k - 1$, $|X| \geq 2^k$. Any scenario $S \in \mathcal{S}_j$ which delays 2^k of these trains, induces 2^k unresolved breaks between cars of different types. Hence, these unresolved breaks cannot be recovered, so B is not recoverable robust.

Conversely, let B satisfy $\sum_{x=1}^{q-1} f(x) \leq 2^p + 2^k - 1$. Consider the following recovery strategy: Let $\tau_1, \ldots, \tau_j$ be the set of cars that are delayed by a scenario. If several cars of the same type are delayed, at most one unresolved break needs to be recovered. Hence, it suffices to use the following recovery strategy: let z be a delayed car with $\tau_{i_x} \leq \tau_z < \tau_{i_{x+1}}$. If there is a car y in chain $[i_x, i_{x+1}]$ with $\tau_y > \tau_z$, we increase the bitstrings of each car y with $\tau_y > \tau_z$.

Since $f(x)$ denotes the number different types that occur in the chain $[i_x, i_{x+1}]$, $x = 1, \ldots, q-1$, at most $2^k - 1$ unresolved breaks between cars of different types can be induced by one scenario. Therefore, the above recovery strategy yields a (p, k)-extension of B for every scenario $S \in \mathcal{S}_j$. □

As a consequence of Lemma 6.5.4, instead of considering $\mathcal{S}_j$, it suffices to consider the chain representation graph for the single cost function $c^S : A \to \mathbb{N}$ with $c^S((x, y)) = \tau_y - \tau_x$ if there is no car $z > y$ with $\tau_z = \tau_y$, and $c^S((x, y)) = \tau_y - \tau_x - 1$ otherwise. Since $c^S(a) \leq n$ for all arcs a of the chain representation graph, this constrained shortest path problem can again be solved in polynomial time, which is finally summarized in the following theorem:

Theorem 6.5.5. *The recoverable robust train classification problem can be solved in polynomial time for the set of scenarios $\mathcal{S}_j$ in the case of singleton trains.*

6.6. Conclusion and Open Issues

We have developed a practically applicable algorithm for deriving robust train classification schedules of minimum length. In contrast to [30], we take into account multiple inbound trains, which allows integrating the most relevant disturbance in form of delayed trains. We have introduced the natural recovery action of (p, k)-extensions, for which we proved that the problem is **NP**-hard for every constant $k \geq 1$. Nevertheless, for the quite general set of scenarios $\mathcal{S}_j$ where each scenario can delay up to j trains, we showed that our generic algorithm of Section 6.3 computes a solution in polynomial time by solving the subproblem of calculating a maximum recoverable set of breaks efficiently. The experimental study of Section 6.4 indicates that the resulting algorithm improves on the current classification practice as it yields shorter schedules and still allows high degrees of robustness. Its flexibility further allows balancing between strictly robust and optimal non-robust schedules and raises potential for increased traffic throughput in classification yards. Finally, we have shown that the problem variant with non-distinct cars is polynomial-time solvable for a constant number of scenarios.

Future Work First of all, it is an open question whether the problem variant with non-distinct cars (Section 6.5) can be solved efficiently for $\mathcal{S}_j$ (or a similar natural set of scenarios). Further practical restrictions, such as a limited number of classification tracks (see [63]), should be considered in the context of robustness. Moreover, the number of cars rolled in presents a secondary objective, which can be additionally minimized for a minimum length. Finally, making the order of inbound trains part of the optimization yields a different robust optimization problem.

Bibliography

[1] E. H. L. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. Wiley, Chichester, UK, 1997.

[2] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[3] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation complexity of min-max (regret) versions of shortest path, spanning tree, and knapsack. In G. S. Brodal and S. Leonardi, editors, *Algorithms – ESA 2005*, volume 3669 of *LNCS*, pages 862–873. Springer, Berlin, 2005.

[4] H. Aissi, C. Bazgan, and D. Vanderpooten. Approximation of min–max and min–max regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179(2):281–290, 2007.

[5] H. Aissi, C. Bazgan, and D. Vanderpooten. Complexity of the min–max (regret) versions of min cut problems. *Discrete Optimization*, 5:66–73, 2008.

[6] Y. P. Aneja, V. Aggarwal, and K. P. K. Nair. Shortest chain subject to side constraints. *Networks*, 13(2):295–302, 1983.

[7] A. Atamtürk and M. Zhang. Two-stage robust network flow and design under demand uncertainty. *Operations Research*, 55(4):662–673, 2007.

[8] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8(1):146–164, 1975.

[9] J. E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41(11):1069–1072, 1990.

[10] R. Bellman. Notes on the theory of dynamic programming IV - maximization over discrete sets. *Naval Research Logistics Quarterly*, 3(1-2):67–70, 1956.

[11] A. Ben-Tal, A. Goryashko, E. Guslitzer, and A. Nemirovski. Adjustable robust solutions of uncertain linear programs. *Mathematical Programming*, 99(2):351–376, 2004.

[12] A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[13] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88(3):411–424, 2000.

[14] D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98(1-3):49–71, 2003.

[15] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[16] D. Bienstock. Histogram models for robust portfolio optimization. *Journal of Computational Finance*, 11(1):1–64, 2007.

[17] D. Bienstock and F. D'Andreagiovanni. Robust wireless network planning. In *Proceedings of the XL Annual Conference of the Italian Operational Research Society (AIRO)*, 2009.

[18] J. R. Birge and F. Louveaux. *Introduction to stochastic programming.* Springer, New York, 1997.

[19] A. Bley and J. Neto. Approximability of 3- and 4-hop bounded disjoint paths problems. In F. Eisenbrand and F. B. Shepherd, editors, *Integer Programming and Combinatorial Optimization*, volume 6080 of *LNCS*, pages 205–218. Springer, Berlin, 2010.

[20] M. Bruglieri, M. Ehrgott, H. W. Hamacher, and F. Maffioli. An annotated bibliography of combinatorial optimization problems with fixed cardinality constraints. *Discrete Applied Mathematics*, 154(9):1344–1357, 2006.

[21] M. Bruglieri, F. Maffioli, and M. Ehrgott. Cardinality constrained minimum cut problems: Complexity and algorithms. *Discrete Applied Mathematics*, 137(3):311–341, 2004.

[22] C. Büsing. The exact subgraph recoverable robust shortest path problem. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 231–248. Springer, Berlin, 2009.

[23] C. Büsing, A. M. C. A. Koster, and M. Kutschka. Recoverable robust knapsacks: The discrete scenario case. *Optimization Letters*, online first, 2011.

[24] C. Büsing and J. Maue. Robust algorithms for sorting railway cars. In M. de Berg and U. Meyer, editors, *Algorithms – ESA 2010*, volume 6346 of *LNCS*, pages 350–361. Springer, Berlin, 2010.

[25] V. Cacchiani, A. Caprara, L. Galli, L. Kroon, G. Maróti, and P. Toth. Recoverable robustness for railway rolling stock planning. In M. Fischetti and P. Widmayer, editors, *ATMOS 2008 - 8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems.* Schloss Dagstuhl - LZI, Dagstuhl, 2008.

[26] A. Caprara, L. Galli, L. Kroon, G. Maróti, and P. Toth. Robust train routing and online re-scheduling. In T. Erlebach and M. Lübbecke, editors, *ATMOS 2010 - 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, pages 24–33. Schloss Dagstuhl - LZI, Dagstuhl, 2010.

[27] C. Caramanis. *Adaptable Optimization: Theory and Algorithms.* PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Eng. and Comp. Science., 2006.

[28] J. Cheriyan and R. Thurimella. Approximating minimum-size k-connected spanning subgraphs via matching. *SIAM Journal on Computing*, 30(2):528–560, 2000.

[29] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and A. Navarra. Robust algorithms and price of robustness in shunting problems. In C. Liebchen, R. K. Ahuja, and J. A. Mesa, editors, *ATMOS 2007 - 7th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization and Systems.* Schloss Dagstuhl - IBFI, Dagstuhl, 2007.

[30] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, A. Navarra, M. Schachtebeck, and A. Schöbel. Recoverable robustness in shunting and timetabling. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 28–60. Springer, Berlin, 2009.

[31] S. Cicerone, G. Di Stefano, M. Schachtebeck, and A. Schöbel. Dynamic algorithms for recoverable robustness problems. In M. Fischetti and P. Widmayer, editors, *ATMOS 2008*

- *8th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems*. Schloss Dagstuhl - LZI, Dagstuhl, 2008.

[32] H. Crowder, E. L. Johnson, and M. Padberg. Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834, 1983.

[33] C. F. Daganzo, R. G. Dowling, and R. W. Hall. Railroad classification yard throughput: The case of multistage triangular sorting. *Transportation Research Part A: General*, 17(2):95–106, 1983.

[34] E. Dahlhaus, F. Manne, M. Miller, and J. Ryan. Algorithms for combinatorial problems related to train marshalling. In L. Branlovic and J. Ryan, editors, *AWOCA 2000 - 11th Australasian Workshop on Combinatorial Algorithms*, pages 7–16. University of Newcastle, Callaghan, NSW, 2000.

[35] G. B. Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2):266–277, 1957.

[36] P. De, E. J. Dunne, J. B. Ghosh, and C. E. Wells. Complexity of the discrete time-cost tradeoff problem for project networks. *Operations Research*, 45(2):302–306, 1997.

[37] K. Dhamdhere, V. Goyal, R. Ravi, and M. Singh. How to pay, come what may: Approximation algorithms for demand-robust covering problems. In J. D. Cantarella, editor, *FOCS 2005 - 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 367–378. IEEE Computer Society, Los Alamitos, California, 2005.

[38] G. Di Stefano and M. L. Koči. A graph theoretical approach to the shunting problem. *Electronic Notes in Theoretical Computer Science*, 92:16–33, 2004.

[39] C. Eggermont, C. A. J. Hurkens, M. Modelski, and G. J. Woeginger. The hardness of train rearrangements. *Operations Research Letters*, 37(2):80–82, 2009.

[40] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[41] L. El Ghaoui, F. Oustry, and H. Lebret. Robust solutions to uncertain semidefinite programs. *SIAM Journal on Optimization*, 9(1):33–52, 1998.

[42] A. L. Erera, J. C. Morales, and M. Savelsbergh. Robust optimization for empty repositioning problems. *Operations Research*, 57(2):468–483, 2009.

[43] J. E. Falk. Exact solutions of inexact linear programs. *Operations Research*, 24(4):783–787, 1976.

[44] U. Feige, K. Jain, M. Mahdian, and V. Mirrokni. Robust combinatorial optimization with exponential scenarios. In M. Fischetti and D. P. Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *LNCS*, pages 439–453, 2007.

[45] C. E. Ferreira, A. Martin, and R. Weismantel. Solving multiple knapsack problems by cutting planes. *SIAM Journal on Optimization*, 6(3):858–877, 1996.

[46] M. Fischetti and M. Monaci. Light robustness. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 61–84. Springer, Berlin, 2009.

[47] H. Flandorffer. Vereinfachte Güterzugbildung. *ETR RT*, 13:114–118, 1953.

[48] A. Frank. A weighted matroid intersection algorithm. *Journal of Algorithms*, 2:328 – 336, 1981.

[49] R. Freling, R. M. Lentink, L. G. Kroon, and D. Huisman. Shunting of passenger train units in a railway station. *Transportation Science*, 39(2):261–272, 2005.

[50] R. Fujita, Y. Kobayashi, and K. Makino. Robust matchings and matroid intersections. In M. de Berg and U. Meyer, editors, *Algorithms – ESA 2010*, volume 6347 of *LNCS*, pages 123–134. Springer, Berlin, 2010.

[51] H. N. Gabow, M. X. Goemans, É. Tardos, and D. P. Williamson. Approximating the smallest k-edge connected spanning subgraph by LP-rounding. *Networks*, 53(4):345–357, 2009.

[52] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, San Francisco, 1979.

[53] M. Gatto, J. Maue, M. Mihalák, and P. Widmayer. Shunting for dummies: An introductory algorithmic survey. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 310–337. Springer, Berlin, 2009.

[54] D. Golovin, V. Goyal, and R. Ravi. Pay today for a rainy day: Improved approximation algorithms for demand-robust min-cut and shortest path problems. In B. Durand and W. Thomas, editors, *STACS 2006*, volume 3884 of *LNCS*, pages 206–217, 2006.

[55] A. Gupta, V. Nagarajan, and R. Ravi. Thresholded covering algorithms for robust and max-min optimization. In S. Abramsky, C. Gavoille, C. Kirchner, F. M. auf der Heide, and P. G. Spirakis, editors, *ICALP 2010*, volume 6198 of *LNCS*, pages 262–274, 2010.

[56] E. Guslitser. Uncertainty-immunized solutions in linear programming. Master's thesis, Technion, Israel Institute of Technology, IE&M faculty, 2002.

[57] R. S. Hansmann and U. T. Zimmermann. Optimal sorting of rolling stock at hump yards. In H.-J. Krebs and W. Jäger, editors, *Mathematics - Key Technology for the Future*, chapter 5, pages 189–203. Springer, Berlin, 2008.

[58] R. Hassin and S. Rubinstein. Robust matchings. *SIAM Journal on Discrete Mathematics*, 15(4):530–537, 2002.

[59] R. Hassin and D. Segev. Robust subgraphs for trees and paths. *ACM Transactions on Algorithms*, 2(2):263–281, 2006.

[60] A. Hauser and J. Maue. Experimental evaluation of approximation and heuristic algorithms for sorting railway cars. In P. Festa, editor, *SEA 2010*, volume 6049 of *LNCS*, pages 154–165. Springer, Berlin, 2010.

[61] H. Iida. A note on the max-min 0-1 knapsack problem. *Journal of Combinatorial Optimization*, 3(1):89–94, 1999.

[62] R. Jacob. On shunting over a hump. Technical Report 576, Institute of Theoretical Computer Science, ETH Zürich, 2007.

[63] R. Jacob, P. Márton, J. Maue, and M. Nunkesser. Multistage methods for freight train classification. *Networks*, 2010. to appear.

[64] H. C. Joksch. The shortest route problem with constraints. *Journal of Mathematical Analysis and Applications*, 14(2):191–197, 1966.

[65] R. Kalaï and D. Vanderpooten. Lexicographic α-robust knapsack problem: Complexity results. In *IEEE International Conference on Service Systems and Service Management (ICSSSM 2006)*, pages 1103–1107, 2006.

[66] K. Kaparis and A. N. Letchford. Local and global lifted cover inequalities for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research*, 186(1):91–103, 2008.

[67] K. Kaparis and A. N. Letchford. Separation algorithms for 0-1 knapsack polytopes. *Mathematical Programming*, 124(1-2):69–91, 2010.

[68] O. E. Karaşan, M. Ç. Pinar, and H. Yaman. The robust shortest path problem with interval data. Technical report, Industrial Engineering Department, Bilkent University, 2001.

[69] R. M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972. Also known as Karp's 21 NP-complete problems.

[70] A. Kasperski and P. Zieliński. On the approximability of minmax (regret) network optimization problems. *Information Processing Letters*, 109(5):262–266, 2009.

[71] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, 2004.

[72] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, 49(2):291–307, 1970.

[73] R. Khandekar, G. Kortsarz, V. Mirrokni, and M. R. Salavatipour. Two-stage robust network design with exponential scenarios. In D. Halperin and K. Mehlhorn, editors, *Algorithms – ESA 2008*, volume 5193 of *LNCS*, pages 589–600. Springer, Berlin, 2008.

[74] S. Khuller and B. Raghavachari. Improved approximation algorithms for uniform connectivity problems. *Journal of Algorithms*, 21(2):434–450, 1996.

[75] G. W. Klau and R. Weiskircher. Robustness and resilience. In U. Brandes and T. Erlebach, editors, *Network Analysis*, volume 3418 of *LNCS*, chapter 15, pages 417–437. Springer, Berlin, 2005.

[76] O. Klopfenstein and D. Nace. A note on polyhedral aspects of a robust knapsack problem. 2007.

[77] B. Korte and J. Vygen. *Combinatorial Optimization: Theory and algorithms*. Springer, Berlin, 2008.

[78] P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, Dordrecht, 1997.

[79] K. Krell. Grundgedanken des Simultanverfahrens. *ETR RT*, 22:15–23, 1962.

[80] E. L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9(1):31–56, 1975.

[81] C. Liebchen, M. E. Lübbecke, R. H. Möhring, and S. Stiller. The concept of recoverable robustness, linear programming recovery, and railway applications. In R. K. Ahuja, R. H. Möhring, and C. D. Zaroliagis, editors, *Robust and Online Large-Scale Optimization*, volume 5868 of *LNCS*, pages 1–27. Springer, Berlin, 2009.

[82] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. Wiley, Chichester, 1990.

[83] J. M. Mulvey, R. J. Vanderbei, and S. A. Zenios. Robust optimization of large-scale systems. *Operations Research*, 43(2):264–281, 1995.

[84] M. W. Padberg. $(1, k)$-configurations and facets for packing problems. *Mathematical Programming*, 18(1):94–99, 1980.

[85] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In D. C. Young, editor, *FOCS 2000 - 41st Annual IEEE Symposium on Foundations of Computer Science*, pages 86–92. IEEE Computer Society, Los Alamitos, California, 2000.

[86] K. J. Pentinga. Teaching simultaneous marshalling. *The Railway Gazette*, pages 590–593, 1959.

[87] R. Ravi and A. Sinha. Hedging uncertainty: Approximation algorithms for stochastic optimization problems. *Mathematical Programming*, 108(1):97–114, 2006.

[88] J. Rosenhead, M. Elton, and S. K. Gupta. Robustness and optimality as criteria for strategic decisions. *Operational Research Quarterly*, 23(4):413–431, 1972.

[89] T. J. Schaefer. The complexity of satisfiability problems. In *STOC 1978 - Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226. ACM, New York, N.Y., 1978.

[90] Y. Shiloach. A polynomial solution to the undirected two paths problem. *Journal of the Association for Computing Machinery*, 27(3):445–456, 1980.

[91] M. W. Siddiqee. Investigation of sorting and train formation schemes for a railroad hump yard. In G. F. Newell, editor, *Proceedings of the 5th International Symposium on the Theory of Traffic Flow and Transportation (ISTTT5)*, pages 377–387. Elsevier, New York, N.Y., 1972.

[92] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

[93] A. L. Soyster. A duality theory for convex programming with set-inclusive constraints. *Operations Research*, 22(4):892–898, 1974.

[94] R. Stephan. Cardinality constrained combinatorial optimization: Complexity and polyhedra. *Discrete Optimization*, 7(3):99–113, 2010.

[95] S. Stiller. *Extending Concepts of Reliability – Network Creation Games, Real-time Scheduling, and Robust Optimization*. PhD thesis, Technische Universität Berlin, 2008.

[96] J.-P. Watson, W. E. Hart, and R. Murray. Formulation and optimization of robust sensor placement problems for contaminant warning systems. In S. G. Buchberger, R. M. Clark, W. M. Grayman, and J. G. Uber, editors, *Proceedings of the 8th Annual Water Distribution Systems Analysis Symposium*. ASCE, Cincinnati, Ohio, 2006.

[97] R. Weismantel. On the 0/1 knapsack polytope. *Mathematical Programming*, 77(3):49–68, 1997.

[98] L. A. Wolsey. Faces for a linear inequality in 0–1 variables. *Mathematical Programming*, 8(1):165–178, 1975.

[99] R. K. Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[100] G. Yu. On the max-min 0-1 knapsack problem with robust optimization applications. *Operations Research*, 44(2):407–415, 1996.

[101] G. Yu and J. Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457–468, 1998.

A. Max-Scenario Problems

The max-scenario problem is a subproblem of several recoverable robust optimization problems, like the k-distance recoverable robust problem or the exact subset recoverable robust problem, and represents the view of a worst-case scenario in such a setting. In Gupta et al. [55] this kind of problems were named max min problems, which contradicts the name for the robust versions of maximization problems also called max min problems. They considered the special demand scenario set in which k elements may not be available in the second stage. We will focus on scenarios that define different cost functions but have no influence on the set of feasible solutions.

Definition A.0.1 (Max-Scenario Problem)**.** Let $(U, \mathcal{F}, c)$ be an LCMin problem and $\mathcal{S}$ be a set of scenarios, each scenario determining a cost function $c^S : U \to \mathbb{N}$. Then $\text{profit}(S) = \min_{F \in \mathcal{F}} c^S(F)$ defines the *profit* of a scenario $S \in \mathcal{S}$. The *max-scenario problem* of this LCMin problem is to find a scenario $S \in \mathcal{S}$ with maximum profit.

Let us consider the three scenario types. If we start with an LCMin problem in **P** and deal with discrete scenarios, then the max-scenario problem is solvable in polynomial time: Given r scenarios $S_1, \ldots, S_r$, we just need to compute the profit for each scenario S_i, $i = 1, \ldots, r$, and choose the best one. Considering interval scenarios defined by lower and upper bounds $\underline{c}$ and $\overline{c}$ on the cost values, the scenario $S_{\max}$ with $c^{S_{\max}} = \overline{c}$ is an optimal solution to any max-scenario problem.

It remains to investigate the set of Γ-scenarios. Recall that this set is described implicitly by lower and upper cost bounds $\underline{c}$ and $\overline{c}$ and a parameter $\Gamma \in \mathbb{N}$. The cost function of each feasible scenario obeys these cost bounds, and at most Γ values may vary from the lower bound. From a combinatorial perspective the max-scenario problem with Γ-scenarios becomes more interesting, since no obvious optimal solution exists. In the following part we prove that the max-scenario versions of the shortest path problem and the minimum (s,t)-cut problem are strongly **NP**-complete. Finally, we consider the max-scenario weighted disjoint hitting set problem and show that it can be solved efficiently.

A.1. Shortest Path Problem

Let $G = (V, A)$ be a directed graph, s, t be two vertices in V and $c : A \to \mathbb{N}$ be a cost function. The shortest path problem is to find a simple (s,t)-path with minimum cost. In the following theorem, we show that the max-scenario version of this problem with Γ-scenarios is strongly **NP**-hard. The reduction is from exactly-one-in-three 3SAT. It is similar to the **NP**-hardness proof for the discrete time-cost trade-off problem with negative processing times and the goal to maximize the makespan [36].

Theorem A.1.1. *The max-scenario shortest path problem with Γ-scenarios is strongly* **NP***-hard.*

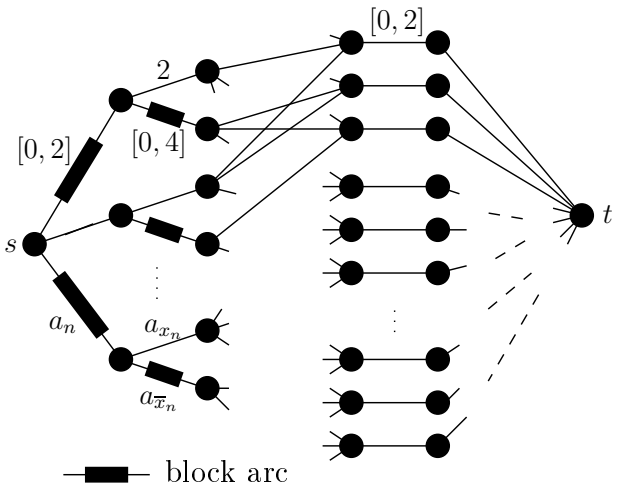


Figure A.1.: The arcs a_n, a_{x_n} and $a_{\overline{x}_n}$ form the fork G_{x_n}. For every clause C_j, $j = 1, \ldots, m$, there exist three clause-arcs a_{j1}, a_{j2} and a_{j3}. Note that all arcs are directed from right to left.

Proof. In order to prove **NP**-hardness of the max-scenario shortest path problem, we show a reduction from the **NP**-hard exactly-one-in-three 3SAT problem [89]. Let I be an exactly-one-in-three 3SAT instance with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$. Each clause C_j consists of three literals $y_{j1}, y_{i2}, y_{j3} \in \{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$, i.e.,

$$C_j = y_{j1} \vee y_{j2} \vee y_{j3}.$$

W.l.o.g. x_i and $\overline{x}_i$ are each contained in a least one clause and no clause contains both of them, for each $i = 1, \ldots, n$. A feasible assignment of I is a vector $x \in \{\text{true}, \text{false}\}^n$, such that exactly one literal in every clause is satisfied. In the following, we construct a max-scenario instance I' based on a shortest path problem defined by a graph G and two vertices s and t, and a Γ-scenario set $\mathcal{S}_\Gamma$ determined by lower and upper cost bounds on the arcs in G and a value $\Gamma \in \mathbb{N}$. We start with a description of the graph. For each variable x_i, $i = 1, \ldots, n$, the graph G contains a fork G_{x_i} with $s_i = s$, the origin vertex in G. A *fork* is a graph G_{x_i} defined by four vertices s_i, y_i, v_{x_i} and $v_{\overline{x}_i}$ and three arcs $a_i, a_{x_i}, a_{\overline{x}_i}$ with $a_i = (s_i, y_i)$, $a_{x_i} = (y_i, v_{x_i})$ and $a_{\overline{x}_i} = (y_i, v_{\overline{x}_i})$. The arcs a_i and $a_{\overline{x}_i}$ are block arcs. A *block arc* (v, w) is an arc representing M parallel (v, w) arcs each having the same properties, e.g., the same lower and upper cost bounds. We call a_i the *handle* of a fork, a_{x_i} the *true arm* of a fork and $a_{\overline{x}_i}$ the *false arm* of a fork (see Figure A.1).

Furthermore, G contains three parallel arcs a_{j1}, a_{j2} and a_{j3} for each clause C_j, $j = 1, \ldots, m$. Each arc represents a true assignment for C_j, where for a_{ji} the ith literal is true, $i = 1, 2, 3$. We call these arcs the *clause arcs*. Each clause arc is connected with t, the destination vertex in G. We finish the construction of G by defining the arcs between the fork arms and clause arcs. Let a_{ji}, $i = 1, 2, 3$, be the clause arcs of the clause $C_j = y_{j1} \vee y_{j2} \vee y_{j3}$, $j = 1, \ldots, m$. For $\ell \in \{1, 2, 3\}$, $\ell \neq i$ and $y_{j\ell} = \overline{x}_k$, $k \in \{1, \ldots, n\}$, we connect the true arm of the fork G_{x_k} with a_{ji}, and if $y_{j\ell} = x_k$, we connect the false arm of G_{x_k} with a_{ji}. In the other two cases, i.e., for $\ell \in \{1, 2, 3\}$, $\ell = i$ and $y_{ji} = x_k$ and $y_{ji} = \overline{x}_k$, we add an arc between a_{ji} and the true as well as the false arm of G_{x_k}. See Figure A.2 for an example.

We continue with the upper and the lower cost bounds in G. The handles, the true arms and the clause arcs get upper cost bounds of 2 and the false arms obtain upper cost bounds of 4. Furthermore, the lower cost bounds of the true arms are set to 2, i.e., the cost of these arcs are fixed in every scenario. Every other cost bound is 0 (see Figure A.1). Finally, for the

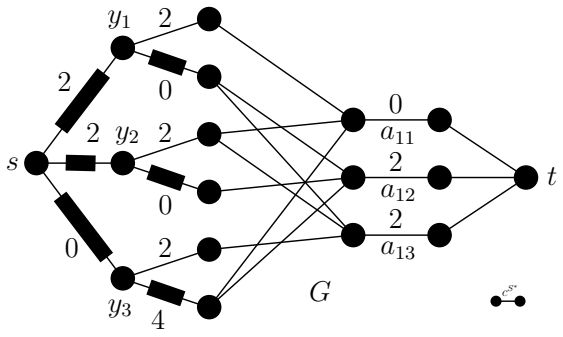


Figure A.2.: This graph G is constructed for the instance I with $C_1 = x_1 \vee \overline{x}_2 \vee x_3$. The scenario S^*, defined according to the feasible assignment $x^* = (\text{true}, \text{true}, \text{false})$, has $\text{profit}(S^*) = 4$. In C_1 the first variable x_1 verifies the clause. Therefore, the cost of a_{11} is not raised.

definition of the scenario set $\mathcal{S}_\Gamma$, we set $M = 2m + 1$ and $\Gamma = Mn + 2m$. Note that the size of the constructed max-scenario instance I' is polynomial in the size of I.

We will prove that there exists a Γ-scenario S^* with $\text{profit}(S^*) = 4$ in I' if and only if I is a yes-instance of exactly-one-in-three 3SAT.

($\Leftarrow$): Let x^* be a feasible assignment of I. We define the cost function of S^* in the following way: If $x_i^* = \text{true}$, $i \in 1, \ldots, n$, then S^* assigns upper cost to the handle of G_{x_i} and lower cost to the false arm. If $x_i^* = \text{false}$, $i = 1, \ldots, n$, the false arm gets the upper cost and the handle the lower cost. Note that any (s,t)-path already has a length of 2 due to this cost assignments. It remains to set the scenario cost for the clause arcs. Since x^* is a feasible assignment, in every clause C_j, $j = 1, \ldots, m$, there exists exactly one literal y_{ji_j}, $i_j \in \{1,2,3\}$, which verifies this clause. The scenario S^* assigns the upper cost bounds to all clause arcs a_{ji} with $i \neq i_j$ and leaves the cost of a_{ji_j} at the lower bound (see Figure A.2). In total S^* changes the cost values of n block arcs and $2m$ clause arcs from the lower cost bounds to the upper bound, i.e., the cost values of $Mn + 2m$ arcs. Therefore, S^* is a feasible Γ-scenario. It remains to show that any (s,t)-path in G has cost 4 w.r.t. the cost function c^{S^*}.

Let us assume that there exists a path p with $c^{S^*}(p) = 2$. This path crosses one true or false arm b of a variable x_ℓ, $\ell \in \{1, \ldots, n\}$, and one clause arc a_{ji}, $j \in \{1, \ldots, m\}$, $i \in \{1,2,3\}$. Due to the construction of c^{S^*}, the cost of the subpath of p starting in s and ending after traversing b are already 2. Hence, $i = i_j$ and $c^{S^*}(a_{ji}) = 0$. We now distinguish two cases: the assignment x_ℓ^* verifies C_j, i.e., $y_{ji_j} \in \{x_\ell, \overline{x}_\ell\}$, or the assignment falsifies C_j. In the first case and $y_{ji_j} = x_\ell$, p traverses the true arm. But since $x_\ell^* = \text{true}$, the cost assigned to the handle of G_{x_ℓ} are set to 2. Hence, p collects cost 4. The same argument works if $y_{ji_j} = \overline{x}_\ell$. In the other case and $\overline{x}_\ell \in C_j$, b is the true arm and $x_\ell^* = \text{true}$. As before, crossing the handle produces cost 2 and thus $c^{S^*}(p) \geq 4$. If $x_\ell \in C_j$, then b is the false arm and since $x_\ell^* = \text{false}$, the path p gets a cost of at least 4. In summary, all paths traversing the arc a_{ji_j} have already a cost 4 before they pass the clause arc. This is a contradiction.

($\Rightarrow$): Let S^* be a Γ-scenario in I' with $\text{profit}(S^*) = 4$. Before we start with a construction of x^*, we need some observations.

Claim. The scenario S^* assigns upper cost bounds to exactly one block arc in every fork.

Proof. Assume that there is a fork G_{x_ℓ}, $\ell \in 1, \ldots, n$, in which the cost in no block arc are all assigned to $\overline{c}$. Then in the handle block arc and in the false arm block arc there exists an arc with cost 0. An (s,t)-path traversing these two arcs has at most cost 2. This is a contradiction to $\text{profit}(S^*) = 4$.

Since $2m < M$, at most n block arcs can obtain cost values on the upper cost bound. $\triangle$

Claim. Exactly two cost values of the clause arcs of each clause are moved to their upper cost bounds.

Proof. We assume that there exists a clause C_j, $j \in 1, \ldots, m$, in which only one clause arc is changed to the upper cost. Each one of the three clause arcs a_{j1}, a_{j2} and a_{j3} is connected to the same forks G_{x_a}, G_{x_b} and G_{x_c}, $a, b, c \in \{1, \ldots, n\}$. Since in every fork one of the block arcs has been assigned to the upper cost, either a shortest path to the end of the true arm or a shortest path to the end of the false arm has cost 4. The other one has cost 2. W.l.o.g. let a_{j1} be the one clause in which the cost have been moved to the upper cost bound. Since the shortest path from s to t has a cost 4 and the other two clause arcs a_{j2} and a_{j3} have cost 0, both must be connected to the three arms with the higher cost (see Figure A.3). This is a contradiction to the construction of G.

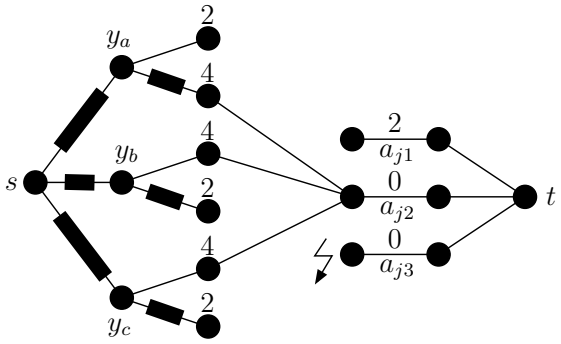


Figure A.3.: If a scenario S moves just one of three clause arcs, then there exists an (s,t)-path in G of length 2.

Since S^* already changed $n \cdot M$ arc cost, there are just $2m$ possibilities left; two for every clause. △

Now we define a solution x^* to the scenario S^* by setting for $i = 1, \ldots, n$ the assignment $x_i^* =$ true if $c^S(a_i) = 2$ and $x_i^* =$ false otherwise. For every clause C_j, $j = 1, \ldots, m$, there is one clause arc a_{ji_j} with cost 0. W.l.o.g. $i_j = 1$. If $y_{j1} = x_\ell$, $\ell \in \{1, \ldots, n\}$, then a_{j1} is connected to the true arm of G_{x_ℓ}. Every path crossing this arm produces cost 4. Therefore, the cost of the handle arc a_ℓ is at the upper bound and hence $x_\ell^* =$ true. The same argument works for $y_{j1} = \overline{x}_\ell$. Furthermore, for $y_{ji} = x_t$ or $y_{ji} = \overline{x}_t$ with $t \in \{1, \ldots, n\}$, $t \neq \ell$, the two variables are set such that they do not satisfy the clause. Hence, x^* is a feasible solution.

This completes the proof for the **NP**-hardness of the max-scenario problem. □

The graph G constructed in the reduction has few structural properties, e.g., is neither series-parallel nor planar nor a grid graph. An open question is whether the max-scenario version of the shortest path problem with Γ-scenarios can be solved efficiently on these special graph classes.

A.2. Minimum (s,t)-Cut Problem

Let $G = (V, A)$ be a directed graph, $s, t \in V$ be two vertices and $c : A \to \mathbb{N}$ be a cost function. A (s,t)-*cut* $\delta^+(X)$ with $X \subsetneq V$, $s \in X$ and $t \notin X$ consists of all arcs (a, b) with $a \in X$ and $b \in V \backslash X$. Furthermore, we assume all (s,t)-cuts to be inclusion minimal, i.e., for a given (s,t)-cut $\delta^+(X)$ there exists no subset $\delta^+(X') \subsetneq \delta^+(X)$, such that $\delta^+(X')$ is an (s,t)-cut for

some $X' \subseteq V$. Then, the minimum (s,t)-cut problem is to find an (s,t)-cut $\delta^+(X)$ with minimum cost $c(\delta^+(X)) = \sum_{a \in \delta^+(X)} c(a)$.

Theorem A.2.1. *The max-scenario minimum (s,t)-cut problem with Γ-scenarios is strongly* **NP***-hard, even on bipartite graphs.*

Proof. We show a reduction from 3SAT. Let I be an instance of 3SAT with n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$, where each clause consists of three literals. We construct a max-scenario instance I' for the minimum (s,t)-cut problem in the following way: The graph $G = (V, A)$ contains for every variable x_i, $i = 1, \ldots, n$, the vertices v_i, $\overline{v}_i$ and w_i, and for every clause C_j, $j = 1, \ldots, m$, a vertex u_j, and finally two vertices s and t. The vertex v_i represents the true-assignment of x_i and $\overline{v}_i$ the false-assignment, $i = 1, \ldots, n$. The vertices w_i, $i = 1, \ldots, n$, are auxiliary vertices. The vertex s is connected to all vertices v_i and $\overline{v}_i$, $i = 1, \ldots, n$, and each of these arcs has a lower cost bound 0 and an upper cost bound $m + 1$. All vertices u_j, $j = 1, \ldots, m$, and w_i, $i = 1, \ldots, n$, are connected to t and have a lower and upper cost bound 1. Furthermore, there is an arc connecting v_i and u_j if and only if $x_i \in C_j$, and an arc connecting $\overline{v}_i$ and u_j if and only if $\overline{x}_i \in C_j$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. These arcs also get lower and upper cost 1. Finally, we link the vertices v_i, $\overline{v}_i$ with w_i and assign upper and lower cost bound 1 (see Figure A.4). The size of this instance I' is clearly polynomial in the size of I.

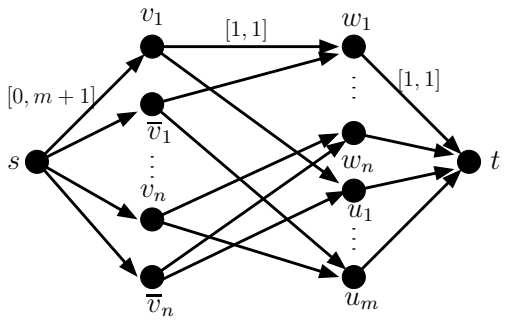


Figure A.4.: The vertices v_i, $\overline{v}_i$ represent the true and false assignment of the variable x_i, $i = 1, \ldots, n$, and the vertices $u_1, \ldots, u_m$ the different clauses $C_1, \ldots, C_m$. All arcs have a fixed cost 1 except for the arcs incident to s, which have lower and upper cost 0 and $m + 1$, respectively.

We will prove that the instance I is a yes-instance if and only if there exists a Γ-scenario S with $\Gamma = n$ and $\text{profit}(S) \geq m + n$.

($\Rightarrow$): Let x^* be an assignment satisfying all clauses. We define

$$A^* = \cup_{i=1}^{n} \{(s, v_i) \mid x_i^* = \text{true}\} \cup_{i=1}^{n} \{(s, \overline{v}_i) \mid x_i^* = \text{false}\}.$$

The scenario S^* assigns cost of the upper bound to all arcs $a \in A^*$. Since $|A^*| = n$, the scenario S^* is feasible. We consider the graph with these cost as capacities and compute a maximum (s,t)-flow. It is well-known that the value of a maximum (s,t)-flow is equivalent to the minimum cost of an (s,t)-cut. We will show that for every arc (u_j, t), $j = 1, \ldots, m$, there exists an (s, u_j)-path in the graph which crosses just arcs with cost greater than 0.

Let C_j be the clause corresponding to the vertex u_j. Since x^* is a valid assignment, there exists a variable in C_j which satisfies this clause. Let x_i^* be that variable with $x_i^* = \text{true}$, $i \in \{1, \ldots, n\}$. Due to the definition of A^*, the arc (s, v_i) has an upper cost $m + 1$ and v_i is

connected to u_j. Hence, this forms a path from s to u_j. The same holds for the path $s\overline{v}_i u_j$ if $x_i^* = \text{false}$. The existence of such an (s, w_i)-path using just arcs with positive cost can also be shown for every (w_i, t)-arc, $i = 1, \ldots, n$. Choosing such a path for every (u_j, t), $j = 1, \ldots m$, and (w_i, t), $i = 1, \ldots, n$, and prolonging it with the arc (u_j, t) or (w_i, t), respectively, the flow sending one unit along each of these paths is feasible: there are at most $m + 1$ paths containing an arc (s, v_i) or $(s, \overline{v}_i)$ and at most one path containing any other arc. The value of this (s, t)-flow is $m + n$. Since $\text{profit}(\delta^+(X)) = m + n$ for $X = V \backslash \{t\}$, the flow is maximum and thus $\text{profit}(S^*) = m + n$.

($\Leftarrow$): Let S^* be a scenario changing the cost of the arcs A^* with $\text{profit}(S^*) \geq m + n$. Hence, there exists a flow sending one unit across every arc (u_j, t) and $(\overline{u}_i, t)$, $i = 1, \ldots, n$, $j = 1, \ldots, m$. Since (w_i, t) is just adjacent to (v_i, w_i) and $(\overline{v}_i, w_i)$, the scenario changes either the cost of the arc (s, v_i) or $(s, \overline{v}_i)$. Since there are at most n arcs with cost $m + 1$, the assignment $x_i^* = \text{true}$ if $(s, v_i) \in A^*$, and $x_i^* = \text{false}$ if $(s, \overline{v}_i) \in A^*$, is well defined, $i = 1, \ldots, n$. Let us assume that there exists a clause C_j, $j \in \{1, \ldots, m\}$, which is not satisfied by x^*. It follows from the considerations above that there exists no path from s to u_j using arcs with positive cost. This is a contradiction to $\text{profit}(S^*) \geq m + n$. □

Note that it is essential for the reduction that G is a directed graph.

A.3. Weighted Disjoint Hitting Set Problem

The weighted disjoint hitting set problem is a special case of the well-known weighted hitting set problem: Let $U = \{u_1, \ldots, u_n\}$ be a set of n elements, $\mathcal{M} = \{M_1, \ldots, M_d\}$ be a set of d pairwise disjoint subsets of U, i.e., $M_i \subseteq U$ and $M_i \cap M_j = \emptyset$ for all $i \neq j$, $i, j \in \{1, \ldots, d\}$, and $c : U \to \mathbb{N}$ be a cost function. A feasible solution $F \subseteq U$ of the *weighted disjoint hitting set problem (WDHS)* contains exactly one element for every set $M \in \mathcal{M}$, i.e., $|F \cap M| = 1$ for all $M \in \mathcal{M}$. Then the weighted disjoint hitting set problem is the following:

Given: A set of n elements $U = \{u_1, \ldots, u_n\}$, a set of d pairwise disjoint subsets $\mathcal{M} = \{M_1, \ldots, M_d\}$ and a cost function $c : U \to \mathbb{N}$.

Task: Find a feasible solution $F \in \mathcal{F}$ minimizing the cost $c(F) = \sum_{u \in F} c(u)$.

This problem has a quite simple structure, and a solution choosing the element with minimum cost from every subset is optimal. Yet, its recoverable robust version turns out to be the key problem showing **NP**-hardness for several recoverable robust LCMin problems. For the sake of completeness (and to prove that not all max-scenario problems are **NP**-complete), we show how this problem can be solved in polynomial time.

Theorem A.3.1. *The max-scenario WDHS problem is solvable in polynomial time.*

Proof. Let U be a set of n elements, $M_1, \ldots, M_d$ be d pairwise disjoint subsets of U, $\underline{c}(u)$ and $\overline{c}(u)$ be lower and upper bounds on all elements $u \in U$ and $\Gamma \in \mathbb{N}$. The max-scenario WDHS problem is to find a scenario $S \in \mathcal{S}_\Gamma$, i.e., a scenario S with $c^S : U \to \mathbb{N}$, $c^S(u) \in [\underline{c}(u), \overline{c}(u)]$ for all $u \in U$ and $|\{u \in U \mid c^S(u) > \underline{c}(u)\}| \leq \Gamma$, with maximum profit. We start by ordering the items of M_i, $i = 1, \ldots, d$, such that $\underline{c}(u_{ik}) \leq \underline{c}(u_{i\ell})$ for $k < \ell$ and $\overline{c}(u_{ik}) \leq \overline{c}(u_{i\ell})$ if $\underline{c}(u_{ik}) = \underline{c}(u_{i\ell})$ for $k < \ell$. Let S^* be an optimal solution moving $\gamma_i(S^*)$ values of the set M_i to the upper cost bound, $i = 1, \ldots, d$. We define

$$c_i^{\gamma_i(S^*)} = \min\{\min_{j=1,\ldots,\gamma_i(S^*)} \overline{c}(u_{ij}),\ \underline{c}(u_{i(\gamma_i(S^*)+1)})\}.$$

Since $\underline{c}(u_{ij}) \leq \underline{c}(u_{i(\gamma_i(S^*)+1)})$ and $\underline{c}(u_{ij}) \leq \overline{c}(u_{ij})$ for any $j \leq \gamma_i(S^*)$,

$$\text{profit}(S^*) = \sum_{i=1}^{d} c_i^{\gamma_i(S^*)}.$$

Thus, it suffices to determine the values $\gamma_i(S^*)$, $i = 1, \ldots, d$, instead of the set of elements, which obtain the upper bound as cost. In the remaining part we assume that any scenario S changing $\gamma_i(S)$ values in M_i, $i = 1, \ldots, d$, chooses the set $\{u_{i1}, \ldots, u_{i\gamma_i(S)}\}$.

We now show how to compute the values $\gamma_i(S^*)$, $i = 1, \ldots, d$, of an optimal Γ-scenario S^* via a constraint longest path problem. First we transform the sets M_i, $i = 1, \ldots, d$, into a graph G. The graph G is an extension of a simple (s,t)-path of length d where the ith arc is replaced by $|M_i|$ parallel arcs, $i = 1, \ldots, d$. Each arc a_{ij} gets two cost values $(w(a_{ij}), \ell(a_{ij})) = (c_i^j, j)$, $i = 1, \ldots, d$, $j = 1, \ldots, |M_i|$. Let $p = a_{1j_1} a_{2j_2} \ldots a_{dj_d}$ be an (s,t)-path in G with $\ell(p) \leq \Gamma$. Then p represents a Γ-scenario S_p, where S_p changes exactly j_i elements to their upper bound in each set M_i, $i = 1, \ldots, d$, and

$$\text{profit}(S_p) = \sum_{i=1}^{d} c_i^{j_i} = w(p).$$

On the other hand, any scenario $S \in \mathcal{S}_\Gamma$ represents an (s,t)-path in G. Thus, a longest (s,t)-path p with $\ell(p) \leq \Gamma$ generates a Γ-scenario with maximum profit. Since the chain graph is acyclic and the values of the second cost function ℓ are polynomially bounded by the input, such a constraint longest path can be computed in polynomial time via a dynamic program introduced by Joksch [64] or the labeling Dijkstra algorithm analyzes by Aneja et al. [6]. □

For several classic combinatorial optimization problems, as the minimum perfect matching problem or the minimum spanning tree problem, the complexity status of their max-scenario version remains open.

B. Cardinality Constrained Minimum (s,t)-Cut Problem

Cardinality constrained combinatorial optimization problems ask for a feasible solution with minimal cost containing either exactly k, nor more than k, or less than k elements, $k \in \mathbb{N}$. These problems have widely be studied with respect to their complexity or their polyhedral structure [20, 94]. Yet, to the best of our knowledge, the complexity status for the cardinality constrained minimum (s,t)-cut problem, where the number of arcs is upper bounded by k, remained an open problem. The constrained versions where the cardinality is fixed to k or is lower bounded by k are known to be strongly **NP**-complete, shown via a reduction from max cut [21]. In the following we prove that this result also holds for the upper bounded case. This is joint work with Rico Zenklusen.

Definition B.0.2 (Upper Bounded Cardinality Constrained Minimum (s,t)-Cut Problem)**.** Let $G = (V, A)$ be a directed graph, s, t be two designated vertices in V, $c : A \to \mathbb{N}$ be a cost function on the arc set A and $k \in \mathbb{N}$ be an upper bound on the cost. An (s,t)-cut is a subset of arcs $\delta^+(X)$ with $X \subseteq V \backslash \{t\}$, $s \in X$, $\delta^+(X) := \{(u,v) \in A \mid u \in X, v \in V \backslash X\}$. The *upper bounded constrained minimum (s,t)-cut problem* is to find an (s,t)-cut $\delta^+(X)$ with minimal cost $c(\delta^+(X))$ such that $|\delta^+(X)| \leq k$.

We assume that all (s,t)-cuts are inclusion minimal, i.e., no subset $\delta^+(X') \subsetneq \delta^+(X)$ forms a feasible (s,t)-cut $\delta^+(X')$ for some $X' \subseteq V$.

Theorem B.0.3. *The upper bounded cardinality constrained (s,t)-cut problem is strongly* **NP***-hard.*

Proof. In order to prove **NP**-hardness we show a reduction from the network interdiction problem with unit removal cost, which is strongly **NP**-hard [99]. Let $G = (V, A)$ be a directed graph, s and t be two designated nodes, $u : A \to \mathbb{N}$ be a capacity function, and B be a budget on the removal cost. The task in the network interdiction problem is to decide for a given parameter $K \in \mathbb{N}$ whether there exists a set of arcs $D \subseteq A$ with $|D| \leq B$ such that the value of a maximum (s,t)-flow in the graph $(V, A \backslash D)$ is smaller than or equal to K. By the Max-Flow-Min-Cut Theorem of Ford and Fulkerson, this is equivalent to finding a set of arcs $D \subseteq A$ with $|D| \leq B$ such that the value of a minimum (s,t)-cut $\delta^+(X)$ in the graph $(V, A \backslash D)$ is smaller than or equal to K. We assume that $\delta^+(X) \cup D$ forms an (s,t)-cut in G. If this is not the case, we can choose a subset D' of D such that $\delta^+(X) \cup D'$ is an (s,t)-cut.

To derive an upper bounded cardinality constrained minimum (s,t)-cut instance I', we modify the graph $G = (V, A)$ and define a cost function c on the arcs in the following way: We subdivide each arc $a = (u,v) \in A$ by an extra vertex v_a and add M parallel (v_a, v) arcs to the set of arcs of the new graph $G' = (V', A')$, $M \in \mathbb{N}$. All these $(M+1)$ parallel arcs obtain cost 0, and the (u, v_a) arc obtains the original capacity as cost, i.e., $c((u, v_a)) = u(a)$ for all $a \in A$. Finally, we set $M = |A|$ and the upper bound on the cardinality $k = B \cdot (M+1) + |A|$. Let $\delta^+(X')$ be an (s,t)-cut in G' with $|c(\delta^+(X'))| \leq K$ and $|\delta^+(X')| \leq k$. Since this cut

contains $(M+1)$ different arcs, for every vertex v_a, $v_a \in V$ and $a \in A$, at most B of these vertices are included in X'. Hence, the set $D = \{a \in A \mid v_a \in X'\}$ contains at most B arcs. Furthermore, the set $\delta^+(X)$ with $X = \{v \in V \mid v \in X'\}$ is an (s,t)-cut in the graph $(V, A \backslash D)$ with $u(\delta^+(X)) = c(\delta^+(X')) \leq K$. Thus, D is a feasible solution of the network interdiction instance.

On the other hand, let D be a subset of A with $|D| \leq B$, such that the (s,t)-cut $\delta^+(X)$ in $(V, A \backslash D)$ produces cost of less than or equal to K, and $D \cup \delta^+(X)$ forms an (s,t)-cut in G. Then $X' = X \cup \{v_a \mid a \in D\}$ is a feasible solution of I' with $c(\delta^+(X')) = u(\delta^+(X))$. Thus, the network interdiction instance is a yes-instance if and only if the constructed upper bounded cardinality constrained minimum (s,t)-cut instance is a yes-instance. □

The upper bounded constrained minimum (s,t)-cut problem is a subproblem in one of the recoverable robust settings introduced in Section 2. The **NP**-completeness of this problem implies that the total cost of a given (s,t)-cut cannot be computed efficiently, unless $\mathbf{P} = \mathbf{NP}$.

Zusammenfassung

Recoverable Robustness ist eine Methode um mit Unsicherheiten in Optimierungsproblemen umzugehen und erweitert das Konzept der robusten Optimierung. Im Gegensatz zur klassischen Robustness darf eine gewählte Lösung durch die Anwendung limitierter Maßnahmen verändert werden. Eine so veränderte Lösung wird auch als Recovery bezeichnet. Gesucht ist eine Lösung, die für jeden eintretenden Fall der Unsicherheitsparameter eine zulässige Lösung im Recovery enthält und die maximal auftretenden Kosten minimiert. Dieses Konzept wurde bisher hauptsächlich auf Probleme aus der Verkehrsoptimierung angewandt, so zum Beispiel auf das Timetabling und das Platforming.

Die vorliegende Arbeit besteht aus zwei Teilen: Der erste Teil beschäftigt sich in drei Kapiteln mit unterschiedlichen Modellen zur Recoverable Robustness für kombinatorische Optimierungsprobleme und der zweite mit der Anwendung dieses Konzepts auf Praxisprobleme.

Die drei theoretischen Modelle unterscheiden sich in der Art der Recovery und in ihren Zielfunktionen. Im ersten Modell, der k-Distance Recoverable Robustness, ist die Recovery auf Lösungen beschränkt, die maximal k neue Elemente beinhalten. Im zweiten Fall, der Rent Recoverable Robustness, ergibt sich durch die Wahl einer ersten Lösung keine Einschränkung an die Recovery. Die Kosten für eine aus der Recovery gewählten Lösung variieren jedoch mit dieser Wahl. Im letzten Modell soll eine kardinalitätsminimale Menge gesucht werden, die für jedes Szenario eine optimale Lösung enthält. Das Hauptaugenmerk in diesem Teil der Arbeit liegt auf der Untersuchung des Komplexitätsstatus und der kombinatorischen Eigenschaften der Modelle in Abhängigkeit des gegebenen Optimierungsproblems und der betrachteten Unsicherheiten.

Im zweiten Teil der Arbeit wird das Knapsack Problem als Spezialfall des Bandwidth Packings aus der Telekommunikation sowie das Train Classification Problem betrachtet. Neben der Modellierung und einer erneuten Analyse der kombinatorischen Eigenschaften, werden im Anschluss die Auswirkungen des Ansatzes der Recoverable Robustness im Vergleich zur reinen Robustness in mehreren auf Praxisdaten basierenden Experimenten ausgewertet.