Algorithms for Robust Combinatorial Optimization with Budgeted Uncertainty and Fair Planning of the Out-of-Hours Service for Pharmacies

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

vorgelegt von

Timo Gersing, M. Sc.

aus Bergisch Gladbach

Berichtende: Univ.-Prof. Dr. rer. nat. Christina Büsing
Univ.-Prof. Dr. Ir. Arie M.C.A. Koster
Univ.-Prof. Dipl.-Ing. Dr. techn. Ulrich Pferschy

Tag der mündlichen Prüfung: 17. Mai 2024

Diese Dissertation ist auf den Internetseiten der Universitätsbibliothek verfügbar.

Timo Gersing
Algorithms for Robust Combinatorial Optimization with Budgeted Uncertainty and Fair Planning of the Out-of-Hours Service for Pharmacies
May 23, 2024

✉ gersing@combi.rwth-aachen.de

Abstract

Practical problems usually require robust or fair solutions. However, optimal solutions for classical optimization problems tend to structurally neglect these two criteria. This is due to the promotion of extreme decisions that exhaust the planning constraints and most profitable options as far as possible. Robustness, though, is usually achieved by spreading risks, and fairness by distributing benefits and burdens, which requires diverse solutions rather than extreme ones. Accordingly, both criteria often have to be explicitly integrated into optimization problems. This poses a challenge both from a modeling and an algorithmic point of view. In this thesis, we want to contribute to overcoming this challenge by studying the following problems.

In the first part of this thesis, we consider robust optimization with budgeted uncertainty, which is one of the most popular approaches for addressing uncertainty in optimization problems. Positive complexity results as well as the existence of a compact reformulation for (mixed-integer) linear programs suggest that these problems are easy to solve. However, the reformulation as well as the algorithms that provide these complexity results do not perform well when solving robust combinatorial problems in practice. To address this, we propose a new class of valid inequalities to strengthen the reformulation. These inequalities are facet-defining in many cases, and are thus among the theoretically strongest inequalities. Furthermore, we develop a branch-and-bound algorithm based on new formulations and structural results. We show in two extensive computational studies that both approaches facilitate the computation of optimal robust solutions. Especially the branch and bound algorithm outperforms all previous approaches by far.

In the second part, we consider the problem of planning the out-of-hours service for pharmacies, which ensures a continuous supply of pharmaceuticals. The problem consists in assigning 24-hour shifts to a subset of pharmacies on each day such that an appropriate supply is guaranteed while the burden on pharmacists is minimized. We present a model for the planning, developed in collaboration with the Chamber of Pharmacists North Rhine, and show that computing a feasible plan is $\mathcal{NP}$-hard. We develop algorithms that nevertheless compute almost optimal plans for the North Rhine area in short time. The computed plans assign fewer shifts in total compared to the real plan of the Chamber of Pharmacists North Rhine, but they exhibit an unfair concentration of shifts. Consequently, we discuss strategies to integrate fairness into the planning. We show theoretical results on fairness in optimization problems, on the basis of which we compute out-of-hours plans that are almost maximally fair.

Zusammenfassung

Praktische Probleme erfordern meist robuste oder faire Lösungen. Optimale Lösungen für klassische Optimierungsprobleme neigen allerdings dazu diese beiden Kriterien strukturell zu vernachlässigen. Ursächlich hierfür ist die Begünstigung extremer Entscheidungen, welche die Planungsbedingungen und rentabelsten Optionen weitestmöglich ausschöpfen. Robustheit wird in der Regel jedoch durch eine Streuung von Risiken sowie Fairness durch eine Verteilung von Nutzen und Lasten erreicht, was eher diverse Lösung anstelle extremer erfordert. Beide Kriterien müssen entsprechend oftmals explizit in Optimierungsprobleme integriert werden. Dies stellt sowohl aus Sicht der Modellierung als auch der Algorithmik eine Herausforderung dar. In dieser Arbeit wollen wir mit dem Studium der folgenden Probleme einen Teil zur Bewältigung dieser Herausforderung beitragen.

Im ersten Teil dieser Arbeit betrachten wir robuste Optimierung mit budgetierter Unsicherheit, einen der populärsten Ansätze zur Berücksichtigung von Unsicherheiten in Optimierungsproblemen. Positive Komplexitätsresultate sowie die Existenz einer kompakten Reformulierung für (gemischt ganzzahlige) lineare Programme suggerieren, dass diese Probleme leicht zu lösen sind. Allerdings zeigen die Reformulierung und die Algorithmen, welche diese Komplexitätsergebnisse liefern, in der Praxis keine gute Performanz beim Lösen robuster kombinatorischer Probleme. Um dem entgegenzuwirken, präsentieren wir eine neue Klasse gültiger Ungleichungen zur Stärkung der Reformulierung. Diese Ungleichungen definieren in vielen Fällen Facetten und gehören somit zu den theoretisch stärksten Ungleichungen. Zudem entwickeln wir einen Branch-and-Bound Algorithmus auf Basis neuer Formulierungen und struktureller Ergebnisse. In zwei umfassenden Rechenstudien zeigen wir, dass beide Ansätze die Berechnung optimaler robuster Lösungen erleichtern. Insbesondere der Branch-and-Bound Algorithmus übertrifft alle bisherigen Ansätze deutlich.

Im zweiten Teil betrachten wir das Problem der Planung des Apothekennotdienstes, welcher eine durchgängige Versorgung mit Arzneimitteln sicherstellt. Das Problem besteht darin, täglich einer Teilmenge der Apotheken 24-Stunden Dienste zuzuweisen, sodass eine angemessene Versorgung gewährleistet und gleichzeitig die Belastung der Apotheken minimiert wird. Wir stellen ein Planungsmodell vor, das in Zusammenarbeit mit der Apothekerkammer Nordrhein entwickelt wurde und zeigen, dass die Berechnung eines zulässigen Plans $\mathcal{NP}$-schwer ist. Wir entwickeln Algorithmen, die dennoch in kurzer Zeit nahezu optimale Pläne für das Gebiet Nordrhein berechnen. Die berechneten Pläne verteilen insgesamt weniger Dienste als der reale Plan der Apothekerkammer Nordrhein, weisen jedoch eine unfaire Konzentration von Diensten auf. Daher diskutieren wir Strategien zur Integration von Fairness in die Planung. Wir zeigen theoretische Resultate zu Fairness in Optimierungsproblemen, auf deren Grundlage wir Notdienstpläne berechnen, die nahezu maximal fair sind.

Acknowledgement

I consider myself fortunate to have been accompanied and supported by many wonderful people on my way to completing this thesis. I am especially grateful to have received guidance from two exceptional supervisors, Christina Büsing and Arie Koster. Both provided me with all the opportunities I could have wished for: They enabled me to attend numerous conferences, go on research trips, and, most importantly, gave me the chance to work on topics I love. In particular, thank you, Christina, for being the optimistic person you are and for having confidence when I did not. I always knew that I could count on your support and foresight, even if times were stressful. Please continue on your path of empowering people – as you have done with me – with your interest in students not only as mathematicians but also as human beings. Also thank you, Arie, for your support and advice, as well as for encouraging me to step out of my comfort zone. These impulses frequently led to invaluable experiences that broadened my horizons and helped me grow as a researcher and person.

In addition to my supervisors, I was fortunate to collaborate with many outstanding people. Thanks to all members of the HealthFaCT team for many joyful project retreats, conferences, and lessons in table soccer. Thank you, Martin, for enriching work in general and the pharmacy planning in particular with your cheerful personality. Thank you, Sascha, for our conversations on robust optimization, which made me tinker again with this topic. And thank you, Sophia, for lending me your ears, your eyes and your good nose all the time I doubted my own thoughts.

Next to my collaborators, I would like to thank all colleagues that I had the pleasure to work with – especially those with whom I shared an office, whether physically or at heart. I am grateful for every hour of brainstorming, chatting, and social activities. All of you may feel guilty in the most positive way for making the university a second home, where I enjoyed spending way too much time.

A special thanks goes to our secretaries Angela, Dina, and Silke as well as our administrators Anto, Christoph, Frank, and Stephan – whether officially or unofficially. This work would not have been possible without your support.

I am also grateful to all my friends who enrich my life with happiness, distraction, and emotional support. I would like to particularly thank those who are partially responsible for my journey. Thank you, Finn, for making me start this quest of becoming a mathematician. Thank you, Karl and Marius, for carrying me through the first semesters at a time before I was able to stand on my own feet. Thank you, Jeff and Sabrina, for bringing joy to my studies and mathematical optimization into my life.

My sincere gratitude extends to my family. You gave and taught me everything I needed to achieve this goal. Thank you for standing by me through all the excitement and the obstacles in the last few years. This is especially to my fiancée Sarah. Thank you for your unconditional love, support, and patience. After completing the chapters below, I cannot wait to open a new chapter of our life – I love you!

Timo Gersing
Aachen, May 23, 2024

Contents

1 Introduction **1**

2 Preliminaries **7**
2.1 Basic Notation and Graphs 7
2.2 Mathematical Optimization 7
2.3 Linear Programming 8
2.4 Mixed-Integer Linear Programming 12

I Algorithms for Robust Combinatorial Optimization with Budgeted Uncertainty **19**

3 Introduction **21**
3.1 Problem Statement 22
3.2 Related Work 25
3.3 Contribution and Outline 26

4 Recycling Valid Inequalities **29**
4.1 A Strong Bilinear Formulation 29
4.2 Recycled Inequalities 32
4.3 Facet-Defining Recycled Inequalities 34
4.4 Separating Recycled Inequalities 42
4.5 Partially Recycling of Non-Recyclable Inequalities 47
4.6 Computational Study 52
4.7 Conclusion 68

5 A Branch and Bound Algorithm **69**
5.1 Strong Linear Formulations for Bounded z 69
5.2 The Basic Branch and Bound Framework 72
5.3 A Reformulation using Cliques in Conflict Graphs 73
5.4 Lagrangean Relaxations 79
5.5 Characterization of Optimal Values for p and z 81
5.6 The Branch and Bound Algorithm 87
5.7 Computational Study 107
5.8 Conclusion 126

II Fair Planning of the Out-of-Hours Service for Pharmacies 129

6 Introduction 131
6.1 Related Work . . . 132
6.2 Contribution and Outline . . . 134

7 Planning the Out-of-Hours Service for Pharmacies 137
7.1 Problem Definition and Notation . . . 137
7.2 Complexity of the OHP . . . 142
7.3 Solution Approaches for the OHP . . . 143
7.4 Case Study . . . 151
7.5 Conclusion . . . 167

8 Integrating Fairness into the Planning of the Out-of-Hours Service 169
8.1 Fairness Concepts for the Planning of the Out-of-Hours Service . . . 169
8.2 Lexicographic Fair Planning of the Out-of-Hours Service . . . 172
8.3 Min-Max Fairness . . . 174
8.4 Aggregated Planning and Water-Draining . . . 180
8.5 Bringing Fairness into Practice . . . 186
8.6 Case Study . . . 190
8.7 Conclusion . . . 208

Bibliography 211

A Computational Results for Chapter 8 217

1 Introduction

Mathematical optimization deals with the question of how to make decisions in the best possible way. For this purpose, decision-making processes are abstracted and mathematically modeled in an optimization problem using variables, constraints, and objective functions. Afterwards, algorithms are developed for computing solutions that are optimal for this model. When carried out successfully, the solutions for the abstracted model can be translated back into the real world, where they provide practical benefits. Optimal solutions can represent an increase in quality compared to solutions that are constructed by hand. For example, parcels can be delivered faster, at lower cost, and with lower emissions by optimizing the routing. In addition to increased quality, the automation of decision-making processes provides a major advantage by saving planning effort. For example, an algorithmic scheduling of workers saves the staff a significant amount of time spent on self-scheduling, allowing them to concentrate on their core tasks.

Despite the great potential of mathematical optimization, academic solutions to real-world problems are, admittedly, often not applicable in practice. This is the case when the abstracted mathematical model lacks aspects that are important in reality. In manual planning, expert decision-makers often intuitively consider soft constraints that are sometimes hard to formalize. Neglecting such constraints in the mathematical model can result in structural deficiencies of solutions. This is especially true if a neglected constraint is in conflict with the objective function. In this case, an optimal solution might accidentally maximize the violation of the constraint to an extent that manual planning would never consider.

A potentially neglected constraint, or rather goal, is the diversity of solutions, which is often hard to grasp. Optimal solutions to classical optimization problems tend to be extreme: The limits imposed by planning constraints and the most profitable options are exhausted as much as possible in order to achieve the best value for the model's objective function. For example, when planning a long-term purchasing strategy, we might focus on a single supplier that offers the cheapest price. However, single-source supply chains are vulnerable to natural and political disasters. Thus, the purchasing strategy may fail if the price of the supposedly cheapest supplier turns out to be higher than expected in the future. Another example involves the assignment of workload: If some employees work more efficiently and to a higher quality, then we may be inclined to assign as many tasks as possible to these employees. However, an uneven assignment of workload might cause dissatisfaction among employees. In both examples, the development of the purchasing strategy and the assignment of workload, it is advisable to strive for solutions that are more robust against uncertain developments and more fair towards stakeholders.

This poses two major challenges for practical mathematical optimization. First, formalizing requirements such as robustness and fairness is not a trivial task. Second, even when formalized, these additional requirements often result in optimization problems that are much more difficult to solve. Hence, there is a need for models that incorporate robustness and fairness in a tractable way as well as efficient algorithms that yield (almost) optimal solutions to these models in reasonable computation time. In this thesis, we face both of these challenges: We develop efficient algorithms for solving a class of generic robust optimization problems, and propose approaches for the practical problem of planning the out-of-hours service for pharmacies with a particular emphasis on fairness. We briefly introduce these two topics in the following.

Robust Optimization with Budgeted Uncertainty

Robust optimization is a popular approach for integrating protection against uncertainty into optimization models. While classical optimization assumes that the input parameters of a model are fixed, robust optimization considers a range of (potentially infinitely many) *scenarios*, each with different input parameters. A *robust solution* remains feasible for all considered scenarios. Furthermore, an *optimal* robust solution has the best objective value with respect to its worst-case scenario. As optimal robust solutions guarantee a certain quality among a range of different scenarios, they favor decisions that are structurally less prone to uncertainties. For example, a supply chain considering a diverse set of suppliers is likely chosen over a single-source supply chain. Therefore, optimal robust solutions tend also to be protected against scenarios that are not explicitly considered in the model, and are thus still viable even when reality does not turn out in our favor.

Different approaches for the construction of the set of scenarios have been proposed in the past, but the concept of robust optimization with *budgeted uncertainty* by Bertsimas and Sim has received particular attention. This is illustrated by the fact that their seminal paper [23] is the most cited document in the literature databases Scopus and Web of Science containing "robust optimization" in its title, keywords or abstract. This popularity is not least due to the scenarios being constructed in an intuitive way, where an *uncertainty budget* is used to control the extent to which we want to hedge against uncertainties. Bertsimas and Sim show theoretically and experimentally that the "price of robustness" is rather small, as the uncertainty budget can be chosen such that a high level of protection against not explicitly considered scenarios is achieved, while the loss in the objective value compared to non-robust solutions is rather small [23]. In addition, the possibility to formulate robust optimization problems with budgeted uncertainty in a compact mathematical model, and the existence of positive results on their theoretical complexity suggest that such problems are easily solvable.

However, despite the amount of research devoted to solving robust optimization problems with budgeted uncertainty, instances of practical size often still pose a considerable challenge, even if the corresponding non-robust problem is relatively easy to solve [67]. We will see

later that this is an inherent challenge in robust optimization, as the trend towards diverse solutions is hindering for the performance of one of the most powerful techniques used in optimization, namely (mixed) integer linear programming. Therefore, while Bertsimas and Sim show that the price of robustness on the objective function is relatively low, the computational price can be so high that the approach becomes intractable for many real-world problems. In this thesis, we aim to address this issue by developing algorithms that facilitate the computation of optimal robust solutions.

Planning the Out-of-Hours Service for Pharmacies

The out-of-hours service of pharmacies is an integral part of the German healthcare system, as it ensures continuous supply with pharmaceuticals at any time of the day. For this, every day a subset of pharmacies is assigned an *out-of-hours shift*, which obliges them to be open for 24 hours. In order to maximize supply, each pharmacy could be assigned an out-of-hours shift on each day. However, these shifts are economically unattractive and place a high workload on the pharmacists. Therefore, it is crucial to strike a balance between an appropriate supply of pharmaceuticals and an acceptable number of shifts per pharmacy when planning the out-of-hours service. This balance may become increasingly difficult to achieve in the future, as the number of pharmacies has been in constant decline in recent years [1], which results in more shifts per pharmacy or a worse supply. This development is especially problematic in rural areas, where we already observe a low density of pharmacies. We thus require sound methods that guarantee an efficient planning of the out-of-hours service.

Until today, it is common practice in Germany to divide the planning area into districts, typically based on administrative borders, in which the out-of-hours service is organized locally as a rotation of the resident pharmacies. This approach has different downsides. First, a lack of synchronization between districts can lead to neighboring pharmacies in different districts having an out-of-hours shift on the same day. This creates an oversupply for the corresponding area and unnecessary shifts. Second, we can also have an undersupply in case that the out-of-hours pharmacies of neighboring districts are located far apart from another. Third, the districting itself can negatively impact the burden of individual pharmacies, since being part of a district with few pharmacies results in many shifts. This is especially unsatisfactory if neighboring pharmacies are in different districts that differ significantly in terms of the implied burden. Justifying such differences on the basis of administrative borders alone is difficult and can lead to frustration among pharmacists who may feel treated unfairly due to their allocation to an unfavorable district. We address these issues with a centralized approach that considers all pharmacies together to obtain an efficient and fair out-of-hours plan.

Contribution and Outline

Before we turn to the main topics of this thesis, we introduce some notation and discuss selected concepts of mathematical optimization in Chapter 2.

In Part I, we consider robustness with budgeted uncertainty for generic combinatorial optimization problems. We focus on uncertainty in the objective function, but most of our results carry over to uncertainty in the constraints. In Chapter 3, we give a formal introduction into the problem and discuss why it is hard to solve in practice. In the following two chapters, we develop approaches that contribute to making the problem more tractable. In Chapter 4, we propose a new class of valid inequalities, namely *recycled inequalities*. Valid inequalities are an integral part of modern (mixed) integer linear programming solvers and can help lowering computation times significantly. We will show that recycled inequalities are often facet-defining, that is, they are among the theoretically best valid inequalities for our problem. Furthermore, they are easy to compute, and thus can yield a significant performance improvement. To demonstrate their practical use, we conduct a computational study on carefully generated robust versions of classical combinatorial problems and real-world instances from MIPLIB 2017 [49]. In Chapter 5, we prove several structural properties and introduce different formulations for robust combinatorial optimization problems with budgeted uncertainty. We combine these results in a specialized branch and bound algorithm and perform an extensive computational study, which reveals that our algorithm outperforms the current state-of-the-art approaches by far. Moreover, we show that our structural results can be used to improve most of these approaches substantially, thus highlighting the relevance of our findings for future research. All implemented algorithms [46] and generated test instances [48] are freely available online for future benchmarking in robust combinatorial optimization with budgeted uncertainty.

In Part II, we study the planning of the out-of-hours service for pharmacies. After giving an introduction in Chapter 6, we propose a planning problem in Chapter 7, which has arisen from a collaboration with the Chamber of Pharmacists North Rhine. The problem consists in minimizing the total number of shifts while guaranteeing a certain quality of coverage for residents. We show that it is hard to compute a feasible out-of-hours plan in theory. However, we also propose algorithms that are capable of computing almost optimal solutions in reasonable time for a real-world instance. The computed out-of-hours plans are more efficient than the real plans, with a reduction of roughly 10% in the total number of shifts, while simultaneously maintaining a higher compliance with planning regulations. The drawback, however, is that the distribution of shifts among pharmacies is not fair. Similar to the assignment of workload discussed above, striving for as few shifts as possible leads to a concentration of shifts on pharmacies in favorable locations that enable an efficient coverage of our planning area. We therefore focus on the computation of fair plans in Chapter 8. For this, we first consider the concept of min-max fairness, which can be seen as the strictest fairness concept, for an idealized planning problem. We generalize and prove several statements from the literature on min-max fairness, which we then use to compute

min-max fair solutions to the idealized problem within seconds. Afterwards, we use these solutions as an orientation for computing fair out-of-hours plans. We show for our real-world instance that we can compute efficient plans that almost match the idealized solution, and are thus almost maximally fair. We furthermore show that the idealized solutions are invaluable within a decision support environment for analyzing and customizing the planning model.

Acknowledgment of Funding and Material

The present work received support from various sources, for which we would like to express our gratitude.

We received funding from the German Federal Ministry of Education and Research (grant no. 05M16PAA) within the project “HealthFaCT - Health: Facility Location, Covering and Transport”. We furthermore received traveling grants from the German Academic Exchange Service (DAAD).

The maps in Part II were created using a visualization and analysis tool provided by our partners Neele Leithäuser and Johanna Schneider from the Fraunhofer Institute for Industrial Mathematics ITWM as well as the free geographical information system QGIS (`qgis.org`). The map data used is provided by OpenStreetMap under ODbL (`openstreetmap.org/copyright`).

Most computational experiments for this thesis have been performed on the computing cluster of the Department of Mathematics of the RWTH Aachen University. We would like to thank Beatrix, Birgit, and Frank for making these experiments possible.

2 Preliminaries

In this chapter, we clarify some basic notation and cover fundamental topics of mathematical optimization. Note that we do not present a self-contained introduction but aim to reflect our take on selected topics that will be important over the course of this thesis. For a broader overview, we refer to the books by Nemhauser and Wolsey [96] as well as Korte and Vygen [64].

2.1 Basic Notation and Graphs

For a set of numbers $S \subseteq \mathbb{R}$, we denote with $S_{\geq 0} = \{s \in S | s \geq 0\}$ the subset of non-negative numbers and with $S_{>0} = \{s \in S | s > 0\}$ the subset of positive numbers. We define $[n] = \{1, \dots, n\}$ for a positive integer $n \in \mathbb{Z}_{>0}$ and for convenience also $[0] = \emptyset$. For a non-negative integer $n \in \mathbb{Z}_{\geq 0}$, we denote $[n]_0 = \{0, \dots, n\}$. The set of all subsets of a set S is written as $2^S = \{S' \subseteq S\}$. Furthermore, we write $\binom{S}{n} = \left\{S' \in 2^S \middle| |S'| = n\right\}$ for the set of subsets of cardinality $n \in \mathbb{Z}_{\geq 0}$. Lastly, we define $x^+ = \max\{x, 0\}$ to be the positive part of a number $x \in \mathbb{R}$.

A *graph* is a tuple $G = (V, E)$ consisting of a set of *nodes* V and a set of *edges* $E \subseteq \binom{V}{2}$ *connecting* the nodes. If two nodes $v, w \in V$ are connected via an edge $\{v, w\} \in E$, then we say that v and w are *neighbors*. The *neighborhood* of a node $v \in V$ is denoted with $N(v) = \{w \in V | \{v, w\} \in E\}$ and the *closed neighborhood* with $N[v] = N(v) \cup \{v\}$. The set of *incident edges* is denoted with $\delta(v) = \{\{v, w\} \in E\}$. We say that a set of nodes $S \subseteq V$ is a *clique* if all pairs of nodes in S are neighbors. Conversely, $S \subseteq V$ is an *independent set* if there exists no edge connecting any two nodes in S.

2.2 Mathematical Optimization

Optimization is about finding the best possible solution to a problem. A solution is usually modeled as a vector of finitely many *decision variables* $x \in \mathbb{R}^n$, which reflect our choice as a collection of elementary decisions. For example, when going into a supermarket, we might decide to buy x_1 apples, x_2 packs of flour, x_3 packs of sugar, and x_4 packs of margarine for baking a delicious apple pie. Surely, not all $x \in \mathbb{R}^n$ represent a feasible solution for our problem. If we bought a negative amount of apples $x_1 < 0$, that is we actually sell $-x_1$, the supermarket would probably refuse to cooperate unless we happen to be owners of an apple

orchard. And although it would be lovely to buy exactly π kilograms of ingredients for our pie, we are most likely forced to purchase whole apples and packages. Thus, we are restricted to a subset of *feasible solutions* $\mathcal{X} \subseteq \mathbb{R}^n$.

To compare solutions, a strict partial order $\prec$ is defined on the set $\mathcal{X}$. That is, if we consider a minimization problem and have $x, x' \in \mathcal{X}$ with $x \prec x'$, then we prefer x over x'. A solution x is called *optimal* if there exists no $x' \in \mathcal{X}$ with $x' \prec x$. In many cases, the order is with respect to an *objective value* $v(x) \in \mathbb{R}$, such as the cost of our purchase. Then we have $x \prec x'$ if $v(x) < v(x')$ holds. However, not all optimization problems are of this form. This is especially when the needs of multiple stakeholders must be considered, like in Part II of this thesis. In this case, we have multiple objectives $(v_1(x), \dots, v_k(x)) \in \mathbb{R}^k$ which need to be minimized simultaneously. Then we might have $x \prec x'$ if $v_i(x) \leq v_i(x')$ holds for all $i \in [k]$ and $v_i(x) < v_i(x')$ for at least one $i \in [k]$.

The set of solutions and the objective should ideally be formulated in a concise way that allows for an exploration of the solution space without performing a pairwise comparison between all elements. In the following, we cover some concepts of mathematical optimization that enable us to model problems concisely.

2.3 Linear Programming

Linear programming is one of the fundamental concepts in mathematical optimization, as it provides us with powerful tools for modeling and solving problems. We give a brief introduction into the topic but refer to the books of Korte and Vygen [64], Schrijver [88], as well as Nemhauser and Wolsey [96] for more details.

A *linear program* (LP) consists of decision variables $x \in \mathbb{R}^n$ to which we assign *objective coefficients* $c \in \mathbb{R}^n$ whose sum $c^\top x = \sum_{i \in [n]} c_i x_i$ is to be minimized. Furthermore, the set of feasible solutions is restricted by linear constraints $Ax \geq b$, given by a matrix $A \in \mathbb{R}^{m \times n}$ and a right-hand side $b \in \mathbb{R}^m$. An LP is written as

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax \geq b \\ & x \in \mathbb{R}^n. \end{aligned}$$

Note that this form also allows for modeling maximization problems by simply using $-c$ instead of c. Likewise, we can model linear constraints $\alpha^\top x \leq \beta$ via $-\alpha^\top x \geq -\beta$. Equality constraints $\alpha^\top x = \beta$ can be modeled by using both $\alpha^\top x \geq \beta$ and $-\alpha^\top x \geq -\beta$.

For each LP, there are three possibilities regarding its solvability. An LP is *infeasible* if there exists no $x \in \mathbb{R}^n$ with $Ax \geq b$. It is *unbounded* if there exists a sequence of solutions $\left(x^k\right)_{k \in \mathbb{Z}_{\geq 0}}$ with $\lim_{k \to \infty} c^\top x^k = -\infty$. If an LP is neither infeasible nor unbounded, then there exists a

finite optimal solution [64, Chapter 3]. Let v be the optimal objective value of an LP, then we write $v = \infty$ for infeasible LPs and $v = -\infty$ for unbounded LPs. However, for the sake of simplicity, we will now only consider LPs that have an optimal solution.

2.3.1 The Simplex Method

LPs are not only powerful for modeling problems but can also be solved in polynomial time. The oldest known polynomial approach is the *ellipsoid algorithm* of Khachiyan [62], followed by Karmarkar's much more efficient *interior point method* [58]. For this thesis, however, we are particularly interested in the *simplex method*, which was proposed by Dantzig [39] as the first method for solving LPs. Simplex algorithms are in practice often faster than interior point methods, although there is currently no version known with a polynomial worst-case complexity. Borgwardt [25] theoretically supports the strength of the simplex method by showing that a version of it is polynomial on average for random instances in some probabilistic model.

To understand the idea of the simplex method, we consider the set of feasible solutions to our LP geometrically. A set $P \subseteq \mathbb{R}^n$ that is constrained by a finite number of linear inequalities, like our set of solutions, is called a *polyhedron*. We are interested in the extreme points of polyhedra, the so-called *vertices*. A vector $x \in P$ is a vertex if it cannot be expressed as a convex combination of other vectors $\left\{x^1, \ldots, x^k\right\} \subseteq P \setminus \{x\}$. Vertices have the special property that if there exists an optimal solution to an LP, then there exists a vertex which is also optimal. Hence, for finding an optimal solution to an LP, it is sufficient to only consider the vertices of P. Indeed, the idea of the simplex method is to start at some vertex and travel across the boundary of P, iteratively visiting further vertices until an optimal solution is reached.

We now consider vertices algebraically in order to see how they are visited in the simplex method. For this, note that each LP can be transformed into the so-called *standard form*

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{R}^n_{\geq 0}. \end{aligned}$$

This can be done by replacing all variables $x \in \mathbb{R}^n$ with their positive part $x^+ \in \mathbb{R}^n_{\geq 0}$ and negative part $x^- \in \mathbb{R}^n_{\geq 0}$, that is, $x = x^+ - x^-$. Furthermore, we can add so-called *slack variables* $s \in \mathbb{R}^n_{\geq 0}$ to the constraints, that is, $Ax \geq b$ is equivalent to $Ax - s = b$ for $s \in \mathbb{R}^m_{\geq 0}$. In the following, we assume that our LP is given in this form and that $\operatorname{rank}(A) = m$ holds. This is without loss of generality, as otherwise some rows are linearly dependent, implying that the constraints are redundant or conflicting. Under these assumptions, we define a subset $B \in \binom{[n]}{m}$ of size m to be a *basis* if the columns $\{A_i | i \in B\}$ of A are linearly independent. Given a set of indices $I \subseteq [n]$, we denote with A_I the submatrix of A consisting of the columns

with index in I. Likewise, we denote with x_I the corresponding subvector of x. The variables x_B are called *basic variables* and $x_{[n]\setminus B}$ *non-basic variables*. Since A_B is non-singular, we know that its inverse A_B^{-1} exists. Then the solution $x \in \mathbb{R}^n$ with $x_B = A_B^{-1}b$ and $x_{[n]\setminus B} = 0$ is feasible if and only if we have $x_B \geq 0$. Solutions of this form are called *feasible basic solutions* and are of particular interest, as they are exactly the vertices of a polyhedron [64, Section 3.1].

In order to solve an LP, one could enumerate all possible sets $B \in \binom{[n]}{m}$, check whether they correspond to a feasible basic solution, and then return the one with the best objective value. Luckily, the simplex method is more efficient and does not consider all possible bases but only a sequence of neighboring feasible basic solutions with improving objective coefficients. Two feasible basic solutions are called *neighbors* if their bases B_1 and B_2 can be transformed into another by swapping one index, that is $|B_1 \setminus B_2| = 1$. A feasible basic solution is optimal if there exists no neighboring solution with a better objective value. This is the case if the so-called *reduced costs* $\bar{c}^\top = c_{[n]\setminus B}^\top - c_B^\top A_B^{-1} A_{[n]\setminus B}$ are non-negative (we will see later why). Otherwise, if there exists an index $i \in [n] \setminus B$ with $\bar{c}_i < 0$, then one of these indices may enter the basis B while a different index $j \in B$ is removed, yielding a new basis B' that corresponds to a solution with a better objective value (neglecting degeneracy, cf. [96, Section I.2.3]). Hence, we can iteratively improve our solution by swapping basic variables with non-basic variables, improve the objective value in each iteration, and end at an optimal solution. For a detailed description on how to perform these swaps, we refer to one of the books cited above.

Note that we require an initial feasible basis to start our walk along the boundary of the polyhedron. This can be done by solving an auxiliary LP for which there exists an obvious feasible basis. The search for an initial basis is known as phase I of the simplex method, while the search for an optimal basis is known as phase II. When a sequence of similar LPs is solved, each emerging from the previous via small manipulations, it is often possible to use the previous basis and directly start in phase II. We will revisit this idea in Section 2.4.2, as it will be relevant for our branch and bound algorithm in Chapter 5.

2.3.2 Duality

In mathematical optimization, one often considers a problem from two sides, namely the *primal* and the *dual* problem. Given a primal minimization problem, the corresponding dual problem asks for lower bounds on the optimal objective of the primal problem. For an LP in standard form $\min \left\{ c^\top x \middle| Ax = b, x \in \mathbb{R}^n_{\geq 0} \right\}$, we can compute lower bounds on the optimal objective value by combining the constraints $Ax = b$ linearly with coefficients $y \in \mathbb{R}^m$. If this

linear combination $y^\top Ax = y^\top b$ is such that $y^\top A \leq c^\top$ holds, then $y^\top b$ is a lower bound on $c^\top x$, due to x being non-negative. Therefore, the dual problem of the above LP reads

$$\begin{aligned} \max\ & y^\top b \\ \text{s.t. } & y^\top A \leq c^\top \\ & y \in \mathbb{R}^n. \end{aligned}$$

Due to the reasoning above, the objective value of a solution to the dual LP is always smaller than the objective value of a solution to the primal LP. This property is called *weak duality*. Duality for LPs is particularly interesting, because if one of the LPs has an optimal solution, then the other one also has an optimal solution with the same value [64, Section 3.4]. This equality of optimal primal and dual objective values is called *strong duality*.

2.3.3 The Dual Simplex Method

Solving either the primal or dual LP does not only yield the optimal objective value for the other problem, but we can even compute optimal primal and dual solutions simultaneously. Let $B \subseteq [n]$ be a basis for the primal problem $\min\left\{c^\top x \middle| Ax = b, x \in \mathbb{R}^n_{\geq 0}\right\}$. We call B *primal feasible* if $A_B^{-1} b \geq 0$ holds, that is, if the corresponding basic solution with $x_B = A_B^{-1} b$ and $x_{[n]\setminus B} = 0$ is feasible for the primal problem. Furthermore, we define the corresponding dual solution as $y^\top = c_B^\top A_B^{-1}$. This solution is feasible for the dual problem if $y^\top A \leq c^\top$ holds, where

$$y^\top A \leq c^\top \Leftrightarrow c_B^\top A_B^{-1} A \leq c^\top \Leftrightarrow c_B^\top A_B^{-1} A_{[n]\setminus B} \leq c_{[n]\setminus B}^\top.$$

Therefore, we call B *dual feasible* if we have $c_{[n]\setminus B}^\top - c_B^\top A_B^{-1} A_{[n]\setminus B} \geq 0$. Moreover, since we have

$$c^\top x = c_B^\top x_B = c_B^\top A_B^{-1} b = y^\top b,$$

it follows together with weak duality that x and y are optimal solutions if B is primal and dual feasible. Note that B is dual feasible if and only if the reduced costs defined in Section 2.3.1 are non-negative, which yields the optimality criterion of the primal simplex method.

The fact that a basis B yields both a primal and a dual optimal solution raises the possibility to solve an LP from two different sides. The *primal simplex method* computes a sequence of primal feasible bases until one is found that is also dual feasible. Conversely, the *dual simplex method* computes a sequence of dual feasible bases and aims for one that is also primal feasible. While the dual simplex is similar to the primal simplex, it is particularly useful in case we already have a basis that is not primal feasible but dual feasible. In this case, the dual simplex can directly start in phase II, and is thus usually faster.

2.4 Mixed-Integer Linear Programming

As already noted in Section 2.2, fractional decisions $x \notin \mathbb{Z}^n$ are not always feasible for real-world applications. If integrality of some variables is required, we rely on the concepts of (mixed) integer linear programming. Again, we only give a brief introduction into the topic and refer to the books of Conforti et al. [37] as well as Nemhauser and Wolsey [96].

Mixed-integer linear programs (MILP) generalize LPs by allowing for integrality restrictions of variables, that is $x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$ instead of $x \in \mathbb{R}^n$ with $n_1 + n_2 = n$. An MILP is written as

$$\begin{aligned} &\min c^\top x \\ &\text{s.t. } Ax \geq b \\ &\quad x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}. \end{aligned}$$

Unlike LPs, MILPs are in theory hard to solve and there exists no polynomial-time algorithm for general instances unless $\mathcal{P} = \mathcal{NP}$. In practice, however, modern MILP solvers are often quite successful in solving even large problems with a million variables. The driving force behind this success is the *branch and bound* paradigm, which was first proposed by Land and Doig [69].

2.4.1 Branch and Bound

The general idea of branch and bound for solving optimization problems of the form $\min\{v(x)|x \in \mathcal{X}\}$ is to partition *(branch)* the set of feasible solutions $\mathcal{X} = \biguplus_{i\in[k]} X_i$ and then solve the corresponding subproblems $\min\{v(x)|x \in X_i\}$ recursively. When considering the subproblems $\min\{v(x)|x \in X\}$ as nodes in a graph that are connected to their parent problem $\min\{v(x)|x \in X'\}$ with $X \subseteq X'$, from which they emerged directly via branching, then we obtain a rooted *branch and bound tree*. We call the original problem $\min\{v(x)|x \in \mathcal{X}\}$ the *root node problem*. In order to avoid a complete enumeration, an easy to obtain *dual bound* $\underline{v}(X) \leq \min\{v(x)|x \in X\}$ is computed for every considered $X \subseteq \mathcal{X}$ and compared with a *primal bound* $\overline{v}$, which is the value of the so far best known feasible solution. If $\underline{v}(X) \geq \overline{v}$ holds, then we can *prune* the branch of the tree considering X, as we will not be able to find a better solution there. To compute the dual bound $\underline{v}(X)$, one usually considers a *relaxed problem* $\min\left\{\underline{v}(x)\middle|x \in \tilde{X}\right\}$ of $\min\{v(x)|x \in X\}$ with $X \subseteq \tilde{X}$ and $\underline{v}(x) \leq v(x)$ for all $x \in X$. Naturally, this relaxed problem should be much easier to solve, so that we can efficiently compute the dual bound $\underline{v}(X)$.

A common relaxation for problems with integrality restrictions is the *continuous relaxation*, for which we drop the integrality constraints. For an MILP $\min\left\{c^\top x\middle|Ax \geq b, x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}\right\}$, we start with solving the LP $\min\left\{c^\top x\middle|Ax \geq b, x \in \mathbb{R}^{n_1+n_2}\right\}$. If the computed optimal solution

$x^* \in \mathbb{R}^{n_1+n_2}$ is *integer-feasible*, that is $x^* \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$, then x^* is also a solution to the MILP. Otherwise, there exists an index $i \in [n_1]$ with $x_i^* \notin \mathbb{Z}$. In this case, we branch on the variable x_i by considering two new LPs, one with the additional constraint $x_i \leq \lfloor x_i^* \rfloor$ and one with $x_i \geq \lceil x_i^* \rceil$. The fractional solution x^* is now no longer feasible for any of the two subproblems, but all integer-feasible solutions remain feasible for one of them. Recursive branching and pruning until no subproblem is left for consideration thus yields and proves an optimal solution.

2.4.2 Warm Starting in Branch and Bound

Branch and bound is in general "only" an intelligent enumeration approach and the number of nodes in the tree cannot be polynomially bounded unless $\mathcal{P} = \mathcal{NP}$. Therefore, we potentially need to solve many LPs in order to solve one MILP. However, most of the LPs can be solved very efficiently in practice: After the root node, we can *warm start* the simplex method for all subsequent nodes in phase II. To see this, let $\min\left\{c^\top x \middle| Ax = b, x \in \mathbb{R}^n_{\geq 0}\right\}$ be an LP in standard form that we consider during the branch and bound. Furthermore, let $B \subseteq [n]$ be the computed optimal, and thus dual feasible, basis. Branching corresponds to adding a new constraint $\alpha^\top x + x_{n+1} = \beta$ with slack variable x_{n+1}. The problem of the child node then reads $\min\left\{\left(c^\top, 0\right) x \middle| \begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix} x = \begin{pmatrix} b \\ \beta \end{pmatrix}, x \in \mathbb{R}^{n+1}_{\geq 0}\right\}$. For $B' = B \cup \{n+1\}$, the matrix $\begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix}_B = \begin{pmatrix} A_B & 0 \\ \alpha_B^\top & 1 \end{pmatrix}$ has rank $m+1$ if and only if A_B has rank m and the corresponding reduced costs

$$\begin{aligned}
&\left(c^\top, 0\right)_{[n+1]\setminus B'} - \left(c^\top, 0\right)_{B'} \begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix}_{B'}^{-1} \begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix}_{[n+1]\setminus B'} \\
=&c_{[n]\setminus B}^\top - \left(c_B^\top, 0\right) \begin{pmatrix} A_B^{-1} & 0 \\ * & 1 \end{pmatrix} \begin{pmatrix} A_{[n]\setminus B} \\ \alpha_{[n]\setminus B}^\top \end{pmatrix} \\
=&c_{[n]\setminus B}^\top - c_B^\top A_B^{-1} A_{[n]\setminus B}
\end{aligned}$$

are exactly the reduced costs of B for the parent problem. Hence, B' is a dual feasible basis for the new LP if and only if B is a dual feasible basis for the original LP. Therefore, B' can be used to warm start the dual simplex method in phase II.

Moreover, the above equivalency implies that if $\alpha^\top x + x_{n+1} = \beta$ is a constraint with basic slack variable x_{n+1}, then removing $n+1$ from the basis yields a dual feasible basis for the LP in which the constraint is removed. We may also increase the objective coefficient c_i of non-basic variables x_i: For $c' \geq c$ with $c'_i = c_i$ for $i \in B$, we have

$$c_{[n]\setminus B}'^\top - c_B'^\top A_B^{-1} A_{[n]\setminus B} \;= c_{[n]\setminus B}'^\top - c_B^\top A_B^{-1} A_{[n]\setminus B} \;\geq\; c_{[n]\setminus B}^\top - c_B^\top A_B^{-1} A_{[n]\setminus B} \;\geq\; 0.$$

All these observations will be important later in Section 5.6.6 and are therefore summarized in the following remark.

Remark 1. Given a dual feasible basis of an LP, the following manipulations result in a dual feasible basis for the new LP

- adding a new constraint $\alpha^\top x + s = \beta$ and setting the slack variable s to basic,
- removing a constraint $\alpha^\top x + s = \beta$ together with its slack variable s if s is basic,
- increasing the objective coefficient c_i of a non-basic variable x_i.

If we also increase the objective coefficients of basic variables, then the dual feasibility of our basis is in general not preserved. However, if we only want to manipulate the objective coefficients of an LP and leave the constraints unchanged, then we can use the primal instead of the dual simplex method. Note that the non-negativity of $x_B = A_B^{-1}b$, and thus the primal feasibility of basis B, is independent of the objective coefficients. Therefore, we can use B to warm start the primal simplex method in phase II. Moreover, we can also remove constraints with basic slack variables from the LP and preserve primal feasibility. That is, if $B \subseteq [n+1]$ with $n+1 \in B$ is a primal feasible basis for $\min\left\{\left(c^\top, 0\right)x \middle| \begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix} x = \begin{pmatrix} b \\ \beta \end{pmatrix}, x \in \mathbb{R}^{n+1}_{\geq 0}\right\}$, then $B' = B \setminus \{n+1\}$ is a primal feasible basis for $\min\left\{c^\top x \middle| Ax = b, x \in \mathbb{R}^n_{\geq 0}\right\}$, since we have

$$0 \leq x_B = \begin{pmatrix} A & 0 \\ \alpha^\top & 1 \end{pmatrix}_B^{-1} \begin{pmatrix} b \\ \beta \end{pmatrix} = \begin{pmatrix} A_{B'}^{-1} & 0 \\ * & 1 \end{pmatrix} \begin{pmatrix} b \\ \beta \end{pmatrix} = \begin{pmatrix} A_{B'}^{-1} b \\ * \end{pmatrix},$$

and thus $A_{B'}^{-1} b \geq 0$. Again, we summarize both observations in the following remark.

Remark 2. Given a primal feasible basis of an LP, the following manipulations result in a primal feasible basis for the new LP

- arbitrarily changing objective coefficients c,
- removing a constraint $\alpha^\top x + s = \beta$ together with its slack variable s if s is basic.

In Section 5.6.6, we will use our observations about primal and dual feasible bases to speed up the LP solving process by warm starting the appropriate simplex method.

2.4.3 Strong Formulations

The number of nodes one needs to consider in a branch and bound tree largely depends on the difference between the original problem and the chosen relaxations. If an optimal solution x to the relaxation of the root node $\min\left\{\underline{v}(x) \middle| x \in \tilde{\mathcal{X}}\right\}$ is also feasible for the original problem $\min\{v(x) | x \in \mathcal{X}\}$ and we have $\underline{v}(x) = v(x)$, then x is also optimal for the original problem. In contrast, if x is far from being feasible for $\mathcal{X}$ or if $\underline{v}(x) \ll v(x)$ holds, then

we potentially require many branching steps until we obtain and prove an optimal solution to the original problem. This implies the need to classify "strong" and "weak" relaxations. Throughout this thesis, we are mostly concerned with linear objective functions, which we also use for the relaxed problem. Therefore, we are particularly interested in the relaxation $\tilde{\mathcal{X}}$ of the set of feasible solutions $\mathcal{X}$.

In the context of mixed-integer optimization, we call a set $\mathcal{F} \subseteq \mathbb{R}^n$ *with* $n = n_1 + n_2$ *a formulation* for a problem $\min\{v(x)|x \in \mathcal{X}\}$ with a set of solutions $\mathcal{X} \subseteq \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}$ if $\mathcal{F} \cap (\mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}) = \mathcal{X}$ holds. Using such a formulation, we can solve the original problem by solving $\min\{v(x)|x \in \mathcal{F}\}$ and branching on the integer variables. If the original problem is an MILP, then one often only considers polyhedral formulations, the most prominent being the continuous relaxation $\mathcal{F} = \{x \in \mathbb{R}^n | Ax \geq b\}$ for $\mathcal{X} = \{x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2} | Ax \geq b\}$. However, we will also consider more general sets $\mathcal{F}$, and thus don't restrict ourselves to polyhedra. Intuitively, a formulation is strong if it is close to the set of feasible solutions $\mathcal{X}$. Therefore, if $\mathcal{F}_1$ and $\mathcal{F}_2$ are two formulations for the same problem, then we say that $\mathcal{F}_1$ is *at least as strong* as $\mathcal{F}_2$ if $\mathcal{F}_1 \subseteq \mathcal{F}_2$ holds.

With polyhedral formulations, which are convex, the closest we can possibly get to $\mathcal{X}$ is its *convex hull* $\operatorname{conv}(\mathcal{X})$, which is the smallest possible convex set containing all points in $\mathcal{X}$. If $\mathcal{X}$ is the solution space of an MILP $\min\left\{c^\top x \middle| Ax \geq b, x \in \mathbb{Z}^{n_1} \times \mathbb{R}^{n_2}\right\}$ with rational constraint matrix $A \in \mathbb{Q}^{m \times n}$ and right-hand side $b \in \mathbb{Q}^m$, then $\operatorname{conv}(\mathcal{X})$ itself is a polyhedron, and thus the best polyhedral formulation [75]. Since the vertices of $\operatorname{conv}(\mathcal{X})$ are all contained in $\mathcal{X}$, an optimal vertex solution of the LP $\min\left\{c^\top x \middle| x \in \operatorname{conv}(\mathcal{X})\right\}$ is also an optimal solution to the MILP. Hence, we can solve any MILP by solving the corresponding LP over $\operatorname{conv}(\mathcal{X})$. This motivates calling $\operatorname{conv}(\mathcal{X})$ a *perfect formulation*. However, this observation also suggests that $\operatorname{conv}(\mathcal{X})$ is in general not easy to describe, since otherwise we had $\mathcal{P} = \mathcal{NP}$. In fact, the convex hull is for most problems described by an exponential number of inequalities. Only in some special cases, the continuous relaxation $\{x \in \mathbb{R}^n | Ax \geq b\}$ already equals $\operatorname{conv}(\mathcal{X})$. An important special case is when the right-hand side b is integer and the matrix A is *totally unimodular*, that is, the determinant of every square submatrix of A is in $\{-1, 0, 1\}$ [96, Section III.1.2].

In practice, we usually rely on formulations that are as strong as possible while also being *compact*, that is, the size of the constraint matrix is polynomial in the size of the problem instance. To achieve this, it is sometimes beneficial to include additional variables in the formulation. Using these auxiliary variables, we might be able to describe a compact formulation, whereas we would otherwise require an exponential number of constraints if we limited ourselves to the original variable space. A set $\mathcal{F}' \subseteq \mathbb{R}^{n+n'}$ is called an *extended formulation* for a problem if its projection $\operatorname{proj}(\mathcal{F}') \subseteq \mathbb{R}^n$ into the original solution space is a formulation for that problem. Then, instead of solving $\min\{v(x)|x \in \mathcal{F}\}$, we solve $\min\{v(x)|(x, y) \in \mathcal{F}'\}$ and project the computed solution $(x, y) \in \mathbb{R}^{n+n'}$ to $x \in \mathbb{R}^n$. We say that an extended formulation $\mathcal{F}'_1$ is at least as strong as an extended formulation $\mathcal{F}'_2$ for the same problem if $\operatorname{proj}(\mathcal{F}'_1) \subseteq \operatorname{proj}(\mathcal{F}'_2)$ holds.

Note that the objective function for the above problem using the extended formulation is defined only on the original variables. To generalize the concept, we call a problem $v' = \min\{v'(x') | x' \in \mathcal{X}'\}$ defined by some $\mathcal{F}' \subseteq \mathbb{R}^{n'_1+n'_2}$ with $\mathcal{F}' \cap \left(\mathbb{Z}^{n'_1} \times \mathbb{R}^{n'_2}\right) = \mathcal{X}'$ a *reformulation in a different variable space* of $v = \min\{v(x) | x \in \mathcal{X}\}$, if both have the same optimum objective value, i.e., $v = v'$, and there exists a polynomial-time computable, objective-preserving mapping $\phi : \mathcal{F}' \to \mathbb{R}^n$ with $\phi(\mathcal{X}') \subseteq \mathcal{X}$. Then, instead of solving the original problem, we can solve the problem over $\mathcal{X}'$ and map an optimal solution $x' \in \mathcal{X}'$ to an optimal solution $\phi(x') \in \mathcal{X}$. To generalize the concept of strong formulations, we say that $\mathcal{F}'_1$ is at least as strong as $\mathcal{F}'_2$ if $\phi_1(\mathcal{F}'_1) \subseteq \phi'_2(\mathcal{F}'_2)$ holds for the corresponding mappings ϕ_1, ϕ_2.

We cannot rank all formulations with the above definitions, as there exists no total order on the set of formulations. A practical possibility to quantify the strengths of different formulations is to consider the so-called *integrality gap*, which measures the relative difference between the optimal objective values of the original problem and the continuous relaxation. Let v be the optimal objective value for a problem and v^{R} be the optimal objective value of the continuous relaxation with respect to the considered formulation. Then we define the integrality gap as $\frac{|v^{\mathrm{R}}-v|}{|v|}$ for $v \neq 0$. For $v = 0$, we define it to be zero if $v^{\mathrm{R}} = 0$ holds and ∞ otherwise. A small integrality gap indicates that an optimal solution to the relaxation might not be far from an optimal solution to the original problem, and should thus imply less computational effort during the branch and bound.

2.4.4 Valid Inequalities and Branch and Cut

Although we usually don't know how to describe the convex hull of feasible solutions for an MILP, we can often improve a formulation by adding *valid inequalities*. If $\mathcal{X}$ is the set of feasible solutions, then we call an inequality $\pi^\top x \leq \pi_0$ valid if it is satisfied by all $x \in \mathcal{X}$. Naturally, not all inequalities are useful for improving a formulation. For example, $0 \leq 1$ is always valid but does not help us describing $\mathrm{conv}(\mathcal{X})$. Instead, we are interested in so-called *cutting planes*, which are valid for $\mathcal{X}$ but not for the current formulation $\mathcal{F}$. That is, we cut off a part of $\mathcal{F}$ but retain all solutions in $\mathcal{X}$ when adding a cutting plane to a formulation.

Cutting planes are an integral part of MILP solvers, which use an extension of branch and bound, the so-called *branch and cut* approach. The idea of branch and cut is to add cutting planes to strengthen a formulation so that fewer nodes need to be considered in the branching tree. For this, one does not add all (potentially exponentially many) known cutting planes to the formulation but only ones that cut off interesting parts of $\mathcal{F}$. Given an optimal solution $x \in \mathcal{F} \setminus \mathcal{X}$ of the continuous relaxation that is not feasible for the original problem, we call a cutting plane $\pi^\top x \leq \pi_0$ a *separating cutting plane for* x if $\pi^\top x > \pi_0$ holds. Then adding $\pi^\top x \leq \pi_0$ to our formulation and reoptimizing over the resulting relaxation yields a new solution x' that is hopefully closer to the set of feasible solutions $\mathcal{X}$.

Note that the reoptimization can be done efficiently in practice by using the dual simplex method. Finding a separating cutting plane for a solution $x \in \mathcal{F} \setminus \mathcal{X}$ is itself an optimization problem and known as the *separation problem*. In theory, it is possible to solve an MILP by iteratively optimizing over the current formulation and adding separating cutting planes until a feasible solution is found. However, since this in general amounts to describing $\operatorname{conv}(\mathcal{X})$, one usually only adds few cutting planes to the formulation and then continues branching.

2.4.5 Strong Valid Inequalities and Facets

Just as with formulations, inequalities can also be (partially) ranked. We say that a valid inequality $\pi^\top x \leq \pi_0$ *dominates* another inequality $\mu^\top x \leq \mu_0$ if there exists a coefficient $\lambda \in \mathbb{R}_{\geq 0}$ such that $\lambda\pi \geq \mu$ and $\lambda\pi_0 \leq \mu_0$ holds. In this case, we have $\left\{x \in \mathbb{R}^n_{\geq 0} \middle| \pi^\top x \leq \pi_0\right\} \subseteq \left\{x \in \mathbb{R}^n_{\geq 0} \middle| \mu^\top x \leq \mu_0\right\}$. Furthermore, we say that $\pi^\top x \leq \pi_0$ *strictly dominates* $\mu^\top x \leq \mu_0$ if we have $\lambda\pi \neq \mu$ or $\lambda\pi_0 \neq \mu_0$ for a coefficient $\lambda > 0$ as above. Since inequalities are not applied isolated, we also define dominance for multiple inequalities. We say that a set of inequalities $\pi^{j\top} x \leq \pi^j_0$ with $j \in [k]$ dominates an inequality $\mu^\top x \leq \mu_0$ if there exist coefficients $\lambda_1, \ldots, \lambda_k \in \mathbb{R}_{\geq 0}$ such that the combination $\left(\sum_{j\in[k]} \lambda_j \pi^j\right)^\top x \leq \sum_{j\in[k]} \lambda_j \pi^j_0$ dominates $\mu^\top x \leq \mu_0$. Strict domination for multiple inequalities is defined analogously.

Consider a polyhedral formulation $\mathcal{F} = \left\{x \in \mathbb{R}^n_{\geq 0} \middle| Ax \leq b\right\}$ and an inequality $\mu^\top x \leq \mu_0$. While it is clear that $\mu^\top x \leq \mu_0$ is no cutting plane if it is dominated by the constraints $Ax \leq b$, it is not trivial to see whether the other direction also holds. The following proposition shows that non-cutting planes are indeed dominated.

Proposition 3 (Nemhauser and Wolsey [96, Section II.1.1])**.** *If $P = \left\{x \in \mathbb{R}^n_{\geq 0} \middle| Ax \leq b\right\}$ is a non-empty polyhedron, then all valid inequalities for P are dominated by the constraints $Ax \leq b$.*

Knowing that an inequality can be omitted if it is dominated by other inequalities in our constraint matrix, we might ask ourselves which inequalities are actually needed to describe a polyhedron. This is especially interesting when the polyhedron itself is not known, as in the case of the convex hull. Intuitively, it is clear that inequalities $\pi^\top x \leq \pi_0$ that cannot be omitted define a hyperplane $\left\{x \in \mathbb{R}^n \middle| \pi^\top x = \pi_0\right\}$ which intersects with the boundary of the polyhedron P. The intersection of the hyperplane with the polyhedron $F(\pi) = \left\{x \in P \middle| \pi^\top x = \pi_0\right\}$ is called the *face* of P defined by $\pi^\top x \leq \pi_0$. The proper faces $\emptyset \subset F \subset P$ are at the boundary of a polyhedron, with the smallest proper faces being the vertices. The largest proper faces, the so-called *facets*, are the most interesting ones and can be characterized via their dimension. Each face is again a polyhedron and the dimension of a polyhedron P' is defined as the maximum number of affinely independent points within P' minus one. The following proposition characterizes facets.

Proposition 4 (Conforti et al. [37, Section 3.9])**.** *A face F of a polyhedron $P \subseteq \mathbb{R}^n$ is a facet if and only if* $\dim(F) = \dim(P) - 1$ *holds.*

We say that an inequality is *facet-defining* if its face is a facet. Facet-defining inequalities are in fact exactly the inequalities needed to describe a polyhedron P in the sense that a minimal representation

$$P = \left\{ x \in \mathbb{R}^n \middle| \begin{array}{l} A^{=}x = b^{=} \\ A^{<}x \leq b^{<} \end{array} \right\},$$

with as few constraints as possible, consists of $n - \dim(P)$ equations $A^{=}x = b^{=}$ and otherwise only proper inequalities $A^{<}x \leq b^{<}$, all of which are facet-defining [37, Section 3.9]. From this, we deduce the following statement, which is an important tool for determining the dimension of a polyhedron.

Proposition 5. *For a polyhedron $P \subseteq \mathbb{R}^n$, the number $n - \dim(P)$ equals the maximum number of linearly independent equations that are met by all elements in P.*

In theory, we would like to know all facet-defining inequalities of the convex hull of integer-feasible solutions in order to strengthen our formulation. In practice, computing these inequalities is in general $\mathcal{NP}$-hard and also not always useful: Some inequalities cut off uninteresting regions of our formulation, and thus have little impact on the performance of a branch and cut approach. Conversely, we might be able to efficiently compute cutting planes for our current formulation that are strictly dominated by the facets of the convex hull but have a positive impact on the branch and cut. Therefore, valid inequalities should not only be evaluated based on whether they define a facet but also based on empirical studies. For this, we cannot only measure the impact of adding cuts on the computation time but also the impact on the integrality gap of our formulation.

Part I

Algorithms for Robust Combinatorial Optimization with Budgeted Uncertainty

3 Introduction

Decision-makers are frequently confronted with problems that are inherently affected by uncertainties in the parameters. Such uncertain parameters may result, for example, from forecasts, estimates, or inaccurate measurements. They should be treated with caution, as variations in the parameters can render a seemingly optimal decision impractical. This is especially in mathematical optimization, where optimal solutions often exhaust as much as possible the limits imposed by the planning constraints. Accordingly, already small fluctuations can cause these constraints to be heavily violated [14]. But even if all parameters in the constraints are certain, a profitable solution may suddenly become costly if parameters in the objective turn out to be different than expected.

A popular approach for integrating protection against uncertainty into optimization problems is *robust optimization*. Here, one considers an *uncertainty set* that consists of (potentially infinitely many) *scenarios* reflecting the uncertain parameters. The goal of robust optimization is to optimize against the *worst-case scenario*. That is, a *robust solution* remains feasible for all considered scenarios and an *optimal robust solution* should still have a reasonable objective value in the worst-case. Robust optimization was proposed first by Soyster [90] in the early 1970s. Kouvelis and Yu [66] considered it for combinatorial optimization problems and discrete uncertainty sets in the 1990s. The concept was then further analyzed by Ben-Tal and Nemirovski [13, 14, 15], and Bertsimas and Sim [22, 23] at the beginning of this century. An overview on the topic is given in [12, 19, 18, 44].

The approach by Bertsimas and Sim of modeling uncertainty with *budgeted uncertainty sets* has proven to be the most popular, with their introductory paper [23] being the most cited document on robust optimization in the literature databases Scopus and Web of Science (search for “robust optimization” in title, keywords, and abstract). This popularity can be attributed to several reasons. One reason is that the modeling of the uncertainty set is quite intuitive, with an *uncertainty budget* controlling the extent to which one wants to protect against uncertainties. Bertsimas and Sim prove that, under certain assumptions, already a low uncertainty budget offers high probabilistic guarantees that a robust solution remains feasible for scenarios that are not contained in the uncertainty set. Simultaneously, they show experimentally that a low uncertainty budget only leads to a small loss in the objective value, thus demonstrating that the “price of robustness” is relatively low [23].

Another reason for the popularity of budgeted uncertainty is the special structure of the emerging robust problems. Most notably, (mixed-integer) linear problems remain (mixed-integer) linear, which constitutes an advantage in terms of tractability compared to earlier uncertainty sets, for example ellipsoidal sets [15]. Moreover, there exists a *reformulation* for

mathematical programming problems which is polynomial in the size of the formulation of the non-robust problem. The structure of the uncertainty set also allows for the development of combinatorial algorithms: Bertsimas and Sim [22] show that robust combinatorial problems with uncertainty in the objective function are theoretically not more complex than their non-robust versions. In particular, the robust counterpart of polynomially solvable combinatorial optimization problems remains polynomially solvable.

Despite these promising algorithmic results, instances from practice can still pose a considerable challenge for modern MILP solvers, even if the non-robust problem is relatively easy to solve, as observed, for example, by Kuhnke et al. [67]. In this thesis, we will show that this disillusioning performance is due to the weakness of the reformulation. We propose new formulations and approaches to address this issue. In this process, we restrict ourselves to combinatorial problems with uncertain objective functions. However, most of our results carry over to uncertain constraints.

3.1 Problem Statement

We start with a standard, so-called *nominal,* combinatorial optimization problem without uncertainties

$$\text{(NOM)} \qquad \begin{aligned} \min \ & \sum_{i\in[n]} c_i x_i \\ \text{s.t. } & Ax \leq b \\ & x \in \{0,1\}^n , \end{aligned}$$

with binary decision variables $x \in \{0,1\}^n$, an objective vector $c \in \mathbb{R}^n$, and a constraint matrix $A \in \mathbb{R}^{m\times n}$ with a right-hand side $b \in \mathbb{R}^m$. Instead of assuming the objective coefficients c_i to be certain, we consider uncertain coefficients c'_i that lie in an interval $c'_i \in [c_i, c_i + \hat{c}_i]$ and can *deviate* from their *nominal value* c_i by up to the *deviation* $\hat{c}_i$. The set of *scenarios* then consists of all vectors of possible objective coefficients $\{c' \in \mathbb{R}^n | c'_i \in [c_i, c_i + \hat{c}_i] \ \forall i \in [n]\}$. Note that for any $x \in \{0,1\}^n$, the objective value is worst if all coefficients c'_i are equal to their maximum value $c_i + \hat{c}_i$. In practice, however, this extreme scenario is usually unlikely, and thus it may be overly conservative to assume that it actually occurs.

To adjust the level of conservatism, Bertsimas and Sim [23] propose a robust counterpart to NOM in which they restrict the set of considered scenarios by defining an *uncertainty budget* $\Gamma \in [0, n)$. Given such a budget, we only consider those scenarios in which at most $\lfloor \Gamma \rfloor$

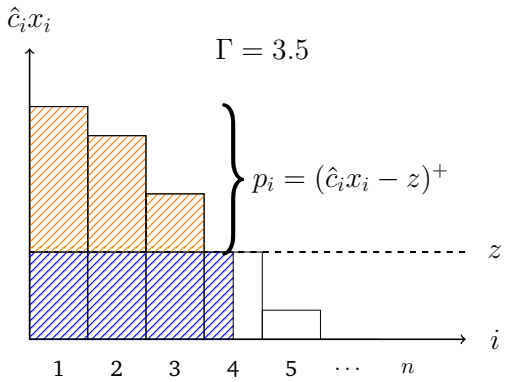


Figure 3.1. The optimal choice of z and p for a given nominal solution x.

coefficients c_i' deviate to $c_i + \hat{c}_i$ and one coefficient deviates to $c_i + (\Gamma - \lfloor\Gamma\rfloor)\,\hat{c}_i$. This robust counterpart of NOM can be stated as the non-linear robust problem

$$\text{(NLR)} \qquad \begin{aligned} \min \; & \sum_{i\in[n]} c_i x_i + \max_{\substack{S\cup\{t\}\subseteq[n]:\\ |S|\leq\lfloor\Gamma\rfloor, t\notin S}} \left((\Gamma - \lfloor\Gamma\rfloor)\,\hat{c}_t x_t + \sum_{i\in S} \hat{c}_i x_i \right) \\ \text{s.t. } & Ax \leq b \\ & x \in \{0,1\}^n . \end{aligned}$$

Bertsimas and Sim [23] show that the non-linearity can be resolved by writing the inner maximization problem as a linear program and dualizing it afterwards. This results in the *robust problem*

$$\text{(ROB)} \qquad \begin{aligned} & \min \Gamma z + \sum_{i\in[n]} c_i x_i + p_i \\ & \text{s.t. } (x,p,z) \in \mathcal{F}^{\text{ROB}}, x \in \{0,1\}^n \end{aligned}$$

with

$$\mathcal{F}^{\text{ROB}} = \left\{ (x,p,z) \middle| \begin{array}{ll} Ax \leq b & \\ p_i + z \geq \hat{c}_i x_i & \forall i \in [n] \\ x \in [0,1]^n, p \in \mathbb{R}^n_{\geq 0}, z \in \mathbb{R}_{\geq 0} & \end{array} \right\}.$$

Figure 3.1 gives a visual idea of why ROB is an equivalent reformulation of NLR. In the graphic, we are given a nominal solution $x \in \{0,1\}^n$ and depict the right-hand side of the *robustness constraints* $p_i + z \geq \hat{c}_i x_i$ as non-increasingly ordered bars of height $\hat{c}_i x_i$. For $\Gamma = 3.5$, we choose z equal to the $\lceil\Gamma\rceil = 4$ highest value $\hat{c}_i x_i$ and p_i as small as possible while fulfilling the robustness constraints, i.e., $p_i = (\hat{c}_i x_i - z)^+$. Then the orange area above the bar at height z corresponds to the term $\sum_{i\in[n]} p_i$ and the blue area below corresponds to Γz, which together constitute the robust part of the objective in ROB. Furthermore, the colored area matches exactly the $\lfloor\Gamma\rfloor$ largest bars and a $(\Gamma - \lfloor\Gamma\rfloor)$-fraction of the next largest bar. As these bars correspond to the set $S \cup \{t\} \subseteq [n]$ that maximizes the inner problem of NLR, we indeed have

$$\Gamma z + \sum_{i\in[n]} p_i = \max_{\substack{S\cup\{t\}\subseteq[n]:\\ |S|\leq\lfloor\Gamma\rfloor, t\notin S}} \left((\Gamma - \lfloor\Gamma\rfloor)\,\hat{c}_t x_t + \sum_{i\in S} \hat{c}_i x_i \right)$$

for our choice of z and p. We will see later in Section 5.5 that choosing z and p as above is always optimal.

Although ROB only has n additional constraints and $n+1$ continuous variables, solving ROB directly as an MILP can require much more time than solving the nominal problem NOM. For example, we observe in our computational study that Gurobi [52] already struggles to solve robust bipartite matching instances for graphs with 100 nodes within a time limit of an hour, although we have a compact perfect formulation for the nominal problem (cf. Section 4.6.5). This degradation in performance is due to the weakness of the formulation $\mathcal{F}^{\text{ROB}}$. In fact, the following example shows that the integrality gap of $\mathcal{F}^{\text{ROB}}$ can be arbitrarily large, even if we have a perfect formulation for the nominal problem.

Example 6. Consider the trivial problem of selecting the cheapest of $n \in \mathbb{Z}_{>0}$ elements

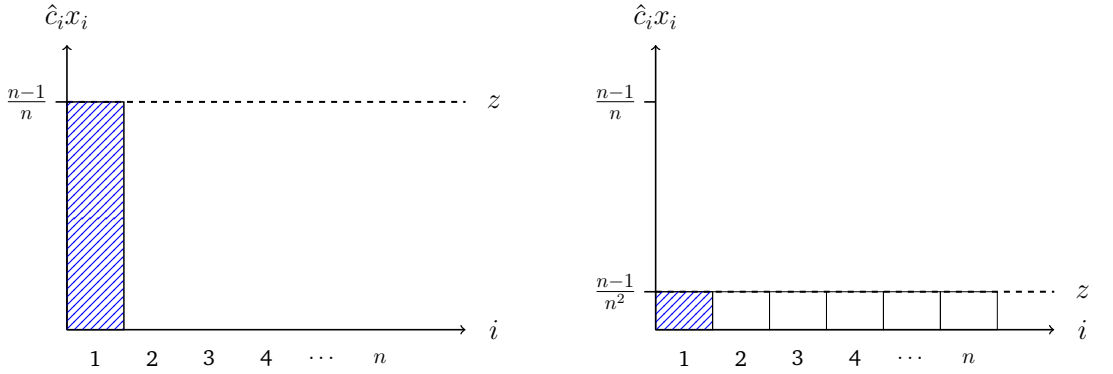
$$\begin{aligned} \min \sum_{i\in[n]} c_i x_i \\ \text{s.t.} \sum_{i\in[n]} x_i = 1 \\ x \in \{0,1\}^n . \end{aligned}$$

The above formulation is perfect, since we have a totally unimodular constraint matrix and an integer right-hand side (cf. Section 2.4.3). Therefore, every optimal integer solution is also an optimal continuous solution, yielding a perfect integrality gap of zero. However, if we consider an instance of the uncertain counterpart ROB with $c \equiv -1$, $\hat{c} \equiv \frac{n-1}{n}$, and $\Gamma = 1$

$$\begin{aligned} \min\ & z + \sum_{i\in[n]} p_i - x_i \\ \text{s.t.}\ & \sum_{i\in[n]} x_i = 1 \\ & p_i + z \geq \frac{n-1}{n} x_i && \forall i \in [n] \\ & x \in \{0,1\}^n, p \in \mathbb{R}^n_{\geq 0}, z \in \mathbb{R}_{\geq 0}, \end{aligned}$$

then $(x,p,z) = \left(\frac{1}{n},\dots,\frac{1}{n},0,\dots,0,\frac{n-1}{n^2}\right)$ is the unique optimal continuous solution of value $-1+\frac{n-1}{n^2}$, while the objective value of every integer solution is $-\frac{1}{n}$. Hence, the integrality gap as defined in Section 2.4.3 is $\frac{\left|-1+\frac{n-1}{n^2}+\frac{1}{n}\right|}{\left|-\frac{1}{n}\right|} = n-2+\frac{1}{n}$, and thus grows towards infinity as $n \to \infty$.

The example indicates that optimal continuous solutions for ROB tend to be highly fractional. Figure 3.2 shows the choice of z for the integer and continuous solution and visualizes that we can cover the right-hand sides $\hat{c}_i x_i$ of the robustness constraints with a fairly small value



(a) Optimal choice of z for an integer solution.

(b) Optimal choice of z for the continuous solution.

Figure 3.2. Difference between optimal integer and continuous solutions of Example 6.

of z when choosing fractional x_i. Instead of paying the highest deviation $\hat{c}_i$ of an element x_i that is part of the solution, we only pay the deviation of a fractional element. On the one hand, this structure of optimal continuous solutions intuitively proves the concept of robust optimization: Instead of choosing extreme solutions, the model favors diverse ones, which are less sensitive to uncertainty. On the other hand, this trend towards highly fractional solutions is problematic for the performance of MILP solvers (cf. Section 2.4.3). This points to an inherent challenge in solving robust optimization problems.

3.2 Related Work

In the literature, several approaches for solving ROB have been developed and evaluated. Bertsimas et al. [20] as well as Fischetti and Monaci [43] test the practical performance of the compact reformulation $\mathcal{F}^{\text{ROB}}$ compared to a separation approach using an alternative formulation with exponentially many inequalities, each one corresponding to the objective coefficients c' of a scenario from the budgeted uncertainty set (cf. Section 5.7.2). Unfortunately, the alternative formulation is, despite its size, theoretically as weak as $\mathcal{F}^{\text{ROB}}$. Joung and Park [57] propose cuts that dominate the classic scenario inequalities. These cuts can be separated by considering the value of the inner maximization problem in NLR as a submodular function in x and greedily solving a maximization problem over the corresponding polymatroid (cf. Section 5.7.2). Atamtürk [9] addresses the issue of the weak formulation by proposing four different problem-independent strong but considerably larger formulations (cf. Section 5.7.2). The strongest of these formulations is theoretically as strong as possible, as it preserves the integrality gap of the nominal problem. In particular, this implies that it describes the convex hull of robust solutions

$$\mathcal{C}^{\text{ROB}} = \text{conv}\left(\left\{(x,p,z) \in \mathcal{F}^{\text{ROB}} \middle| x \in \{0,1\}^n\right\}\right)$$

if the corresponding nominal formulation

$$\mathcal{F}^{\text{NOM}} = \{x \in [0,1]^n | Ax \leq b\}$$

itself equals the convex hull of nominal solutions

$$\mathcal{C}^{\text{NOM}} = \operatorname{conv}\left(\left\{x \in \mathcal{F}^{\text{NOM}} \middle| x \in \{0,1\}^n\right\}\right).$$

A famous approach to completely avoid issues arising from the weak formulation $\mathcal{F}^{\text{ROB}}$ is to solve ROB via resorting to its nominal counterpart. We already noted above that for a fixed $x \in \{0,1\}^n$, it is optimal to choose z equal to the $\lceil \Gamma \rceil$ largest value $\hat{c}_i x_i$. Hence, there always exists an optimal solution (x, p, z) to ROB such that $z \in \{\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_n\}$, with $\hat{c}_0 = 0$. Bertsimas and Sim [22] use this observation together with the fact that the optimal value $p_i = (\hat{c}_i x_i - z)^+$ equals $(\hat{c}_i - z)^+ x_i$ for binary x_i. When fixing $z \in \{\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_n\}$, the term $(\hat{c}_i - z)^+ x_i$ becomes linear, and thus ROB can be written as an instance of its nominal counterpart

$$\text{(NOS}\,(z)\text{)} \qquad \begin{aligned} &\min \Gamma z + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - z)^+\right) x_i \\ &\text{s.t. } x \in \mathcal{F}^{\text{NOM}}, x \in \{0,1\}^n. \end{aligned}$$

Hence, solving ROB reduces to solving up to $|\{\hat{c}_0, \hat{c}_1, \ldots, \hat{c}_n\}| \leq n+1$ *nominal subproblems NOS* (z), implying that the robust counterpart of a polynomially solvable nominal problem is again polynomially solvable. However, if the number of distinct deviations $|\{\hat{c}_0, \ldots, \hat{c}_n\}|$ is large, then solving all nominal subproblems may require too much time. Hence, it is beneficial to discard as many non-optimal choices for z as possible. For $\Gamma \in \mathbb{Z}$, Álvarez-Miranda et al. [6] as well as Park and Lee [82] showed that there exists a subset $\mathcal{Z} \subseteq \{\hat{c}_0, \ldots, \hat{c}_n\}$ containing an optimal choice for z with $|\mathcal{Z}| \leq n+2-\Gamma$, or $|\mathcal{Z}| \leq n+1-\Gamma$ respectively. This result was later improved by Lee and Kwon [70], who prove that $\mathcal{Z}$ can be chosen such that $|\mathcal{Z}| \leq \left\lceil \frac{n-\Gamma}{2} \right\rceil + 1$ holds. Hansknecht et al. [53] propose a divide and conquer approach for the robust shortest path problem that also aims to reduce the number of nominal subproblems to be solved. Their algorithm, which can as well be used to solve general problems ROB, successively divides the set of deviations $\{\hat{c}_0, \ldots, \hat{c}_n\}$ into intervals and chooses in each iteration a value z from the most promising interval for solving the nominal subproblem *NOS* (z). Furthermore, non-optimal choices of z are identified and discarded by using a relation between the optimal objective values of *NOS* (z) for different z (cf. Section 5.6.1.1).

3.3 Contribution and Outline

Roughly summarized, there are two general directions for solving ROB in the literature: strong formulations on the one hand and fixing z on the other hand. Our first approach

for solving ROB, presented in Chapter 4, follows the first direction by proposing the new class of *recycled inequalities* for ROB. These inequalities are based on valid inequalities for the nominal problem NOM, and thus enable us to use model constraints as well as cuts for well studied nominal problems a second time. Our proposed inequalities are easy to compute and often define facets of $\mathcal{C}^{\text{ROB}}$, that is, they are among the best inequalities to describe the feasible solutions of ROB. After a theoretical analysis of recycled inequalities, we conduct an extensive computational study on carefully generated robust versions of classical combinatorial problems and real-world instances from MIPLIB 2017 [49]. Our study verifies that recycled inequalities can substantially strengthen the formulation $\mathcal{F}^{\text{ROB}}$, which is expressed by drastic reductions of the integrality gap. Together with the efficient separation of recycled inequalities, this leads to a significant improvement of solving times.

Our second approach for solving ROB, presented in Chapter 5, combines the two general directions from the literature, that is, implying restrictions on the variable z and using strong formulations. However, instead of tentatively fixing $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$ to single values and solving many nominal subproblem NOS (z), we only restrict z to a subset $Z \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$. We show that such restrictions can be used to obtain a stronger formulation for ROB. Moreover, we demonstrate that the variable z and restrictions on it have a large impact on the structure of optimal solutions to ROB. We use these insights in a branch and bound approach in which we branch on the variable z. We again perform an extensive computational study, which reveals that our branch and bound algorithm outperforms the current state-of-the-art approaches by far. To the best of our knowledge, this is the first computational study that compares many sophisticated approaches from the literature on a broad set of instances. Moreover, we show that our structural results can be used to improve most of these approaches substantially, highlighting the relevance of our findings also for future research.

The results from Chapter 4 have been partially published in the proceedings of IPCO 2023 [29]. An extended paper is currently in revision at Mathematical Programming [30]. Most results from Chapter 5 have been published at Mathematical Programming Computation [31]. This thesis extends the published work with results on Lagrangean relaxations from Sections 5.4 and 5.6.5, the warm starting described in Section 5.6.6, and an updated computational study in Section 5.7. All described results have been produced by myself together with my supervisors Christina Büsing and Arie M.C.A. Koster.

All algorithms proposed in this part are implemented together along with many approaches from the literature and are freely available online [46]. Furthermore, all generated test instances are published and can be used for future benchmarking of algorithms for robust optimization with budgeted uncertainty [48].

4 Recycling Valid Inequalities

We have seen that the standard reformulation $\mathcal{F}^{\text{ROB}}$ is weak, which can lead to a poor performance of MILP solvers. In this chapter, we propose the new class of *recycled inequalities* to compensate for this weakness. For this, we first introduce a strong bilinear formulation, which will be an invaluable foundation for the results in this first part of the thesis. Afterwards, we analyze recycled inequalities both theoretically and experimentally.

4.1 A Strong Bilinear Formulation

Before we propose our approaches for solving ROB, let us first recall why formulation $\mathcal{F}^{\text{ROB}}$ is weak. In Example 6, we have seen that choosing fractional values for x_i is favorable for the continuous relaxation, even if $\mathcal{F}^{\text{NOM}}$ is a perfect formulation, as we are then able to meet the inequalities $p_i + z \geq \hat{c}_i x_i$ with a small value of z and $p_i = 0$. This structural advantage of the continuous relaxation does not carry over to the nominal subproblems NOS (z) for fixed z. Here, optimal integer solutions are also optimal for the continuous relaxation if $\mathcal{F}^{\text{NOM}}$ is perfect. This implies that the formulation is strengthened during the transformation from ROB to NOS (z). Indeed, the inequality $p_i \geq (\hat{c}_i - z)^+ x_i$, which is the basis for the substitution $p_i = (\hat{c}_i - z)^+ x_i$, is not implied by the robustness constraint $p_i + z \geq \hat{c}_i x_i$ for $x_i \notin \{0, 1\}$. Instead, it is equivalent to $p_i + z x_i \geq \hat{c}_i x_i$ for $x_i \in [0, 1]$, given that $p_i \geq 0$.

The *bilinear inequality* $p_i + z x_i \geq \hat{c}_i x_i$ is valid for all integer solutions of ROB, since it becomes $p_i \geq 0$ for $x_i = 0$ and is equivalent to the original constraint for $x_i = 1$. The multiplication of z with x_i has the benefit that choosing a lower value for x_i also reduces the left-hand side. Thus, it is no longer possible to cover all inequalities $p_i + z x_i \geq \hat{c}_i x_i$ with a small value of z and $p_i = 0$. When applying the bilinear inequalities to Example 6, we would now choose $z = \frac{n-1}{n}$ for $x = \left(\frac{1}{n}, \ldots, \frac{1}{n}\right)$, leading to the same objective value as the integer solution. In fact, we will see in the following that when applying the *bilinear problem*

$$\text{(BIL)} \qquad \begin{aligned} &\min \Gamma z + \sum_{i \in [n]} (c_i x_i + p_i) \\ &\text{s.t. } (x, p, z) \in \mathcal{F}^{\text{BIL}}, x \in \{0, 1\}^n \end{aligned}$$

over the *bilinear formulation*

$$\mathcal{F}^{\text{BIL}} = \left\{ (x, p, z) \middle| \begin{array}{ll} p_i + z x_i \geq \hat{c}_i x_i & \forall i \in [n] \\ x \in \mathcal{F}^{\text{NOM}}, p \in \mathbb{R}^n_{\geq 0}, z \in \mathbb{R}_{\geq 0} & \end{array} \right\}$$

together with a perfect nominal formulation $\mathcal{F}^{\text{NOM}}$, then the optimal integer objective value always equals the optimal continuous objective value. Note that we could also multiply the variable p_i with x_i in order to obtain an even stronger formulation with $p_i x_i + z x_i \geq \hat{c}_i x_i$. However, as we would never use this strengthening in this thesis, we only consider $\mathcal{F}^{\text{BIL}}$ for the sake of simplicity.

In order to show the strength of the bilinear formulation $\mathcal{F}^{\text{BIL}}$, we compare it to the strong formulations of Atamtürk [9]. Atamtürk proposes four problems RP1 - RP4 that are equivalent to ROB, using different (extended) formulations $\mathcal{F}^{\text{RP1}}, \dots, \mathcal{F}^{\text{RP4}}$. The theoretical strength (cf. Section 2.4.3) of the four formulations exceeds that of $\mathcal{F}^{\text{ROB}}$ by far. More precisely, when neglecting trivial cases, we have $\text{proj}\left(\mathcal{F}^{\text{RP4}}\right) \subsetneq \text{proj}\left(\mathcal{F}^{\text{RP1}}\right) = \mathcal{F}^{\text{RP2}} = \text{proj}\left(\mathcal{F}^{\text{RP3}}\right) \subsetneq \mathcal{F}^{\text{ROB}}$. The problem

$$\text{(RP4)} \qquad \begin{aligned} &\min \Gamma z + \sum_{i \in [n]} c_i x_i + p_i \\ &\text{s.t. } (x, p, z, \omega, \lambda) \in \mathcal{F}^{\text{RP4}}, \omega \in \{0,1\}^{n \times (n+1)} \end{aligned}$$

over the strongest formulation

$$\mathcal{F}^{\text{RP4}} = \left\{ (x, p, z, \omega, \lambda) \middle| \begin{aligned} &\sum_{k \in [n]_0} \lambda_k = 1 && \\ &A\omega^k \leq \lambda_k b && \forall k \in [n]_0 \\ &\omega_i^k \leq \lambda_k && \forall i \in [n], k \in [n]_0 \\ &\sum_{k \in [n]_0} \omega_i^k = x_i && \forall i \in [n] \\ &z \geq \sum_{k \in [n]_0} \hat{c}_k \lambda_k && \\ &p_i \geq \sum_{k \in [n]_0} (\hat{c}_i - \hat{c}_k)^+ \omega_i^k && \forall i \in [n] \\ &x \in [0,1]^n, p \in \mathbb{R}^n_{\geq 0}, z \in \mathbb{R}_{\geq 0}, && \\ &\omega \in [0,1]^{n \times n+1}, \lambda \in [0,1]^{n+1} && \end{aligned} \right\},$$

is especially interesting. Remember that we defined $\hat{c}_0 = 0$. Then for every vertex $(x, p, z, \omega, \lambda)$ of the polyhedron $\mathcal{F}^{\text{RP4}}$, we have $\lambda_{k^*} = 1$ for an index $k^* \in [n]_0$ and $\lambda_k = 0$ for $k \neq k^*$. Choosing λ in such a way reduces RP4 to solving the nominal subproblem NOS $(\hat{c}_{k^*})$. Thus, RP4 essentially combines the nominal subproblems NOS (z) that are solved in the Bertsimas and Sim approach for all possible values $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$ into one problem. As a result, RP4 preserves the integrality gap of the nominal problem. That is, if there exists an $\alpha \geq 0$ such that the integrality gap of NOM is bounded by α for all objective coefficients $c \in \mathbb{R}^n$, then the integrality gap of RP4 is also bounded by α for all $c, \hat{c} \in \mathbb{R}^n$ and $\Gamma \geq 0$ [9]. Accordingly, formulation $\mathcal{F}^{\text{RP4}}$ is the strongest possible polyhedral formulation, as we cannot improve the integrality gap beyond that of NOM without changing the nominal formulation $\mathcal{F}^{\text{NOM}}$.

The disadvantage of all formulations $\mathcal{F}^{\text{RP1}}, \ldots, \mathcal{F}^{\text{RP4}}$ is that they become too large for practical purposes, as we will see in our computational study in Section 5.7. In contrast, formulation $\mathcal{F}^{\text{BIL}}$ is non-linear but has the same size as the original formulation $\mathcal{F}^{\text{ROB}}$ and is at least as strong as $\mathcal{F}^{\text{RP4}}$, as stated in the following theorem.

Theorem 7. *We have* $\mathcal{F}^{BIL} \subseteq \operatorname{proj}\left(\mathcal{F}^{RP4}\right)$.

Proof. Let $(x, p, z) \in \mathcal{F}^{\text{BIL}}$ and assume without loss of generality that $0 = \hat{c}_0 \leq \hat{c}_1 \leq \ldots \leq \hat{c}_n$ holds. First, consider the case in which we have $z \leq \hat{c}_n$. Then there exists an index $j \in [n-1]_0$ and a value $\varepsilon \in [0, 1]$ with $z = \varepsilon \hat{c}_j + (1 - \varepsilon)\, \hat{c}_{j+1}$. We define $\lambda_k = 0$ for $k \notin \{j, j+1\}$ and $\lambda_j = \varepsilon$ as well as $\lambda_{j+1} = 1 - \varepsilon$. Furthermore, we set $\omega_i^k = \lambda^k x_i$ for all $i \in [n]\,, k \in [n]_0$ and show that $(x, p, z, \omega, \lambda) \in \mathcal{F}^{\text{RP4}}$. The first five constraints of formulation $\mathcal{F}^{\text{RP4}}$ are trivially satisfied by the definition of ε, λ, and ω. For the last constraint, we have

$$\begin{aligned}
\sum_{k \in [n]_0} (\hat{c}_i - \hat{c}_k)^+ \omega_i^k &= (\hat{c}_i - \hat{c}_j)^+ \varepsilon x_i + (\hat{c}_i - \hat{c}_{j+1})^+ (1 - \varepsilon)\, x_i \\
&\overset{(*)}{=} \left((\hat{c}_i - \hat{c}_j)\, \varepsilon + (\hat{c}_i - \hat{c}_{j+1})\, (1 - \varepsilon)\right)^+ x_i \\
&= (\hat{c}_i - z)^+ x_i \\
&\leq p_i
\end{aligned}$$

for all $i \in [n]$, where equality $(*)$ holds since $(\hat{c}_i - \hat{c}_j)$ and $(\hat{c}_i - \hat{c}_{j+1})$ are either both non-positive if we have $i \leq j$ or both non-negative if we have $i \geq j + 1$.

For the case $z > \hat{c}_n$, we define $\lambda_k = 0$ for $k \in [n-1]_0$ and $\lambda_n = 1$ as well as $\omega_i^k = \lambda^k x_i$ for all $i \in [n]$ and $k \in [n]_0$. Again, $(x, p, z, \omega, \lambda)$ satisfies the first five constraints trivially. Furthermore, we have

$$\sum_{k \in [n]_0} (\hat{c}_i - \hat{c}_k)^+ \omega_i^k = (\hat{c}_i - \hat{c}_n)^+ \omega_i^n = 0 \leq p_i$$

and thus $(x, p, z, \omega, \lambda) \in \mathcal{F}^{\text{RP4}}$, which completes the proof. □

It directly follows that the bilinear formulation also preserves the integrality gap of the nominal formulation.

Corollary 8. *If there exists an* $\alpha \geq 0$ *such that the integrality gap of NOM is bounded by* α *for all objective coefficients* $c \in \mathbb{R}^n$*, then the integrality gap of BIL is also bounded by* α *for all* $c, \hat{c} \in \mathbb{R}^n$ *and* $\Gamma \geq 0$.

Although formulation $\mathcal{F}^{\text{BIL}}$ is strong and compact, its bilinearity is rather hindering when solving instances in practice. Nevertheless, this formulation will be the theoretical foundation for the approaches that we develop in this as well as the following chapter.

4.2 Recycled Inequalities

In this section, we show that structural properties encoded in valid inequalities for the nominal problem can be used to carry over the strength of the bilinear formulation $\mathcal{F}^{\text{BIL}}$ to a linear formulation.

Multiplying linear inequalities with variables as an intermediate step in order to achieve a stronger linear formulation is not a new approach. For the Reformulation-Linearization-Technique by Sherali and Adams [89], one multiplies constraints with variables and linearizes the resulting products afterwards via substitution with auxiliary variables. When taken to the extreme, where all constraints are multiplied with all possible combinations of variables, one obtains a formulation with exponentially many variables and constraints, whose projection on the original variables equals the convex hull of integer-feasible solutions. Our approach is different in the sense that we don't directly linearize the bilinear inequalities, and thus don't create auxiliary variables. Instead, we combine several of the bilinear inequalities in order to estimate the non-linear terms against a linear term, using a valid inequality for the corresponding nominal problem.

Theorem 9. *Let $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ be a valid inequality for $\mathcal{C}^{NOM}$ with $\pi \in \mathbb{R}^{n+1}_{\geq 0}$. Then the inequality*

$$\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i \tag{4.1}$$

is valid for $\mathcal{C}^{ROB}$.

Proof. Summing the bilinear constraints $p_i + x_i z \geq \hat{c}_i x_i$, each with a weight of π_i, we obtain

$$\sum_{i\in[n]} \pi_i p_i + \sum_{i\in[n]} \pi_i x_i z \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i,$$

which is a valid inequality for $\mathcal{C}^{\text{ROB}}$ due to $\pi \geq 0$. Now, since $z \geq 0$ holds, we have $\sum_{i\in[n]} \pi_i x_i z \leq \pi_0 z$, and thus the validity of (4.1). □

As we reuse the valid inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ to strengthen the formulation $\mathcal{F}^{\text{ROB}}$, we call inequality (4.1) the *recycled inequality* of $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$. In accordance with the requirements of Theorem 9, we call $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ *recyclable* if it is valid for $\mathcal{C}^{\text{NOM}}$ and $\pi \geq 0$.

In the following, we will only consider nominal inequalities $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ consisting exclusively of variables with uncertain objective coefficients, i.e., $\pi_i = 0$ for all $i \in [n]$ with $\hat{c}_i = 0$. We call inequalities and their corresponding coefficients π with this property *uncertainty-exclusive*. Note that uncertainty-exclusive inequalities are the only interesting ones for recycling, since we can always recycle $\sum_{i\in[n]\setminus\{j\}} \pi_i x_i \leq \pi_0$ when $\hat{c}_j = 0$ holds. While this inequality is weaker than $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ for the nominal problem, the corresponding

recycled inequality is stronger. This is because we remove $\pi_i p_i$ from the left-hand side while the right-hand side does not change due to $\pi_i \hat{c}_i x_i = 0$. By focusing on uncertainty-exclusive inequalities, we obtain the following statement.

Proposition 10. *Let $\pi \in \mathbb{R}^{n+1}$ be uncertainty-exclusive. If $\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i$ is valid for $\mathcal{C}^{ROB}$, then $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is a recyclable inequality.*

Proof. First, note that the validity of $\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i$ already implies $\pi \geq 0$, since p and z are unbounded while the right-hand side $\sum_{i\in[n]} \pi_i \hat{c}_i x_i$ is not. Second, note that $\hat{c}_i x_i = 0$ implies $\pi_i x_i = 0$ for all $i \in [n]$ due to the uncertainty-exclusiveness. Assume that $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is invalid, i.e., there exists a vector $\tilde{x} \in \mathcal{C}^{\text{NOM}}$ with $\sum_{i\in[n]} \pi_i \tilde{x}_i > \pi_0$. Then there exists an index $i \in [n]$ with $\hat{c}_i \tilde{x}_i > 0$, as otherwise $\pi_0 < \sum_{i\in[n]} \pi_i \tilde{x}_i = 0$. We define $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{C}^{\text{ROB}}$ with $\tilde{z} = \min\{\hat{c}_i | i \in [n], \hat{c}_i \tilde{x}_i > 0\}$ as well as $\tilde{p}_i = (\hat{c}_i - \tilde{z})^+ \tilde{x}_i$ for all $i \in [n]$. Then we have $\tilde{z} > 0$ and $\pi_i \tilde{p}_i = \pi_i (\hat{c}_i - \tilde{z}) \tilde{x}_i$ for all $i \in [n]$, which implies

$$\pi_0 \tilde{z} + \sum_{i\in[n]} \pi_i \tilde{p}_i = \tilde{z} \left(\pi_0 - \sum_{i\in[n]} \pi_i \tilde{x}_i \right) + \sum_{i\in[n]} \pi_i \hat{c}_i \tilde{x}_i < \sum_{i\in[n]} \pi_i \hat{c}_i \tilde{x}_i,$$

and thus proves that $\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i$ cannot be valid. □

The above shows that we can actually obtain all non-dominated valid inequalities of the form $\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i$ by recycling a valid nominal inequality. We also get a first understanding of the strength of the recycling approach. If there exists an inequality of the same form as the recycled inequality (4.1) that is stronger than the recycled one, then this does not mean that the recycling procedure is weak but that there exists a better nominal inequality to recycle.

In order to see in which cases recycled inequalities are useful, let us consider how they compare to the bilinear inequalities over the course of their construction. First, note that the sum of the bilinear inequalities is weaker than the bilinear inequalities themselves. Hence, when separating a recycled inequality to cut off a fractional solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{NOM}}$, our inequality to be recycled should only support indices $i \in [n]$ with $\pi_i > 0$ for which the bilinear inequality $\tilde{p}_i + \tilde{x}_i \tilde{z} \geq \hat{c}_i \tilde{x}_i$ is violated or tight. A second potential weakening occurs when applying the estimation $\sum_{i\in[n]} \pi_i x_i z \leq \pi_0 z$. This implies that recycling $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is especially interesting if it is binding for $(\tilde{x}, \tilde{p}, \tilde{z})$.

Reconsider Example 6, for which we can recycle the valid inequality $\sum_{i\in[n]} x_i \leq 1$ implied by $\sum_{i\in[n]} x_i = 1$. The recycled inequality $z + \sum_{i\in[n]} p_i \geq \sum_{i\in[n]} \frac{n-1}{n} x_i$ yields $z + \sum_{i\in[n]} p_i \geq \frac{n-1}{n}$, and thus the optimal objective value of the continuous relaxation is now equal to the optimal integer objective value. This intuitively highlights the strength of the recycled inequalities in the case where both properties, a binding recyclable valid inequality and the violation of corresponding bilinear inequalities, coincide. In the next section, we will investigate the strength of recycled inequalities from a polyhedral point of view.

4.3 Facet-Defining Recycled Inequalities

In this section, we show that recycled inequalities often define facets of the convex hull of the robust problem $\mathcal{C}^{\text{ROB}}$. Recall that facet-defining inequalities are the best inequalities to describe a polyhedron $P \subseteq \mathbb{R}^n$ and are characterized via the dimension of their face $F(\pi) = \left\{x \in P \middle| \sum_{i\in[n]} \pi_i x = \pi_0\right\}$. That is, $\dim(F(\pi)) = \dim(P) - 1$ holds for a facet-defining inequality $\sum_{i\in[n]} \pi_i x \leq \pi_0$ (cf. Section 2.4.5). Consequently, in order to prove that recycled inequalities can be facet-defining, we first determine the dimension of $\mathcal{C}^{\text{ROB}}$. For the sake of simplicity, we assume for the rest of this chapter that the solution sets $\mathcal{C}^{\text{NOM}}$ and $\mathcal{C}^{\text{ROB}}$ are non-empty.

Lemma 11. *We have* $\dim\left(\mathcal{C}^{ROB}\right) = \dim\left(\mathcal{C}^{NOM}\right) + n + 1$.

Proof. According to Proposition 5, the number $n - \dim(P)$ equals the maximum number of linearly independent equations that are met by all elements in a polyhedron $P \subseteq \mathbb{R}^n$. Let $\sum_{i\in[n]} (\omega_i x_i + \omega_{n+i} p_i) + \omega_{2n+1} z = \omega_0$ be satisfied by all $(x, p, z) \in \mathcal{C}^{\text{ROB}}$. Since p and z can be raised arbitrarily and $\mathcal{C}^{\text{ROB}} \neq \emptyset$, we have $\omega_{n+1} = \cdots = \omega_{2n+1} = 0$, and thus $\sum_{i\in[n]} \omega_i x_i = \omega_0$. Hence, the equations that are met by all $(x, p, z) \in \mathcal{C}^{\text{ROB}}$ are exactly the equations that are met by all $x \in \mathcal{C}^{\text{NOM}}$, which implies

$$\dim\left(\mathcal{C}^{\text{ROB}}\right) = 2n + 1 - \left(n - \dim\left(\mathcal{C}^{\text{NOM}}\right)\right) = \dim\left(\mathcal{C}^{\text{NOM}}\right) + n + 1. \qquad \square$$

Knowing the dimension of $\mathcal{C}^{\text{ROB}}$, we are now able to study facet-defining recycled inequalities. Remember that we only consider uncertainty-exclusive valid inequalities, as these are the only inequalities yielding non-dominated recycled inequalities. The following theorem states the dimension of a recycled inequality's face based on the projection of the nominal inequality's face on the supported variables. Together with Lemma 11, this yields a characterization of facet-defining recycled inequalities.

Theorem 12. *Let* $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ *be a recyclable, uncertainty-exclusive inequality and let* $S = \{i \in [n] | \pi_i > 0\}$. *Then the face of the recycled inequality (4.1) has dimension*

$$\dim\left(\text{proj}_S\left(F(\pi)\right)\right) + \dim\left(\mathcal{C}^{NOM}\right) + 1 + n - |S|,$$

where $F(\pi)$ *is the face with respect to* $\mathcal{C}^{NOM}$ *and* proj_S *is the projection on the supported variables* $\{x_i | i \in S\}$. *Hence, the recycled inequality (4.1) is facet-defining for* $\mathcal{C}^{ROB}$ *if and only if*

$$\dim\left(\text{proj}_S\left(F(\pi)\right)\right) = |S| - 1$$

holds.

Proof. There always exist $\dim\left(\mathcal{C}^{\text{NOM}}\right)+1+n-|S|$ affinely independent vectors $(x,p,z)\in\mathcal{C}^{\text{ROB}}$ satisfying (4.1) with equality. To see this, let $\left\{x^0,\ldots,x^{\dim\left(\mathcal{C}^{\text{NOM}}\right)}\right\}\subseteq\mathcal{C}^{\text{NOM}}$ be affinely independent. For each $j\in\left[\dim\left(\mathcal{C}^{\text{NOM}}\right)\right]_0$, we choose $(x^j,\hat{c}\odot x^j,0)$, where $\hat{c}\odot x^j$ refers to the component-wise multiplication, i.e., $\left(\hat{c}\odot x^j\right)_i=\hat{c}_i x_i^j$. By definition, $(x^j,\hat{c}\odot x^j,0)$ is within $\mathcal{C}^{\text{ROB}}$ and we have

$$\pi_0 0+\sum_{i\in[n]}\pi_i\left(\hat{c}\odot x^j\right)_i=\sum_{i\in[n]}\pi_i\hat{c}_i x_i^j.$$

Additionally, we choose $(x^0,\hat{c}\odot x^0+e^j,0)$ for each $j\in[n]\setminus S$, with $e^j\in\mathbb{R}^n$ being the j-th unit vector. Again, this vector is within $\mathcal{C}^{\text{ROB}}$ and due to $\pi_j=0$ it follows

$$\pi_0 0+\sum_{i\in[n]}\pi_i\left(\hat{c}\odot x^0+e^j\right)_i=\pi_j+\sum_{i\in[n]}\pi_i\hat{c}_i x_i^0=\sum_{i\in[n]}\pi_i\hat{c}_i x_i^0.$$

We extend the $\dim\left(\mathcal{C}^{\text{NOM}}\right)+1+n-|S|$ vectors above with a set of additional vectors $\{(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)|j\in[k]\}\subseteq\mathcal{C}^{\text{ROB}}$ for some $k\in\mathbb{Z}_{\geq 0}$. We say that such an *extension* is *valid* if the additional vectors satisfy the recycled inequality (4.1) with equality and are affinely independent together with the vectors above. Let $k'\in\mathbb{Z}_{\geq 0}$ be the maximum number of vectors in a valid extension. In order to prove the theorem, we show that $k'=\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)+1$ holds. For this, we can restrict ourselves without loss of generality to extensions with binary $\tilde{x}^j$. If $\tilde{x}^j$ is not binary, then $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$ is a convex combination of integer-feasible solutions within $\mathcal{C}^{\text{ROB}}$. Convex combinations are a special case of affine combinations, and thus one of these integer-feasible solutions must be affinely independent and can replace $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$. Otherwise, $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$ itself would not be affinely independent.

To show $k'=\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)+1$, we claim that an extension $\{(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)|j\in[k]\}$ is valid if and only if the following four properties hold:

1. $\tilde{p}_i^j=\left(\hat{c}_i-\tilde{z}^j\right)\tilde{x}_i^j$ for all $j\in[k]$, $i\in S$,
2. $\tilde{z}^j>0$ for all $j\in[k]$,
3. $\{\text{proj}_S\left(\tilde{x}^j\right)|j\in[k]\}$ are affinely independent,
4. $\{\tilde{x}^j|j\in[k]\}\subseteq F\left(\pi\right)$.

Assume that the claim is true. Then properties 3 and 4 imply $k'\leq\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)+1$. In order to prove $k'\geq\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)+1$, let $\{\tilde{x}^j|j\in[\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)]_0\}\subseteq F\left(\pi\right)$ be affinely independent in the components $\{x_i|i\in S\}$. These vectors fulfill properties 3 and 4, and thus it suffices to construct $\tilde{p}^j,\tilde{z}^j$ for each $j\in[\dim\left(\text{proj}_S\left(F\left(\pi\right)\right)\right)]_0$ such that $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$ satisfies properties 1 and 2. We choose $\tilde{z}^j=\min\left\{\hat{c}_i|i\in[n],\hat{c}_i>0\right\}$ and $\tilde{p}_i^j=\left(\hat{c}_i-\tilde{z}\right)^+\tilde{x}_i^j$ for all $i\in[n]$. Then $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$ is by definition within $\mathcal{C}^{\text{ROB}}$ and satisfies $\tilde{z}^j>0$. Since π is uncertainty-exclusive, we have $\pi_i=0$ for all $\hat{c}_i<\tilde{z}^j$, and thus $\tilde{p}_i^j=\left(\hat{c}_i-\tilde{z}^j\right)\tilde{x}_i^j$ for all $i\in S$. Therefore, all $(\tilde{x}^j,\tilde{p}^j,\tilde{z}^j)$ satisfy properties 1 to 2, and thus constitute a valid extension.

It remains to show the equivalency between the validity of extensions and the properties 1 to 4. For this, let $\{(\tilde{x}^j, \tilde{p}^j, \tilde{z}^j) | j \in [k]\}$ be a valid extension of arbitrary size $k \in \mathbb{Z}_{\geq 0}$. Since we can assume $\tilde{x}^j \in \{0,1\}^n$ for all $j \in [k]$, we have $\tilde{p}_i^j \geq (\hat{c}_i - \tilde{z}^j)\, \tilde{x}_i^j$ for all $i \in [n]$. If property 1 did not hold, then there would exist indices $j \in [k]$ and $i \in S$ with $\tilde{p}_i^j > (\hat{c}_i - \tilde{z}^j)\, \tilde{x}_i^j$, and thus $\pi_i \tilde{p}_i^j > \pi_i\, (\hat{c}_i - \tilde{z}^j)\, \tilde{x}_i^j$, since $\pi_i > 0$ holds for $i \in S$. This is a contradiction due to

$$\begin{aligned}
\pi_0 \tilde{z}^j + \sum_{i \in [n]} \pi_i \tilde{p}_i^j &> \pi_0 \tilde{z}^j + \sum_{i \in [n]} \pi_i \left(\hat{c}_i - \tilde{z}^j\right) \tilde{x}_i^j \\
&\geq \sum_{i \in [n]} \pi_i \tilde{x}_i^j \tilde{z}^j + \sum_{i \in [n]} \pi_i \left(\hat{c}_i - \tilde{z}^j\right) \tilde{x}_i^j \\
&= \sum_{i \in [n]} \pi_i \hat{c}_i \tilde{x}_i^j.
\end{aligned}$$

The affine independency of the extension is equivalent to the linear independency of the $\dim\left(\mathcal{C}^{\text{NOM}}\right) + n - |S| + k$ vectors obtained when subtracting $(x^0, \hat{c} \odot x^0, 0)$ of all others. We write these into the following matrix, using property 1 and with indices ordered such that $S = \{1, \ldots, |S|\}$

$$\left(\begin{array}{ccc|ccc|ccc}
\ldots & x_1^j - x_1^0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{x}_1^j - x_1^0 & \ldots \\
 & \vdots & & \vdots & & \vdots & & \vdots & \\
\ldots & x_n^j - x_n^0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{x}_n^j - x_n^0 & \ldots \\
\hline
\ldots & \hat{c}_1 \left(x_1^j - x_1^0\right) & \ldots & 0 & \ldots & 0 & \ldots & \left(\hat{c}_1 - \tilde{z}^j\right) \tilde{x}_1^j - \hat{c}_1 x_1^0 & \ldots \\
 & \vdots & & \vdots & & \vdots & & \vdots & \\
 & \vdots & & 0 & \ldots & 0 & & \left(\hat{c}_{|S|} - \tilde{z}^j\right) \tilde{x}_{|S|}^j - \hat{c}_{|S|} x_{|S|}^0 & \\
 & \vdots & & 1 & & 0 & & \tilde{p}_{|S|+1}^j - \hat{c}_{|S|+1} x_{|S|+1}^0 & \\
 & \vdots & & & \ddots & & & \vdots & \\
\ldots & \hat{c}_n \left(x_n^j - x_n^0\right) & \ldots & 0 & & 1 & \ldots & \tilde{p}_n^j - \hat{c}_n x_n^0 & \ldots \\
\hline
\ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{z}^j & \ldots
\end{array}\right).$$

For each $i \in [n]$, we subtract row i from row $n + i$ with factor $\hat{c}_i$ and obtain

$$\left(\begin{array}{ccc|ccc|ccc}
\ldots & x_1^j - x_1^0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{x}_1^j - x_1^0 & \ldots \\
 & \vdots & & \vdots & & \vdots & & \vdots & \\
\ldots & x_n^j - x_n^0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{x}_n^j - x_n^0 & \ldots \\
\hline
\ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots & -\tilde{z}^j \tilde{x}_1^j & \ldots \\
 & \vdots & & \vdots & & \vdots & & \vdots & \\
 & \vdots & & 0 & \ldots & 0 & & -\tilde{z}^j \tilde{x}_{|S|}^j & \\
 & \vdots & & 1 & & 0 & & \tilde{p}_{|S|+1}^j - \hat{c}_{|S|+1} \tilde{x}_{|S|+1}^j & \\
 & \vdots & & & \ddots & & & \vdots & \\
\ldots & 0 & \ldots & 0 & & 1 & \ldots & \tilde{p}_n^j - \hat{c}_n \tilde{x}_n^j & \ldots \\
\hline
\ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{z}^j & \ldots
\end{array}\right).$$

We use the middle columns containing the unit vectors to eliminate the corresponding rows. Furthermore, since $\tilde{x}^j - x^0$ is linearly dependent of $\left\{x^1 - x^0, \ldots, x^{\dim(\mathcal{C}^{\text{NOM}})} - x^0\right\}$ due to the dimension of $\mathcal{C}^{\text{NOM}}$, we can eliminate the first n rows in the last k columns and obtain

$$\left(\begin{array}{ccc|ccc|ccc} \ldots & x_1^j - x_1^0 & \ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots \\ & \vdots & & \vdots & & \vdots & & \vdots & \\ \ldots & x_n^j - x_n^0 & \ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots \\ \hline \ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots & -\tilde{z}^j \tilde{x}_1^j & \ldots \\ & \vdots & & \vdots & & \vdots & & \vdots & \\ & \vdots & & 0 & \ldots & 0 & & -\tilde{z}^j \tilde{x}_{|S|}^j & \\ & \vdots & & 1 & & 0 & & 0 & \\ & \vdots & & & \ddots & & & \vdots & \\ \ldots & 0 & \ldots & 0 & & 1 & \ldots & 0 & \ldots \\ \hline \ldots & 0 & \ldots & 0 & \ldots & 0 & \ldots & \tilde{z}^j & \ldots \end{array}\right).$$

These columns are linearly independent if and only if $\tilde{z}^j \neq 0$ holds for all $j \in [k]$ and $\{\text{proj}_S(\tilde{x}^j) | j \in [k]\}$ are affinely independent. Thus, if property 1 holds, then the affine independency of the extension is equivalent to properties 2 and 3. Since we have already shown above that property 1 is implied, we know that the validity of the extension implies properties 1 to 3. Conversely, properties 1 to 3 imply the affine independency of the extension. Furthermore, given properties 1 and 2, we have

$$\pi_0 \tilde{z}^j + \sum_{i \in [n]} \pi_i \tilde{p}_i^j = \sum_{i \in [n]} \pi_i \hat{c}_i \tilde{x}_i^j \;\Leftrightarrow\; \pi_0 \tilde{z}^j = \sum_{i \in [n]} \pi_i \tilde{z}^j \tilde{x}_i^j \;\Leftrightarrow\; \pi_0 = \sum_{i \in [n]} \pi_i \tilde{x}_i^j,$$

and thus $\{\tilde{x}^j | j \in [k]\} \subseteq F(\pi)$. Hence, property 4 holds if and only if $\{(\tilde{x}^j, \tilde{p}^j, \tilde{z}^j) | j \in [k]\}$ fulfill the recycled inequality (4.1) with equality. In summary, this shows that the validity of the extension is equivalent to properties 1 to 4. □

A powerful implication of Theorem 12 is that recycling an uncertainty-exclusive inequality yields always a facet-defining inequality if $\dim(F(\pi)) = n - 1$ holds. This is because there exist n affinely independent vectors satisfying $\sum_{i \in [n]} \pi_i x = \pi_0$, of which $|S|$ must be affinely independent when projected on the variables $\{x_i | i \in S\}$. Note that $\dim(F(\pi)) = n - 1$ holds if $F(\pi)$ is either a facet of a full-dimensional polyhedron $\mathcal{C}^{\text{NOM}}$ or if $\sum_{i \in [n]} \pi_i x \leq \pi_0$ is actually an equation with $F(\pi) = \mathcal{C}^{\text{NOM}}$ and $\dim\left(\mathcal{C}^{\text{NOM}}\right) = n - 1$. This is summarized in the following corollary.

Corollary 13. *Let $\sum_{i \in [n]} \pi_i x_i \leq \pi_0$ be a recyclable, uncertainty-exclusive inequality. The recycled inequality (4.1) is facet-defining for $\mathcal{C}^{ROB}$ if one of the following holds:*

- *$\mathcal{C}^{NOM}$ is full-dimensional and $F(\pi)$ is a facet of $\mathcal{C}^{NOM}$,*
- *$\dim\left(\mathcal{C}^{NOM}\right) = n - 1$ and $F(\pi) = \mathcal{C}^{NOM}$.*

Contrary to first intuition, it is also possible to obtain facet-defining inequalities by recycling weaker inequalities that are neither facet-defining nor equations. This is because the dimension of the face $F(\pi)$ can shrink by less than $n - |S|$ when projected on $\{x_i | i \in S\}$. Thus, inequalities defining low-dimensional faces can also yield facet-defining recycled inequalities if $\pi_i = 0$ holds for many $i \in [n]$. For example, consider an independent set problem on a graph with nodes $V = [n]$. For some independent set $I \subseteq V$ and an index $i \in V$, let $x_i = 1$ if and only if $i \in I$. If $Q \subseteq V$ is a clique, then the *clique inequality* $\sum_{i \in Q} x_i \leq 1$ is valid for all independent sets. Furthermore, the inequality strictly dominates all inequalities $\sum_{i \in Q'} x_i \leq 1$ with $Q' \subsetneq Q$ and it is facet-defining if and only if Q is a maximal clique with respect to inclusion [37, Section 3.9]. However, if $\hat{c}_i > 0$ holds for all $i \in Q$, then the recycled inequality $z + \sum_{i \in Q'} p_i \geq \sum_{i \in Q'} \hat{c}_i x_i$ defines a facet of the corresponding $\mathcal{C}^{\text{ROB}}$ for all cliques $Q' \subseteq Q$. This is because the projection $\text{proj}_{Q'}(F(\pi))$, that is $S = Q'$, contains $\{e^j | j \in Q'\}$ and thus has dimension $|Q'| - 1$.

Note that the recycling of clique inequalities not only yields facet-defining inequalities for the independent set problem but for arbitrary problems ROB as long as $x_i = 0$ is not valid for $\mathcal{C}^{\text{NOM}}$ for some $i \in Q$. Other examples of interesting classical inequalities include odd hole inequalities for the independent set problem [81] and minimal cover inequalities for the knapsack problem [37, Section 7.1]. These are in general not facet-defining for their respective nominal problems but yield facet-defining recycled inequalities for the robust counterpart.

One now might raise the question whether the recycling of dominated inequalities is actually of practical interest. After all, the resulting inequalities might not matter due to the special structure of the objective function, with all p_i having an objective coefficient of 1. The following example demonstrates that it can be beneficial to weaken an inequality before it is recycled.

Example 14. Consider the robust problem

$$\begin{aligned} \min\ & 2z + \sum_{i \in [5]} -x_i + p_i \\ \text{s.t. } & 3x_5 + \sum_{i \in [4]} x_i \leq 3 \\ & z + p_i \geq x_i && \forall i \in [5] \\ & x \in \{0,1\}^5, p \in \mathbb{R}^5_{\geq 0}, z \in \mathbb{R}_{\geq 0}. \end{aligned}$$

Choosing $x = \left(\frac{3}{4}, \ldots, \frac{3}{4}, 0\right)$, $p = 0$, and $z = \frac{3}{4}$ yields an optimal solution for the continuous relaxation with objective value $-\frac{3}{2}$. The constraint $3x_5 + \sum_{i \in [4]} x_i \leq 3$ can be recycled to $3z + 3p_5 + \sum_{i \in [4]} p_i \geq 3x_5 + \sum_{i \in [4]} x_i$. After adding the inequality, an optimal solution is given by $x = \left(\frac{3}{4}, \ldots, \frac{3}{4}, 0\right)$, $p = \left(0, \ldots, 0, \frac{1}{4}\right)$, and $z = \frac{3}{4}$, with an objective value of $-\frac{5}{4}$. Note that we now choose $p_5 > 0$ even though $x_5 = 0$ holds. Since the bilinear inequality $p_5 + x_5 z \geq \hat{c}_5 x_5$ now has a slack of $\frac{1}{4}$, our observation from the last section suggests that it may be beneficial

to drop x_5 from the valid inequality for recycling. In fact, when recycling the dominated inequality $\sum_{i\in[4]} x_i \leq 3$ instead of the model constraint, we obtain $3z + \sum_{i\in[4]} p_i \geq \sum_{i\in[4]} x_i$ and an optimal solution is now given by $x = (1, 1, 1, 0, 0)$, $p = 0$, and $z = 1$, which yields an objective value of -1. Hence, by only recycling the dominated inequality, we are able to strengthen the formulation such that an optimal integer solution is also an optimal continuous solution.

After discussing the strong implications of Theorem 12, let us now consider its limitations. The next example shows that it is indeed necessary that the inequality to be recycled is uncertainty-exclusive.

Example 15. Consider the full-dimensional polyhedron

$$\mathcal{C}^{\text{ROB}} = \operatorname{conv}\left(\left\{(x,p,z) \in \{0,1\}^3 \times \mathbb{R}^4_{\geq 0} \middle| \begin{array}{r} x_1 + x_2 + x_3 \leq 2 \\ z + p_i \geq \hat{c}_i x_i \quad \forall i \in [3] \end{array}\right\}\right),$$

with $\hat{c}_1 = \hat{c}_2 = 1$ and $\hat{c}_3 = 0$. The constraint $x_1 + x_2 + x_3 \leq 2$ is facet-defining for the corresponding $\mathcal{C}^{\text{NOM}}$, and thus meets all requirements of Corollary 13 with the exception that it is not uncertainty-exclusive. Indeed, the recycled inequality $2z+p_1+p_2+p_3 \geq x_1+x_2$ is not facet-defining, as it is dominated by the sum of the constraints $z + p_1 \geq x_1$ and $z + p_2 \geq x_2$. Note that this does not change when recycling the corresponding uncertainty-exclusive inequality $x_1 + x_2 \leq 2$ instead.

The observation in the example above is quite intuitive. While we can always transform an arbitrary recyclable inequality into an uncertainty-exclusive one by dropping all x_i with $\hat{c}_i = 0$, we loose information during this process and cannot expect to obtain a facet of $\mathcal{C}^{\text{ROB}}$. Less obvious is the importance of the dimension of the nominal polyhedron $\mathcal{C}^{\text{NOM}}$, which is a prerequisite for Corollary 13. In the following, we study lower-dimensional problems, for which we first consider another example.

Example 16. Consider the four-dimensional polyhedron with five variables

$$\mathcal{C}^{\text{ROB}} = \operatorname{conv}\left(\left\{(x,p,z) \in \{0,1\}^2 \times \mathbb{R}^3_{\geq 0} \middle| \begin{array}{r} x_1 + x_2 = 1 \\ z + p_i \geq x_i \quad \forall i \in [2] \end{array}\right\}\right).$$

The inequality $2x_1 + x_2 \leq 2$ defines a facet for the corresponding $\mathcal{C}^{\text{NOM}}$. However, the recycled inequality $2z+2p_1+p_2 \geq 2x_1+x_2$ is not facet-defining for $\mathcal{C}^{\text{ROB}}$, as it is the sum of $z+p_1 \geq x_1$ and $z + p_1 + p_2 \geq x_1 + x_2$, where the latter is recycled from $x_1 + x_2 = 1$.

The example shows that in the lower dimensional case, inequalities recycled from facet-defining inequalities are not necessarily facet-defining. Note that the mapping from the set of recyclable inequalities to the corresponding recycled inequalities is a homomorphism in the sense that the recycled inequality of $\sum_{i\in[n]} (\pi^1_i + \pi^2_i)\, x_i \leq \pi^1_0 + \pi^2_0$ equals the sum of the

recycled inequalities of $\sum_{i\in[n]} \pi_i^1 x_i \leq \pi_0^1$ and $\sum_{i\in[n]} \pi_i^2 x_i \leq \pi_0^2$. As recycled inequalities are proper inequalities, their sum is weaker than the separate inequalities and it is better to recycle each inequality individually. This applies to the example, where $2x_1 + x_2 \leq 2$ is the sum of the recyclable inequality $x_1 \leq 1$ and the recyclable equation $x_1 + x_2 = 1$. Although both inequalities have the same face, recycling $x_1 \leq 1$ yields a facet-defining inequality, while recycling $2x_1 + x_2 \leq 2$ does not. Hence, for lower dimensional $\mathcal{C}^{\text{NOM}}$, one cannot decide whether recycling yields a facet-defining inequality by solely relying on the recyclable inequality's face.

However, we can eliminate equations from an inequality and recycle the resulting one, which is equivalent for $\mathcal{C}^{\text{NOM}}$. We call two inequalities $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$ and $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ *equivalent for a polyhedron* P if there exist equations $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$, satisfied by all $x \in P$, such that $\left\{\pi', \omega^1, \ldots, \omega^\ell\right\}$ are linearly independent and $\lambda_0 \pi' + \sum_{k\in[\ell]} \lambda_k \omega^k = \pi$ holds for some $\lambda_0 > 0$ and $\lambda_k \in \mathbb{R}$ with $k \in [\ell]$ [96, Section I.4.3].

Algorithmus 1 : Procedure for eliminating equations from π.

Input : A recyclable, uncertainty-exclusive inequality $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$ and equations $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$ for $k \in [\ell]$, satisfied by all $x \in \mathcal{C}^{\text{NOM}}$, such that $\left\{\pi', \omega^1, \ldots, \omega^\ell\right\}$ are linearly independent

Output : An equivalent recyclable, uncertainty-exclusive inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ with $|\{i \in [n] | \pi_i = 0\}| \geq \ell$

1 Set $\pi = \pi'$
2 **for** $k \in [\ell]$ **do**
3 Choose $i^* \in \operatorname{argmin} \left\{ \left| \frac{\pi_i}{\omega_i^k} \right| \middle| i \in [n], \omega_i^k \neq 0, \right\}$
4 Update $\pi \leftarrow \pi - \frac{\pi_{i^*}}{\omega_{i^*}^k} \omega^k$
5 **for** $k' \in \{k+1, \ldots, \ell\}$ **do**
6 Update $\omega^{k'} \leftarrow \omega^{k'} - \frac{\omega_{i^*}^{k'}}{\omega_{i^*}^k} \omega^k$
7 **return** $\sum_{i\in[n]} \pi_i x \leq \pi_0$

We use Algorithm 1 to transform a recyclable inequality $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$ into an equivalent inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ that satisfies the conditions of Theorem 12. For given equations $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$, the algorithm performs a special Gaussian elimination on $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$. We already noted above that having many zero coefficients is beneficial for obtaining facet-defining recycled inequalities. Therefore, for each equation given by ω^k, Algorithm 1 subtracts in line 4 a multiple $\frac{\pi_{i^*}}{\omega_{i^*}^k}$ of ω^k from π such that the coefficient π_{i^*} becomes or stays zero. The index i^* is chosen with respect to a bottleneck condition in line 3, which ensures that $\frac{\pi_{i^*}}{\omega_{i^*}^k}$ is the multiple with the smallest absolute value for all $i \in [n]$ with $\omega_i^k \neq 0$. This has two desirable implications. First, if π_i was positive before, then it will be non-negative after subtracting $\frac{\pi_{i^*}}{\omega_{i^*}^k} \omega_i^k$. Second, if π_i was already zero before, then it will still be zero afterwards. Hence, if $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$ is recyclable and uncertainty-exclusive, then this also holds for the resulting $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$. After eliminating π_{i^*}, we make sure in line 6 that $\omega_{i^*}^k = 0$ holds

for all remaining equations such that i^* will be different in every iteration. By doing so, we guarantee that we have at least ℓ indices $i \in [n]$ with $\pi_i = 0$ at the end of Algorithm 1.

The following proposition uses Algorithm 1 to generalize Corollary 13 for lower-dimensional problems.

Proposition 17. *Let $\sum_{i\in[n]} \pi'_i x_i \leq \pi'_0$ be a recyclable, uncertainty-exclusive inequality such that $F(\pi)$ is either a facet of $\mathcal{C}^{NOM}$ or $F(\pi) = \mathcal{C}^{NOM}$. Let furthermore ℓ be the maximum number of equations $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$ for $k \in [\ell]$, satisfied by all $x \in \mathcal{C}^{NOM}$, such that $\left\{\pi', \omega^1, \ldots, \omega^\ell\right\}$ are linearly independent. Then Algorithm 1 computes an equivalent recyclable inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ whose recycled inequality (4.1) is facet-defining for $\mathcal{C}^{ROB}$.*

Proof. Since the returned inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ from Algorithm 1 is recyclable and uncertainty-exclusive, it only remains to show that $\dim(\mathrm{proj}_S(F(\pi))) = |S| - 1$ holds for $S = \{i \in [n] | \pi_i > 0\}$ according to Theorem 12. Due to the choice of ℓ, we have

$$\ell = \begin{cases} n - \dim\left(\mathcal{C}^{\text{NOM}}\right) - 1, & \text{if } F(\pi) = \mathcal{C}^{\text{NOM}} \\ n - \dim\left(\mathcal{C}^{\text{NOM}}\right), & \text{otherwise.} \end{cases}$$

Hence, there exists a set $\left\{x^1, \ldots, x^{n-\ell}\right\} \subseteq F(\pi)$ of affinely independent vectors. Let furthermore $T \subseteq [n] \setminus S$ consist of the ℓ indices i^* that were chosen in Algorithm 1. We show that the vectors $\left\{\mathrm{proj}_{[n]\setminus T}(x^1), \ldots, \mathrm{proj}_{[n]\setminus T}\left(x^{n-\ell}\right)\right\}$ are affinely independent, i.e., $\dim\left(\mathrm{proj}_{[n]\setminus T}(F(\pi))\right) \geq n - |T| - 1$. Since $S \subseteq [n]\setminus T$ holds, this implies $\dim(\mathrm{proj}_S(F(\pi))) \geq |S| - 1$. Furthermore, since the equation induced by π is only on the variables $\{x_i | i \in S\}$, we have $\dim(\mathrm{proj}_S(F(\pi))) < |S|$, which then proves the proposition.

Assume that the projections $\mathrm{proj}_{[n]\setminus T}(x^j)$ are not affinely independent. Then there exist coefficients $\lambda \in \mathbb{R}^{n-\ell}$ with $\lambda \neq 0$, $\sum_{j\in[n-\ell]} \lambda_j = 0$, and $\sum_{j\in[n-\ell]} \lambda_j x_i^j = 0$ for all $i \in [n] \setminus T$. Consider a fixed but arbitrary index $i^* \in T$. Since i^* was chosen in Algorithm 1, there exists an equation $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$ with $\omega_{i^*}^k \neq 0$. Without loss of generality, we can assume $\omega_{i^*}^k = 1$ and obtain

$$x_{i^*}^j = \omega_0^k - \sum_{i\in[n]\setminus\{i^*\}} \omega_i^k x_i^j$$

for all $j \in [n - \ell]$, and thus

$$\begin{aligned} \sum_{j\in[n-\ell]} \lambda_j x_{i^*}^j &= \sum_{j\in[n-\ell]} \lambda_j \left(\omega_0^k - \sum_{i\in[n]\setminus\{i^*\}} \omega_i^k x_i^j\right) \\ &= \omega_0^k \underbrace{\sum_{j\in[n-\ell]} \lambda_j}_{=0} - \sum_{i\in[n]\setminus\{i^*\}} \omega_i^k \underbrace{\sum_{j\in[n-\ell]} \lambda_j x_i^j}_{=0} = 0. \end{aligned}$$

However, as this applies for all $i^* \in T$, we have $\sum_{j\in[n-\ell]} \lambda_j x_i^j = 0$ for all $i \in [n]$. This implies that the vectors $\left\{x^1, \ldots, x^{n-\ell}\right\}$ are affinely dependent, which contradicts their choice and completes the proof. □

Note that we do not always know the dimension of $\mathcal{C}^{\text{NOM}}$ in practice, let alone all equations $\sum_{i\in[n]} \omega_i^k x_i = \omega_0^k$. We tested Algorithm 1 using the already present equations in the constraint matrix $Ax \leq b$ of the robust instances generated from the MIPLIB 2017, which we use in our computational study in Section 4.6.7. Interestingly, we observed no improvement in the dual bound provided by the continuous relaxation compared to the setting in which we didn't use Algorithm 1. Thus, Algorithm 1 is more of a theoretical tool for Proposition 17.

Now that we have established a good theoretical understanding of the strength of recycled inequalities, we discuss in the next section how to use them in practice.

4.4 Separating Recycled Inequalities

In the previous section, we have seen that recycling can yield a vast number of facet-defining inequalities. For example, in the case of the independent set problem, every clique inequality can be recycled to a facet-defining inequality. Therefore, potentially exponentially many facet-defining recycled inequalities exist, which raises the need for an efficient separation.

4.4.1 Separation of Recycled Constraints

A straightforward separation approach is to recycle the constraints $Ax \leq b$ of the nominal problem. Given a row $\sum_{i\in[n]} a_{ji} x_i \leq b_j$ of the constraint matrix, we first remove all negative entries on the left-hand side. Since the variables x_i are binary,

$$\sum_{i\in[n]:a_{ji}\geq 0} a_{ji} x_i \leq b_j - \sum_{i\in[n]:a_{ji}<0} a_{ji}$$

is a recyclable inequality for $\mathcal{C}^{\text{ROB}}$. We may either add the corresponding recycled inequalities directly to the formulation $\mathcal{F}^{\text{ROB}}$ or precalculate and store them for later separation during branch and cut. In both cases, we restrict ourselves to inequalities with

$$\sum_{i\in[n]:a_{ji}\geq 0} a_{ji} > b_j - \sum_{i\in[n]:a_{ji}<0} a_{ji},$$

as the corresponding recycled inequality is otherwise dominated by the robustness constraints $p_i + z \geq \hat{c}_i x_i$. When using the precalculated recyclable inequalities to cut off a fractional solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{ROB}}$, we also make sure to only include variables x_i with $\hat{c}_i \tilde{x}_i - \tilde{p}_i \geq 0$.

This maximizes the violation $\sum_{i\in[n]} \pi_i \left(\hat{c}_i\tilde{x}_i - \tilde{p}_i\right) - \pi_0\tilde{z}$ of the recycled inequality. Accordingly, we iterate over every row in the constraint matrix $Ax \leq b$ and recycle

$$\sum_{\substack{i\in[n]:a_{ji}\geq 0,\\ \hat{c}_i\tilde{x}_i-\tilde{p}_i\geq 0}} a_{ji}x_i \leq b_j - \sum_{i\in[n]:a_{ji}<0} a_{ji}$$

if the resulting recycled inequality is violated. We will see in our computational study that this simple approach already improves the solver's performance drastically in many cases.

4.4.2 Separation of Recycled Cuts

Another approach is to benefit from the research on the nominal problem and recycle well studied cutting planes, i.e., inequalities that are valid for the convex hull $\mathcal{C}^{\text{NOM}}$ but not for the continuous relaxation $\mathcal{F}^{\text{NOM}}$. Let $\Pi \subseteq \mathbb{R}^{n+1}_{\geq 0}$ be such that $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is a recyclable inequality for all $\pi \in \Pi$. When separating inequalities to cut off a fractional solution to the nominal problem $\tilde{x} \in \mathcal{F}^{\text{NOM}}$, we usually search for a $\pi \in \Pi$ with positive violation $\sum_{i\in[n]} \pi_i\tilde{x}_i - \pi_0 > 0$. When separating recycled inequalities for a given solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{ROB}}$, we require $\sum_{i\in[n]} \pi_i \left(\hat{c}_i\tilde{x}_i - \tilde{p}_i\right) - \pi_0\tilde{z} > 0$ instead. Note that the coefficients $\hat{c}_i\tilde{x}_i - \tilde{p}_i$ and $\tilde{z}$ are fixed in this case. Therefore, the same algorithms for finding a violated nominal inequality can be applied for separating violated recycled inequalities, provided these do not rely on some special structure of the objective function of the separation problem. In our computational study, we will test a heuristic separation of recycled clique inequalities for the robust independent set problem. We will show that these facet-defining recycled inequalities improve the formulation significantly.

4.4.3 Exact Separation via Recycling Combined Constraints

An exact separation of violated recycled inequalities is $\mathcal{NP}$-hard in general. For example, in the case of recycled clique inequalities, an exact separation requires solving the $\mathcal{NP}$-hard maximum weighted clique problem [59]. However, we show below how we can separate recycled inequalities from valid inequalities for $\mathcal{F}^{\text{NOM}}$ in polynomial time in the size of the constraint matrix $Ax \leq b$ via solving an LP. In particular, if we already know the convex hull of the nominal problem, i.e., $\mathcal{C}^{\text{NOM}} = \mathcal{F}^{\text{NOM}}$, then an exact separation of recycled inequalities can be done in polynomial time. To see this, recall that an inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is valid for $\mathcal{F}^{\text{NOM}}$ if and only if it can be expressed as a conic combination of the rows in $Ax \leq b$ as well as the box constraints $x_i \leq 1$ and $-x_i \leq 0$ (cf. Section 2.4.5). That is, we have $\nu_i - \mu_i + \sum_{j\in[m]} a_{ji}\lambda_j = \pi_i$ and $\sum_{i\in[n]} \nu_i + \sum_{j\in[m]} b_j\lambda_j = \pi_0$ for some $\lambda \in \mathbb{R}^m_{\geq 0},\ \mu, \nu \in \mathbb{R}^n_{\geq 0}$. Hence, there exists a recyclable inequality for $\mathcal{F}^{\text{NOM}}$ whose recycled inequality is violated by $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{ROB}}$ if and only if the following separation LP has a solution with positive objective value

$$
\begin{aligned}
\max \ & \sum_{i\in[n]} \pi_i \left(\hat{c}_i \tilde{x}_i - \tilde{p}_i\right) - \pi_0 \tilde{z} \\
\text{s.t. } & \nu_i - \mu_i + \sum_{j\in[m]} a_{ji}\lambda_j = \pi_i && \forall i \in [n] \\
& \sum_{i\in[n]} \nu_i + \sum_{j\in[m]} b_j \lambda_j = \pi_0 \\
& \pi \in \mathbb{R}^{n+1}_{\geq 0},\ \lambda \in \mathbb{R}^m_{\geq 0},\ \mu, \nu \in \mathbb{R}^n_{\geq 0}.
\end{aligned}
\tag{SLP}
$$

As the optimal objective value of SLP is either zero or unbounded due to arbitrary scaling, we normalize the recyclable inequality $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ by fixing $\pi_0 = 1$. This imposes no restriction on finding violated recycled inequalities, as we always have $\pi_0 > 0$ for all relevant recyclable inequalities and can thus achieve $\pi_0 = 1$ via scaling. This is because the left-hand side $\sum_{i\in[n]} \pi_i x_i$ is non-negative, and thus $\pi_0 = 0$ implies $x_i = 0$ for all $x \in \mathcal{F}^{\text{NOM}}$ and $i \in [n]$ with $\pi_i > 0$. In this case, the right-hand side of the recycled inequality $\sum_{i\in[n]} \pi_i \hat{c}_i \tilde{x}_i$ would always be zero, which renders the inequality void.

Remember that recycling is a homomorphism on the set of recyclable inequalities. Hence, recycling a conic combination of inequalities, as given by a solution to SLP, is only reasonable if some of the combined inequalities are not recyclable themselves. Accordingly, if we have $\pi = \pi^1 + \pi^2$, with π^1, π^2 linearly independent and both define recyclable inequalities, then it is better to recycle each of these inequalities separately. In practice, this is achieved by fixing $\pi_0 = 1$, as a combination of π^1 and π^2 is never a vertex of SLP and can only be an optimal solution if the violation of their respective recycled inequalities is equal.

We observe two issues when using SLP for separation in practice. First, solving SLP is relatively time consuming if the number of inequalities is large. Second, we obtain only one optimal solution when solving SLP, and can thus only separate one recycled inequality at a time. However, MILP solvers usually perform better when several cuts are added at once. The following proposition helps in this regard, showing that we can partition the constraints into sets that can be considered independently for combination. Doing so, we can solve one smaller LP for each set of the partition, yielding multiple (possibly violated) recycled inequalities within the same separation round.

Proposition 18. *Let $A = (a_{ji})_{j\in[m],i\in[n]}$ be the left-hand side of the constraints $Ax \leq b$ (not including $0 \leq x \leq 1$) and let $G = (V, E)$ be the graph with nodes $V = [m]$ as well as edges $E = \left\{\{j, j'\} \in \binom{V}{2} \middle| \exists i \in [n] : a_{ji} < 0 < a_{j'i}\right\}$. Let $\{C_1, \ldots, C_k\} \subseteq 2^V$ be the connected components in G. Then every inequality that is recycled from a valid inequality for $\mathcal{F}^{NOM}$ is dominated by inequalities recycled from recyclable inequalities of the form*

$$
\sum_{i\in[n]} \left(\nu^\ell_i - \mu^\ell_i + \sum_{j\in C_\ell} a_{ji}\lambda_j\right) x_i \leq \sum_{i\in[n]} \nu^\ell_i + \sum_{j\in C_\ell} b_j \lambda_j \tag{4.2}
$$

with $\ell \in [k]$, $\lambda \in \mathbb{R}^m_{\geq 0}$, and $\mu^\ell, \nu^\ell \in \mathbb{R}^n_{\geq 0}$.

Proof. We write $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ as

$$\sum_{i\in[n]} \left(\nu_i - \mu_i + \sum_{j\in[m]} a_{ji}\lambda_j \right) x_i \leq \sum_{i\in[n]} \nu_i + \sum_{j\in[m]} b_j\lambda_j,$$

which is a conic combination of $Ax \leq b$ and $-x \leq 0$ as well as $x \leq 1$ with coefficients $\lambda \in \mathbb{R}^m_{\geq 0},\ \mu, \nu \in \mathbb{R}^n_{\geq 0}$. We show that if $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is recyclable, then it is also a conic combination of the recyclable inequalities from the statement.

Note that we can assume $\nu_i = 0$ or $\mu_i = 0$ for all $i \in [n]$, since we can otherwise decrease both and then recycle $\sum_{i\in[n]} \pi_i x_i \leq \pi_0 - \sum_{i\in[n]} \min\{\nu_i, \mu_i\}$ instead. We also assume $\nu_{i'} = \left(-\sum_{j\in[m]} a_{ji'}\lambda_j\right)^+$ for all $i' \in [n]$, as otherwise $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is a combination of $x_{i'} \leq 1$ and the recyclable inequality obtained by decreasing $\nu_{i'}$ to $\left(-\sum_{j\in[m]} a_{ji'}\lambda_j\right)^+$. It follows that $\mu_{i'} = 0$ holds for all $i' \in [n]$ with $\sum_{j\in[m]} a_{ji'}\lambda_j < 0$, because we have $\nu_{i'} = -\sum_{j\in[m]} a_{ji'}\lambda_j > 0$ in this case. If $\sum_{j\in[m]} a_{ji'}\lambda_j \geq 0$ holds, then we can assume $\mu_{i'} \in \left\{0, \sum_{j\in[m]} a_{ji'}\lambda_j\right\}$. This is because both values result in recyclable inequalities and all values in between imply that $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ is a convex combination of the two inequalities obtained by choosing $\mu_{i'} = 0$ or $\mu_{i'} = \sum_{j\in[m]} a_{ji'}\lambda_j$. We conclude that we only choose $\mu_{i'} > 0$ or $\nu_{i'} > 0$ if we want to obtain $\pi_{i'} = 0$. This implies $\mu_{i'} = 0$ for $\pi_{i'} > 0$ and $\mu_{i'} = \left(\sum_{j\in[m]} a_{ji'}\lambda_j\right)^+$ for $\pi_{i'} = 0$. Together with $\nu_{i'} = \left(-\sum_{j\in[m]} a_{ji'}\lambda_j\right)^+$ from above, we can rewrite $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ as

$$\begin{aligned}
&\sum_{i\in[n]} \left(\left(-\sum_{j\in[m]} a_{ji}\lambda_j \right)^+ + \sum_{j\in[m]} a_{ji}\lambda_j \right) x_i - \sum_{i\in[n]:\pi_i=0} \left(\sum_{j\in[m]} a_{ji}\lambda_j \right)^+ x_i \\
&\leq \sum_{i\in[n]} \left(-\sum_{j\in[m]} a_{ji}\lambda_j \right)^+ + \sum_{j\in[m]} \lambda_j b_j.
\end{aligned}$$

Note that $\sum_{j\in[m]} a_{ji}\lambda_j$ and $\sum_{j\in C_\ell} a_{ji}\lambda_j$ have the same sign for all $i \in [n]$ and $\ell \in [k]$. Otherwise, there existed $i \in [n]$ and $\ell, \ell' \in [k]$ with $\sum_{j\in C_\ell} a_{ji}\lambda_j < 0$ and $\sum_{j\in C_{\ell'}} a_{ji}\lambda_j > 0$. Then there exists $j \in C_\ell$ and $j' \in C_{\ell'}$ such that $a_{ji} < 0 < a_{j'i}$ holds. However, this implies that constraints j and j' are adjacent in the graph G, and thus $\ell = \ell'$. It follows $\left(\pm\sum_{j\in[m]} a_{ji}\lambda_j\right)^+ = \sum_{\ell\in[k]} \left(\pm\sum_{j\in C_\ell} a_{ji}\lambda_j\right)^+$, and thus we can rewrite $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ again as

$$\sum_{\ell\in[k]} \left(\sum_{i\in[n]} \left(\nu^\ell_i - \mu^\ell_i + \sum_{j\in C_\ell} a_{ji}\lambda_j \right) x_i \right) \leq \sum_{\ell\in[k]} \left(\sum_{i\in[n]} \nu^\ell_i + \sum_{j\in C_\ell} b_j\lambda_j \right),$$

with $\mu^\ell_i = \left(\sum_{j\in C_\ell} a_{ji}\lambda_j\right)^+$ for $\pi_i = 0$ and $\mu^\ell_i = 0$ for $\pi_i > 0$ as well as $\nu^\ell_i = \left(-\sum_{j\in C_\ell} a_{ji}\lambda_j\right)^+$.

Thus, the above inequality decomposes into the k inequalities (4.2), all of which are recyclable since $\nu^\ell_i - \mu^\ell_i + \sum_{j\in C_\ell} a_{ji}\lambda_j \geq 0$ holds by the definition of ν^ℓ, μ^ℓ. □

In the special case where all constraints are recyclable, the graph G from the proposition above contains no edges. Thus, we don't have to consider any combinations of constraints but can solely rely on recycling constraints as in Section 4.4.1.

Corollary 19. *Let all constraints in $Ax \leq b$ be recyclable. Then every inequality that is recycled from a valid inequality for $\mathcal{F}^{\text{NOM}}$ is dominated by the recycled inequalities from $\sum_{i \in I} a_{ji} x_i \leq b_j$ for $j \in [m]$ and $I \subseteq [n]$.*

In the case where we have $\mathcal{F}^{\text{NOM}} = \mathcal{C}^{\text{NOM}}$ and all constraints are recyclable, the above corollary shows together with Proposition 10 that we can separate inequalities of the form $\pi_0 z + \sum_{i \in [n]} \pi_i p_i \geq \sum_{i \in [n]} \pi_i \hat{c}_i x_i$ exactly in linear time. Given such favorable conditions, one might ask whether we even obtain the convex hull $\mathcal{C}^{\text{ROB}}$ by separating recycled inequalities. This seems especially plausible since $\mathcal{F}^{\text{NOM}} = \mathcal{C}^{\text{NOM}}$ implies that ROB can be solved in polynomial time according to the famous result of Bertsimas and Sim [22]. However, the following example shows that we do not obtain $\mathcal{C}^{\text{ROB}}$ for a robust bipartite matching problem, although all constraints are recyclable in this case and we have $\mathcal{F}^{\text{NOM}} = \mathcal{C}^{\text{NOM}}$ for the bipartite matching polytope [64, Section 11.1].

Example 20. Consider a complete bipartite graph $G = (V, E)$ with $V = \{1, 2\} \cup \{3, 4\}$ as well as $E = \{e_1, \ldots, e_4\} = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}\}$. The convex hull of the robust weighted matching problem with variables $x_i \in \{0, 1\}$ and deviations $\hat{c}_i = i$ for every $e_i \in E$ is

$$\mathcal{C}^{\text{ROB}} = \operatorname{conv}\left(\left\{(x, p, z) \in \{0,1\}^4 \times \mathbb{R}^5_{\geq 0} \;\middle|\; \begin{array}{ll} x_1 + x_2 \leq 1 & \\ x_1 + x_3 \leq 1 & \\ x_2 + x_4 \leq 1 & \\ x_3 + x_4 \leq 1 & \\ p_i + z \geq \hat{c}_i x_i & \forall i \in [4] \end{array}\right\}\right).$$

We use PORTA [36] to compute a representation of $\mathcal{C}^{\text{ROB}}$ for this problem and see that the inequality $p_2 + p_3 + z \geq 2x_2 + x_3$ is facet-defining for $\mathcal{C}^{\text{ROB}}$. The validity is easily verified, because the inequality is implied by $p_2 + z \geq 2x_2$ for $x_3 = 0$ and by $p_3 + z \geq 3x_3$ for $x_3 = 1$ due to $2x_2 + x_3 \leq 3 = 3x_3$. To see that it is also facet-defining, one verifies that the following $\dim\left(\mathcal{C}^{\text{ROB}}\right) = 9$ vectors are affinely independent and satisfy the inequality with equality

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ p_1 \\ p_2 \\ p_3 \\ p_4 \\ z \end{pmatrix} \in \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 3 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 4 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 2 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 3 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 2 \end{pmatrix} \right\}.$$

Note that $p_2 + p_3 + z \geq 2x_2 + x_3$ is not a recycled inequality, since the quotient of the coefficients of x_3 and p_3 is not $\hat{c}_3 = 3$.

In our computational study, we show that recycled inequalities nevertheless almost completely close the gap between optimal integer and continuous solutions for some robust bipartite matching problems. This highlights the large potential of recycled inequalities when all constraints are recyclable and $\mathcal{F}^{\text{NOM}} = \mathcal{C}^{\text{NOM}}$ holds.

Even if not all constraints are recyclable, an optimal solution to SLP often corresponds to an already recyclable constraint in $Ax \leq b$ in practice. Hence, we observe that it is beneficial to first check whether we can separate violated recycled inequalities from constraints, as described in Section 4.4.1. Only if none of these are violated, we solve SLP to check whether there exists a violated recycled inequality from a combined inequality. We will see in our tests on robust instances generated from the MIPLIB 2017 that solving SLP sometimes yields very strong recycled inequalities, even if recycling the constraints in $Ax \leq b$ has no effect at all (cf. Section 4.6.7).

4.5 Partially Recycling of Non-Recyclable Inequalities

Let $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ be a non-recyclable valid inequality. In the previous section, we transformed such inequalities into $\sum_{i\in[n]:\pi_i>0} \pi_i x_i \leq \pi_0 - \sum_{i\in[n]:\pi_i<0} \pi_i$ for recycling, by estimating $\pi_i x_i \geq \pi_i$ for $\pi_i < 0$. Intuitively, the resulting recycled inequality seems to be unnecessarily weak if the estimated terms $\pi_i \tilde{x}_i$ are actually (near to) zero for a continuous solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{ROB}}$ to be cut off. To resolve this, we propose another procedure, using the recyclable part of generally non-recyclable inequalities.

Note that $\sum_{i\in[n]:\pi_i\geq 0} \pi_i x_i \leq \pi_0$ is a recyclable inequality for the restricted nominal solution space $\left\{x \in \mathcal{C}^{\text{NOM}} \middle| x_i = 0 \; \forall i \in [n] : \pi_i < 0\right\}$, and can thus be recycled to a valid inequality for $\left\{(x,p,z) \in \mathcal{C}^{\text{ROB}} \middle| x_i = 0 \; \forall i \in [n] : \pi_i < 0\right\}$. In order to obtain a valid inequality for $\mathcal{C}^{\text{ROB}}$, we can *lift* the fixed variables into the recycled inequality. For this, we compute *lifting coefficients* $\alpha_i \in \mathbb{R}$ for $i \in [n]$ with $\pi_i < 0$ such that

$$\pi_0 z + \sum_{i\in[n]:\pi_i>0} \pi_i p_i \geq \sum_{i\in[n]:\pi_i>0} \pi_i \hat{c}_i x_i + \sum_{i\in[n]:\pi_i<0} \alpha_i x_i$$

is a valid inequality.

In general, one wants to choose maximal lifting coefficients α, such that the *lifted inequality* is as strong as possible. Whether one obtains a facet-defining inequality is not trivial to say, as this not only depends on the inequality to be lifted and the maximality of the lifting coefficients but also on the considered polyhedron. However, roughly speaking, lifting is more likely to yield a facet-defining inequality if the original inequality is facet-defining for

the restricted solution space, where the variables to be lifted are fixed to zero [50, 95, 98]. Using Theorem 12, we can state in which case this applies for our recycled inequalities.

Corollary 21. *Let $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ be valid for $\mathcal{C}^{NOM}$ with $\pi_0 \geq 0$. Let $S^+ = \{i \in [n] | \pi_i > 0\}$ as well as $S^- = \{i \in [n] | \pi_i < 0\}$. The recycled inequality*

$$\pi_0 z + \sum_{i\in S^+} \pi_i p_i \geq \sum_{i\in S^+} \pi_i \hat{c}_i x_i$$

is facet-defining for the restricted solution space $\left\{(x,p,z) \in \mathcal{C}^{ROB} \middle| x_i = 0 \; \forall i \in S^-\right\}$ if $\hat{c}_i > 0$ holds for all $i \in S^+$, i.e., it is uncertainty-exclusive on $\{x_i | i \in S^+\}$, and

$$\dim\left(\mathrm{proj}_{S^+}\left(\{x \in F(\pi) | x_i = 0 \; \forall i \in S^-\}\right)\right) = \left|S^+\right| - 1.$$

Hence, the approach of fixing, recycling, and lifting is promising if the original inequality is strong on the variables $\{x_i | i \in S^+\}$ and if we are able to compute high lifting coefficients α. Computing maximal lifting coefficients involves solving multiple optimization problems that are often $\mathcal{NP}$-hard. This is because we need to optimize over a set of solutions that almost equals that of the original problem. For example, when only lifting the variable x_ℓ into the recycled inequality $\pi_0 z + \sum_{i\in S^+} \pi_i p_i \geq \sum_{i\in S^+} \pi_i \hat{c}_i x_i$, then we need to solve

$$\begin{aligned} \min\ & \pi_0 z + \sum_{i\in S^+} \pi_i p_i - \sum_{i\in S^+} \pi_i \hat{c}_i x_i \\ \text{s.t. }\ & (x,p,z) \in \mathcal{C}^{\text{ROB}}, x_\ell = 1. \end{aligned}$$

That is, we minimize the slack of the inequality to be lifted while fixing $x_\ell = 1$. This (in our case non-positive) slack is then the maximal lifting coefficient of x_ℓ. The theoretical complexity of lifting implies the need for an efficient heuristic approach. The following proposition shows how to compute lifting coefficients by solving a sequence of easy continuous knapsack problems.

Proposition 22. *Let $\sum_{i\in[n]} \pi_i x_i \leq \pi_0$ be valid for $\mathcal{C}^{NOM}$ with $\pi_0 \geq 0$. Let $S^+ = \{i \in [n] | \pi_i > 0\}$ as well as $S^- = \{i \in [n] | \pi_i < 0\}$. Consider the continuous knapsack problem*

$$f(\beta) = \max\left\{ \sum_{i\in S^+} \pi_i \hat{c}_i x_i \middle| \sum_{i\in S^+} \pi_i x_i \leq \beta, x \in [0,1]^n \right\}$$

for a capacity $\beta \in \mathbb{R}_{\geq 0}$ and let $\alpha_i = f(\pi_0) - f(\pi_0 - \pi_i)$ for $i \in S^-$. Then

$$\pi_0 z + \sum_{i\in S^+} \pi_i p_i \geq \sum_{i\in S^+} \pi_i \hat{c}_i x_i + \sum_{i\in S^-} \alpha_i x_i$$

is a valid inequality for $\mathcal{C}^{ROB}$.

Proof. We sequentially lift the variables $\{x_{i_1}, \dots, x_{i_k}\} = \{x_i | i \in S^-\}$. After lifting x_{i_ℓ}, this yields a valid inequality for the restricted solutions $\left\{(x,p,z) \in \mathcal{C}^{\text{ROB}} \middle| x_{i_{\ell+1}} = \dots = x_{i_k} = 0\right\}$. Assume that we already lifted variables $x_{i_1}, \dots, x_{i_{\ell-1}}$ with coefficients $\alpha_1, \dots, \alpha_{\ell-1}$ and consider the problem of lifting variable x_{i_ℓ}

$$\begin{aligned}\min\ & \pi_0 z + \sum_{i \in S^+} \pi_i p_i - \sum_{i \in S^+} \pi_i \hat{c}_i x_i - \sum_{j \in [\ell-1]} \alpha_j x_{i_j} \\ \text{s.t. }& (x,p,z) \in \mathcal{C}^{\text{ROB}},\ x_{i_\ell} = 1,\ x_{i_{\ell+1}} = \dots = x_{i_k} = 0.\end{aligned}$$

We can assume that the bilinear constraints $p_i + x_i z \geq \hat{c}_i x_i$ are met, since it is sufficient to only consider integer-feasible solutions for the lifting problem. We relax the lifting problem by only considering the bilinear constraints as well as the reduced constraint $\sum_{i \in S^+} \pi_i x_i \leq \pi_0 - \pi_{i_\ell} - \sum_{j \in [\ell-1]} \pi_{i_j} x_{i_j}$ obtained from the original constraint $\sum_{i \in [n]} \pi_i x_i \leq \pi_0$ and $x_{i_{\ell+1}} = \dots = x_{i_k} = 0$. Furthermore, we allow all variables but $x_{i_1}, \dots, x_{i_\ell}$ to be fractional. By assuming that $S \subseteq [\ell - 1]$ defines an optimal choice for the already lifted variables $x_{i_1}, \dots, x_{i_{\ell-1}}$, with $x_{i_j} = 1$ iff $j \in S$, we obtain the following relaxed lifting problem

$$\text{(RLP)} \qquad \begin{aligned}\min\ & \pi_0 z + \sum_{i \in S^+} \pi_i p_i - \sum_{i \in S^+} \pi_i \hat{c}_i x_i - \sum_{j \in S} \alpha_j \\ \text{s.t. }& \sum_{i \in S^+} \pi_i x_i \leq \beta \\ & p_i + x_i z \geq \hat{c}_i x_i && \forall i \in S^+ \\ & x \in [0,1]^{S^+}, p \in \mathbb{R}^{S^+}_{\geq 0}, z \in \mathbb{R}_{\geq 0}\end{aligned}$$

with $\beta = \pi_0 - \pi_{i_\ell} - \sum_{j \in S} \pi_{i_j}$. We will first show that the optimal solution value of RLP equals $f(\pi_0) - f(\beta) - \sum_{j \in S} \alpha_j$ for all $\beta \geq \pi_0$. Afterwards, we show that $S = \emptyset$ is an optimal choice, which proves that $f(\pi_0) - f(\pi_0 - \pi_{i_\ell}) = \alpha_\ell$ is a feasible lifting coefficient.

Since $f(\beta)$ is a continuous knapsack problem with capacity β, values $\pi_i \hat{c}_i$, and weights π_i, we can compute an optimal solution by sorting the variables with respect to $\frac{\pi_i \hat{c}_i}{\pi_i} = \hat{c}_i$ and greedily fill the knapsack until the capacity is reached [61, Section 2.3]. Let x^* be such an optimal greedy solution to $f(\beta)$. We show that x^*, together with appropriate p^*, z^*, is also an optimal solution to RLP. For this, let (x,p,z) be an optimal solution to RLP. We can assume $p_i = (\hat{c}_i - z)^+ x_i$, and thus obtain

$$\pi_0 z + \sum_{i \in S^+} \pi_i p_i - \sum_{i \in S^+} \pi_i \hat{c}_i x_i - \sum_{j \in S} \alpha_j = \pi_0 z - \sum_{i \in S^+} \pi_i \min\{\hat{c}_i, z\} x_i - \sum_{j \in S} \alpha_j.$$

Hence, when fixing z, RLP reduces to a continuous knapsack problem with values $\pi_i \min\{\hat{c}_i, z\}$ and weights π_i. The above greedy solution x^* is optimal for this continuous knapsack problem, since sorting with respect to $\hat{c}_i$ also yields a sorting with respect to $\frac{\pi_i \min\{\hat{c}_i, z\}}{\pi_i} = \min\{\hat{c}_i, z\}$.

Now, we choose

$$z^* = \min\left\{ z \in \{0, \hat{c}_1, \dots, \hat{c}_n\} \middle| \sum_{i \in S^+, \hat{c}_i > z} \pi_i x^*_i \leq \pi_0 \right\}$$

together with $p_i^* = (\hat{c}_i - z^*)^+ x_i^*$. We first show that the value of this solution equals $f(\pi_0) - f(\beta) - \sum_{j \in S} \alpha_j$ and show afterwards that it is optimal.

If $\sum_{i \in S^+} \pi_i \leq \pi_0$ holds, then we have $z^* = 0$, and thus

$$\begin{aligned} &\pi_0 z^* + \sum_{i \in S^+} \pi_i p_i^* - \sum_{i \in S^+} \pi_i \hat{c}_i x_i^* - \sum_{j \in S} \alpha_j \\ =&\sum_{i \in S^+} \pi_i (\hat{c}_i - 0)^+ x_i^* - \sum_{i \in S^+} \pi_i \hat{c}_i x_i^* - \sum_{j \in S} \alpha_j = -\sum_{j \in S} \alpha_j. \end{aligned}$$

As the capacity π_0 is non-restrictive for the knapsack problem, an increase up to $\beta \geq \pi_0$ has no effect on the objective value. Hence, we also have $f(\pi_0) - f(\beta) - \sum_{j \in S} \alpha_j = -\sum_{j \in S} \alpha_j$.

If $\sum_{i \in S^+} \pi_i > \pi_0$ holds, then we assume $0 = \hat{c}_0 \leq \hat{c}_1 \leq \cdots \leq \hat{c}_n$ and let $j^* \in [n]_0$ be the smallest index such that $\sum_{i \in S^+ : i > j^*} \pi_i \leq \pi_0$. It follows $z^* = \hat{c}_{j^*}$ by the definition of z^* and we can assume $x_i^* = 1$ for all $i \in S^+$ with $i > j^*$ by the definition of our greedy solution. This implies that (x^*, p^*, z^*) is a solution to RLP of value

$$\begin{aligned} &\pi_0 z^* + \sum_{i \in S^+} \pi_i p_i^* - \sum_{i \in S^+} \pi_i \hat{c}_i x_i^* - \sum_{j \in S} \alpha_j \\ =&\pi_0 \hat{c}_{j^*} + \sum_{i \in S^+ : i > j^*} \pi_i (\hat{c}_i - \hat{c}_{j^*}) - f(\beta) - \sum_{j \in S} \alpha_j \\ =&\left(\pi_0 - \sum_{i \in S^+ : i > j^*} \pi_i \right) \hat{c}_{j^*} + \sum_{i \in S^+ : i > j^*} \pi_i \hat{c}_i - f(\beta) - \sum_{j \in S} \alpha_j \\ =&f(\pi_0) - f(\beta) - \sum_{j \in S} \alpha_j. \end{aligned}$$

Here, the last equation holds since $x_i^* = 1$ for all $i \in S^+$ with $i > j^*$ and

$$x_{j^*}^* = \frac{\pi_0 - \sum\limits_{i \in S^+ : i > j^*} \pi_i}{\pi_{j^*}}$$

is an optimal solution to $f(\pi_0)$.

To see that the choice of p^*, z^* is optimal, first consider $z' > z^*$ and p' with $p_i' \geq (\hat{c}_i - z') x_i^*$. By definition of z^*, we have $\sum_{i \in S^+ : \hat{c}_i > z^*} \pi_i x_i^* \leq \pi_0$, and thus

$$\begin{aligned} \pi_0 z^* + \sum_{i \in S^+} \pi_i p_i^* &= \pi_0 z^* + \sum_{i \in S^+ : \hat{c}_i > z^*} \pi_i (z' - z^* + \hat{c}_i - z') x_i^* \\ &\leq \pi_0 z^* + \pi_0 (z' - z^*) + \sum_{i \in S^+ : \hat{c}_i > z^*} \pi_i (\hat{c}_i - z') x_i^* \\ &\leq \pi_0 z' + \sum_{i \in S^+ : \hat{c}_i > z'} \pi_i (\hat{c}_i - z') x_i^* \\ &\leq \pi_0 z' + \sum_{i \in S^+} \pi_i p_i'. \end{aligned}$$

Second, consider $z' < z^*$ with an appropriate p'. Due to the minimality of z^*, we have $\sum_{i \in S^+ : \hat{c}_i \geq z^*} \pi_i x_i^* > \pi_0$, and thus

$$\begin{aligned}
\pi_0 z^* + \sum_{i \in S^+} \pi_i p_i^* &= \pi_0 \left(z' + z^* - z'\right) + \sum_{i \in S^+ : \hat{c}_i \geq z^*} \pi_i \left(\hat{c}_i - z^*\right) x_i^* \\
&< \pi_0 z' + \sum_{i \in S^+ : \hat{c}_i \geq z^*} \pi_i \left(z^* - z' + \hat{c}_i - z^*\right) x_i^* \\
&\leq \pi_0 z' + \sum_{i \in S^+ : \hat{c}_i > z'} \pi_i \left(\hat{c}_i - z'\right) x_i^* \\
&\leq \pi_0 z' + \sum_{i \in S^+} \pi_i p_i',
\end{aligned}$$

which shows the optimality of z^* and p^*.

We have shown that $f(\pi_0) - f\left(\pi_0 - \pi_{i_\ell} - \sum_{j \in S} \pi_{i_j}\right) - \sum_{j \in S} \alpha_j$ is the optimal value of RLP for some $S \subseteq [\ell - 1]$. Thus, it only remains to show that $S = \emptyset$ is optimal. To see this, note that f is submodular due to the diminishing utility of additional capacity. That is, we have $f(\beta' + \varepsilon) - f(\beta') \geq f(\beta + \varepsilon) - f(\beta)$ for $\beta' \leq \beta$ and $\varepsilon \geq 0$. Since all π_{i_j} are negative, this implies

$$\sum_{j \in S} \left(f\left(\pi_0 - \pi_{i_j}\right) - f(\pi_0)\right) \geq f\left(\pi_0 - \sum_{j \in S} \pi_{i_j}\right) - f(\pi_0),$$

and thus we have

$$\begin{aligned}
& f(\pi_0) - f\left(\pi_0 - \pi_{i_\ell} - \sum_{j \in S} \pi_{i_j}\right) - \sum_{j \in S} \alpha_j \\
=& f(\pi_0) - f\left(\pi_0 - \pi_{i_\ell} - \sum_{j \in S} \pi_{i_j}\right) + \sum_{j \in S} \left(f\left(\pi_0 - \pi_{i_j}\right) - f(\pi_0)\right) \\
\geq& f\left(\pi_0 - \sum_{j \in S} \pi_{i_j}\right) - f\left(\pi_0 - \pi_{i_\ell} - \sum_{j \in S} \pi_{i_j}\right) \\
\geq& f(\pi_0) - f(\pi_0 - \pi_{i_\ell}),
\end{aligned}$$

which proves the statement. □

In practice, when cutting off a fractional solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\mathrm{ROB}}$ with a lifted recycled inequality, we again drop all variables x_i from the inequality with $\pi_i > 0$ and $\hat{c}_i \tilde{x}_i < \tilde{p}_i$, as these negatively impact the violation of the recycled inequality. We do this before lifting the variables x_i with $\pi_i < 0$, as doing so restricts the lifting problem RLP, and thus yields potentially better lifting coefficients.

Note that we require $\pi_0 \geq 0$ in the above proposition, as a negative coefficient of the unbounded variable z would imply infinite lifting coefficients α for obtaining a valid inequality. Hence, if the original inequality has $\pi_0 < 0$, we first have to estimate some $\pi_i x_i \geq \pi_i$ with $\pi_i < 0$ to obtain a non-negative right-hand side. This raises the question of which variables

should be estimated and which should be lifted. Moreover, even if $\pi_0 \geq 0$ holds, it is reasonable to check whether lifting or estimating a variable yields a higher violation. For example, when cutting off a fractional solution $(\tilde{x}, \tilde{p}, \tilde{z})$ with $\tilde{z} = 0$, we obtain a higher violation by estimating $x_i \pi_i \geq \pi_i$ and recycling $\sum_{i \in S^+} \pi_i x_i \leq \pi_0 - \sum_{i \in S^-} \pi_i$, as the higher coefficient of z in the recycled inequality is irrelevant in this case. Contrary to that, if $\tilde{z} > 0$ and $\tilde{x}_i = 0$ hold, then it is preferable to lift x_i.

Since we add $\alpha_i \tilde{x}_i$ to the violation when lifting and $\pi_i \tilde{z}$ when estimating, we want to lift those variables with $\alpha_i \tilde{x}_i > \pi_i \tilde{z}$. However, if we decide to estimate $\pi_i x_i \geq \pi_i$ for $i \in S \subseteq S^-$, then we obtain a new inequality with a greater right-hand side $\pi_0 - \sum_{i \in S} \pi_i$, which influences the lifting coefficients $\alpha_j(S) = f(\pi_0 - \sum_{i \in S} \pi_i) - f(\pi_0 - \sum_{i \in S} \pi_i - \pi_j)$ of the other variables x_j, and thus our lifting decision. Let $\{i_1, \ldots, i_k\} = S^-$ such that $\tilde{x}_{i_1} \geq \cdots \geq \tilde{x}_{i_k}$. Since variables with higher solution values $\tilde{x}_i$ are less preferable for lifting, it is reasonable to assume that a good decision for S consists of variables $x_{i_1}, \ldots, x_{i_j}$ for some $j \in [k]$. Therefore, we first set $S = \emptyset$ and assume that all variables will be lifted. Afterwards, we greedily decide for $i \in \{i_1, \ldots, i_k\}$ whether x_i should better not be lifted and instead added to S. For this, we check whether

$$\pi_i \tilde{z} + \sum_{j \in \{i_1, \ldots, i_k\} \setminus (S \cup \{i\})} \alpha_j(S \cup \{i\}) \tilde{x}_j \geq \sum_{j \in \{i_1, \ldots, i_k\} \setminus S} \alpha_j(S) \tilde{x}_j$$

holds, i.e., whether the change of the violation is positive when not lifting x_i. Note that the values $\alpha_j(S \cup \{i\})$ can be updated efficiently from $\alpha_j(S)$ by greedily extending the solutions of the corresponding continuous knapsack problems.

We use this approach in the following computational study, which shows the practical relevance of recycling in general and also indicates the potential of partially recycling.

4.6 Computational Study

In this section, we assess the performance of recycled inequalities computationally. We first discuss numerical pitfalls that can occur in practice when using recycled inequalities and present a remedy for these in Section 4.6.1. Afterwards, we lay out our methodology for measuring an algorithm's performance in Section 4.6.2. Furthermore, we evaluate in Section 4.6.3 how the parameters $\hat{c}$ and Γ should be chosen for converting a nominal problem NOM into a hard robust problem ROB. Using these insights, we construct robust instances for different classes of combinatorial optimization problems in order to test different aspects of recycling inequalities.

We study the robust independent set problem in Section 4.6.4 to examine the contribution of recycling problem-specific cuts. For this, we heuristically separate recycled clique inequalities, which are always facet-defining for the robust problem (cf. Section 4.3). In Section 4.6.5, we test the recycling of model constraints for the robust bipartite matching problem. Since the

standard formulation of the nominal version consists exclusively of recyclable inequalities and also describes the convex hull $\mathcal{C}^{\text{NOM}}$ [64, Section 11.1], every non-dominated recycled inequality corresponds to a model constraint (cf. Corollary 19). This allows us to test the influence of recycled inequalities to their limits. In Section 4.6.6, we consider the robust bipartite matching problem with penalties, in which we allow the violation of matching constraints at the cost of a penalty. Using the adapted model constraints, which are no longer recyclable, we test the partially recycling of non-recyclable inequalities.

After considering the combinatorial problems above, we evaluate the practical relevance of recycling on a broad set of robustified real world instances from the MIPLIB 2017 [49] in Section 4.6.7. For these instances, we also test the generic approach of separating recycled inequalities via solving SLP.

All experiments are implemented in Java 11 and performed on a single core of a Linux machine with an Intel® Core™ i7-5930K CPU @ 3.50GHz with 2 GB RAM reserved for each calculation. We use Gurobi version 9.5.0 [52] in single thread mode and all other settings at default to solve LPs and MILPs. Furthermore, we use a time limit of 3,600 seconds for each algorithm and instance.

All implemented algorithms [46] and generated test instances [48] are freely available online.

4.6.1 Dealing with Numerical Issues

MILP solvers that rely on numerical arithmetic constantly face the threat of numerical instability, leading to inconsistent results. One source of numerical instability is a constraint matrix $Ax \leq b$ with a high range in the order of magnitude of coefficients a_{ij}, e.g., with $a_{11} = 10^{-4}$ and $a_{12} = 10^{10}$. In fact, the Gurobi documentation recommends that the range of coefficients in the constraint matrix should be within six orders of magnitude [51]. In the case of recycled inequalities $\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \hat{c}_i x_i$, the coefficients $\pi_i \hat{c}_i$ on the right-hand side might violate this desirable property if $\hat{c}_i$ and π_i are both very large or both very small. As a consequence, we observed for three instances in our computational study on the MIPLIB that sub-optimal solutions were reported as optimal.

To tackle this problem, we scale the deviations $\hat{c}_i$ as well as the variables p, z in an attempt to reduce the range of coefficients in the recycled inequalities. Let $\hat{c}_{\max} = \max\{\hat{c}_1, \ldots, \hat{c}_n\}$ and $\hat{c}_{\min} = \min\{\hat{c}_i | i \in [n], \hat{c}_i > 0\}$ be the maximum and minimum (proper) deviations. If $\hat{c}_{\max}$ is very large and $\hat{c}_{\min}$ simultaneously very small, then our problem is predisposed to be numerically unstable anyway. However, if both are either very large or very small, then we can scale the deviations such that $\hat{c}_{\max}$ and $\hat{c}_{\min}$ are closer to one. For this, we divide all deviations $\hat{c}_i$ by $\lambda = \sqrt{\hat{c}_{\max}\hat{c}_{\min}}$. This implies $\frac{\hat{c}_{\max}}{\lambda}\frac{\hat{c}_{\min}}{\lambda} = 1$, i.e., the scaled maximum and minimum deviation have the same distance to one in orders of magnitudes. To compensate

this change, we multiply z and p in the objective function with λ. Thus, our new problem, which is equivalent to ROB, reads

$$\begin{aligned}
&\min \lambda\Gamma z + \sum_{i\in[n]} (c_i x_i + \lambda p_i) \\
&\text{s.t. } Ax \leq b \\
&\quad p_i + z \geq \frac{\hat{c}_i}{\lambda} x_i && \forall i \in [n] \\
&\quad x \in \{0,1\}^n, p \in \mathbb{R}^n_{\geq 0}, z \in \mathbb{R}_{\geq 0}
\end{aligned}$$

and the recycled inequalities are

$$\pi_0 z + \sum_{i\in[n]} \pi_i p_i \geq \sum_{i\in[n]} \pi_i \frac{\hat{c}_i}{\lambda} x_i.$$

This small change resolves the observed issues for the MIPLIB instances. For comparability, we will always use the scaled problem when solving ROB. However, for the sake of simplicity, we will only write down the non-scaled problem in the following sections.

4.6.2 Performance Indicators

Rating the performance of generic algorithms is not trivial, as different use cases imply different requirements for an algorithm. While we aim to find an optimal solution as fast as possible for some practical problems, it is important to find any good solution within seconds for other problems. Therefore, we need performance indicators that appropriately reflect the spectrum of use cases.

The standard performance indicator for optimization algorithms is the computation time it takes to solve an instance. This indicator reflects the aim of solving problems as fast as possible to optimality. However, a major drawback is that we might not know how long it takes to solve very hard instances, as we often need to terminate algorithms prematurely after a predefined time limit. A common workaround is to set the computation time to the time limit and pretend that all instances are solved at this time at the latest. This obviously leads to unfair comparisons in favor of algorithms reaching the time limit more frequently and also completely ignores the optimality gap at termination. Moreover, the computation time provides no information on how an algorithm performed up to the point at which it is terminated. That is, one algorithm can prove optimality later than other algorithms but provides better solutions in the first few seconds, which makes it more practicable in operational planning.

To take this into account, we additionally measure the *primal-dual integral*, which was proposed by Berthold [16] with the aim that this metric "reflects the development of the solution quality over the complete optimization process". The primal-dual integral is defined to be the integral of the gap between the current primal and dual bound for each point in

time. Since the optimality gap, as reported by Gurobi, is in general not bounded and at the start even infinite, the primal-dual integral is defined over an adapted gap. Let $\overline{v}(t)$ be the primal bound and $\underline{v}(t)$ be the dual bound at time t, with $\overline{v}(t) = \infty$ or $\underline{v}(t) = -\infty$ respectively if no bound is known. We define the step function

$$g(t) = \begin{cases} 1, & \text{if } \overline{v}(t) = \infty \text{ or } \underline{v}(t) = -\infty \text{ or } \underline{v}(t) \cdot \overline{v}(t) < 0, \\ 0, & \text{if } \underline{v}(t) = \overline{v}(t), \\ \frac{\overline{v}(t) - \underline{v}(t)}{\max\{|\overline{v}(t)|, |\underline{v}(t)|\}}, & \text{else,} \end{cases}$$

with respect to the piecewise constant bounds $\overline{v}(t), \underline{v}(t)$, for which we can easily compute the primal-dual integral

$$G(T) = \int_{t=0}^{T} g(t) \, \mathrm{d}\, t.$$

Here, T is the time at which the algorithm is terminated or finishes. The primal-dual integral reflects improvements of the gap over the whole computation process, and is thus a reasonable additional performance indicator alongside the computation time.

We aggregate performance indicators using the *shifted geometric mean* [2], which is defined as $\left(\Pi_{i \in [k]} (v_i + s)^{1/k}\right) - s$ for values $v_1, \ldots, v_k \in \mathbb{R}_{\geq 0}$ and a *shifting parameter* $s \in \mathbb{R}_{\geq 0}$. The advantage of the shifted geometric mean over the geometric or arithmetic mean is that it is not overly sensitive to very small or very large values. The geometric mean considers the difference between 0.1 and 0.2 equally significant as the difference between 1,000 and 2,000. Conversely, differences in small values are barely noticeable in the arithmetic mean when larger values are present. In the following, we always use the shifted geometric mean with shifting parameters of $s = 1$ second for computation times and $s = 100\%$ for primal-dual integrals. The latter corresponds to the integral of one second at maximum gap. Besides computation times and primal-dual integrals, we will also report integrality gaps to compare the strength of the continuous relaxation with and without recycled inequalities. For aggregating these, we use the shifted geometric mean with $s = 1\%$.

4.6.3 Generating Hard Robust Instances

In the following, we empirically evaluate the random generation of robust problems ROB based on given nominal problems NOM. In order to avoid a bias towards certain combinatorial problems, we do this for nominal problems of the diverse MIPLIB 2017 [49]. However, our findings will also be valuable for generating robust instances for specific problem classes.

To transform a given nominal problem into a robust problem, we have to decide which objective coefficients c_i are uncertain, that is $\hat{c}_i > 0$, how large the corresponding deviations $\hat{c}_i$ are, and what our uncertainty budget Γ is. In real-world applications, a coefficient is uncertain if, for example, it is the result of a forecast or a measurement. In [14, 20, 23], a coefficient is expected to be a result of such procedures, and thus uncertain, if it is an "ugly" number. In particular, integer values are considered "non-ugly" and are assumed to be certain.

However, since many MIPLIB instances only contain integer values, treating all integer objective coefficients as certain would leave us with few instances for our study. Therefore, we take a middle course by considering c_i to be certain only if we have $c_i \in \{-1, 0, 1\}$ in the nominal instance. We consider it unlikely that c_i is the result of a forecast or measurement in this case. This is because coefficients $c_i \in \{-1, 1\}$ usually do not represent a numerical objective value for x_i but are for counting the number of chosen variables. Moreover, $c_i = 0$ suggests that the choice of x_i has no direct effect on the objective at all.

The value of the deviations is often chosen to be a fixed percentage of the absolute nominal coefficients. That is, $\hat{c}_i = \xi |c_i|$ for uncertain objective coefficients, where ξ ranges from from 0.01% to 2% across different studies [14, 20, 23, 43]. Furthermore, the uncertainty budget Γ is chosen from a predefined set of arbitrarily fixed values [20, 23, 43]. However, these studies not only consider uncertain objective coefficients but also uncertainties in the constraints. Bearing this in mind, the above choices may be appropriate in the respective settings for illustrating the effect of uncertainty [14, 23] and the creation of sufficiently hard instances [20, 43]. Nevertheless, we advocate for a different choice of $\hat{c}$ and Γ in order to construct instances with which we can test our algorithms to their limits. In the following, we study the impact of $\hat{c}$ and Γ on the integrality gap of ROB to evaluate how they should be chosen to obtain hard instances.

Just like in the literature, we define our deviations $\hat{c}_i = \xi_i |c_i|$ with respect to the nominal coefficients. However, the factor ξ_i is chosen independently for each uncertain coefficient from an interval $\left[\underline{\xi}, \overline{\xi}\right]$. In order to investigate whether a strong correlation between $\hat{c}_i$ and c_i raises the integrality gap, we test different ranges $\left[\underline{\xi}, \overline{\xi}\right]$ with a fixed middle value $\left(\underline{\xi} + \overline{\xi}\right)/2$. We also test much higher values ξ_i, compared to the values chosen in [14, 20, 23, 43], since large deviations result in more difficult problems and deviations of even more than 100% are relevant in practice, as observed by Koster and Kutschka [65].

The choice of Γ must be made with particular care. For a problem ROB and an arbitrary optimal solution $\left(x^{\text{ROB}}, p^{\text{ROB}}, z^{\text{ROB}}\right)$, let $u\left(x^{\text{ROB}}\right) = \left|\left\{i \in [n] \middle| x_i^{\text{ROB}} = 1, \hat{c}_i > 0\right\}\right|$ be the number of uncertain variables contributing to the solution. If $\Gamma = 0$ or $\Gamma \geq u\left(x^{\text{ROB}}\right)$ holds, then either none or all coefficients of the chosen uncertain variables deviate to their maximum. This not only leads the idea of budgeted robust optimization to absurdity but also results in a relatively small integrality gap. Hence, Γ should be somewhere between 0 and $u\left(x^{\text{ROB}}\right)$ to obtain a difficult instance. Choosing Γ from a fixed set of values for all instances is therefore not appropriate for our purpose. While $\Gamma = 100$ may be suitable for large instances, it is too high for the smaller ones. Obviously, we cannot choose Γ with respect to $u\left(x^{\text{ROB}}\right)$, as we do not know the exact value in advance. Furthermore, in contrast to a practitioner solving a real problem, we have no insight into the structure of the diverse problems from the MIPLIB 2017. Hence, our best bet is to solve the nominal problem first, count the number $u\left(x^{\text{NOM}}\right)$ of uncertain variables appearing in the obtained optimal solution x^{NOM}, and choose Γ relative to $u\left(x^{\text{NOM}}\right)$. We will see in the following that for the choice $\Gamma = \gamma u\left(x^{\text{NOM}}\right)$, there is a correlation between $\gamma \geq 0$ and the integrality gap of ROB.

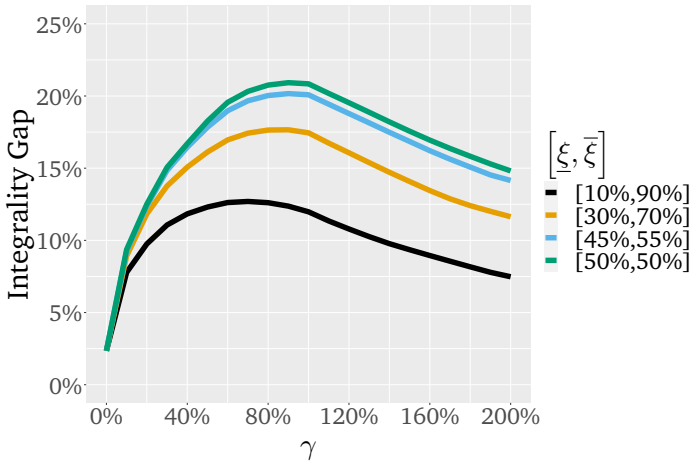


Figure 4.1. Shifted geometric mean of integrality gaps with shifting parameter $s = 1\%$ for different choices of γ and $\left[\underline{\xi}, \overline{\xi}\right]$.

Before determining the integrality gap of ROB for different choices of $\hat{c}$ and Γ, we have to select the nominal instances to be transformed into robust problems. Naturally, not all instances from the MIPLIB 2017 are suitable for this transformation. There are 1,065 instances available, of which we consider the ones that are labeled to be feasible, have an objective function, and consist only of binary variables. Furthermore, we only consider instances that have the "easy" label, as we cannot expect to solve the robust counterpart of hard instances. After this first selection, we try to solve the remaining 123 nominal instances within one hour using Gurobi. Of the instances that were solved to optimality, we select those whose computed optimal solution contains at least ten uncertain variables, i.e., $u\left(x^{\text{NOM}}\right) \geq 10$. This ensures that variables with uncertain coefficients have an impact on the optimal solution. From the remaining instances, we also had to exclude *pb-fit2d* and *supportcase11* due to numerical issues. After this final selection, we are left with 67 nominal instances for our computational study.

We construct robust problems from all these 67 instances by choosing $\Gamma = \left\lceil \gamma u\left(x^{\text{NOM}}\right) \right\rceil$, with $\gamma \in \{0\%, 10\%, \ldots, 200\%\}$, and $\hat{c}_i = \xi_i |c_i|$, where ξ_i is an independent and uniformly distributed random integer percentage in $\left[\underline{\xi}, \overline{\xi}\right] \in \{[10\%, 90\%], [30\%, 70\%], [45\%, 55\%], \{50\%\}\}$. We then solve the continuous relaxation, try to compute an optimal integer solution, and determine the integrality gap for each resulting robust problem. We use the branch and bound algorithm from Chapter 5 to compute the integer solutions, as it is our best performing algorithm, solving the highest number of instances. For a fair comparison of the integrality gap with respect to different choices of γ and $\left[\underline{\xi}, \overline{\xi}\right]$, we only consider the 44 underlying nominal instances for which we were able to compute an optimal solution for all combinations of γ and $\left[\underline{\xi}, \overline{\xi}\right]$.

Figure 4.1 shows for all combinations of γ and $\left[\underline{\xi}, \overline{\xi}\right]$ the shifted geometric mean of integrality gaps over all considered instances. The integrality gap first increases monotonically in γ, peaks at latest at $\gamma = 90\%$ and decreases afterwards for all choices of $\left[\underline{\xi}, \overline{\xi}\right]$. This suggests that the maximum integrality gap is usually achieved for some $\Gamma \in \left[0, u\left(x^{\text{NOM}}\right)\right]$. We cover this

spectrum by choosing $\gamma \in \{10\%, 40\%, 70\%, 100\%\}$ in our computational study on robustified MIPLIB instances in Section 4.6.7. Note that the higher values of Γ are most likely way too conservative for a practical problem. However, we are not interested in constructing meaningful practical instances but instances where uncertainty contributes to the difficulty of the problem.

Figure 4.1 shows higher integrality gaps for narrow intervals $\left[\underline{\xi}, \overline{\xi}\right]$, which suggests that a strong correlation between $\hat{c}_i$ and c_i results in hard robust instances. Although choosing $\underline{\xi} = \overline{\xi}$ seems to be beneficial in this regard, we chose $\underline{\xi} \neq \overline{\xi}$ for our computational study, since fixing ξ_i may result in structural properties that lead to a biased performance of the tested algorithms. For example, Monaci and Pferschy [76] showed that an adaptation of the classical greedy heuristic for the binary knapsack problem has a better worst-case performance for the robust knapsack if $\max\{\xi_i/\xi_j | i, j \in [n]\}$ is small. Moreover, choosing $\underline{\xi} = \overline{\xi}$ usually provides fewer different deviations $\{\hat{c}_1, \ldots, \hat{c}_n\}$, and thus fewer possible optimal values for z (cf. Section 5.5). This benefits algorithms exploring these possible choices for z, just like our branch and bound algorithm from Chapter 5. Therefore, we choose $\left[\underline{\xi}, \overline{\xi}\right] = [45\%, 55\%]$. In addition, we take smaller and larger deviations into account by also considering $\left[\underline{\xi}, \overline{\xi}\right] \in \{[5\%, 15\%]\,[95\%, 105\%]\}$ in Section 4.6.7.

4.6.4 Robust Independent Set

Before we evaluate the performance of our recycling approaches on the robustified MIPLIB instances, we study their potential on robust versions of classical combinatorial optimization problems. We first show the effect of recycling a class of well-known valid inequalities in a separation procedure. For this, we consider the robust maximum weighted independent set problem on a graph $G = (V, E)$. The robust counterpart of the standard formulation with decision variables $x_v \in \{0, 1\}$ for each node $v \in V$ and *edge constraints* $x_v + x_w \leq 1$ for $\{v, w\} \in E$ reads

$$
\begin{aligned}
\max \sum_{v \in V} c_v x_v - \left(\Gamma z + \sum_{v \in V} p_v \right) & \\
\text{s.t. } x_v + x_w \leq 1 & \qquad \forall \{v, w\} \in E \\
p_v + z \geq \hat{c}_v x_v & \qquad \forall v \in V \\
x \in \{0,1\}^V, p \in \mathbb{R}^V_{\geq 0}, z \in \mathbb{R}_{\geq 0}. &
\end{aligned}
$$

As seen in Section 4.3, recycling a clique inequality $\sum_{v \in Q} x_v \leq 1$ yields a facet-defining inequality for all cliques $Q \subseteq V$. We compare the separation of recycled clique inequalities in the root node of the branching tree against the robust default formulation $\mathcal{F}^{\text{ROB}}$, which solely uses the constraints $p_i + z \geq \hat{c}_i x_i$. For this, we use Gurobi's callback to add the recycled inequalities as user cuts [52]. Every time Gurobi invokes the callback in the root node and reports a current optimal fractional solution $(\tilde{x}, \tilde{p}, \tilde{z}) \in \mathcal{F}^{\text{ROB}}$, we heuristically separate cliques

Table 4.1. Computational results for 230 instances of the robust maximum weighted independent set problem. We use different nominal formulations and test with Gurobi's own cuts enabled or disabled.

		DEF				RECsepClq			
formulation	GCuts	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap
edge	enable	22	26.13	14.12	1427.09%	21	33.22	16.99	1206.91%
	disable	42	51.23	22.91		20	20.93	11.72	
clique	enable	61	116.13	67.66	135.50%	64	135.15	77.06	56.49%
	disable	74	168.47	83.30		53	82.07	49.97	

$Q \subseteq V$ for which the recycled inequality $z + \sum_{v \in Q} p_v \geq \sum_{v \in Q} \hat{c}_v x_v$ is violated. We do so as in Section 4.4.2, that is, we heuristically solve maximum weighted clique problems on G with weights $\hat{c}_v \tilde{x}_v - \tilde{p}_v$. To separate many recycled inequalities at once, we extend each node $v \in V$ with $\hat{c}_v \tilde{x}_v - \tilde{p}_v > 0$ greedily to a clique $Q_v \subseteq V$ with $v \in Q_v$. For this, we start with $Q_v = \{v\}$ and then iteratively extend the clique with nodes $w \in \bigcap_{u \in Q_v} N(u)$ such that $\hat{c}_w \tilde{x}_w - \tilde{p}_w$ is maximal and non-negative. Finally, we return the corresponding recycled inequality to Gurobi if its violation is positive.

We use the graphs of the second DIMACS implementation challenge on the clique problem [56] as a basis for our test instances. We select those 46 out of the 66 DIMACS graphs that have at most 500 nodes, as otherwise the nominal problem is already very hard. We generate independent and uniformly distributed weights $c_v \in \{900, \dots, 1000\}$ for each $v \in V$. The deviations $\hat{c}_v$ and the uncertainty budget Γ are chosen in accordance with our observations from the previous section. We choose $\hat{c}_v = \lceil \xi_v c_v \rceil$, where $\xi_v \in [0.45, 0.55]$ are independent and uniformly distributed random variables. Since Γ should be somewhere between 0 and $u\left(x^{\text{ROB}}\right)$, that is the number of variables in an optimal solution with $x_i^{\text{ROB}} = 1$, we greedily compute an inclusion-wise maximal independent set $S \subseteq V$ and define $\Gamma = \left\lfloor \frac{|S|}{2} \right\rfloor$. We randomly generate five robust independent set problems for each of the 46 DIMACS graphs, leaving us with 230 robust instances.

Detailed computational results can be found in [47]. We show aggregated results for the default formulation (DEF) and the separation of recycled clique inequalities (RECsepClq) in Table 4.1. The table shows the number of instances that could not be solved within the time limit (timeout), the shifted geometric mean of the computation times (time), the shifted geometric mean of the primal-dual integrals (P-D integral) and the shifted geometric mean of the integrality gaps (int gap).

We see that the shifted geometric mean of the integrality gaps is reduced absolutely by roughly 220 percentage points from 1427.09% to 1206.91% when using recycled clique inequalities. For computing these gaps, we use the best computed primal bound as well as the dual bound obtained by heuristically separating recycled clique inequalities for subsequent continuous relaxations until no violated inequalities are found. While the absolute reduction of the integrality gap is quite impressive, the relative reduction does not adequately reflect the strength of the recycled inequalities. This is due to the large integrality gap of the nominal problem, which constitutes a major part of the total gap.

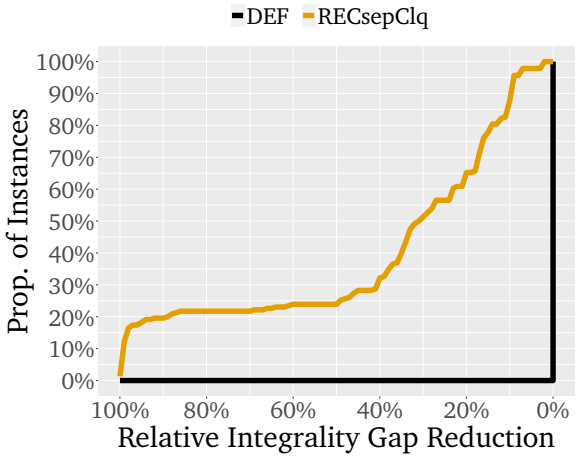


Figure 4.2. Cumulative distribution of integrality gap reductions using the clique formulation.

To reduce the integrality gap of the nominal problem, we test an additional formulation for the independent set problem. Here, we replace every constraint $x_v + x_w \leq 1$ for an edge $\{v, w\} \in E$ with a constraint $\sum_{v \in Q} x_v \leq 1$ for a clique $Q \subseteq V$ with $\{v, w\} \subseteq Q$. This *clique formulation* is valid and much stronger than the previous *edge formulation*, since all edge constraints are dominated by their corresponding clique constraint. Thus, the clique formulation reduces the contribution of the nominal problem to the integrality gap. Indeed, Table 4.1 shows that separating recycled clique inequalities yields a relative reduction of the integrality gaps by 58.3% when using the clique formulation. Figure 4.2 gives a more detailed view on the improvement by showing for how many instances the integrality gap is reduced by at least a specific percentage when comparing RECsepClq with DEF. Here, we see that recycling cliques reduces the integrality gap by at least 30% for more than 50% of all instances. Moreover, we have a reduction of 90% for almost 20% of the instances.

The clique formulation is not of practical interest apart from the analysis of the integrality gap, as Gurobi seems to be better trained on the edge formulation. Table 4.1 shows for the edge formulation that we solve one more instance when recycling clique inequalities but have an increase in the computation time and the primal-dual integral. This seems to be due to some interference with Gurobi's own cutting planes. When Gurobi's cutting planes are disabled, then recycling is much better than using the default formulation, as it approximately halves the computation time and the primal-dual integral. In fact, disabling Gurobi's cuts and using recycled clique inequalities (RECsepClq-noGCuts) is the overall best performing approach, solving the most instances in the least amount of computation time. This is supported by Figure 4.3, which shows for each approach the proportion of instances whose computation time or primal-dual integral are below a specific value. While DEF, RECsepClq, and RECsepClq-noGCuts solve roughly the same number of instances within the first 280 seconds and up to a primal-dual integral of 160, RECsepClq-noGCuts clearly performs better afterwards on the harder instances.

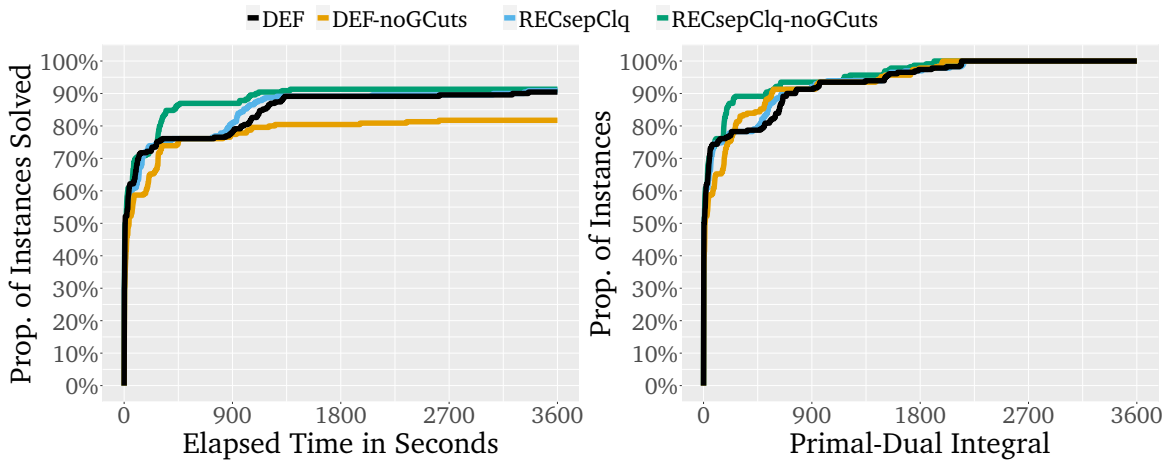


Figure 4.3. Cumulative distribution of computation times and primal-dual integrals when using the edge formulation.

4.6.5 Robust Bipartite Matching

To study the recycling of model constraints, we now consider the robust maximum weighted matching problem on a bipartite graph with nodes V and edges E. Using decision variables $x_e \in \{0,1\}$ for each edge $e \in E$, the robust problem reads

$$\begin{aligned}
\max \ & \sum_{e\in E} c_e x_e - \left(\Gamma z + \sum_{e\in E} p_e\right) \\
\text{s.t.} \ & \sum_{e\in\delta(v)} x_e \leq 1 && \forall v \in V \\
& p_e + z \geq \hat{c}_e x_e && \forall e \in E \\
& x \in \{0,1\}^E, p \in \mathbb{R}^E_{\geq 0}, z \in \mathbb{R}_{\geq 0}.
\end{aligned}$$

As mentioned above, the bipartite matching problem has the interesting property that the constraints $\sum_{e\in\delta(v)} x_e \leq 1$ for $v \in V$ and $x_e \geq 0$ for $e \in E$ already define the convex hull of the nominal problem [64, Section 11.1]. Moreover, since all constraints are recyclable, the properties from Corollary 19 are fulfilled, which allows for an exact separation of recycled inequalities in linear time, and thus enables us to test their strength to the limit.

We randomly generate instances by first dividing a set of nodes $V = [n]$ into two partitions $U = [\lceil \frac{n}{2} \rceil]$ and $W = \{\lceil \frac{n}{2} \rceil + 1, \dots, n\}$. Afterwards, we sample for each node $u \in U$ a random number $\phi_u \in [0,1]$ that models the probability with which an edge incident to u exists. Then for every $w \in W$, we add the edge $\{u,w\}$ with probability ϕ_u. Given the constructed graph, we generate weights c_e and deviations $\hat{c}_e$ analogously to the independent set problem. Every weight is a random number $c_e \in \{900, \dots, 1000\}$ and the correlated deviations are $\hat{c}_e = \lceil \xi_e c_e \rceil$ with $\xi_e \in [0.45, 0.55]$. Finally, as the number of edges in a solution will most likely be near to $\frac{n}{2}$, we set $\Gamma = \lfloor \frac{n}{4} \rfloor$. We use this procedure to generate ten instances each for different numbers of nodes $n \in \{50, 100, 150\}$.

Table 4.2. Computational results for the robust maximum weighted bipartite matching problem. We generate ten instances per number of nodes and test with Gurobi's own cuts enabled or disabled.

		DEF				RECcons				RECconsSepCons			
nodes	GCuts	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap
50	enable	0	1.73	0.04	19.532%	0	0.48	0.04	0.326%	0	0.77	0.04	0.319%
	disable	10	3600.00	192.48		0	0.25	0.05		0	0.35	0.05	
100	enable	9	2269.14	3.49	22.820%	0	4.50	0.16	0.319%	0	6.28	0.17	0.316%
	disable	10	3600.00	550.96		0	15.16	0.21		0	16.26	0.20	
150	enable	7	2223.68	2.56	23.660%	0	150.40	0.59	0.269%	0	168.12	0.61	0.265%
	disable	10	3600.00	635.91		6	1887.56	2.51		8	1960.65	2.65	

Detailed computational results can be found in [47]. Table 4.2 shows aggregated results for the robust default formulation (DEF) and two different approaches for using recycled inequalities. The first approach directly recycles all constraints $\sum_{e\in\delta(v)} x_e \leq 1$ for $v \in V$ (RECcons). The second approach additionally separates violated inequalities $\sum_{e\in E'} x_e \leq 1$ with $E' \subseteq \delta(v)$ for $v \in V$ in the root node of the branch and bound tree (RECconsSepCons).

It is evident that recycling inequalities is significantly better than solely using the default formulation. We observe a considerable strengthening of the formulation, leading to a reduction of the integrality gaps by 98.9% for $n = 150$ nodes. This strength also translates to a higher number of instances solved and much lower computation times. For $n = 150$ with Gurobi's cuts enabled, RECcons has 93.2% lower computation times than DEF. Still, the primal-dual integral is quite low for DEF, suggesting that the solver is very close to optimality from the beginning. This changes once we disable Gurobi's cuts. In this case, DEF is not even able to solve any instance. Furthermore, the primal-dual integrals are 253-times as large as those of RECcons for $n = 150$.

The recycling of dominated inequalities $\sum_{e\in E'} x_e \leq 1$ compared to the sole recycling of constraints $\sum_{e\in\delta(v)} x_e \leq 1$ yields an improvement of the integrality gap. However, as the recycled constraints already perform well for these instances, the improvement in the continuous relaxation is very small. In fact, the minor strengthening of the continuous relaxation cannot compensate for the computational load imposed by the additional inequalities, which leads to higher computation times. Later, our study on the MIPLIB instances will reveal that recycling dominated inequalities can have a much greater effect on the integrality gap, and thus lead to lower computation times.

4.6.6 Robust Bipartite Matching with Penalties

Until now, we only considered problems for which all valid inequalities are recyclable. In order to test our approach of partially recycling from Section 4.5, we alter the bipartite matching problem from above so that none of the constraints are recyclable. To this end, we allow a solution to match each node $v \in V$ up to two times. However, when matching v

Table 4.3. Computational results for the robust maximum weighted bipartite matching with penalties problem. We generate ten instances per number of nodes and test with Gurobi's own cuts enabled or disabled.

		DEF				RECsepEst				RECsepPart			
nodes	GCuts	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap
50	enable	0	65.67	0.35	18.14%	0	67.20	0.37	10.58%	0	27.07	0.17	1.67%
	disable	10	3600.00	289.70		9	2907.90	72.20		0	127.79	0.48	
100	enable	10	3600.00	17.79	20.89%	10	3600.00	17.73	11.02%	10	3600.00	13.84	2.02%
	disable	10	3600.00	524.65		10	3600.00	252.57		10	3600.00	35.51	
150	enable	10	3600.00	28.15	20.80%	10	3600.00	27.00	9.81%	10	3600.00	21.27	2.03%
	disable	10	3600.00	575.09		10	3600.00	267.58		10	3600.00	45.15	

more than once, we have to pay a penalty $c_v > 0$. We introduce decision variables $y_v \in \{0,1\}$ indicating whether node $v \in V$ is matched twice and obtain the following robust problem

$$\begin{aligned}
\max \ & \sum_{e\in E} c_e x_e - \sum_{v\in V} c_v y_v - \left(\Gamma z + \sum_{e\in E} p_e\right) \\
\text{s.t.} \ & \sum_{e\in\delta(v)} x_e - y_v \leq 1 && \forall v \in V \\
& p_e + z \geq \hat{c}_e x_e && \forall e \in E \\
& x \in \{0,1\}^E, y \in \{0,1\}^V, p \in \mathbb{R}^E_{\geq 0}, z \in \mathbb{R}_{\geq 0}.
\end{aligned}$$

We use the same graphs and parameters as in the previous section with random penalties $c_v \in \{450, \dots, 500\}$. This is on average half the value $c_e \in \{900, \dots, 1000\}$ of the edges $e \in E$, and thus the benefit of matching two already matched nodes is on average equal to the received penalty. Note that we do not consider uncertainties on the penalty coefficients.

Detailed computational results can be found in [47]. Table 4.3 shows aggregated results for the robust default formulation (DEF) as well as the separation of recycled inequalities via estimating $\sum_{e\in\delta(v)} x_e \leq 2$ (RECsepEst), as in Section 4.4.1, and the separation of partially recycled inequalities (RECsepPart), as in Section 4.5. Again, we only separate within the root node of the branch and bound tree.

RECsepEst still significantly improves the formulation, reducing the integrality gap by 52.8% for $n = 150$. However, the effect is clearly weaker compared to the improvement for the original matching problem. RECsepPart is considerably stronger, reducing the integrality gap by 90.2%. We have a closer look at the computed solutions in order to assess whether this observed reduction is meaningful. Note that partially recycled inequalities are especially strong when the variables to be lifted are zero. Hence, if we had $y_v = 0$ for all $v \in V$, then the reduction might only be due to a favorable problem generation. However, we observe that we have $y_v = 1$ for approximately three-fourths of all nodes $v \in V$ in the computed solutions, showing that the instance generation is not in our favor.

Even with the partially recycling procedure, the matching with penalties is apparently much harder to solve. We cannot compare computation times, since all approaches always hit the time limit for $n \in \{100, 150\}$. However, we still see that the partially recycling results in

significantly smaller primal-dual integrals compared to the other approaches, especially when Gurobi's cuts are disabled.

4.6.7 Robustified MIPLIB Instances

We have seen so far that, given the right setting, recycling inequalities can have a significant impact on the strength of formulations, and thus on the computational performance. To evaluate the use of recycled inequalities for practical instances, we also perform tests on the broad set of 804 robust MIPLIB instances that we generated in Section 4.6.3.

We consider four different approaches for integrating recycled inequalities in the optimization process. Note that we do not have any insight into the structure of the nominal problems of our test instances, and thus only recycle inequalities in a generic fashion based on the constraints $Ax \leq b$. Our first two approaches are as described in Section 4.4.1. We test the direct addition of recycled constraints to the default formulation (RECcons) and the separation of violated recycled constraints in the root node of the branch and bound tree (RECsepCons). For the third approach, we first separate recycled constraints as for RECsepCons. Once we do not find any violated recycled constraints, we solve SLP from Section 4.4.3 for a more refined separation (RECsepLP). The fourth approach is as RECsepCons, but we also consider partially recyclable constraints as in Section 4.5 (RECsepPart).

The relaxation value used for computing the integrality gap is again computed by separating inequalities until no violated inequality is found. We use a time limit of 3,600 seconds, which leads to a termination of the separation for some instances although we are still able to find violated inequalities. As a result, we may compute a worse relaxation value for RECsepCons than for RECcons. As the relaxation value provided by RECsepCons is in theory stronger than that of RECcons, we report the maximum computed value of both for RECsepCons. The same holds for RECsepLP and RECsepPart, which are in theory stronger than RECsepCons. For these, we report the maximum relaxation value of RECcons, RECsepCons, and the respective algorithm.

Detailed computational results can be found in [47]. Table 4.4 shows aggregated results for the four recycling approaches and the default formulation. As for the combinatorial problems in the last sections, recycling inequalities is very effective when Gurobi's cuts are disabled. In this setting, the recycling approaches require between 39.5% and 45.1% less time in the shifted geometric mean over all instances. When only considering the affected instances, that are the instances for which at least one of the recycling approaches provides a better integrality gap compared to the default formulation, the speed-up is even higher. Out of the 804 instances in our test set, 608 were affected by recycling. For these, the recycling approaches require between 49.5% and 55.2% less time, which clearly highlights the practical potential of recycling inequalities.

Table 4.4. Computational results for robustified MIPLIB instances. We test with Gurobi's own cuts enabled or disabled and show results aggregated for all 804 instances as well as only the 608 on which at least one recycling approach had an effect.

		DEF				RECcons				RECsepCons			
instances	GCuts	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap
all	enable	348	226.51	35.35	15.90%	333	198.96	35.95	8.61%	309	193.71	31.97	7.87%
	disable	497	670.44	99.48		394	368.21	51.14		409	395.63	51.69	
affected	enable	273	303.20	47.59	24.64%	254	252.92	45.17	11.15%	235	246.40	41.63	9.93%
	disable	398	842.90	147.33		295	377.53	60.50		311	416.76	61.85	

		RECsepLP				RECsepPart			
instances	GCuts	timeout	time	P-D integral	int Gap	timeout	time	P-D integral	int Gap
all	enable	308	199.96	33.16	6.60%	316	195.51	32.33	7.86%
	disable	403	405.30	52.97		408	400.15	52.53	
affected	enable	234	251.87	42.65	7.92%	242	247.77	41.99	9.92%
	disable	305	425.28	62.88		310	423.32	63.24	

This performance boost is due to the substantially strengthened continuous relaxations. RECcons already cuts the integrality gap nearly in half, from 15.90% to 8.61%. RECsepCons yields an even better integrality gap, since we also recycle dominated inequalities in this approach. Note that the relative reduction of the integrality gap using RECsepCons instead of RECcons (8.6% relative reduction) is much larger compared to our observations for the robust bipartite matching problem (1.5% relative reduction for $n = 150$ nodes). This indicates that recycling dominated inequalities is more important when the coefficients in the constraints are not all the same. This is in line with our observation in Example 14 from Section 4.3.

RECsepPart yields nearly no improvement of the integrality gap compared to RECsepCons. While we were able to prove the great potential of this approach for the matching with penalties, the considered instances of the MIPLIB apparently do not contain many constraints of the necessary structure with both positive and negative coefficients on the left-hand side. In contrast, RECsepLP yields another substantial improvement of the integrality gap down to 6.60%. This is a relative reduction of 58.5% compared to the integrality gap of the default formulation. When only considering the affected instances, we even see an improvement from 24.64% to 7.92%, which is a relative reduction of 67.9%.

To get a better understanding of the improvement on the affected instances, we show in Figure 4.4 for how many of these the integrality gap is reduced by at least a specific percentage. Note that RECsepCons and RECsepPart have nearly identical lines, as they mostly compute the same cuts for these instances. Of the 608 affected instances, RECcons, RECsepCons, and RECsepPart close the integrality gap completely for 19 and RECsepLP even for 22 instances. Interestingly, this includes not only instances with a low default integrality gap but 12 instances with a default gap of more than 10%, of which one is even 74,417%. Moreover, these 12 instances are based on 7 different nominal instances from the MIPLIB 2017. That is, for more than every tenth nominal instance, there is at least one corresponding robust instance for which we close the integrality gap from over 10% down to zero.

In addition to these extreme cases, we see that RECsepLP is able to halve the integrality gap for 51% of the instances. Furthermore, RECsepLP achieves a reduction for some problems on which RECcons, RECsepCons, and RECsepPart have no effect. This gives hope that

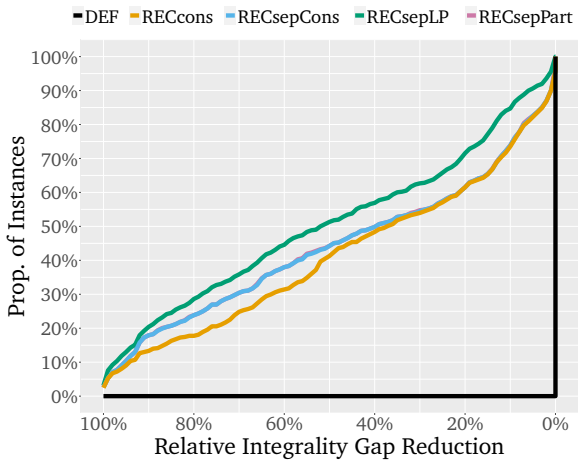


Figure 4.4. Cumulative distribution of integrality gap reductions for affected instances.

practitioners with a good understanding of their problem might be able to benefit from problem specific fast separations of recycled inequalities that do not correspond directly to the constraints $Ax \leq b$.

The strong continuous relaxations also translate to an improved performance when Gurobi's cuts are enabled, with all recycling approaches solving more instances in shorter time. Table 4.4 shows that RECsepCons has the lowest shifted geometric mean for the computation time and primal-dual integral. Compared to the default formulation, the computation times are 14.5% lower for all instances and 18.7% for the affected ones. Moreover, the lower primal-dual integral implies that separating recycled constraints improves the performance across the whole optimization process. RECsepLP solves one instance more but is on average slightly slower than RECsepCons. This is because the overhead of handling and solving SLP only pays off for specific instances. RECsepPart performs worse than RECsepCons, as both compute almost the same cuts, with RECsepPart requiring more time doing so. RECcons is overall slower compared to the other recycling approaches because many of the added recycled constraints are actually uninteresting for strengthening the continuous relaxation, and thus impose unnecessary computational load due to the bigger constraint matrix. Nevertheless, we will see in the following that RECcons can actually be very useful in practice.

Just like for the integrality gap, Figure 4.5 shows the cumulative distribution of performance indicators for each approach on the set of affected instances. We see that each recycling approach solves at any point in time more instances than DEF. The same holds for the primal-dual integral with the exception of RECcons, which has a high primal-dual integral for more instances than DEF. While the differences in the primal-dual integrals appear to be small, a paired Wilcoxon signed-rank test [93] reveals that for all algorithms but RECcons the improvement in the computation time and the primal-dual integral is significant with a confidence level of at least 98%.

Figure 4.6 displays the performance indicators of our recycling approaches relative to the default formulation. In the left graphic, we see that all recycling approaches are 8-times as

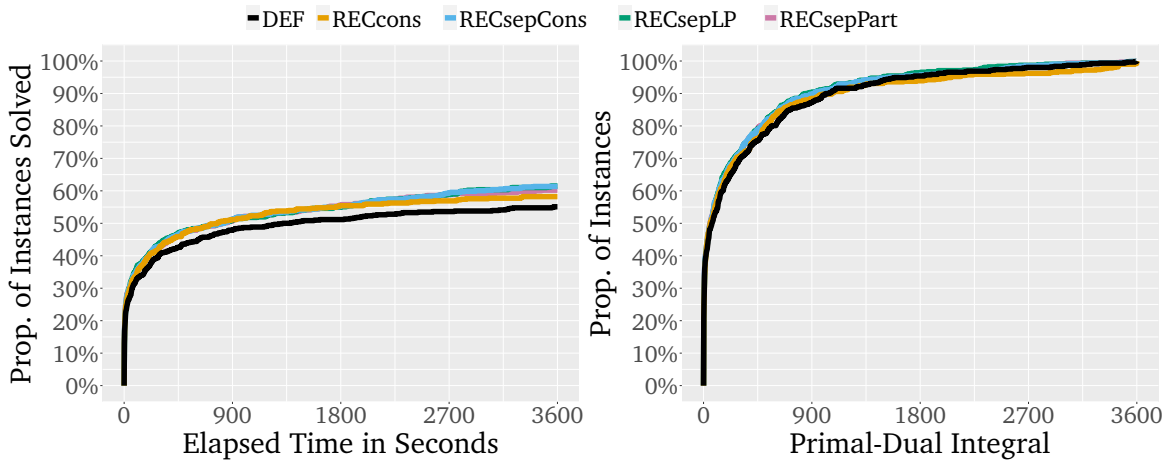


Figure 4.5. Cumulative distribution of computation times and primal-dual integrals for the set of affected instances.

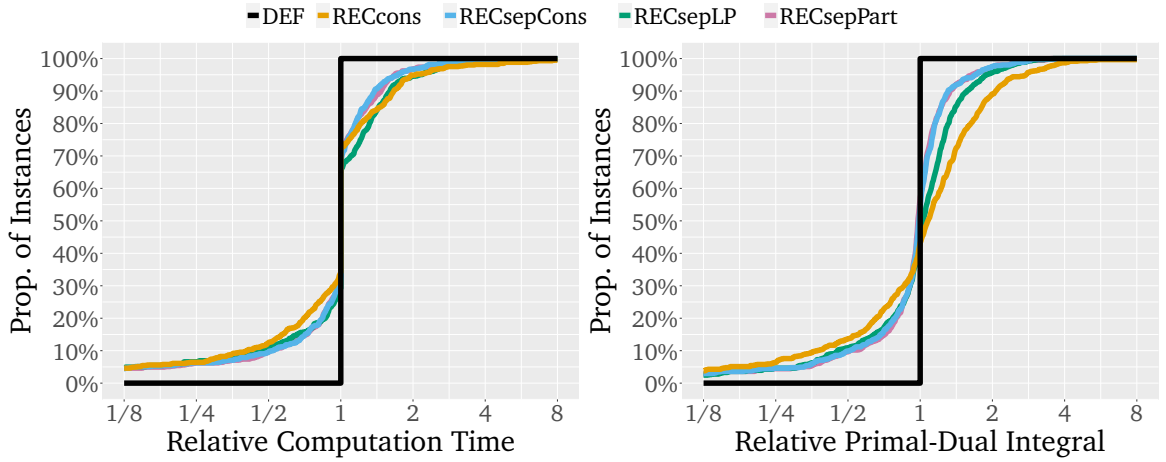


Figure 4.6. Cumulative distribution of computation times and primal-dual integrals relative to the default formulation for the set of affected instances.

fast for roughly 5%, while requiring 8-times as much time for only 0.5% of the instances. The most balanced of all approaches is RECsepCons, which is 2-times as fast for 9.5% and halve as fast for 3.3% of the instances. Similar observations can be made for the primal-dual integral in the right graphic. However, the most interesting observation about Figure 4.6 is that RECcons' and RECsepLP's performance is quite extreme. Both approaches perform badly for more instances than RECsepCons, but the number of instances on which they perform very well is also higher.

The extreme performance is no surprise for RECsepLP, as we already observed above that the higher effort invested in the separation pays off for some specific instances. For RECcons, we see that, given the proper problem structure, recycling constraints directly is not only the most easy approach but also very efficient. This is good news for the practical use of recycled inequalities, as practitioners will often know whether their optimization problem contains promising recyclable constraints. This may, for example, include clique constraints or (almost) binding capacity restrictions. Recycling precisely these constraints, and not all

as we do here for RECcons, might result in a good speed-up for the respective problem. In comparison to RECsepCons, this yields the advantage that the added recycled inequalities are present from the beginning of the optimization process, which is beneficial because the solver can use the additional information for preprocessing. Future engineering might enable us to combine the stable performance of RECsepCons with the performance peaks of RECcons.

4.7 Conclusion

After identifying in the previous chapter that the standard formulation for robust combinatorial optimization problems with budgeted uncertainty is weak, we introduced a compact bilinear formulation, which is as strong as the strongest possible polyhedral formulation. To benefit from this in practice, we combined the bilinear formulation's strength with structural properties provided by valid inequalities for the nominal problem to obtain the new class of recycled inequalities.

Given a valid knapsack inequality for the nominal problem, the corresponding recycled inequality can be derived in linear time. This yields the possibility to reuse model-constraints and well known classes of valid inequalities in order to strengthen the continuous relaxation of the robust problem. We highlighted the theoretical strength of such recycled inequalities by proving that they often define facets of the convex hull of the robust problem, even when the underlying valid inequality is dominated.

To make recycled inequalities usable in practice, we discussed different separation procedures that either depend on separation algorithms for classical cutting planes or simply work on the constraint matrix in a generic fashion. One of these separation procedures even implies that recycled inequalities can be separated exactly in polynomial time if the convex hull of the nominal problem is known. Furthermore, we showed that inequalities that are not of the knapsack type can be partially recycled on a restricted solution space and lifted afterwards to obtain a valid inequality for the robust problem.

To test the strength of recycled inequalities and the practicability of their separation, we conducted an extensive computational study on robust versions of three classes of combinatorial problems and a carefully generated set of hard robust instances based on real-world problems from the MIPLIB 2017. Our experiments show that recycled inequalities are not only interesting from a theoretical point of view but can also yield a significant speed-up in the optimization process.

For future research, it would be interesting to further analyze the recycling of non-knapsack inequalities and evaluate whether one can obtain facet-defining robust inequalities from specific classes of nominal inequalities. Furthermore, the effect of recycling should be tested for robust problems with uncertain constraints.

5 A Branch and Bound Algorithm

In the last chapter, we proposed a new class of valid inequalities, which can be used easily in the standard approach of solving ROB directly as an MILP. In this chapter, we follow a slightly more combinatorial approach, which can be seen as a more sophisticated version of solving ROB via the nominal subproblems NOS (z) for $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$. For this, we do not fix z to one value, but split its domain in a specialized branch and bound algorithm. We will show several structural results while introducing our algorithm and later demonstrate its performance in a computational study.

5.1 Strong Linear Formulations for Bounded z

We already noted in the previous chapter that the bilinear formulation $\mathcal{F}^{\text{BIL}}$ will also be the foundation for the algorithms proposed in this chapter. To understand how we can make further use of it, remember from Section 4.1 that $\mathcal{F}^{\text{BIL}}$ is closely related to the nominal subproblems NOS (z). When fixing z, the bilinear formulation becomes not only linear, but it also holds $p_i = (\hat{c}_i - z)^+ x_i$ for all $i \in [n]$ in an optimal solution (x, p, z). Hence, the problem of optimizing over $\mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times \{z\})$ is equivalent to

$$\begin{aligned} \min\ & \Gamma z + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - z)^+ \right) x_i \\ \text{s.t. } & Ax \leq b \\ & x \in [0,1]^n , \end{aligned}$$

which is the continuous relaxation of the nominal subproblem NOS (z). The strength of the linearization for fixed z suggests that we may also derive strong linearizations of $\mathcal{F}^{\text{BIL}}$ for general restrictions on z, that is $z \in Z \subseteq \mathbb{R}_{\geq 0}$. In this section, we introduce such a linearization, which will be a key component of our branch and bound algorithm.

Remember that $\{\hat{c}_0, \dots, \hat{c}_n\}$, with $\hat{c}_0 = 0$, always contains an optimal value for z and let $Z \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$. With some abuse of notation, we will write $\underline{z} = \min(Z)$ and $\overline{z} = \max(Z)$ as well as $\underline{z}' = \min(Z')$ as well as $\overline{z}' = \max(Z')$ for some $Z' \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$ for the remainder of this chapter. Assuming that there exists an optimal solution (x, p, z) to ROB with $z \in Z$, we can restrict ourselves to searching for solutions in the domain $\mathbb{R}^{2n} \times [\underline{z}, \overline{z}]$. We use this restriction to obtain a linear relaxation of the restricted bilinear formulation $\mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$.

Lemma 23. *The linear inequalities*

$$p_i + z \geq (\hat{c}_i - \underline{z})^+ x_i + \underline{z} \tag{5.1}$$

and

$$p_i \geq (\hat{c}_i - \overline{z})^+ x_i \tag{5.2}$$

are valid for all $(x, p, z) \in \mathcal{F}^{BIL} \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$.

Proof. Since $p_i + zx_i \geq \hat{c}_i x_i$ and $p_i \geq 0$ hold for $(x, p, z) \in \mathcal{F}^{\text{BIL}}$, the bound $\underline{z} \leq z$ yields

$$\begin{aligned} p_i + zx_i \geq \hat{c}_i x_i &\Leftrightarrow p_i + (z - \underline{z} + \underline{z})\, x_i \geq \hat{c}_i x_i \\ &\Rightarrow p_i + z - \underline{z} + \underline{z} x_i \geq \hat{c}_i x_i \\ &\Leftrightarrow p_i + z \geq (\hat{c}_i - \underline{z})^+ x_i + \underline{z}. \end{aligned}$$

Furthermore, due to $z \leq \overline{z}$, we obtain

$$\begin{aligned} p_i + zx_i \geq \hat{c}_i x_i &\Leftrightarrow p_i \geq (\hat{c}_i - z)^+ x_i \\ &\Rightarrow p_i \geq (\hat{c}_i - \overline{z})^+ x_i. \end{aligned}$$

□

Note that the inequalities (5.1) and (5.2) are stronger than the original robustness constraints $p_i + z \geq \hat{c}_i x_i$ and $p_i \geq 0$ of $\mathcal{F}^{\text{ROB}}$ in the case of $\underline{z} > 0$ and $\hat{c}_i > \overline{z}$ respectively. Both inequalities address the problem of the original formulation, which is that one can decrease x_i in a fractional solution down to $x_i \leq \frac{z}{\hat{c}_i}$ in order to choose $p_i = 0$, even if we have $\hat{c}_i > z$. Given a lower bound $z \geq \underline{z}$, inequality (5.1) reduces the benefit of decreasing x_i, as the right-hand side only decreases with the factor $(\hat{c}_i - \underline{z})^+$ instead of $\hat{c}_i$. For an upper bound $z \leq \overline{z}$, inequality (5.2) guarantees that p_i is not zero for $\hat{c}_i > \overline{z}$ and $x_i > 0$ by using the fact that the value of p_i is at least $\hat{c}_i - \overline{z}$ if we have $\hat{c}_i > \overline{z}$ and $x_i = 1$.

Using these strengthened inequalities, we obtain the *robust subproblem*

$$\text{(ROB}\,(Z)\text{)} \qquad \begin{aligned} &\min \Gamma z + \sum_{i \in [n]} c_i x_i + p_i \\ &\text{s.t. } (x, p, z) \in \mathcal{F}\,(Z)\,, x \in \{0,1\}^n \end{aligned}$$

over the linear formulation

$$\mathcal{F}\,(Z) = \left\{ (x,p,z) \middle| \begin{array}{ll} Ax \leq b & \\ p_i + z \geq (\hat{c}_i - \underline{z})^+ x_i + \underline{z} & \forall i \in [n] \\ p_i \geq (\hat{c}_i - \overline{z})^+ x_i & \forall i \in [n] \\ x \in [0,1]^n\,, p \in \mathbb{R}^n, z \in [\underline{z}, \overline{z}] & \end{array} \right\}.$$

As shown in Lemma 23, formulation $\mathcal{F}(Z)$ is a relaxation of the restricted bilinear formulation $\mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$. Note that $\mathcal{F}(Z)$ becomes stronger, the narrower the bounds of Z are, i.e., for Z, Z' with $[\underline{z}, \overline{z}] \subsetneq [\underline{z}', \overline{z}']$, we have $\mathcal{F}(Z) \subsetneq \mathcal{F}(Z') \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$ for non-trivial cases. The following statement shows that $\mathcal{F}(Z)$ is even as strong as $\mathcal{F}^{\text{BIL}}$ in the case where z equals one of the bounds $\underline{z}, \overline{z}$.

Proposition 24. *It holds* $\mathcal{F}(Z) \cap (\mathbb{R}^{2n} \times \{\underline{z}, \overline{z}\}) = \mathcal{F}^{BIL} \cap (\mathbb{R}^{2n} \times \{\underline{z}, \overline{z}\})$.

Proof. Consider a solution $(x, p, z) \in \mathcal{F}(Z) \cap (\mathbb{R}^{2n} \times \{\underline{z}, \overline{z}\})$. For $z = \underline{z}$, we have

$$p_i + zx_i = p_i + z - z + zx_i \geq (\hat{c}_i - \underline{z})^+ x_i + \underline{z} - z + zx_i \geq (\hat{c}_i - \underline{z})\, x_i + \underline{z} - \underline{z} + \underline{z}x_i = \hat{c}_i x_i$$

and for $z = \overline{z}$, it holds

$$p_i + zx_i \geq (\hat{c}_i - \overline{z})^+ x_i + zx_i \geq (\hat{c}_i - \overline{z})\, x_i + \overline{z}x_i = \hat{c}_i x_i.$$

This implies $(x, p, z) \in \mathcal{F}^{\text{BIL}}$, and thus $\mathcal{F}(Z) \cap (\mathbb{R}^{2n} \times \{\underline{z}, \overline{z}\}) \subseteq \mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times \{\underline{z}, \overline{z}\})$. The other direction follows from Lemma 23. □

The improved formulation $\mathcal{F}(Z)$ comes with the cost of a larger constraint matrix compared to $\mathcal{F}^{\text{ROB}}$, as we have $p_i \geq (\hat{c}_i - \overline{z})^+ x_i$ instead of $p_i \geq 0$. This is a disadvantage in practice, as more constraints result in larger simplex bases, more calculations in each simplex iteration, and thus a higher computational effort. However, we can overcome this issue by substituting $p_i = p'_i + (\hat{c}_i - \overline{z})^+ x_i$ and $z = z' + \underline{z}$. We then obtain the equivalent *substituted robust subproblem*

$$(\text{ROB}^{\text{S}}(Z)) \qquad \begin{aligned} &\min \Gamma \underline{z} + \Gamma z' + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - \overline{z})^+ \right) x_i + p'_i \\ &\text{s.t. } (x, p', z') \in \mathcal{F}^{\text{S}}(Z), x \in \{0,1\}^n \end{aligned}$$

over the *substituted formulation*

$$\mathcal{F}^{\text{S}}(Z) = \left\{ (x, p', z') \middle| \begin{array}{ll} Ax \leq b & \\ p'_i + z' \geq (\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+ x_i & \forall i \in [n] \\ x \in [0,1]^n, p' \in \mathbb{R}^n_{\geq 0}, z' \in [0, \overline{z} - \underline{z}] & \end{array} \right\}.$$

Our computational experiments confirm that the substituted robust subproblem $\text{ROB}^{\text{S}}(Z)$ is indeed computationally preferable to the non-substituted robust subproblem $\text{ROB}(Z)$. Furthermore, $\text{ROB}^{\text{S}}(Z)$ is also interesting from a theoretical point of view: Since $z' \leq \overline{z} - \underline{z}$ holds for all optimal solutions, $\text{ROB}^{\text{S}}(Z)$ is equivalent to ROB for an instance with objective coefficients $c_i + (\hat{c}_i - \overline{z})^+$, deviations $(\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+$, and an added constant $\Gamma \underline{z}$. This will be useful in subsequent sections, as properties that we prove for ROB carry over directly to

$\text{ROB}^{\text{S}}(Z)$ and $\text{ROB}(Z)$. We will often consider $\text{ROB}(Z)$ and $\text{ROB}^{\text{S}}(Z)$ as interchangeable, depending on the use case. In the next section, we show how to use the robust subproblems in a branch and bound algorithm for solving ROB.

5.2 The Basic Branch and Bound Framework

The general idea of our branch and bound approach, sketched in Algorithm 2, is to solve ROB by branching the set of possible values $\{\hat{c}_0, \dots, \hat{c}_n\}$ for z into subsets $Z \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$, for which we then consider the robust subproblems $\text{ROB}(Z)$. For each considered subset Z, we store a dual bound $\underline{v}(Z)$ on the optimal solution value of $\text{ROB}(Z)$. This dual bound will primarily be based on a dual bound on the optimal objective value $v(\text{ROB}(Z'))$ for the parent superset $Z' \supseteq Z$. This is obtained, e.g., by computing the optimal objective value of the continuous relaxation $v^{\text{R}}(\text{ROB}(Z'))$ using the strong formulation from the previous section (lines 9 and 10). If the dual bound $\underline{v}(Z)$ is greater than or equal to the current primal bound $\overline{v}$, then we can prune Z (line 5). If Z cannot be pruned, we first asses the strength of formulation $\mathcal{F}(Z)$ (line 6), which converges towards the strength of $\mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$ and achieves equality at latest for $|Z| = 1$ according to Proposition 24. If $\mathcal{F}(Z)$ is almost as strong as $\mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times [\underline{z}, \overline{z}])$, then we directly solve the robust subproblem $\text{ROB}(Z)$ (line 7). This has the advantage that we solve multiple problems that are much easier than ROB but do not have to consider the nominal subproblems $\text{NOS}(z)$ for all $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$. If $\mathcal{F}(Z)$ is too weak for solving $\text{ROB}(Z)$, we continue computing dual bounds and branching into subsets $Z = Z_1 \cup Z_2$ (lines 9 to 11).

Algorithmus 2 : The Basic Branch and Bound Framework

Input : An instance of ROB
Output : An optimal solution (x^*, p^*, z^*) of value $\overline{v}$

```
1  Initialize 𝒩 = {{ĉ_0, ..., ĉ_n}}
2  Set dual bound v̲({ĉ_0, ..., ĉ_n}) = −∞ and primal bound v̄ = ∞
3  while 𝒩 ≠ ∅ do
4      Choose Z ∈ 𝒩 and remove 𝒩 ← 𝒩 \ {Z}
5      if v̲(Z) < v̄ then
6          if ℱ(Z) is "strong enough" then
7              Solve ROB(Z) and update (x*, p*, z*) and v̄ if a new best solution is found
8          else
9              Compute new dual bound v̲(Z) on the optimal solution value v(ROB(Z))
10             Divide Z = Z_1 ∪ Z_2 and set v̲(Z_i) ← v̲(ROB(Z)) for i = 1, 2
11             Insert 𝒩 ← 𝒩 ∪ {Z_1, Z_2}
12 return (x*, p*, z*)
```

Note that the framework given in Algorithm 2 only serves for getting a basic intuition, as many components are described vaguely. For example, we leave open for now how to evaluate whether we can stop branching Z due to $\mathcal{F}(Z)$ being "strong enough". We

will describe all components of our algorithm in detail in Section 5.6. There, we will not only discuss whether and how we should branch Z (cf. Section 5.6.5) and how to choose the next $Z \in \mathcal{N}$ (cf. Section 5.6.4) but also improve on the computation of dual bounds (cf. Section 5.6.1) and primal bounds (cf. Section 5.6.2) as well as discuss an efficient pruning strategy (cf. Section 5.6.3). Before doing so, we first establish some theoretical background in the following sections that will be crucial for the design of our algorithm.

5.3 A Reformulation using Cliques in Conflict Graphs

In this section, we propose a reformulation in a different variable space (cf. Section 2.4.3) of ROB, which is not only stronger than $\mathcal{F}^{\text{ROB}}$ but also contains fewer variables and constraints. The reformulation is indifferent of bounds $\underline{z}, \overline{z}$ on z and can be easily combined with the formulations $\mathcal{F}(Z)$ and $\mathcal{F}^{\text{S}}(Z)$ from Section 5.1. We start with the theoretical introduction of the reformulation and then give technical details on how we construct it in practice.

5.3.1 The Clique Reformulation

Assume that $x_i + x_j \leq 1$ is a valid inequality for $\mathcal{C}^{\text{NOM}}$, that is, x_i and x_j cannot both be equal to one and are therefore said to be in *conflict*. Such conflicts can be modeled within a so-called *conflict graph*, consisting of a node for every binary variable x_i and edges $\{x_i, x_j\}$ between two nodes if there exists no solution with $x_i = x_j = 1$ [10]. Since every solution to the original problem corresponds to an independent set within the conflict graph, all valid inequalities for the independent set problem on the conflict graph are also valid for the original problem. We have already seen in Section 4.3 that clique inequalities $\sum_{i \in Q} x_i \leq 1$, implied by a clique $\{x_i | i \in Q\}$ in the conflict graph, are especially interesting. This is because their recycled inequalities $\sum_{i \in Q} p_i + z \geq \sum_{i \in Q} \hat{c}_i x_i$ are always facet defining, provided that $x_i = 0$ is not valid for $\mathcal{C}^{\text{NOM}}$ for some $i \in Q$. The idea of our reformulation is to replace the robustness constraints $p_i + z \geq \hat{c}_i x_i$ with recycled clique inequalities and afterwards aggregate all variables p_i within one recycled clique inequality. Note that in general, we cannot omit the original robustness constraints when adding recycled clique inequalities, as $\{x_i\}$ is a clique itself, and thus $p_i + z \geq \hat{c}_i x_i$ is facet-defining as long as $x_i = 0$ is not valid for $\mathcal{C}^{\text{NOM}}$. However, we will see that we obtain a valid reformulation for ROB if the underlying cliques of the recycled inequalities form a partition of the variables. To ease notation, we call a subset $Q \subseteq [n]$ a clique if the variables $\{x_i | i \in Q\}$ form a clique in the conflict graph.

Proposition 25. *Let $\mathcal{Q}$ be a partition of $[n]$ into cliques. Then the problem*

$$\text{(ROB}(\mathcal{Q})\text{)} \qquad \begin{aligned} \min\ & \Gamma z + \sum_{i \in [n]} c_i x_i + \sum_{Q \in \mathcal{Q}} p'_Q \\ \text{s.t.}\ & (x, p', z) \in \mathcal{F}^{\text{ROB}}(\mathcal{Q}), x \in \{0,1\}^n \end{aligned}$$

over the clique formulation

$$\mathcal{F}^{ROB}(\mathcal{Q}) = \left\{ (x, p', z) \middle| \begin{array}{ll} Ax \leq b & \\ p'_Q + z \geq \sum_{i \in Q} \hat{c}_i x_i & \forall Q \in \mathcal{Q} \\ x \in [0,1]^n, p' \in \mathbb{R}^{\mathcal{Q}}_{\geq 0}, z \in \mathbb{R}_{\geq 0} & \end{array} \right\}.$$

is a reformulation in a different variable space that is at least as strong as ROB.

Proof. First, note that for any solution (x, p, z) to ROB, the corresponding solution (x, p', z) with $p'_Q = \sum_{i \in Q} p_i$ for all $Q \in \mathcal{Q}$ is a solution to ROB $(\mathcal{Q})$, since

$$p'_Q + z = \sum_{i \in Q} p_i + z \geq \sum_{i \in Q} \hat{c}_i x_i$$

is the recycled inequality of $\sum_{i \in Q} x_i \leq 1$. Moreover, both solutions (x, p, z) and (x, p', z) have the same objective value.

Hence, it only remains to show that every solution $(x, p', z) \in \mathcal{F}^{\text{ROB}}(\mathcal{Q}) \cap \left(\mathbb{Z}^n \times \mathbb{R}^{|\mathcal{Q}|+1}\right)$ has a corresponding solution $\phi(x, p', z) \in \mathcal{F}^{\text{ROB}} \cap (\mathbb{Z}^n \times \mathbb{R}^{n+1})$ of the same objective value. We define the image of $(x, p', z) \in \mathcal{F}^{\text{ROB}}(\mathcal{Q})$ as $\phi(x, p', z) = (x, p, z)$ and consider two different cases for the definition of $p \in \mathbb{R}^n$. For cliques $Q \in \mathcal{Q}$ with $\sum_{j \in Q} \hat{c}_j x_j > 0$, we define $p_i = \frac{\hat{c}_i x_i p'_Q}{\sum_{j \in Q} \hat{c}_j x_j}$ for all $i \in Q$. Then $p_i + z \geq \hat{c}_i x_i$ holds, since we have

$$p_i + z = \frac{\hat{c}_i x_i p'_Q}{\sum_{j \in Q} \hat{c}_j x_j} + z \geq \frac{\hat{c}_i x_i \left(p'_Q + z\right)}{\sum_{j \in Q} \hat{c}_j x_j} \geq \frac{\hat{c}_i x_i \sum_{j \in Q} \hat{c}_j x_j}{\sum_{j \in Q} \hat{c}_j x_j} = \hat{c}_i x_i.$$

For cliques $Q \in \mathcal{Q}$ with $\sum_{j \in Q} \hat{c}_j x_j = 0$, we choose p_i arbitrarily such that $p'_Q = \sum_{j \in Q} p_j$, as $p_i + z \geq 0 = \hat{c}_i x_i$ holds for all $p_i \geq 0$. This shows not only $\phi\left(\mathcal{F}^{\text{ROB}}(\mathcal{Q}) \cap \left(\mathbb{Z}^n \times \mathbb{R}^{|\mathcal{Q}|+1}\right)\right) \subseteq \mathcal{F}^{\text{ROB}} \cap (\mathbb{Z}^n \times \mathbb{R}^{n+1})$, but also proves the strength of ROB $(\mathcal{Q})$, as we did not use the integrality of x, and thus have $\phi\left(\mathcal{F}^{\text{ROB}}(\mathcal{Q})\right) \subseteq \mathcal{F}^{\text{ROB}}$. □

We consider again Example 6 from Section 3.1 to see that reformulation ROB $(\mathcal{Q})$ is not only equal but actually stronger. In the example, $[n]$ is a clique, and we thus have

$$v^{\text{R}}(\text{ROB}(\mathcal{Q})) = z + p_{[n]} - \sum_{i \in [n]} x_i \geq \sum_{i \in [n]} \frac{n-1}{n} x_i - \sum_{i \in [n]} x_i = -\sum_{i \in [n]} \frac{1}{n} x_i = -\frac{1}{n},$$

compared to $v^{\text{R}}(\text{ROB}) = -1 + \frac{n-1}{n^2}$. In fact, $v^{\text{R}}(\text{ROB}(\mathcal{Q}))$ equals the optimal integer solution value.

As mentioned in Section 5.1, the improvement of ROB can directly be applied to ROB$^{\text{S}}(Z)$ due to the equivalency of both problems. Given a clique partition $\mathcal{Q}$ of $[n]$, we obtain the restricted reformulation

$$\text{(ROB}^{\text{S}}\,(Z,\mathcal{Q}))\qquad \begin{aligned} \min\ & \Gamma\underline{z}+\Gamma z'+\sum_{i\in[n]}\left(c_i+(\hat{c}_i-\overline{z})^+\right)x_i+\sum_{Q\in\mathcal{Q}}p'_Q \\ \text{s.t. }\ & (x,p',z')\in\mathcal{F}^{\text{S}}\,(Z,\mathcal{Q})\,,x\in\{0,1\}^n \end{aligned}$$

with

$$\mathcal{F}^{\text{S}}\,(Z,\mathcal{Q})=\left\{(x,p',z')\middle|\begin{array}{ll} Ax\leq b & \\ p'_Q+z'\geq\sum_{i\in Q}\left(\min\left\{\hat{c}_i,\overline{z}\right\}-\underline{z}\right)^+x_i & \forall Q\in\mathcal{Q} \\ x\in[0,1]^n\,,p'\in\mathbb{R}^{\mathcal{Q}}_{\geq 0},z'\in[0,\overline{z}-\underline{z}] & \end{array}\right\}.$$

Note that we have to choose between using formulation $\mathcal{F}^{\text{S}}\,(Z,\mathcal{Q})$ and the recycling of general inequalities, as the aggregation of variables p_i is not compatible with the recycling approach from Chapter 4. For our branch and bound algorithm, we choose to use the formulation $\mathcal{F}^{\text{S}}\,(Z,\mathcal{Q})$ over the recycling, as this reduces the number of variables and constraints instead of adding additional inequalities. In our computational study in Section 5.7, we observe that the reduction of the formulation's size is quite important, since solving the continuous relaxations is often surprisingly time consuming.

Although we use substitution in practice, we will often refer to the non-substituted version

$$\text{(ROB}\,(Z,\mathcal{Q}))\qquad \begin{aligned} \min\ & \Gamma z+\sum_{i\in[n]}c_ix_i+\sum_{Q\in\mathcal{Q}}p'_Q \\ \text{s.t. }\ & (x,p',z)\in\mathcal{F}\,(Z,\mathcal{Q})\,,x\in\{0,1\}^n \end{aligned}$$

with

$$\mathcal{F}\,(Z,\mathcal{Q})=\left\{(x,p',z)\middle|\begin{array}{ll} Ax\leq b & \\ p'_Q+z\geq\sum_{i\in Q}\left(\hat{c}_i-\underline{z}\right)^+x_i+\underline{z} & \forall Q\in\mathcal{Q} \\ p'_Q\geq\sum_{i\in Q}\left(\hat{c}_i-\overline{z}\right)^+x_i & \forall Q\in\mathcal{Q} \\ x\in[0,1]^n\,,p'\in\mathbb{R}^{\mathcal{Q}}_{\geq 0},z\in[\underline{z},\overline{z}] & \end{array}\right\}$$

or even ROB (Z) in the remainder of this chapter. This greatly simplifies the notation in the following sections. However, before proceeding to the next theoretical results, we first describe how we compute $\mathcal{F}\,(Z,\mathcal{Q})$ in practice.

5.3.2 Implementation of Conflict Graphs and Clique Partitions

In order to obtain formulation $\mathcal{F}\,(Z,\mathcal{Q})$, we first have to compute a conflict graph and a clique partition $\mathcal{Q}$ of $[n]$. Since finding conflicts and cliques is an integral part of preprocessing routines in modern MILP solvers [3, 5, 24], we could directly use the solver's conflict graph and clique table if our algorithm was natively implemented. Unfortunately, we cannot access these in Gurobi [52], the solver of our choice, and therefore compute them ourselves.

Ideally, the partition $\mathcal{Q}$ contains few cliques that are as large as possible. However, finding a partition of minimum cardinality is equivalent to computing a minimum clique cover, which is $\mathcal{NP}$-hard in general [59]. Moreover, building the whole conflict graph itself is also $\mathcal{NP}$-hard [27]. Consequently, we have to restrict ourselves to a subgraph of the whole conflict graph. Instead of answering the difficult question whether there exists a solution $x \in \mathcal{C}^{\text{NOM}}$ with $x_{i_1} = x_{i_2} = 1$, we resort to a simpler problem, for which we only consider single constraints of the nominal problem.

Let $\sum_{i\in[n]} a_{ji}x_i \leq b_j$ be a row of the constraint matrix $Ax \leq b$. If

$$\min\left\{\sum_{i\in[n]} a_{ji}x_i \middle| x \in \{0,1\}^n, x_{i_1} = x_{i_2} = 1\right\} > b_j \tag{5.3}$$

holds, then x_{i_1} and x_{i_2} cannot both be equal to one and we can add an edge $\{x_{i_1}, x_{i_2}\}$ to the conflict graph. In order to evaluate inequality (5.3) efficiently for all pairs $i_1, i_2 \in [n]$, we first compute

$$b'_j = b_j - \min\left\{\sum_{i\in[n]} a_{ji}x_i \middle| x \in \{0,1\}^n\right\},$$

that is the maximum possible slack of the constraint, by setting $x_i = 0$ if $a_{ji} \geq 0$ and $x_i = 1$ if $a_{ji} < 0$. We have

$$\begin{aligned}
&\min\left\{\sum_{i\in[n]} a_{ji}x_i \middle| x \in \{0,1\}^n, x_{i_1} = x_{i_2} = 1\right\} \\
=&\min\left\{\sum_{i\in[n]} a_{ji}x_i \middle| x \in \{0,1\}^n\right\} + \max\{a_{ji_1}, 0\} + \max\{a_{ji_2}, 0\} \\
=&b_j - b'_j + \max\{a_{ji_1}, 0\} + \max\{a_{ji_2}, 0\},
\end{aligned}$$

and thus it is sufficient to evaluate whether

$$b'_j < \max\{a_{ji_1}, 0\} + \max\{a_{ji_2}, 0\}$$

holds. In order to find conflicting variables x_{i_1}, x_{i_2} having this property, we use Algorithm 3, which is similar to one presented by Brito and Santos [27]. Instead of performing a pairwise evaluation, Algorithm 3 directly searches for subsets $Q \subseteq [n]$ corresponding to cliques in the conflict graph. This is not only faster than a pairwise evaluation but also beneficial for storing the conflict graph. Assume that a row of the constraint matrix implies conflicts of a large clique Q. Atamtürk et al. [10] mention that storing the $\binom{|Q|}{2}$ implied conflicts as a set of edges or using adjacency lists consumes too much memory. Instead, one should use a separate structure to store these conflicts. Here, we store conflicts implied by cliques Q with $|Q| = 2$ in adjacency lists, while cliques with $|Q| > 2$ are stored directly as a list of variables. To guarantee fast access to the cliques, we maintain for each variable a list of references to cliques in which it is contained. Thus, the memory requirement of adding a clique Q is only

$\mathcal{O}(|Q|)$ instead of $\mathcal{O}\left(|Q|^2\right)$. In the following, we think of the conflict graph as a hypergraph $G = (V, H)$, with $V = \{x_1, \ldots, x_n\}$ being the nodes and $H \subseteq 2^V$ being a set of hyperedges representing cliques in the conflict graph.

Algorithm 3 constructs this hypergraph by iterating over all constraints $\sum_{i\in[n]} a_{ji}x_i \leq b_j$. For each constraint, we first compute b'_j (line 3) and the highest coefficient a^* occurring in the constraint (line 4). If we have $\max\{a^*, 0\} \leq \frac{b'_j}{2}$, then the constraint will imply no conflicts, which allows for a fast skipping of uninteresting constraints (line 5). Otherwise, we construct a list of candidates x_{i_l} for which we have $\max\{a^*, 0\} + \max\{a_{ji_l}, 0\} > b'_j$, that is, the variable x_{i_l} either defines a^* or is in conflict with the variable defining a^* (line 6). If the list $(x_{i_1}, \ldots, x_{i_k})$ consists of more than one variable, then we have found a conflict (line 7). We sort the list of candidates (line 8) and find the lowest index l^* such that $x_{i_{l^*}}$ and $x_{i_{l^*+1}}$ are in conflict (line 9). Due to the ordering, all variables $\{x_{i_{l^*}}, \ldots, x_{i_k}\}$ are in conflict with each other and can thus be added as a hyperedge to the conflict graph (line 10). For all variables x_{i_p} that do not belong to the hyperedge, we search for the lowest index l' such that x_{i_p} and $x_{i_{l'}}$ are in conflict (line 12). Such an index exists, since x_{i_p} is in conflict with x_{i_k}. Again, due to the ordering of the list, the variables $\left\{x_{i_p}, x_{i_{l'}}, \ldots, x_{i_k}\right\}$ are in conflict and can be added to our graph (line 13).

Algorithmus 3 : Algorithm for computing hyperedges for the conflict graph.

Input : Constraints $Ax \leq b$ with $A \in \mathbb{R}^{m\times n}$ and uncertain variables V
Output : A hypergraph $G = (V, H)$ with hyperedges $H \subseteq 2^V$

1 Initialize hyperedges $H = \emptyset$
2 **for** $j \in [m]$ **do**
3 Compute $b'_j = b_j - \min\left\{\sum_{i\in[n]} a_{ji}x_i \middle| x \in \{0,1\}^n\right\}$
4 Let $a^* = \max\{a_{ji} | i \in [n]\}$
5 **if** $\max\{a^*, 0\} > \frac{b'_j}{2}$ **then**
6 Let $L = (x_{i_1}, \ldots, x_{i_k})$ be a list of uncertain variables with $\max\{a^*, 0\} + \max\{a_{ji_l}, 0\} > b'_j$ for $l \in [k]$
7 **if** $k > 1$ **then**
8 Sort L non-decreasingly w.r.t. $\max\{a_{ji}, 0\}$
9 Let $l^* = \text{argmin}\left\{l \in [k-1] \middle| \max\{a_{ji_l}, 0\} + \max\{a_{ji_{l+1}}, 0\} > b'_j\right\}$
10 Add $H \leftarrow H \cup \{\{x_{i_{l^*}}, \ldots, x_{i_k}\}\}$
11 **for** $p \in [l^* - 1]$ **do**
12 Let $l' = \text{argmin}\left\{l \in [k] \middle| \max\{a_{ji_p}, 0\} + \max\{a_{ji_l}, 0\} > b'_j\right\}$
13 Add $H \leftarrow H \cup \left\{\left\{x_{i_p}, x_{i_{l'}}, \ldots, x_{i_k}\right\}\right\}$
14 **return** $G = (V, H)$

The hyperedge added in line 13 consists of a subset of the first hyperedge added to the constraint, starting at index l', and an additional variable x_{i_p}. Brito and Santos [27] point out that this can be used to store the graph more efficiently. Instead of storing the whole hyperedge

$\left\{x_{i_p}, x_{i_{l'}}, \ldots, x_{i_k}\right\}$, we only store the variable x_{i_p}, a reference to the hyperedge $\{x_{i_{l^*}}, \ldots, x_{i_k}\}$, and the index l', from which we can construct the hyperedge $\left\{x_{i_p}, x_{i_{l'}}, \ldots, x_{i_k}\right\}$.

After constructing the conflict graph, we use Algorithm 4 to compute a partition of V into cliques. Algorithm 4 is a greedy heuristic building on the idea that a minimum clique cover consists without loss of generality only of cliques that are maximal with respect to inclusion. We first initialize a set of cliques $\mathcal{Q} = \emptyset$ and remaining nodes $V' = V$ (line 1). While there are nodes left for partition (line 2), we iteratively select an arbitrary remaining node $v' \in V'$ (line 3) and construct a maximal clique $Q \subseteq V'$ containing v'. We initialize Q as the largest hyperedge on the remaining nodes $h \cap V'$ containing v' (line 4 and 5). This speeds up the construction of Q, since we do not have to check whether the variables in h are in conflict. We then expand Q by iteratively adding remaining nodes that are contained in the neighborhood of all current clique members (lines 6 to 10). Afterwards, we add this clique to our partition (line 11), remove the contained nodes from the remaining nodes (line 12), and proceed until no nodes are left.

Algorithmus 4 : Greedy heuristic for clique partitioning in a conflict graph.

Input : A conflict graph $G = (V, H)$ consisting of nodes and hyperedges
Output : A partition $\mathcal{Q}$ of V into cliques

1 Initialize the partition into cliques $\mathcal{Q} = \emptyset$ and set of remaining nodes $V' = V$.
2 **while** $V' \neq \emptyset$ **do**
3 Choose $v' \in V'$
4 Choose $h \in \text{argmax}\left\{|h \cap V'||h \in \left(H \cup \{v'\}\right), v' \in h\right\}$
5 Initialize new clique $Q = h \cap V'$
6 Compute candidates $N = V' \bigcap_{v \in Q} N(v)$
7 **while** $N \neq \emptyset$ **do**
8 Choose any $v \in N$
9 Add candidate $Q \leftarrow Q \cup \{v\}$
10 Update candidates $N \leftarrow N \cap N(v)$
11 Add clique $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{Q\}$
12 Remove from remaining nodes $V' \leftarrow V' \backslash Q$
13 **return** $\mathcal{Q}$

Remember from Theorem 7 that the bilinear formulation $\mathcal{F}^{\text{BIL}}$ is stronger than any polyhedral formulation for a given nominal formulation $\mathcal{F}^{\text{NOM}}$. However, our computed clique partition $\mathcal{Q}$ may contain cliques Q for which the corresponding clique inequality $\sum_{i \in Q} x_i \leq 1$ is not directly contained in the constraint matrix $Ax \leq b$ describing $\mathcal{F}^{\text{NOM}}$. Thus, when using such cliques, we incorporate information into the clique reformulation $\text{ROB}(Z, \mathcal{Q})$ that is not available for the bilinear formulation $\mathcal{F}^{\text{BIL}}$. This can result in $\mathcal{F}(Z, \mathcal{Q})$ being stronger than $\mathcal{F}^{\text{BIL}}$, which is undesirable for our branch and bound algorithm, since we approximate $\mathcal{F}^{\text{BIL}}$ and can therefore end up with a weaker formulation after branching. Therefore, we always add the corresponding clique inequalities of our clique partition to the constraint matrix $Ax \leq b$ if they are not already contained.

5.4 Lagrangean Relaxations

We already noted in the previous section that solving the continuous relaxation of ROB (Z) can be relatively time consuming due to the additional robustness constraints and variables. Although the substituted clique reformulation ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q})$ already contains fewer variables and constraints, we still observe for some instances that solving the continuous relaxation can require much more time than solving an integer nominal subproblem NOS (z). This counteracts the idea of the branch and bound algorithm, which is to solve the theoretically easy relaxations in order to discard non-optimal values for z and focus on interesting subproblems ROB (Z). The computational effort needed to solve the continuous relaxations is especially high for the first nodes of our branch and bound tree, when Z still contains a wide range of possible values for z. After some branching steps, yielding tighter bounds $\underline{z}, \overline{z}$, the relaxations become easier to solve. This effect is not surprising for increasing lower bounds $\underline{z}$, since many variables p'_Q and constraints $p'_Q + z' \geq \sum_{i \in Q} \left(\min\left\{\hat{c}_i, \overline{z}\right\} - \underline{z}\right)^+ x_i$ become redundant for $\underline{z} \geq \max\left\{\hat{c}_i | i \in Q\right\}$. Even for decreasing upper bounds $\overline{z}$, we observe a reduction of the computation time, which might be due to the diminishing impact of the robustness constraints on the problem structure.

On the basis of these observations, it appears reasonable to consider a relaxed problem of ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q})$ that omits the robustness constraints as long as the bounds $\underline{z}, \overline{z}$ are wide. After some branching steps are performed, we can then revert to solving the continuous relaxation of ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q})$. To this end, we consider a *Lagrangean relaxation* of ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q})$. The idea of Lagrangean relaxations is to remove complicating constraints from the constraint matrix so that they do not have to be fulfilled, but their violation is penalized in the objective function. For an arbitrary problem $\min\left\{v(x) | A'x \leq b', x \in \mathcal{X}\right\}$ with $\mathcal{X} \subseteq \mathbb{R}^n$ and a complicating constraint matrix $A' \in \mathbb{R}^{m \times n}$ with right-hand side $b' \in \mathbb{R}^m$, the Lagrangean relaxation with *Lagrange multipliers* $\lambda \in \mathbb{R}^m_{\geq 0}$ is defined as $\min\left\{v(x) + \lambda^\top (A'x - b') \middle| x \in \mathcal{X}\right\}$. Note that this is indeed a relaxation, since $\lambda^\top (A'x - b') \leq 0$ holds for all feasible solutions of the original problem.

The value of the Lagrangean relaxation depends on the multipliers λ. Choosing a large value λ_j rewards solutions with a positive slack in the j-th constraint, while $\lambda_j = 0$ allows for an unpenalized violation. The problem of finding optimal multipliers, yielding a Lagrangean relaxation with the highest objective value, is known as the *Lagrangean dual problem*. Optimal multipliers are in general difficult to obtain, but if they are achieved and the original problem is a linear program, then the value of the optimal Lagrangean relaxation is exactly the value of the original problem [64, Section 5.6]. Hence, if we were able to compute optimal Lagrange multipliers, then we could compute the value of the continuous relaxation $v^{\mathrm{R}}\left(\text{ROB}^{\mathrm{S}}(Z, \mathcal{Q})\right)$ by solving the Lagrangean relaxation of the continuous relaxation of ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q})$.

Since we want to relax the robustness constraints $p'_Q + z' \geq \sum_{i \in Q} \left(\min\left\{\hat{c}_i, \overline{z}\right\} - \underline{z}\right)^+ x_i$ of the continuous relaxation of ROB$^{\text{S}}(Z, \mathcal{Q})$, we need to choose Lagrange multipliers $\lambda_Q \geq 0$ for all cliques $Q \in \mathcal{Q}$. Given such multipliers, we obtain

$$\begin{aligned} \min\ & \Gamma \underline{z} + \Gamma z' + \sum_{i \in [n]} \left(c_i + \left(\hat{c}_i - \overline{z}\right)^+\right) x_i + \sum_{Q \in \mathcal{Q}} p'_Q \\ & + \sum_{Q \in \mathcal{Q}} \lambda_Q \left(\sum_{i \in Q} \left(\min\left\{\hat{c}_i, \overline{z}\right\} - \underline{z}\right)^+ x_i - p'_Q - z' \right) \\ \text{s.t. } & x \in \mathcal{F}^{\text{NOM}}. \end{aligned}$$

Since p' and z' are unbounded, their objective coefficients need to be non-negative, which is equivalent to $\lambda \in [0,1]^{\mathcal{Q}}$ and $\sum_{Q \in \mathcal{Q}} \lambda_Q \leq \Gamma$. Moreover, as p' and z' are no longer contained in the constraint matrix, they will always be zero in an optimal solution. Therefore, we have the following *Lagrangean relaxed continuous robust subproblem*

$$\text{(LRR}(Z, \mathcal{Q}, \lambda)\text{)} \qquad \begin{aligned} \min\ & \Gamma \underline{z} + \sum_{i \in [n]} \left(c_i + \left(\hat{c}_i - \overline{z}\right)^+\right) x_i + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q} \left(\min\left\{\hat{c}_i, \overline{z}\right\} - \underline{z}\right)^+ x_i \\ \text{s.t. } & x \in \mathcal{F}^{\text{NOM}}. \end{aligned}$$

Note that if each index $i \in [n]$ is its own clique, i.e., $\mathcal{Q} = \{\{1\}, \ldots, \{n\}\}$, then the objective coefficients obtained by choosing $\lambda \in [0,1]^{\mathcal{Q}}$ with $\sum_{Q \in \mathcal{Q}} \lambda_Q \leq \Gamma$ correspond to a convex combination of the possible objective coefficients c' in our uncertainty set (cf. Section 3.1). Consequently, the Lagrangean relaxation is stronger than solely using some scenario if we are able to merge indices into actual cliques Q.

In our branch and bound algorithm, we solve LRR$(Z, \mathcal{Q}, \lambda)$ instead of the continuous relaxation of ROB$^{\text{S}}(Z, \mathcal{Q})$ for large sets Z. As we want to reduce the computational time spent for solving the relaxations, we do not compute optimal Lagrange multipliers λ but choose some heuristically. In general, one needs to be careful when doing so, as high values of λ may overcompensate a positive slack of the relaxed constraints, resulting in a weak dual bound. Fortunately, we do not face this problem here, as choosing $\lambda \in [0,1]^{\mathcal{Q}}$ with $\sum_{Q \in \mathcal{Q}} \lambda_Q \leq \Gamma$ is always at least as good as simply removing all robustness constraints. Let $\mathcal{Q}' = \{Q \in \mathcal{Q} | \max\{\hat{c}_i | i \in Q\} > \underline{z}\}$ be the subset of cliques for which the corresponding robustness constraints are not redundant. In our implementation, we choose $\lambda_Q = \min\left\{1, \frac{\Gamma}{|\mathcal{Q}'|}\right\}$ for $Q \in \mathcal{Q}'$ and $\lambda_Q = 0$ otherwise. This yields a balanced objective function, which considers every Lagrangean relaxed robustness constraint equally, and thus yields no obvious loophole that may be exploited by an optimal solution.

The approach leaves room for further engineering, such as choosing other Lagrangean multipliers or additionally considering recycled inequalities for relaxation. However, our simple choice of λ already proves the concept of using Lagrangean relaxations, as they yield

a speed-up of our branch and bound algorithm. Therefore, we leave more sophisticated approaches for future research.

5.5 Characterization of Optimal Values for p and z

The central idea of our branch and bound algorithm for solving ROB is to restrict the value of z to strengthen our formulation and then trying to find an optimal corresponding nominal solution $x \in \mathcal{F}^{\text{NOM}}$. In this section, however, we want to consider the opposite direction. Given a nominal solution $x \in \mathcal{F}^{\text{NOM}}$, what are the optimal values for p and z? We already took a glimpse at this question in the discussion of Figure 3.1 from Chapter 3. We will now have a closer look, as the answer to it will deepen our understanding of the structural properties of ROB and is of practical use in many ways. First, we will generalize the result of Lee and Kwon [70], who showed for $\Gamma \in \mathbb{Z}$ that there exists a subset $\mathcal{Z} \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$, with $|\mathcal{Z}| \leq \left\lceil \frac{n-\Gamma}{2} \right\rceil + 1$, containing an optimal choice for z. This reduction is relevant for our branch and bound algorithm, as we only have to consider subsets $Z \subseteq \mathcal{Z}$. Second, given a choice of z, we will be able to restrict our search for a corresponding nominal solution $x \in \mathcal{F}^{\text{NOM}}$ to those for which the chosen z is optimal. We will extensively use this idea within our branch and bound algorithm, especially in Section 5.6.1, where we describe further dual bounding strategies. Third, as we prove the characterization of optimal z for (potentially fractional) solutions within $\mathcal{F}^{\text{BIL}}$, we can compute for any $x \in \mathcal{F}^{\text{NOM}}$ the corresponding objective value for the optimization problem over $\mathcal{F}^{\text{BIL}}$. This provides an upper bound on the optimal objective value over $\mathcal{F}^{\text{BIL}}$, which we compare to the optimal objective value $v^{\text{R}}\left(\text{ROB}\left(Z, \mathcal{Q}\right)\right)$ of the continuous relaxation of $\text{ROB}\left(Z, \mathcal{Q}\right)$ in order to obtain an indicator of the strength of $\mathcal{F}\left(Z, \mathcal{Q}\right)$. We use this indicator in our branch and bound algorithm to decide whether $\text{ROB}\left(Z, \mathcal{Q}\right)$ should be solved directly as an MILP or whether Z needs to be shrunk further, as explained in Section 5.6.5. The following theorem states the characterization of optimal values for p and z.

Theorem 26. *Let $x \in \mathcal{F}^{NOM}$ be a (fractional) solution to NOM. We define*

$$\underline{z}\left(x\right) = \min\left\{ z \in \{\hat{c}_0, \dots, \hat{c}_n\} \middle| \sum_{i \in [n]: \hat{c}_i > z} x_i \leq \Gamma \right\}$$

and

$$\overline{z}\left(x\right) = \max\left(\{0\} \cup \left\{ z \in \{\hat{c}_0, \dots, \hat{c}_n, \infty\} \middle| \sum_{i \in [n]: \hat{c}_i \geq z} x_i \geq \Gamma \right\} \right).$$

The values $z \in [\underline{z}\left(x\right), \overline{z}\left(x\right)]$ are together with $p_i = \left(\hat{c}_i - z\right)^+ x_i$ for $i \in [n]$ exactly the optimal values satisfying $(x, p, z) \in \mathcal{F}^{BIL}$ and minimizing $\Gamma z + \sum_{i \in [n]} p_i$.

For integer solutions $x \in \mathcal{F}^{\text{NOM}}$, the lower bound $\underline{z}\left(x\right)$ indicates that z should be large enough such that there are at most Γ indices $i \in [n]$ with $x_i = 1$ and $\hat{c}_i > z$. Otherwise,

we may increase z while simultaneously decreasing p_i for more than Γ indices, leading to an improvement of the objective value. Conversely, the upper bound $\overline{z}(x)$ implies that z should be small enough such that there exist at least Γ indices $i \in [n]$ with $x_i = 1$ and $\hat{c}_i \geq z$. Otherwise, we may decrease z and would have to increase p_i for fewer than Γ indices, also yielding an improvement of the objective value. Obviously, if Γ is so large that $\sum_{i\in[n]} x_i < \Gamma$ holds, then we should choose z as small as possible, i.e., $z = 0$. To prove that the bounds $\underline{z}(x), \overline{z}(x)$ from Theorem 26 are exact, we first characterize them in an additional way.

Lemma 27. *For $x \in \mathbb{R}^n$, we have*

$$\underline{z}(x) = \max\left(\{0\} \cup \left\{ z \in \{\hat{c}_0, \ldots, \hat{c}_n\} \middle| \sum_{i\in[n]:\hat{c}_i \geq z} x_i > \Gamma \right\}\right) \tag{5.4}$$

and

$$\overline{z}(x) = \min\left(\{\infty\} \cup \left\{ z \in \{\hat{c}_0, \ldots, \hat{c}_n\} \middle| \sum_{i\in[n]:\hat{c}_i > z} x_i < \Gamma \right\}\right). \tag{5.5}$$

Proof. We first prove equation (5.4). If we have $\underline{z}(x) = 0$, then the definition of $\underline{z}(x)$ in Theorem 26 implies for all $z > 0$ that

$$\sum_{i\in[n]:\hat{c}_i \geq z} x_i \leq \sum_{i\in[n]:\hat{c}_i > 0} x_i \leq \Gamma$$

holds, and thus we have

$$\max\left(\{0\} \cup \left\{ z \in \{\hat{c}_0, \ldots, \hat{c}_n\} \middle| \sum_{i\in[n]:\hat{c}_i \geq z} x_i > \Gamma \right\}\right) = 0.$$

In the case of $\underline{z}(x) > 0$, there exists an index $j \in [n]$ with $\hat{c}_j = \underline{z}(x)$ and $\hat{c}_{j-1} < \underline{z}(x)$. It holds $\sum_{i\in[n]:\hat{c}_i \geq \underline{z}(x)} x_i > \Gamma$, as otherwise we would have

$$\sum_{i\in[n]:\hat{c}_i > \hat{c}_{j-1}} x_i = \sum_{i\in[n]:\hat{c}_i \geq \underline{z}(x)} x_i \leq \Gamma,$$

which contradicts the minimality of $\underline{z}(x)$. Now, assume that there exists a value $z > \underline{z}(x)$ with $\sum_{i\in[n]:\hat{c}_i \geq z} x_i > \Gamma$. Then we would have

$$\sum_{i\in[n]:\hat{c}_i > \underline{z}(x)} x_i \geq \sum_{i\in[n]:\hat{c}_i \geq z} x_i > \Gamma,$$

contradicting the definition of $\underline{z}(x)$.

We continue with proving equation (5.5). If $\overline{z}(x) = \infty$ holds, then the definition of $\overline{z}(x)$ in Theorem 26 yields

$$\Gamma \leq \sum_{i\in[n]:\hat{c}_i \geq \infty} x_i = 0,$$

and thus

$$\min\left(\{\infty\}\cup\left\{z\in\{\hat{c}_0,\dots,\hat{c}_n\}\middle|\sum_{i\in[n]:\hat{c}_i>z}x_i<\Gamma\right\}\right)=\infty.$$

For simplicity, we denote $\hat{c}_{n+1}=\infty$. Then in the case of $\overline{z}(x)<\infty$, there exists an index $j\in[n]_0$ with $\hat{c}_j=\overline{z}(x)$ and $\hat{c}_{j+1}>\overline{z}(x)$. It holds $\sum_{i\in[n]:\hat{c}_i>\overline{z}(x)}x_i<\Gamma$, as otherwise we would have

$$\sum_{i\in[n]:\hat{c}_i\geq\hat{c}_{j+1}}x_i=\sum_{i\in[n]:\hat{c}_i>\overline{z}(x)}x_i\geq\Gamma,$$

which contradicts the maximality of $\overline{z}(x)$. Now, assume that there exists a value $z<\overline{z}(x)$ with $\sum_{i\in[n]:\hat{c}_i>z}x_i<\Gamma$. Then we would have

$$\sum_{i\in[n]:\hat{c}_i\geq\overline{z}(x)}x_i\leq\sum_{i\in[n]:\hat{c}_i>z}x_i<\Gamma,$$

contradicting the definition of $\overline{z}(x)$. □

Using the above lemma, we are able to prove Theorem 26.

Proof of Theorem 26. The interval $[\underline{z}(x),\overline{z}(x)]$ is well-defined. This is because $\sum_{i\in[n]:\hat{c}_i>z}x_i\leq\Gamma$ is a weaker requirement than $\sum_{i\in[n]:\hat{c}_i>z}x_i<\Gamma$, and we thus have $\underline{z}(x)\leq\overline{z}(x)$ by definition of $\underline{z}(x)$ and equation (5.5). Furthermore, choosing $p_i=(\hat{c}_i-z)^+x_i$ is optimal for given x and z, as we minimize and have $p_i\geq(\hat{c}_i-z)x_i$ and $p_i\geq0$ for all $(x,p,z)\in\mathcal{F}^{\text{BIL}}$.

Now, let $z\geq\underline{z}(x)$ and consider another value $z'>z$ together with an appropriate p' such that $(x,p',z')\in\mathcal{F}^{\text{BIL}}$. By definition of $\underline{z}(x)$, we have

$$\sum_{i\in[n]:\hat{c}_i>z}x_i\leq\sum_{i\in[n]:\hat{c}_i>\underline{z}(x)}x_i\leq\Gamma,$$

and thus

$$\begin{aligned}
\Gamma z+\sum_{i\in[n]}(\hat{c}_i-z)^+x_i=&\Gamma z+\sum_{i\in[n]:\hat{c}_i>z}(z'-z)x_i+\sum_{i\in[n]:\hat{c}_i>z}(\hat{c}_i-z')x_i\\
\overset{(*)}{\leq}&\Gamma z+(z'-z)\Gamma+\sum_{i\in[n]:\hat{c}_i>z}(\hat{c}_i-z')x_i\\
=&\Gamma z'+\sum_{i\in[n]:\hat{c}_i>z'}(\hat{c}_i-z')x_i+\sum_{i\in[n]:z'\geq\hat{c}_i>z}(\hat{c}_i-z')x_i\\
\leq&\Gamma z'+\sum_{i\in[n]:\hat{c}_i>z'}(\hat{c}_i-z')x_i\\
\leq&\Gamma z'+\sum_{i\in[n]}p'_i.
\end{aligned}$$

Hence, the objective value is non-decreasing for $z \geq \underline{z}(x)$. Moreover, if $z' > \overline{z}(x)$ holds, then we have $\overline{z}(x) < \infty$, and thus $\sum_{i\in[n]:\hat{c}_i>\overline{z}(x)} x_i < \Gamma$ by equation (5.4). Then for $z = \overline{z}(x)$, it follows that $(*)$ is a proper inequality and all choices $z' > \overline{z}(x)$ are non-optimal.

Now, let $z \leq \overline{z}(x)$ and consider $z' < z$. This implies $\overline{z}(x) > 0$ and together with the definition of $\overline{z}(x)$, we obtain

$$\sum_{i\in[n]:\hat{c}_i\geq z} x_i \geq \sum_{i\in[n]:\hat{c}_i\geq\overline{z}(x)} x_i \geq \Gamma,$$

and thus

$$\begin{aligned}
\Gamma z + \sum_{i\in[n]} (\hat{c}_i - z)^+ x_i &= \Gamma z' + (z - z')\,\Gamma + \sum_{i\in[n]:\hat{c}_i\geq z} (\hat{c}_i - z)\, x_i \\
&\overset{(**)}{\leq} \Gamma z' + \sum_{i\in[n]:\hat{c}_i\geq z} (z - z')\, x_i + \sum_{i\in[n]:\hat{c}_i\geq z} (\hat{c}_i - z)\, x_i \\
&= \Gamma z' + \sum_{i\in[n]:\hat{c}_i\geq z} (\hat{c}_i - z')\, x_i \\
&= \Gamma z' + \sum_{i\in[n]:\hat{c}_i\geq z'} (\hat{c}_i - z')\, x_i - \sum_{i\in[n]:z>\hat{c}_i\geq z'} (\hat{c}_i - z')\, x_i \\
&\leq \Gamma z' + \sum_{i\in[n]:\hat{c}_i\geq z'} (\hat{c}_i - z')\, x_i \\
&\leq \Gamma z' + \sum_{i\in[n]} p'_i.
\end{aligned}$$

Hence, the objective value is non-increasing for $z \leq \overline{z}(x)$, which shows that all values $z \in [\underline{z}(x), \overline{z}(x)]$ are optimal. Furthermore, if $z' < \underline{z}(x)$ holds, then we have $0 < \underline{z}(x)$, and thus $\sum_{i\in[n]:\hat{c}_i\geq\underline{z}(x)} x_i > \Gamma$ by equation (5.4). Then for $z = \underline{z}(x)$, it follows that $(**)$ is again a proper inequality and all choices $z' < \underline{z}(x)$ are non-optimal. □

As already mentioned, Lee and Kwon [70] showed for $\Gamma \in \mathbb{Z}$ that the number of different values for z to be considered can be decreased from $n+1$ to $\left\lceil \frac{n-\Gamma}{2} \right\rceil + 1$. To see this, it is helpful to sort the deviations $\hat{c}_i$. Therefore, for the remainder of this chapter, we assume without loss of generality that $\hat{c}_0 \leq \ldots \leq \hat{c}_n$ holds. The first observation leading to the reduction of Lee and Kwon is that the values $z \in \{\hat{c}_{n+1-\Gamma}, \ldots, \hat{c}_n\}$ are no better than the value $z = \hat{c}_{n-\Gamma}$, i.e., $\hat{c}_{n-\Gamma} \geq \underline{z}(x)$ for all solutions $x \in \mathcal{F}^{\text{NOM}}$. The second observation is that if the value $z = \hat{c}_i$ is optimal, then $z \in \{\hat{c}_{i-1}, \hat{c}_{i+1}\}$ also contains an optimal choice. To put it in terms of Theorem 26: if we have $\hat{c}_i \in [\underline{z}(x), \overline{z}(x)]$, then it also holds $\{\hat{c}_{i-1}, \hat{c}_{i+1}\} \cap [\underline{z}(x), \overline{z}(x)] \neq \emptyset$. This implies that $\mathcal{Z} = \{\hat{c}_0, \hat{c}_2, \hat{c}_4, \ldots, \hat{c}_{n-\Gamma}\}$ contains an optimal choice for z. The following statement generalizes the first observation to $\Gamma \in \mathbb{R}_{\geq 0}$. Furthermore, both observations are strengthened by using conflicts and a clique partition, which we already compute to obtain the strengthened formulations from Section 5.3.1, to reduce the set $\mathcal{Z}$.

Proposition 28. *Let $\mathcal{Q}$ be a partition of $[n]$ into cliques and $q : [n] \to \mathcal{Q}$ be the mapping that assigns each index $j \in [n]$ its corresponding clique $Q \in \mathcal{Q}$ with $j \in Q$. For*

$$i^{\max} = \min\left(\{n\} \cup \{i \in [n-1]_0 | \, |\{q(i+1), \ldots, q(n)\}| \leq \Gamma\}\right),$$

we have $\hat{c}_{i^{\max}} \geq \underline{z}(x)$ for all solutions $x \in \mathcal{F}^{NOM} \cap \{0,1\}^n$ and there exists an optimal solution (x, p, z) to ROB with $z \in \{\hat{c}_0, \ldots, \hat{c}_{i^{\max}}\}$.

Furthermore, let $G = ([n], E)$ be a conflict graph for ROB and $\Gamma \in \mathbb{Z}$. Let $\mathcal{Z} \subseteq \{\hat{c}_0, \ldots, \hat{c}_{i^{\max}}\}$ such that $\hat{c}_{i^{\max}} \in \mathcal{Z}$ and for every $i \in [i^{\max} - 1]_0$ it holds

- *$\hat{c}_i \in \mathcal{Z}$ or*
- *there exists an index $k < i$ with $\hat{c}_k \in \mathcal{Z}$ and for all $j \in \{k+1, \ldots, i-1\}$ there exists an edge $\{j, i\} \in E$ in the conflict graph G.*

Then there exists an optimal solution (x, p, z) to ROB with $z \in \mathcal{Z}$.

Proof. The first part of the statement is easy to see when considering Theorem 26, as we have

$$\sum_{i \in [n]: \hat{c}_i > \hat{c}_{i^{\max}}} x_i \leq \Gamma$$

for all $x \in \mathcal{F}^{\text{NOM}} \cap \{0,1\}^n$, and thus $\hat{c}_{i^{\max}} \geq \underline{z}(x)$. Since the objective value is non-decreasing for $z \geq \underline{z}(x)$, we do not have to consider $z \in \{\hat{c}_{i^{\max}+1}, \ldots, \hat{c}_n\}$.

For the second part, let $x \in \mathcal{F}^{\text{NOM}} \cap \{0,1\}^n$ be an arbitrary solution to NOM and $\mathcal{Z}$ be a set fulfilling the above properties. It suffices to show $\mathcal{Z} \cap [\underline{z}(x), \overline{z}(x)] \neq \emptyset$. Note that $0 = \hat{c}_0 \in \mathcal{Z}$ holds, as there exists no index $k < 0$. Hence, we can assume that $\overline{z}(x) > 0$ holds, as there is nothing left to show otherwise. Assume that $\overline{z}(x) \geq \hat{c}_{i^{\max}}$ holds. As we have $\hat{c}_{i^{\max}} \geq \underline{z}(x)$, it follows $\hat{c}_{i^{\max}} \in [\underline{z}(x), \overline{z}(x)]$, and thus $\mathcal{Z} \cap [\underline{z}(x), \overline{z}(x)] \neq \emptyset$. We are therefore left with $0 < \overline{z}(x) < \hat{c}_{i^{\max}}$. Since

$$\begin{aligned}\overline{z}(x) &= \min\left\{ z \in \{\hat{c}_1, \ldots, \hat{c}_{i^{\max}-1}\} \,\middle|\, \sum_{i \in [n]: \hat{c}_i > z} x_i < \Gamma \right\} \\ &= \max\left\{ z \in \{\hat{c}_1, \ldots, \hat{c}_{i^{\max}-1}\} \,\middle|\, \sum_{i \in [n]: \hat{c}_i \geq z} x_i \geq \Gamma \right\}\end{aligned}$$

is well-defined, there exists an index $i^* \in [i^{\max} - 1]$ with $\sum_{i=i^*}^n x_i = \Gamma$ and $\sum_{i=i^*+1}^n x_i = \Gamma - 1$. It holds $\hat{c}_{i^*} = \overline{z}(x)$, as we have

$$\sum_{i \in [n]: \hat{c}_i > \hat{c}_{i^*}} x_i \leq \sum_{i=i^*+1}^{n} x_i = \Gamma - 1 < \Gamma,$$

which implies $\hat{c}_{i^*} \geq \overline{z}(x)$, and

$$\sum_{i \in [n]: \hat{c}_i \geq \hat{c}_{i^*}} x_i \geq \sum_{i=i^*}^{n} x_i = \Gamma,$$

implying $\hat{c}_{i^*} \leq \overline{z}(x)$. If we have $\hat{c}_{i^*} \in \mathcal{Z}$, then there is nothing left to show. Otherwise, there exists an index $k < i^*$ with $\hat{c}_k \in \mathcal{Z}$ and an edge $\{j, i^*\} \in E$ in the conflict graph for all $j \in \{k+1, \dots, i^*-1\}$. Since $x_{i^*} = 1$ holds, we have $\sum_{i=k+1}^{i^*-1} x_i = 0$, and thus

$$\sum_{i \in [n]: \hat{c}_i > \hat{c}_k} x_i \leq \sum_{i=k+1}^{n} x_i = \sum_{i=k+1}^{i^*-1} x_i + \sum_{i=i^*}^{n} x_i = \sum_{i=i^*}^{n} x_i = \Gamma,$$

which implies $\hat{c}_k \geq \underline{z}(x)$. Together with $\hat{c}_k \leq \hat{c}_{i^*} \leq \overline{z}(x)$, we obtain $\hat{c}_k \in [\underline{z}(x), \overline{z}(x)]$. □

Proposition 28 determines the structure of Algorithm 5, which we use to compute a inclusion-wise minimal set $\mathcal{Z}$ satisfying the requested properties. We first compute the index $i^{\max}$ (lines 1 to 5). Afterwards, we add the mandatory deviation $\hat{c}_0$ (line 6) and check whether $\Gamma \in \mathbb{Z}$ holds (line 7). If so, we evaluate for all $i \in \{1, \dots, i^{\max}-1\}$ if deviation $\hat{c}_i$ has to be added according to Proposition 28 (lines 8 to 12). Lastly, we add deviation $\hat{c}_{i^{\max}}$, which always needs to be considered (line 13). Note that the second part of Proposition 28 only holds for $\Gamma \in \mathbb{Z}$, as we have $\underline{z}(x) = \overline{z}(x)$ otherwise, which makes it impossible to skip deviations. Hence, in the case of $\Gamma \notin \mathbb{Z}$, we only use the first part of Proposition 28 (line 15).

Algorithmus 5 : Procedure for filtering possible optimal values of z.

Input : An uncertainty budget Γ, sorted deviations $\{\hat{c}_0, \dots, \hat{c}_n\}$, a conflict graph $G = ([n], E)$, a clique partition $\mathcal{Q} \subseteq 2^{[n]}$, and a corresponding mapping $q : [n] \mapsto \mathcal{Q}$

Output : A subset $\mathcal{Z} \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$ containing an optimal value of z

```
1  Initialize i^max = n + 1 and Q' = ∅
2  while i^max > 0 and |Q'| ≤ Γ do
3      reduce i^max ← i^max − 1
4      if i^max > 0 then
5          Add Q' ← Q' ∪ {q(i^max)}
6  Initialize Z = {ĉ_0}
7  if Γ ∈ ℤ then
8      for i = 1, ..., i^max − 1 do
9          if ĉ_i ∉ Z then
10             Let k = max{j ∈ {0, ..., i − 1} | ĉ_j ∈ Z}
11             if {k + 1, ..., i − 1} ⊊ N(i) then
12                 Add Z ← Z ∪ {ĉ_i}
13     Add Z ← Z ∪ {c_{i^max}}
14 else
15     Add Z ← Z ∪ {ĉ_1, ..., ĉ_{i^max}}
16 return Z
```

5.6 The Branch and Bound Algorithm

After paving the way with the theoretical results of the previous sections, we now introduce the components of our branch and bound algorithm in detail. That is, we describe our approach for computing dual and primal bounds, our pruning rules as well as our node selection, branching, and warm starting strategies. A summary of the components, merged into one algorithm, is given in Section 5.6.7. An overview on different strategies regarding the components of branch and bound algorithms is provided by Morrison et al. [77].

For the remainder of this chapter, $\mathcal{Z} \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$ will be a set of possible values for z, as constructed by Algorithm 5. To ease notation, we will refer to the considered subsets $Z \subseteq \mathcal{Z}$ as *nodes* in a rooted *branching tree*, where $\mathcal{Z}$ is the *root node* and Z' is a *child node* of Z if it emerges directly via branching. Furthermore, we denote with $\mathcal{N} \subseteq 2^{\mathcal{Z}}$ the set of *active nodes*, that are the not yet pruned leaves of our branching tree, which are still to be considered.

5.6.1 Dual Bounding

The focus of this work is primarily on the computation of strong dual bounds $\underline{v}(Z)$. We introduced the strong clique reformulation $\text{ROB}(Z, \mathcal{Q})$ and the smaller Lagrangean relaxation $\text{LRR}(Z, \mathcal{Q}, \lambda)$, yielding dual bounds $\underline{v}(Z) = v^{\text{R}}(\text{ROB}(Z, \mathcal{Q}))$ and $\underline{v}(Z) = v(\text{LRR}(Z, \mathcal{Q}, \lambda))$ respectively. In the following, we show that we can obtain even better bounds by restricting ourselves to solutions fulfilling the optimality criterion in Theorem 26.

5.6.1.1. Deriving Dual Bounds from ROB (Z)

Imagine that we just solved a mixed-integer robust subproblem $\text{ROB}(Z)$ and observed that the optimal objective value $v(\text{ROB}(Z))$ is significantly higher than the current primal bound $\overline{v}$. Furthermore, imagine that there exists a yet to be considered value z' in an active node $Z' \in \mathcal{N}$ that is very close to one of the just considered values $z \in Z$. Note that the nominal subproblem $\text{NOS}(z)$, arising from fixing z, differs only slightly in its objective function $\Gamma z + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - z)^+\right) x_i$ from the nominal subproblem $\text{NOS}(z')$. This suggests that the objective value $v(\text{NOS}(z'))$ is probably not too far from $v(\text{NOS}(z))$. Since $v(\text{ROB}(Z))$ is higher than $\overline{v}$ and also a dual bound on $v(\text{NOS}(z))$, we might be able to prune z' without considering $\text{ROB}(Z')$ if we are able to carry over some information from $\text{NOS}(z)$ to $\text{NOS}(z')$. In fact, Hansknecht et al. [53] showed that there exists a relation between the optimal solution values $v(\text{NOS}(z))$ for different values z.

Lemma 29 (Hansknecht et al. [53])**.** *For $z' \leq z$, we have*

$$v(\textit{NOS}(z')) \geq v(\textit{NOS}(z)) - \Gamma(z - z').$$

Proof. The objective function $\Gamma z + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - z)^+ \right) x_i$ of NOS (z) is non-increasing in z when omitting the constant term Γz. It follows $v\left(\text{NOS}\left(z'\right)\right) - \Gamma z' \geq v\left(\text{NOS}\left(z\right)\right) - \Gamma z$, which proves the statement. □

Accordingly, in addition to the dual bound $\underline{v}\left(Z'\right)$ for a node $Z' \in \mathcal{N}$, we can also maintain individual dual bounds $\underline{v}\left(z'\right)$ on $v\left(\text{NOS}\left(z'\right)\right)$ with $\underline{v}\left(z'\right) = v\left(\text{ROB}\left(Z\right)\right) - \Gamma\left(\underline{z} - z'\right)$ for $z' < \underline{z}$ after solving ROB (Z). The dual bound for a node Z' is then the maximum of the relaxation value $v^{\text{R}}\left(\text{ROB}\left(Z', \mathcal{Q}\right)\right)$ or $v\left(\text{LRR}\left(Z', \mathcal{Q}, \lambda\right)\right)$ and the minimum of all individual bounds $\min\left\{\underline{v}\left(z'\right) | z' \in Z'\right\}$.

While this already strengthens the dual bounds in our branch and bound algorithm, we can improve the results of Hansknecht et al. even more by using the optimality criterion from Theorem 26 and the clique partition $\mathcal{Q}$ from Section 5.3.1. Since we are solely interested in optimal solutions to ROB, it is sufficient to only consider solutions to NOS (z') that fulfill the optimality criterion, i.e., solutions x' with $z' \in \left[\underline{z}\left(x'\right), \overline{z}\left(x'\right)\right]$. If an optimal solution to NOS (z') does not fulfill this property, then z' is no optimal choice in the first place and can therefore be pruned. Accordingly, we establish an improved bound that is not a dual bound on $v\left(\text{NOS}\left(z'\right)\right)$ but a dual bound on the objective value of all solutions to NOS (z') fulfilling the optimality criterion.

Let x' be such a solution to NOS (z') with objective value v'. Note that x' is also a feasible solution to NOS (z) and let $v \geq v\left(\text{NOS}\left(z\right)\right)$ be the corresponding objective value. For $z' < z$, the value v' is decreased by $\delta^{\text{dec}} = \Gamma\left(z - z'\right)$ compared to v but increased by $\delta^{\text{inc}} = \sum_{i \in [n]} \left(\left(\hat{c}_i - z'\right)^+ - \left(\hat{c}_i - z\right)^+ \right) x'_i$. This yields the estimation

$$v' = v - \delta^{\text{dec}} + \delta^{\text{inc}} \geq v\left(\text{NOS}\left(z\right)\right) - \delta^{\text{dec}} + \delta^{\text{inc}} \tag{5.6}$$

of the objective value v'. The decrease by δ^{dec} is taken into account in the estimation of Lemma 29 but the increase δ^{inc} is not. Obviously, δ^{inc} can be zero if we have $x'_i = 0$ for all $i \in [n]$ with $\hat{c}_i > z'$. However, if x' fulfills the optimality criterion and $z' > 0$ holds, then we know from Theorem 26 that there exist at least Γ indices with $\hat{c}_i \geq z'$ and $x'_i = 1$. Assuming that there do not exist Γ indices with $\hat{c}_i = z'$, there must exist at least one $i \in [n]$ with $\hat{c}_i > z'$ and $x'_i = 1$, yielding a positive lower bound on δ^{inc}. Taking conflicts between variables x_i into account, we might even deduce that there must exist some indices with $x'_i = 1$ and very high $\hat{c}_i$, which improves the bound on δ^{inc}.

For $z' > z$, Lemma 29 provides no bound on NOS (z'), although we can apply similar arguments to this case. Observe that inequality (5.6) still holds, with $\delta^{\text{inc}} \leq 0$ and $\delta^{\text{dec}} < 0$. Unfortunately, if we have $x'_i = 1$ for all $i \in [n]$ with $\hat{c}_i > z$, then $\delta^{\text{inc}} < 0$ might have a large absolute value, leading to a weak estimation. However, if x' fulfills the optimality criterion, then we know from Theorem 26 that there exist at most Γ indices with $\hat{c}_i > z'$ and $x'_i = 1$. From this, we can again deduce a lower bound on δ^{inc}, which can also be improved by taking conflicts between variables x_i into account.

Theorem 30. *Let $\mathcal{Q}$ be a partition of $[n]$ into cliques, $z, z' \in \mathbb{R}_{\geq 0}$, and x' be an arbitrary solution to NOS (z') of value v' satisfying $z' \in [\underline{z}(x), \overline{z}(x)]$. It holds $v' \geq v(\text{NOS}(z)) - \delta_z(z')$, where the* estimator $\delta_z(z')$ *is defined as*

$$\delta_z(z') = \begin{cases} (\Gamma - \lfloor \Gamma \rfloor)(z - z') + \sum\limits_{\substack{Q \in \mathcal{Q}: \\ \exists i \in Q: z < \hat{c}_i \leq z'}} \max\{\hat{c}_i - z | i \in Q, z < \hat{c}_i \leq z'\} & \text{for } z' > z, \\ (\Gamma - \lceil \Gamma \rceil)(z - z') + \max\limits_{\substack{\mathcal{Q}' \subseteq \mathcal{Q}: \\ |\mathcal{Q}'| \leq \lceil \Gamma \rceil}} \left\{ \sum\limits_{Q \in \mathcal{Q}'} \max\{z - \hat{c}_i | i \in Q, \hat{c}_i \geq z'\} \right\} & \text{for } 0 < z' < z, \\ \Gamma z & \text{for } z' = 0. \end{cases}$$

Proof. For $z' = 0$, the statement follows from Lemma 29. Otherwise, we obtain an estimation

$$v' \geq v(\text{NOS}(z)) - \left(\Gamma(z - z') + \sum_{i \in [n]} \left((\hat{c}_i - z)^+ - (\hat{c}_i - z')^+ \right) x'_i \right),$$

as in inequality (5.6), by considering the difference in the objective functions of NOS (z') and NOS (z). Consider the case $z' > z$. Since $z' \geq \underline{z}(x')$ holds, it follows from the definition of $\underline{z}(x')$ in Theorem 26 that we have

$$\sum_{i \in [n]: \hat{c}_i > z'} x'_i \leq \sum_{i \in [n]: \hat{c}_i > \underline{z}(x')} x'_i \leq \lfloor \Gamma \rfloor.$$

We obtain

$$\begin{aligned} & \Gamma(z - z') + \sum_{i \in [n]} \left((\hat{c}_i - z)^+ - (\hat{c}_i - z')^+ \right) x'_i \\ =& \Gamma(z - z') + \sum_{i \in [n]: z < \hat{c}_i \leq z'} (\hat{c}_i - z) x'_i + \sum_{i \in [n]: \hat{c}_i > z'} (z' - z) x'_i \\ \leq& \Gamma(z - z') + \sum_{i \in [n]: z < \hat{c}_i \leq z'} (\hat{c}_i - z) x'_i + (z' - z) \lfloor \Gamma \rfloor \\ =& (\Gamma - \lfloor \Gamma \rfloor)(z - z') + \sum_{i \in [n]: z < \hat{c}_i \leq z'} (\hat{c}_i - z) x'_i \\ \leq& (\Gamma - \lfloor \Gamma \rfloor)(z - z') + \max \left\{ \sum_{i \in [n]: z < \hat{c}_i \leq z'} (\hat{c}_i - z) x''_i \middle| x'' \in \mathcal{F}^{\text{NOM}} \cap \{0,1\}^n \right\} \\ \leq& (\Gamma - \lfloor \Gamma \rfloor)(z - z') + \max \left\{ \sum_{i \in [n]: z < \hat{c}_i \leq z'} (\hat{c}_i - z) x''_i \middle| \begin{array}{ll} \sum\limits_{i \in Q} x''_i \leq 1 & \forall Q \in \mathcal{Q} \\ x'' \geq 0 & \end{array} \right\} \\ =& (\Gamma - \lfloor \Gamma \rfloor)(z - z') + \sum_{\substack{Q \in \mathcal{Q}: \\ \exists i \in Q: z < \hat{c}_i \leq z'}} \max\{\hat{c}_i - z | i \in Q, z < \hat{c}_i \leq z'\}, \end{aligned}$$

where the last equality holds since $\mathcal{Q}$ is a partition of $[n]$.

Now, let $0 < z' < z$. Since $z' \leq \overline{z}(x')$ holds, it follows from Theorem 26 that we have

$$\sum_{i\in[n]:\hat{c}_i\geq z'} x'_i \geq \sum_{i\in[n]:\hat{c}_i\geq \overline{z}(x')} x'_i \geq \lceil\Gamma\rceil .$$

It holds

$$\begin{aligned}
&\Gamma(z-z') + \sum_{i\in[n]} \left((\hat{c}_i - z)^+ - (\hat{c}_i - z')^+\right) x'_i \\
=&\Gamma(z-z') + \sum_{i\in[n]:z'\leq\hat{c}_i<z} (z'-\hat{c}_i)\, x'_i + \sum_{i\in[n]:\hat{c}_i\geq z} (z'-z)\, x'_i \\
=&\Gamma(z-z') + \sum_{i\in[n]:\hat{c}_i\geq z'} (z'-\min\{z,\hat{c}_i\})\, x'_i \\
\leq&\Gamma(z-z') + \max\left\{ \sum_{i\in[n]:\hat{c}_i\geq z'} (z'-\min\{z,\hat{c}_i\})\, x''_i \,\middle|\, \begin{array}{r} \sum_{i\in[n]:\hat{c}_i\geq z'} x''_i \geq \lceil\Gamma\rceil \\ x''\in\mathcal{F}^{\mathrm{NOM}}\cap\{0,1\}^n \end{array} \right\} \\
\leq&\Gamma(z-z') + \max\left\{ \sum_{i\in[n]:\hat{c}_i\geq z'} (z'-\min\{z,\hat{c}_i\})\, x''_i \,\middle|\, \begin{array}{r} \sum_{i\in[n]:\hat{c}_i\geq z'} x''_i = \lceil\Gamma\rceil \\ \sum_{i\in Q} x''_i \leq 1\ \forall Q\in\mathcal{Q} \\ x''\geq 0 \end{array} \right\} \\
=&(\Gamma-\lceil\Gamma\rceil)(z-z') + \max\left\{ \sum_{i\in[n]:\hat{c}_i\geq z'} (z-\min\{z,\hat{c}_i\})\, x''_i \,\middle|\, \begin{array}{r} \sum_{i\in[n]:\hat{c}_i\geq z'} x''_i = \lceil\Gamma\rceil \\ \sum_{i\in Q} x''_i \leq 1\ \forall Q\in\mathcal{Q} \\ x''\geq 0 \end{array} \right\} \\
=&(\Gamma-\lceil\Gamma\rceil)(z-z') + \max_{\substack{\mathcal{Q}'\subseteq\mathcal{Q}:\\ |\mathcal{Q}'|=\lceil\Gamma\rceil}} \left\{ \sum_{Q\in\mathcal{Q}'} \max\{z-\min\{z,\hat{c}_i\} | i\in Q, \hat{c}_i\geq z'\} \right\} \\
=&(\Gamma-\lceil\Gamma\rceil)(z-z') + \max_{\substack{\mathcal{Q}'\subseteq\mathcal{Q}:\\ |\mathcal{Q}'|\leq\lceil\Gamma\rceil}} \left\{ \sum_{Q\in\mathcal{Q}'} \max\{z-\hat{c}_i | i\in Q, \hat{c}_i\geq z'\} \right\},
\end{aligned}$$

which concludes the proof. □

The above statement enables us not only to compute bounds for $z' > z$ but also stronger bounds for $0 < z' < z$. Note that for $z' = 0$, we have to use the dual bound from Lemma 29, since Theorem 26 provides no statement on the required structure of x' in this case.

In our branch and bound algorithm, we use the estimators $\delta_{\underline{z}}(z')$ for all $z' < \underline{z}$ and $\delta_{\overline{z}}(z')$ for $z' > \overline{z}$ after solving ROB(Z). Accordingly, we define for $Z \subseteq \mathcal{Z}$ the estimators

$$\delta_Z(z') = \begin{cases} \delta_{\underline{z}}(z') & \text{for } z' < \underline{z}, \\ \delta_{\overline{z}}(z') & \text{for } z' > \overline{z}. \end{cases}$$

The improved bounds $v\left(\text{ROB}\left(Z\right)\right) - \delta_Z\left(z'\right)$ come at the cost of a higher computational effort compared to the bounds from Lemma 29. However, the additional overhead is marginal, as we can solve the involved maximization problems in linear time and compute all estimators $\delta_{\underline{z}}\left(z'\right)$, or $\delta_{\overline{z}}\left(z'\right)$ respectively, simultaneously. Algorithm 6 describes our approach for computing the estimators for a set $\mathcal{Z}' \subseteq \mathcal{Z}$ of remaining values z'.

Algorithmus 6 : Procedure for computing estimators $\delta_z\left(z'\right)$.

Input : A set $\mathcal{Z}' = \left\{z'_1, \ldots, z'_p\right\} \subseteq \mathcal{Z}$ with $z'_1 < \ldots < z'_p$, values $\underline{z}, \overline{z} \in \mathbb{R}_{\geq 0}$, an uncertainty budget Γ, sorted deviations $\{\hat{c}_0, \ldots, \hat{c}_n\}$, a clique partition $\mathcal{Q} \subseteq 2^{[n]}$, and a corresponding mapping $q : [n] \mapsto \mathcal{Q}$

Output : Estimators $\delta_{\underline{z}}\left(z'\right)$ for $z' < \underline{z}$ and $\delta_{\overline{z}}\left(z'\right)$ for $z' > \overline{z}$

1 Let $\ell = \min\left\{i \in [p] | z'_i > \overline{z}\right\}$ and $k = \min\left\{i \in [n]_0 | \hat{c}_i > \overline{z}\right\}$
2 Initialize estimator $\delta = 0$, set of considered cliques $\mathcal{Q}' = \emptyset$, and mapping to the corresponding index $q^{-1} : \mathcal{Q}' \to [n]$
3 **for** $j = \ell, \ldots, p$ **do**
4 **while** $\hat{c}_k \leq z'_j$ **do**
5 **if** $q\left(k\right) \in \mathcal{Q}'$ **then**
6 Update $\delta \leftarrow \delta - \left(\hat{c}_{q^{-1}(q(k))} - \overline{z}\right)$
7 Add $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{q\left(k\right)\}$ and set $q^{-1}\left(q\left(k\right)\right) = k$
8 Update $\delta \leftarrow \delta + \left(\hat{c}_k - \overline{z}\right)$ and increase $k \leftarrow k + 1$
9 Set $\delta_{\overline{z}}\left(z'_j\right) = \left(\Gamma - \lfloor\Gamma\rfloor\right)\left(\overline{z} - z'_j\right) + \delta$
10 Let $\ell = \max\left\{i \in [p] | z'_i < \underline{z}\right\}$ and $k = \max\left\{i \in [n]_0 | \hat{c}_i < \underline{z}\right\}$
11 Set $\delta = 0$, $\mathcal{Q}' = \emptyset$, and initialize empty list L
12 **for** $j = \ell, \ldots, 1$ **do**
13 **if** $z'_j = 0$ **then**
14 Set $\delta_{\underline{z}}\left(z'_j\right) = \Gamma z$
15 **else**
16 **while** $\hat{c}_k \geq z'_j$ **do**
17 **if** $q\left(k\right) \in \mathcal{Q}'$ **then**
18 Update $\delta \leftarrow \delta - \left(\underline{z} - \hat{c}_{q^{-1}(q(k))}\right)$ and remove $q^{-1}\left(q\left(k\right)\right)$ from L
19 Add $\mathcal{Q}' \leftarrow \mathcal{Q}' \cup \{q\left(k\right)\}$, set $q^{-1}\left(q\left(k\right)\right) = k$, and append k to L
20 Update $\delta \leftarrow \delta + \left(\underline{z} - \hat{c}_k\right)$ and decrease $k \leftarrow k - 1$
21 **while** $|L| > \lceil\Gamma\rceil$ **do**
22 Update $\delta \leftarrow \delta - \left(\underline{z} - \hat{c}_{L[0]}\right)$
23 Remove $\mathcal{Q}' \leftarrow \mathcal{Q}' \backslash \{q\left(L\left[0\right]\right)\}$ and delete $L\left[0\right]$ from L
24 Set $\delta_{\underline{z}}\left(z'_j\right) = \left(\Gamma - \lceil\Gamma\rceil\right)\left(\underline{z} - z'_j\right) + \delta$
25 **return** $\delta_{\underline{z}}$ *and* $\delta_{\overline{z}}$

We first compute $\delta_{\overline{z}}\left(z'\right)$ for $z' \in \{z' \in \mathcal{Z}' | z' > \overline{z}\}$ (lines 1 to 9). For computing $\delta_{\overline{z}}\left(z'_j\right)$, we consider all deviations $\hat{c}_k$ with $\overline{z} < \hat{c}_k \leq z'_j$ (line 4) and add the corresponding value $\hat{c}_k - \overline{z}$ (line 8). Furthermore, we mark the clique $q\left(k\right)$ containing k as considered by adding it to the set $\mathcal{Q}'$ and we associate the clique $q\left(k\right)$ with the index k by maintaining a mapping $q^{-1} : \mathcal{Q}' \to [n]$ (line 7). However, if $q\left(k\right)$ was already contained within $\mathcal{Q}'$, then we considered

an index $k' = q^{-1}(q(k))$ with $q(k) = q(k')$ before k and counted the value $\hat{c}_{k'} - \overline{z}$ towards $\delta_{\overline{z}}(z')$. Hence, either $\hat{c}_k - \overline{z}$ or $\hat{c}_{k'} - \overline{z}$ has to be subtracted, as we only count the highest value per clique. Since we iterate over the deviations in a non-decreasing order, it holds $\hat{c}_k - \overline{z} \geq \hat{c}_{k'} - \overline{z}$, which is why we subtract $\hat{c}_{k'} - \overline{z}$ (line 6). Note that we do not have to consider all values $\left\{\hat{c}_k \middle| \overline{z} < \hat{c}_k \leq z'_j\right\}$ for computing $\delta_{\overline{z}}\left(z'_j\right)$ if we already considered the values $\left\{\hat{c}_k \middle| \overline{z} < \hat{c}_k \leq z'_{j-1}\right\}$ for $\delta_{\overline{z}}\left(z'_{j-1}\right)$. Instead, we construct $\delta_{\overline{z}}\left(z'_j\right)$ on the basis of $\delta_{\overline{z}}\left(z'_{j-1}\right)$ and only iterate over $\left\{\hat{c}_k \middle| z'_{j-1} < \hat{c}_k \leq z'_j\right\}$.

The computation of $\delta_{\underline{z}}(z')$ for $z' \in \{z' \in \mathcal{Z}' | z' < \underline{z}\}$ is almost analogous (lines 10 to 24). The difference here is that we only consider up to $\lceil \Gamma \rceil$ values $\underline{z} - \hat{c}_k$. Hence, we not only maintain the set $\mathcal{Q}'$ and the mapping q^{-1} but also a list containing the indices of currently added values $\underline{z} - \hat{c}_k$. The list is updated every time we subtract (line 18) or add (line 20) a value $\underline{z} - \hat{c}_k$. Furthermore, since we iterate reversely over $\left\{\hat{c}_k \middle| z'_j \leq \hat{c}_k < \underline{z}\right\}$, the list is ordered non-decreasing with respect to $\underline{z} - \hat{c}_k$. Hence, before assigning $\delta_{\underline{z}}\left(z'_j\right)$, we check whether L contains more than $\lceil \Gamma \rceil$ elements and, if necessary, remove the first $\lceil \Gamma \rceil - |L|$ indices together with their value $\underline{z} - \hat{c}_k$ and their clique $q(k)$ (lines 21 to 23).

5.6.1.2. Optimality-Cuts

Consider a node $Z \subseteq \mathcal{Z}$ of our branching tree and assume that (x, p, z) is a solution to ROB(Z) with $[\underline{z}(x), \overline{z}(x)] \cap Z = \emptyset$. We know from Theorem 26 that it is needless to consider x for the subset Z, as there exists a different set $Z' \subseteq \mathcal{Z}$ with $[\underline{z}(x), \overline{z}(x)] \cap Z' \neq \emptyset$ if x is part of a globally optimal solution. Nevertheless, it is possible that (x, p, z) is an optimal solution to ROB(Z), resulting in an unnecessarily weak dual bound $\underline{v}(Z)$. Using the following theorem, we are able to strengthen our formulations so that we only consider solutions (x, p, z) with $[\underline{z}(x), \overline{z}(x)] \cap Z \neq \emptyset$, and thus raise the dual bound $\underline{v}(Z)$.

Theorem 31. *Let $x \in \mathcal{F}^{NOM} \cap \{0,1\}^n$ be a solution to NOM and $\underline{c} \leq \overline{c}$ be some bounds on z. Then $[\underline{z}(x), \overline{z}(x)] \cap [\underline{c}, \overline{c}] \neq \emptyset$ holds if and only if x satisfies*

$$\sum_{i \in [n]: \hat{c}_i > \overline{c}} x_i \leq \lfloor \Gamma \rfloor \tag{5.7}$$

and in the case of $\underline{c} > 0$ also

$$\sum_{i \in [n]: \hat{c}_i \geq \underline{c}} x_i \geq \lceil \Gamma \rceil . \tag{5.8}$$

Proof. We have $[\underline{z}(x), \overline{z}(x)] \cap [\underline{c}, \overline{c}] \neq \emptyset$ if and only if $\underline{z}(x) \leq \overline{c}$ and $\underline{c} \leq \overline{z}(x)$ holds. We first show that $\underline{z}(x) \leq \overline{c}$ holds if and only if x fulfills inequality (5.7). We know from Theorem 26 that $\sum_{i \in [n]: \hat{c}_i > \underline{z}(x)} x_i \leq \Gamma$ holds. Then $\underline{z}(x) \leq \overline{c}$ implies

$$\sum_{i \in [n]: \hat{c}_i > \overline{c}} x_i \leq \sum_{i \in [n]: \hat{c}_i > \underline{z}(x)} x_i \leq \Gamma,$$

and thus inequality (5.7) due to x being binary. Conversely, x cannot fulfill inequality (5.7) if we have $\overline{c} < \underline{z}(x)$, as this contradicts the minimality in the definition of $\underline{z}(x)$.

It is clear to see that $\underline{c} \leq \overline{z}(x)$ applies if we have $\underline{c} = 0$. Hence, it remains to show that for $0 < \underline{c}$, it holds $\underline{c} \leq \overline{z}(x)$ if and only if x fulfills inequality (5.8). We know from Theorem 26 that $\sum_{i \in [n]: \hat{c}_i \geq \overline{z}(x)} x_i \geq \Gamma$ holds. Then $\underline{c} \leq \overline{z}(x)$ implies

$$\sum_{i \in [n]: \hat{c}_i \geq \underline{c}} x_i \geq \sum_{i \in [n]: \hat{c}_i \geq \overline{z}(x)} x_i \geq \Gamma,$$

and thus inequality (5.7). Conversely, x cannot fulfill inequality (5.8) if we have $\overline{z}(x) < \underline{c}$, as this contradicts the maximality in the definition of $\overline{z}(x)$. □

From now on, we add $\underline{c}, \overline{c}$ to the arguments of our problems and formulations when using the above inequalities (5.7) and (5.8) for bounds $\underline{c} \leq \underline{z}$ and $\overline{z} \leq \overline{c}$, e.g., ROB$(Z, \underline{c}, \overline{c})$ and $\mathcal{F}(Z, \underline{c}, \overline{c})$. When solving relaxations in our branch and bound algorithm, we use $\underline{c} = \underline{z}$ and $\overline{c} = \overline{z}$, and thus solve the continuous relaxations of ROB$(Z, Q, \underline{z}, \overline{z})$ and the Lagrangean relaxations LRR$(Z, Q, \lambda, \underline{z}, \overline{z})$. However, optimality-cuts can cause several problems when added to integer robust subproblems ROB(Z), especially with respect to the dual bounds of the previous section. Note that in the proof of Theorem 30, we require x', the solution to NOS(z'), to be feasible for NOS(z) in order to show that $v(\text{NOS}(z)) - \delta_z(z')$ is a dual bound. Analogously, we require x' to be a feasible solution to ROB$(Z, \underline{c}, \overline{c})$ in order to derive a dual bound from $v(\text{ROB}(Z, \underline{c}, \overline{c}))$. That is, if x' does not meet the optimality-cuts, then we cannot derive any dual bounds from $v(\text{ROB}(Z, \underline{c}, \overline{c}))$. However, if $[\underline{z}(x'), \overline{z}(x')] \cap [\underline{c}, \overline{c}] \neq \emptyset$ holds, then x' is according to Theorem 31 a feasible solution to ROB$(Z, \underline{c}, \overline{c})$. This leads to the following generalization of Theorem 30.

Corollary 32. *Let $Z \subseteq \mathbb{R}_{\geq 0}$ and $\underline{c} \leq \overline{c}$ with $Z \subseteq [\underline{c}, \overline{c}]$. Furthermore, let $z' \in [\underline{c}, \overline{c}]$ and x' be an arbitrary solution to NOS(z') of value v' that satisfies $z' \in [\underline{z}(x'), \overline{z}(x')]$. Then $v' \geq v(\text{ROB}(Z, \underline{c}, \overline{c})) - \delta_Z(z')$ holds.*

Accordingly, there is a trade-off in the choice of $\underline{c}, \overline{c}$. On the one hand, the optimal objective value $v(\text{ROB}(Z, \underline{c}, \overline{c}))$, and thus the derived dual bounds for other $z' \in [\underline{c}, \overline{c}]$, increases if the bounds $\underline{c}, \overline{c}$ are narrow. On the other hand, we want to derive dual bounds for as many z' as possible. Furthermore, the optimality-cuts can be hindering for finding good primal bounds. We resolve this trade-off by adding loose optimality-cuts, corresponding to wide bounds $\underline{c}, \overline{c}$, in the beginning and gradually strengthening them as we consider more robust subproblems. Let $\mathcal{Z}^* \subseteq \mathcal{Z}$ be the union of all nodes $Z^* \subseteq \mathcal{Z}$ for which we already solved a robust subproblem ROB$(Z^*, \underline{c}^*, \overline{c}^*)$ and let $\mathcal{Z}' = \bigcup_{Z \in \mathcal{N}} Z$ be the union of all active nodes. For a node $Z \in \mathcal{N}$, we choose $\underline{c}, \overline{c} \in \mathcal{Z}'$ in our branch and bound algorithm as wide as possible around Z such that there exists no $z^* \in \mathcal{Z}^*$ in between, i.e.,

$$\underline{c} = \min\left\{z' \in \mathcal{Z}' \middle| \nexists z^* \in \mathcal{Z}^* : z' \leq z^* < \underline{z}\right\}$$

and

$$\overline{c} = \max\left\{z' \in \mathcal{Z}' \middle| \nexists z^* \in \mathcal{Z}^* : \overline{z} < z^* \leq z'\right\}.$$

In order to see that it is not reasonable to expand the interval $[\underline{c}, \overline{c}]$, consider a value $z' \in \mathcal{Z}' \setminus [\underline{c}, \overline{c}]$. By definition, we already considered a robust subproblem ROB $(Z^*, \underline{c}^*, \overline{c}^*)$ with $z' \in [\underline{c}^*, \overline{c}^*]$ for some node Z^* that contains a value z^* with either $z' < z^* < \underline{z}$ or $\overline{z} < z^* < z'$. Since z^* is closer to z', we have $\delta_{Z^*}(z') \leq \delta_{z^*}(z') < \delta_Z(z')$, and thus already derived a dual bound $\underline{v}(z') = v(\text{ROB}(Z^*, \underline{c}^*, \overline{c}^*)) - \delta_{Z^*}(z')$ that is probably better than a potential dual bound derived from ROB $(Z, \underline{c}, \overline{c})$. Thus, expanding $[\underline{c}, \overline{c}]$ tends to be meaningless for obtaining new dual bounds. Now, assume that there exists a nominal solution x with $[\underline{z}(x), \overline{z}(x)] \cap [\underline{c}, \overline{c}] = \emptyset$ such that (x, p, z) is feasible for ROB (Z) and also defines an improving primal bound $\overline{v}$. In this case, it would be beneficial to expand $[\underline{c}, \overline{c}]$ such that x fulfills the optimality-cuts and we obtain a new incumbent. However, we have seen in the proof of Theorem 26 that the objective value of (x, p, z) is non-increasing for $z \leq \overline{z}(x)$ and non-decreasing for $z \geq \underline{z}(x)$ with the appropriate $p = (\hat{c}_i - z)^+ x_i$. Using the arguments from above, we should have already found a solution (x, p^*, z^*) that is at least as good as (x, p, z) for a previous subproblem ROB $(Z^*, \underline{c}^*, \overline{c}^*)$. Accordingly, expanding $[\underline{c}, \overline{c}]$ is also uninteresting for obtaining new primal bounds.

In the next Section, we show what else we can do except for choosing appropriate bounds on z in order to guide the branch and bound algorithm in the search for primal bounds.

5.6.2 Primal Bounding

The trend towards highly fractional optimal solutions for the continuous relaxation of ROB is not only hindering for obtaining strong dual bounds but also for computing feasible solutions. This is because nearly integer-feasible solutions can often be used to find a nearby feasible solution, e.g., by using the feasibility pump [42]. Hence, we have to provide guidance for the solver in order to consistently obtain strong primal bounds. As the focus of this work is on the robustness structures of ROB, and not the corresponding nominal problem NOM, we implement no heuristics that explicitly compute feasible solutions x to NOM. Nevertheless, our branch and bound algorithm naturally aids in the search for optimal solutions by quickly identifying non-promising values of z. This allows us early on to focus on nodes $Z \subseteq \mathcal{Z}$ containing (nearly) optimal choices for z, for which solving ROB $(Z, \underline{c}, \overline{c})$ is much easier and yields (nearly) optimal solutions to ROB.

Furthermore, even when considering ROB $(Z, \underline{c}, \overline{c})$ for a node $Z \subseteq \mathcal{Z}$ that contains no optimal choice for z, we can potentially derive good primal bounds or even optimal solutions to ROB. In many cases, an optimal solution (x, p, z) to ROB $(Z, \underline{c}, \overline{c})$ does not meet the optimality criterion $z \in [\underline{z}(x), \overline{z}(x)]$, which leaves potential for improving the primal bound provided by $v(\text{ROB}(Z, \underline{c}, \overline{c}))$. Since $\overline{z}(x)$ is easily computable, we can obtain a better primal bound $\overline{v}(x)$ provided by the solution value of $(x, p', \overline{z}(x))$, with $p'_i = (\hat{c}_i - \overline{z}(x))^+ x_i$. Moreover, we cannot only compute $\overline{v}(x)$ for an optimal solution x to ROB $(Z, \underline{c}, \overline{c})$ but any feasible solution the solver reports while solving ROB $(Z, \underline{c}, \overline{c})$. This increases the chance of finding good primal bounds, as an improved sub-optimal solution may provide an even better bound

than an optimal solution to ROB $(Z, \underline{c}, \overline{c})$. We will see in our computational study that our branch and bound algorithm quickly finds optimal solutions to ROB, often while considering the very first robust subproblem. Additionally, the possibility to derive strong primal bounds from sub-optimal solutions, which may be found early on while solving ROB $(Z, \underline{c}, \overline{c})$, will be relevant for our pruning strategy in the next section.

5.6.3 Pruning

In theory, a problem is solved to optimality if the primal bound $\overline{v}$ is equal to a proven dual bound $\underline{v}$. In practice, however, it is neither always necessary to prove $\overline{v} = \underline{v}$, nor is it always possible due to numerical issues. Instead, one considers a problem to be solved if $\overline{v}$ is sufficiently close to $\underline{v}$, that is, it either holds $\overline{v} - \underline{v} \leq t^{\text{abs}}$ or $\frac{\overline{v}-\underline{v}}{|\overline{v}|} \leq t^{\text{rel}}$, where $t^{\text{abs}} > 0$ is the *absolute tolerance* and $t^{\text{rel}} > 0$ is the *relative tolerance*. The concept of "sufficiently solved" problems is also applied to the pruning of nodes $Z \subseteq \mathcal{Z}$ within our branching tree. More specifically, we prune Z not only if $\underline{v}(Z) \geq \overline{v}$ holds but as soon as we have $\overline{v} - \underline{v}(Z) \leq t^{\text{abs}}$ or $\frac{\overline{v}-\underline{v}(Z)}{|\overline{v}|} \leq t^{\text{rel}}$. In our computational study, we choose $t^{\text{abs}} = 10^{-10}$ and $t^{\text{rel}} = 10^{-4}$, which are the default tolerances used by Gurobi [52]. Note that for $\overline{v} = 0$ and $\overline{v} > \underline{v}$, the *relative gap* $\frac{\overline{v}-\underline{v}}{|\overline{v}|}$ is defined to be ∞. If $\underline{v} \geq \overline{v} = 0$ holds, then the relative gap does not matter, since we have $\overline{v} - \underline{v} \leq t^{\text{abs}}$. To simplify notation, we define $\text{tol}(\underline{v}, \overline{v}) = 1$ if the dual and primal bounds $\underline{v}, \overline{v}$ are within tolerance, and thus strong enough for pruning, and $\text{tol}(\underline{v}, \overline{v}) = 0$ otherwise.

Recall that the dual bound $\underline{v}(Z)$ for $Z \subseteq \mathcal{Z}$ is the maximum of the continuous relaxation value $v^{\text{R}}(\text{ROB}(Z, \mathcal{Q}, \underline{z}, \overline{z}))$ or $v(\text{LRR}(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}))$ and the worst individual dual bound $\min\{\underline{v}(z) | z \in Z\}$ from Section 5.6.1.1. Even if $\underline{v}(Z)$ is too weak for pruning the whole node Z, i.e., $\text{tol}(\underline{v}(Z), \overline{v}) = 0$, we might have $\text{tol}(\underline{v}(z), \overline{v}) = 1$ for a value $z \in Z$. Therefore, we apply a further pruning step in addition to the pruning of Z. Every time we consider a node Z, we check for each $z \in Z$ whether it can be pruned according to its individual dual bound $\underline{v}(z)$. This is beneficial, as we obtain a stronger formulation for the resulting subset of Z. Furthermore, before solving a robust subproblem ROB $(Z, \underline{c}, \overline{c})$, we check for all remaining $z' \in \bigcup_{Z \in \mathcal{N}} Z$ whether $\text{tol}(\underline{v}(z'), \overline{v}) = 1$ holds, so that the bounds $\underline{c}, \overline{c}$, as chosen in Section 5.6.1.2, are as narrow as possible.

Once we consider a mixed-integer robust subproblem ROB $(Z, \underline{c}, \overline{c})$, we can monitor the best known dual bound $\underline{v}(\text{ROB}(Z, \underline{c}, \overline{c}))$ and terminate the subproblem as soon as we have $\text{tol}(\underline{v}(\text{ROB}(Z, \underline{c}, \overline{c})), \overline{v}) = 1$. This is especially important for robust subproblems ROB $(Z, \underline{c}, \overline{c})$ corresponding to nodes Z containing values that are far from being optimal. In this case, we are usually aware of a primal bound $\overline{v}$ that is substantially smaller than the optimal solution value $v(\text{ROB}(Z, \underline{c}, \overline{c}))$, allowing for a fast termination. Such a primal bound can either come from a previously considered robust subproblem or from a solution to ROB $(Z, \underline{c}, \overline{c})$ that we improved, as described in the previous section.

Unfortunately, terminating ROB $(Z, \underline{c}, \overline{c})$ prematurely is problematic regarding the dual bounds $v(\text{ROB}(Z, \underline{c}, \overline{c})) - \delta_Z(z')$ computed in Section 5.6.1.1. Note that in practice, we do not nec-

essarily know the optimal solution value $v\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$, and thus use the best known dual bound $\underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ instead. Hence, there is a trade-off in saving time by terminating $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$ early and generating strong dual bounds $\underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right) - \delta_Z\left(z'\right)$. We resolve this trade-off by computing the estimators $\delta_Z\left(z'\right)$ before solving $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$ and constantly evaluating whether improving $\underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ can potentially lead to the pruning of additional values z'. Let $\mathcal{Z}' \cap [\underline{c}, \overline{c}]$ be the remaining values of z for which we computed the estimators $\delta_Z\left(z'\right)$. Furthermore, let $\overline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ be the currently best known primal bound for $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$. For evaluating whether $z' \in \mathcal{Z}' \cap [\underline{c}, \overline{c}]$ can potentially be pruned, we consider three different cases.

Case 1. If $\text{tol}\left(\max\left\{\underline{v}\left(z'\right), \underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right) - \delta_Z\left(z'\right)\right\}, \overline{v}\right) = 1$ holds, then z' can already be pruned.

Case 2. Otherwise, if $\text{tol}\left(\overline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right) - \delta_Z\left(z'\right), \overline{v}\right) = 1$ holds, then z' can be pruned if we manage to increase $\underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ up to $\overline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$.

Case 3. Otherwise, z' can only be pruned if we find a better global primal bound $\overline{v}$.

If Case 1 applies, then z' is irrelevant to the question whether we should terminate $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$ early, as it will be pruned anyway. In contrast, it is unlikely that z' will be pruned if Case 3 applies. We already stated in the previous section that our branch and bound algorithm usually finds (nearly) optimal solutions to ROB while solving the first robust subproblem. Hence, most of the time, the primal bound $\overline{v}$ will not be improved, leaving little chance for z' to be pruned. Accordingly, in our implementation, we continue solving $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$ as long as there exists a value $z' \in \mathcal{Z}' \cap [\underline{c}, \overline{c}]$ for which Case 2 applies. However, since closing the gap between $\underline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ and $\overline{v}\left(\text{ROB}\left(Z, \underline{c}, \overline{c}\right)\right)$ can potentially require much time, we use an additional termination criterion. In our implementation, we also terminate $\text{ROB}\left(Z, \underline{c}, \overline{c}\right)$ if no $z' \in \mathcal{Z}' \cap [\underline{c}, \overline{c}]$ switched to Case 1 within the last 10 seconds. That is, raising the dual bound did not lead to a pruning of an additional z' within this time. Of course, this criterion is highly arbitrary, but it leads to an improvement of our algorithm's performance in our computational study. As parameter tuning is beyond the scope of this work, we leave a detailed analysis of this component and its potential for future research.

5.6.4 Node Selection

The node selection strategy determines the order in which we explore nodes within our branching tree, and thus directly impacts the number of nodes we consider before finding an optimal solution. Hence, a good node selection strategy is critical to the performance of any branch and bound algorithm, as finding an optimal (or at least good) solution quickly enables us to prune more efficiently. A review of different strategies in the context of mixed-integer programming is given by Linderoth and Savelsbergh [72]. A survey on machine learning for node selection is given by Lodi and Zarpellon [73].

Two meta strategies, from which many other strategies emerge as a combination, are *depth-first* and *best-first search*. Depth-first search is based on the last-in-first-out principle, and thus follows a path down the branching tree until a prunable node is reached. In contrast, best-first search ranks the nodes of the branching tree by assigning a value to each node and always picks a node with the best value. Here, we consider the case where the ranking value is equal to the node's dual bound. In this case, best-first search is also called *best-bound search*.

Naturally, both strategies, depth-first and best-bound search, have advantages and disadvantages, as discussed by Linderoth and Savelsbergh [72]. An advantage of depth-first search is that it requires less memory, as the number of active nodes $|\mathcal{N}|$ in the branching tree is relatively small. Furthermore, depth-first search usually finds feasible solutions quickly, as integer-feasible solutions tend to be located deep in the branching tree, where many variables are fixed due to branching. An obvious drawback, however, is that depth-first search may explore many unnecessary nodes and can get stuck in unpromising subtrees if the current primal bound is far from the optimal solution value. In contrast, best-bound search tends to minimize the number of nodes in the branching tree. This is because best-bound search will never select a node whose dual bound is worse than the optimal solution value. However, the drawback of best-bound search is that it may require more memory, as the number of active nodes in the branching tree grows large if there exist many nodes with similar bounds. This can also prevent the algorithm from finding feasible solutions early, since deeper levels of the branching tree are explored late.

The strategy for our branch and bound algorithm can be seen as a hybrid of depth-first and best-bound search. Note that our algorithm switches back and forth between two phases. In phase one, we branch the set $\mathcal{Z}$ into subsets Z and obtain dual bounds from solving linear subproblems. In phase two, we stick to a node $Z \subseteq \mathcal{Z}$ and solve the mixed-integer robust subproblem ROB$(Z, \underline{c}, \overline{c})$. Phase two can be seen as a leaning towards depth-first search, since we focus on the chosen values in Z until the problem ROB$(Z, \underline{c}, \overline{c})$ is either solved to optimality or terminated, as described in the previous section. Since ROB$(Z, \underline{c}, \overline{c})$ is potentially a hard problem, it would be beneficial to only solve it for promising nodes Z, presumably leading to good solutions. We use the dual bound $\underline{v}(Z)$ of a node $Z \subseteq \mathcal{Z}$ as an indicator for the node's potential to contain good solutions (x, p, z) with $z \in Z$, and thus perform a best-bound search in phase one. That is, for the set of active tree nodes $\mathcal{N}$, we always process a node $Z \in \mathcal{N}$ for which the current dual bound $\underline{v}(Z)$ is minimum among all nodes, i.e., $Z \in \operatorname{argmin}\{\underline{v}(Z) | Z \in \mathcal{N}\}$. Fortunately, the drawbacks of best-bound search described above are not critical in our case. Since the number of active nodes that we generate is at most $|\mathcal{N}| \leq |\mathcal{Z}| \leq \left\lceil \frac{n-\Gamma}{2} \right\rceil + 1$, neither memory consumption nor the exploration of too many nodes before finding a feasible solution should be a problem.

5.6.5 Branching and Choice of Relaxations

Much research has been devoted to the question of how to branch efficiently in integer linear programming, see, for example, Achterberg et al. [4] or Linderoth and Savelsbergh [72]. However, the main question that is addressed there is on which integer-infeasible variable to branch. Obviously, this question is uninteresting in our case, since we solely branch on the variable z and hand the robust subproblems to the chosen MILP solver, which manages the branching within the subproblem on its own. Instead, we have to address the question of how to divide a node $Z \subseteq \mathcal{Z}$ so that the branching is efficient. After this decision is taken, we have to choose between solving the continuous relaxation of ROB $(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ or the Lagrangean relaxation LRR $(Z', \mathcal{Q}, \lambda, \underline{z}', \overline{z}')$ for the child nodes $Z' \subseteq Z$. Furthermore, we need to decide whether a node Z should be branched in the first place or whether we solve the corresponding robust subproblem ROB $(Z, \mathcal{Q}, \underline{c}, \overline{c})$ directly as an MILP.

To decide the latter, let $(x, p, z) \in \mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ be an optimal solution to the continuous relaxation of ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$. Note that we consider ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$ instead of the equivalent ROB$^{\mathrm{S}}$ $(Z, \mathcal{Q}, \underline{z}, \overline{z})$ for the sake of simplicity. Since x meets the optimality-cuts from Section 5.6.1.2 with $\underline{c} = \underline{z}$ and $\overline{c} = \overline{z}$, there exists a value $z' \in [\underline{z}, \overline{z}] \cap [\underline{z}(x), \overline{z}(x)]$. The bilinear solution $(x, p', z') \in \mathcal{F}^{\mathrm{BIL}}$ with $p'_i = (\hat{c}_i - z')^+ x_i$ provides an upper bound on the optimal objective value over all solutions in $\mathcal{F}^{\mathrm{BIL}}$ fulfilling the optimality-cuts. This upper bound is easily computable, as we have $v((x, p', z')) = v((x, p'', \underline{z}(x)))$, with $p''_i = (\hat{c}_i - \underline{z}(x))^+ x_i$, and $\underline{z}(x)$ can be determined in linear time. Now, imagine that the objective values $v((x, p, z))$ and $v((x, p', z'))$ are nearly identical. Since $\mathcal{F}^{\mathrm{BIL}}$ is the strongest possible formulation for ROB, there is not much potential for improving the integrality gap of ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$ via branching. Although this does not necessarily imply for $[\underline{c}, \overline{c}] \supsetneq [\underline{z}, \overline{z}]$ that the integrality gap of ROB $(Z, \mathcal{Q}, \underline{c}, \overline{c})$ is also small enough, we use the relation between the objective values $v((x, p, z))$ and $v((x, p', z'))$ as an indicator and stop branching Z once they are sufficiently close to each other. In our implementation, we consider the two values to be sufficiently close if their gap is in the relative or absolute tolerance, that is $\mathrm{tol}(v((x, p, z)), v((x, p', z'))) = 1$, as defined in Section 5.6.3.

If we decide to stop branching Z, then we do not solve ROB $(Z, \mathcal{Q}, \underline{c}, \overline{c})$ right away but first reinsert the node Z into the set of active nodes $\mathcal{N}$. This is because Z was not selected with respect to the just computed dual bound but a dual bound based on the relaxation value of its parent node. Once Z is chosen again with respect to its new (potentially significantly improved) dual bound, we solve the robust subproblem ROB $(Z, \mathcal{Q}, \underline{c}, \overline{c})$ directly as an MILP.

If we decide to branch, then our decision process for whether to consider the continuous relaxation of ROB $(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ or the Lagrangean relaxation LRR $(Z', \mathcal{Q}, \lambda, \underline{z}', \overline{z}')$ for the child nodes $Z' \subseteq Z$ is similar as above. We aim for using the easier to solve Lagrangean relaxations in the beginning, starting at the root node $\mathcal{Z}$, and only revert to the continuous relaxation of ROB $(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ once we are confident that its formulation is close to the bilinear one. Since $v(\mathrm{LRR}(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z})) \leq v^{\mathrm{R}}(\mathrm{ROB}(Z, \mathcal{Q}, \underline{z}, \overline{z})) \leq v^{\mathrm{R}}(\mathrm{ROB}(Z', \mathcal{Q}, \underline{z}', \overline{z}'))$ holds, we take our decision by comparing $v(\mathrm{LRR}(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}))$ to the value of a bilinear

solution. Let $x \in \mathcal{F}^{\text{NOM}}$ be an optimal solution to the Lagrangean relaxation and $(x, p', \underline{z}(x))$ the corresponding bilinear solution with $p'_i = (\hat{c}_i - \underline{z}(x))^+ x_i$. We choose to revert to the continuous relaxation of ROB$(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ if $v(\text{LRR}(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}))$ and $v((x, p', \underline{z}(x)))$ are close enough within tolerance, as defined in Section 5.6.3 but with 10^{-2} instead of 10^{-4} as the relative tolerance. Note that, if the values are even within relative tolerance 10^{-4}, then we can directly stop branching as above, since we know that the formulation $\mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ is sufficiently close to the bilinear formulation.

Assume that we have decided to not stop branching but to split the node $Z \subseteq \mathcal{Z}$ further into child nodes Z_1, Z_2. The child nodes Z_1, Z_2 should form intervals, i.e., $[\underline{z}_1, \overline{z}_1] \cap [\underline{z}_2, \overline{z}_2] = \emptyset$, as otherwise the bounds on z would be unnecessarily wide, leading to weaker formulations. Therefore, we search for a *branching point* $\theta \in [\underline{z}, \overline{z})$ defining $Z_1 = \{z' \in Z | z' \leq \theta\}$ and $Z_2 = \{z' \in Z | z' > \theta\}$. This branching point should be chosen so that the branching is *effective*. That is, if we solved the continuous relaxation of ROB$(Z, \mathcal{Q}, \underline{z}, \overline{z})$, then the computed optimal solution $(x, p, z) \in \mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ should neither be contained in $\mathcal{F}(Z_1, \mathcal{Q}, \underline{z}_1, \overline{z}_1)$, nor in $\mathcal{F}(Z_2, \mathcal{Q}, \underline{z}_2, \overline{z}_2)$. Whereas, if we solved the Lagrangean relaxation LRR$(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z})$, then the objective value of the computed optimal solution $x \in \mathcal{F}^{\text{NOM}}$ should increase for both LRR$(Z_1, \mathcal{Q}, \lambda, \underline{z}_1, \overline{z}_1)$ and LRR$(Z_2, \mathcal{Q}, \lambda, \underline{z}_2, \overline{z}_2)$. The following proposition shows that there exists an effective branching point if the objective value of our computed solution is lower than the objective value of the corresponding bilinear solution. This is always the case, since we otherwise would have stopped branching. For showing the effectiveness, we only use the formulation $\mathcal{F}(Z, \mathcal{Q})$ and the problem LRR$(Z, \mathcal{Q}, \lambda)$, that is, the result even holds when not taking the strengthening of the optimality-cuts into account.

Proposition 33. *Let $(x, p, z) \in \mathcal{F}(Z, \mathcal{Q})$ such that $v((x, p, z)) < v((x, p', \underline{z}(x)))$ holds with $p'_i = (\hat{c}_i - \underline{z}(x))^+ x_i$. Then $\theta = z$ is an effective branching point.*

Furthermore, let $x \in \mathcal{F}^{NOM}$ be an optimal solution to LRR$(Z, \mathcal{Q}, \lambda)$. Then a branching point $\theta \in [\underline{z}, \overline{z})$ is effective if the following inequalities hold

$$\sum_{Q \in \mathcal{Q}} (1 - \lambda_Q) \sum_{i \in Q : \hat{c}_i > \theta} x_i > 0, \tag{5.9}$$

$$\sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q : \hat{c}_i > \theta} x_i < \Gamma. \tag{5.10}$$

If we have $v(LRR(Z, \mathcal{Q}, \lambda)) < v((x, p', \underline{z}(x)))$, with p' as above, then such an effective branching point exists.

Proof. To prove the first statement, remember that $\mathcal{F}(Z) \cap (\mathbb{R}^{2n} \times \{\overline{z}\}) = \mathcal{F}^{\text{BIL}} \cap (\mathbb{R}^{2n} \times \{\overline{z}\})$ holds by Proposition 24. Thus, if we had $z = \overline{z}$, then $v((x, p, z)) < v((x, p', \underline{z}(x)))$ could not hold. Hence, $\theta = z < \overline{z}$ is a feasible branching point with $Z_1, Z_2 \neq \emptyset$. If $z \notin Z$ holds, then we have $\overline{z}_1 < z$, and thus $(x, p, z) \notin \mathcal{F}(Z_1, \mathcal{Q})$. In the case of $z \in Z$, we have $z = \overline{z}_1$, and thus $(x, p, z) \in \mathcal{F}(Z_1, \mathcal{Q})$ would again contradict Proposition 24. Moreover, we always have $z < \underline{z}_2$, and thus $(x, p, z) \notin \mathcal{F}(Z_2, \mathcal{Q})$, which shows the first statement.

For the second statement, note that for $|Z| = 1$, the Lagrangean relaxation $\mathrm{LRR}(Z, \mathcal{Q}, \lambda)$ is equivalent to the continuous relaxation of $\mathrm{ROB}(Z)$, and therefore by Proposition 24 also equivalent to the problem over $\mathcal{F}^{\mathrm{BIL}} \cap (\mathbb{R}^{2n} \times Z)$. Hence, the down-branching is effective if $Z_1 = \{\underline{z}\}$ and the up-branching is effective if $Z_2 = \{\overline{z}\}$. For the general case, we first show that a branching point $\theta \in [\underline{z}, \overline{z})$ is effective if it fulfills inequalities (5.9) and (5.10). We also show that the branching points leading to $Z_1 = \{\underline{z}\}$ and $Z_2 = \{\overline{z}\}$ each meet one of the inequalities. Afterwards, we will prove that there exists a common effective branching point.

Remember that the objective function of $\mathrm{LRR}(Z, \mathcal{Q}, \lambda)$ is

$$\Gamma \underline{z} + \sum_{i \in [n]} \left(c_i + (\hat{c}_i - \overline{z})^+ \right) x_i + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{z} \right)^+ x_i.$$

Let $\overline{z}_1 \leq \theta < \overline{z}$ be the new upper bound for Z_1 and consider the increase in the objective

$$\begin{aligned}
&\sum_{i \in [n]} \left((\hat{c}_i - \overline{z}_1)^+ - (\hat{c}_i - \overline{z})^+ \right) x_i + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q} \left((\min\{\hat{c}_i, \overline{z}_1\} - \underline{z})^+ - (\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+ \right) x_i \\
&= \sum_{\substack{i \in [n]: \\ \hat{c}_i > \overline{z}_1}} (\min\{\hat{c}_i, \overline{z}\} - \overline{z}_1)\, x_i + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{\substack{i \in Q: \\ \hat{c}_i > \overline{z}_1}} (\overline{z}_1 - \min\{\hat{c}_i, \overline{z}\})\, x_i \\
&= \sum_{Q \in \mathcal{Q}} (1 - \lambda_Q) \left(\sum_{\substack{i \in Q: \\ \hat{c}_i > \overline{z}_1}} (\min\{\hat{c}_i, \overline{z}\} - \overline{z}_1)\, x_i \right).
\end{aligned}$$

All summands are non-negative, since we have $\lambda_Q \leq 1$ for all $Q \in \mathcal{Q}$. As $\min\{\hat{c}_i, \overline{z}\} - \overline{z}_1 > 0$ holds for all $\hat{c}_i > \overline{z}_1$, the increase is positive if and only if

$$\sum_{Q \in \mathcal{Q}} (1 - \lambda_Q) \sum_{i \in Q: \hat{c}_i > \overline{z}_1} x_i > 0$$

holds. As the latter is implied by inequality (5.9) and $\overline{z}_1 \leq \theta$, it follows that the increase is positive, and thus the down-branching is effective. Conversely, we know that the increase is positive for $Z_1 = \{\underline{z}\}$, i.e., $\overline{z}_1 = \underline{z}$. Hence, $\theta = \underline{z}$ fulfills inequality (5.9).

Let $\underline{z}_2 > \theta \geq \underline{z}$ be the new lower bound for Z_2 and consider the increase in the objective

$$\begin{aligned}
&\Gamma(\underline{z}_2 - \underline{z}) + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q} \left((\min\{\hat{c}_i, \overline{z}\} - \underline{z}_2)^+ - (\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+ \right) x_i \\
=&\Gamma(\underline{z}_2 - \underline{z}) - \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q: \hat{c}_i \geq \underline{z}} (\min\{\hat{c}_i, \underline{z}_2\} - \underline{z})\, x_i \\
=&\left(\Gamma - \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q: \hat{c}_i \geq \underline{z}} x_i \right) (\underline{z}_2 - \underline{z}) + \sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q: \underline{z}_2 > \hat{c}_i \geq \underline{z}} (\underline{z}_2 - \hat{c}_i)\, x_i.
\end{aligned}$$

All summands are non-negative, since we have $\sum_{i \in Q} x_i \leq 1$ for all cliques $Q \in \mathcal{Q}$ and $\sum_{Q \in \mathcal{Q}} \lambda_Q \leq \Gamma$ holds due to the choice of λ, implying $\sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q: \hat{c}_i \geq \underline{z}} x_i \leq \Gamma$. Assume that $\sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q: \hat{c}_i \geq \underline{z}} x_i < \Gamma$ holds. Then the increase is positive, and thus the up-branching effective for all branching points $\theta \in [\underline{z}, \overline{z})$. Hence, all branching points θ fulfilling

inequality (5.9) are effective. Moreover, $\theta = \underline{z}$ fulfills inequalities (5.9) and (5.10), showing the existential statement from the proposition.

Accordingly, we can assume for the rest of the proof that $\sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}} x_i = \Gamma$ holds. In this case, the increase resulting from the up-branching is positive if and only if we have

$$\sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\underline{z}_2>\hat{c}_i\geq\underline{z}} x_i > 0.$$

This is equivalent to

$$\sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}_2} x_i < \Gamma,$$

which is implied by inequality (5.10) and $\underline{z}_2 > \theta$. This shows that the down-branching is effective. Conversely, we know that the increase is positive for $Z_2 = \{\overline{z}\}$, i.e., $\underline{z}_2 = \overline{z}$. Hence, $\theta = \max\{\hat{c}_i | i \in [n], \hat{c}_i < \overline{z}\}$ fulfills inequality (5.10).

It remains to show that there exists a branching point fulfilling both inequalities (5.9) and (5.10). It is sufficient to consider points $\theta \in \{\hat{c}_0, \ldots, \hat{c}_n\}$, since $\theta' = \max\{\hat{c}_i | \hat{c}_i \leq \theta\}$ yields the same branching as θ. Let $\underline{\theta}, \overline{\theta} \in \{\hat{c}_0, \ldots, \hat{c}_n\} \cap [\underline{z}, \overline{z})$ be such that $\overline{\theta}$ is the maximum branching point fulfilling inequality (5.9) and $\underline{\theta}$ is the minimum branching point fulfilling inequality (5.10). Then every $\theta \in \left[\underline{\theta}, \overline{\theta}\right]$ is effective. We have already shown that $\underline{\theta}, \overline{\theta}$ exist, and thus it only remains to show $\underline{\theta} \leq \overline{\theta}$. For this, we establish three properties. Remember that we can assume that $\sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}} x_i = \Gamma$ holds *(property 1)*. Then there exist no $Q' \in \mathcal{Q}$ and $j \in Q'$ with $\underline{z} \leq \hat{c}_j < \underline{\theta}$ and $x_j \lambda_{Q'} > 0$ *(property 2)*. Otherwise, we would have

$$\sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i>\hat{c}_j} x_i \leq -x_j\lambda_{Q'} + \sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}} x_i < \Gamma,$$

which contradicts the minimality of $\underline{\theta}$. If we have $\overline{\theta} < \underline{\theta}$, Then inequality (5.9) is not fulfilled for $\theta = \underline{\theta}$ due to the maximality of $\overline{\theta}$. Hence, $(1 - \lambda_Q) x_i = 0$, and thus $\lambda_Q x_i = x_i$ *(property 3),* holds for all $Q \in \mathcal{Q}$ and $i \in Q$ with $\hat{c}_i > \underline{\theta}$. Using the three properties, we obtain

$$\begin{aligned}
v\left(\mathrm{LRR}\left(Z, \mathcal{Q}, \lambda\right)\right) =& \Gamma\underline{z} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \overline{z})^+\right) x_i + \sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{z}\right)^+ x_i \\
=& \Gamma\underline{z} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \overline{z})^+\right) x_i + \sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{z}\right) x_i \\
\overset{(1)}{=}& \Gamma\underline{\theta} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \overline{z})^+\right) x_i + \sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{z}} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{\theta}\right) x_i \\
\overset{(2)}{=}& \Gamma\underline{\theta} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \overline{z})^+\right) x_i + \sum_{Q\in\mathcal{Q}} \lambda_Q \sum_{i\in Q:\hat{c}_i\geq\underline{\theta}} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{\theta}\right) x_i \\
\overset{(3)}{=}& \Gamma\underline{\theta} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \overline{z})^+\right) x_i + \sum_{i\in[n]:\hat{c}_i\geq\underline{\theta}} \left(\min\{\hat{c}_i, \overline{z}\} - \underline{\theta}\right) x_i \\
=& \Gamma\underline{\theta} + \sum_{i\in[n]} \left(c_i + (\hat{c}_i - \underline{\theta})^+\right) x_i.
\end{aligned}$$

However, then $v\left(\mathrm{LRR}\left(Z, \mathcal{Q}, \lambda\right)\right)$ equals $v\left(\left(x, p'', \underline{\theta}\right)\right)$ with $p_i'' = \left(\hat{c}_i - \underline{\theta}\right)^+ x_i$. This contradicts our assumption $v\left(\mathrm{LRR}\left(Z, \mathcal{Q}, \lambda\right)\right) < v\left(\left(x, p', \underline{z}\left(x\right)\right)\right)$, since $v\left(x, p'', \underline{\theta}\right) \geq v\left(x, p', \underline{z}\left(x\right)\right)$ holds. It follows $\underline{\theta} \leq \overline{\theta}$, and thus the existence of an effective branching point. □

While $\theta = z$ yields a theoretically effective branching after solving the continuous relaxation of $\mathrm{ROB}\left(Z, \mathcal{Q}, \underline{z}, \overline{z}\right)$, it is usually not a good choice in practice. We observe that the computed value z is often near to one of the bounds $\underline{z}, \overline{z}$. This leads to an unbalanced branching, where the optimal continuous relaxation value $v^{\mathrm{R}}\left(\mathrm{ROB}\left(Z', \mathcal{Q}, \underline{z}', \overline{z}'\right)\right)$ for one child node rises significantly, while the other remains nearly unchanged. This problem is also observed in the context of spatial branch and bound, which is a common approach for solving non-linear optimization problems. In spatial branch and bound, a convex relaxation of a non-linear formulation is considered to obtain lower bounds on the optimal objective value. This relaxation is then strengthened via branching on (continuous) variables occurring in non-linear terms, which is similar to the branching we perform on z to obtain stronger relaxations $\mathcal{F}\left(Z\right)$ of the bilinear formulation $\mathcal{F}^{\mathrm{BIL}}$.

A common choice for the branching point in spatial branch and bound is a convex combination of the variable's value in the current solution and the middle point of the variable's domain [91]. In our case, this translates to choosing $\alpha z + \left(1 - \alpha\right)\left(\underline{z} + \overline{z}\right)/2$ with $\alpha \in [0, 1]$. This value is then often projected into a subinterval to ensure that θ is not at the boundaries of its domain, i.e., $\theta \in \left[\underline{z} + \beta\left(\overline{z} - \underline{z}\right), \overline{z} - \beta\left(\overline{z} - \underline{z}\right)\right]$ with $\beta \in [0, 0.5]$. However, this does not necessarily lead to an effective branching point in our case. To ensure efficiency, we compute the lowest and highest effective branching points $\underline{\theta}, \overline{\theta} \in Z$. Then, instead of using the bounds $\underline{z}, \overline{z}$ to determine our branching point, we use the effective bounds $\underline{\theta}, \overline{\theta}$. That is, we first set the branching point to $\alpha z + \left(1 - \alpha\right)\left(\underline{\theta} + \overline{\theta}\right)/2$ and then project it into the interval $\left[\underline{\theta} + \beta\left(\overline{\theta} - \underline{\theta}\right), \overline{\theta} - \beta\left(\overline{\theta} - \underline{\theta}\right)\right]$, which ensures efficiency. In summary, we obtain

$$\theta = \max\left\{\underline{\theta} + \beta\left(\overline{\theta} - \underline{\theta}\right), \min\left\{\overline{\theta} - \beta\left(\overline{\theta} - \underline{\theta}\right), \alpha z + \left(1 - \alpha\right)\left(\underline{\theta} + \overline{\theta}\right)/2\right\}\right\}.$$

The choice of parameters α and β leaves room for engineering and differ between solvers. For example, SCIP [92] and COUENNE [11] choose θ with default values $\alpha = 0.25$ and $\beta = 0.2$, while ANTIGONE ($\alpha = 0.75$, $\beta = 0.1$) and BARON ($\alpha = 0.7$, $\beta = 0.01$) choose a significantly higher value for α, according to [91]. Again, we don't dive too deep into the engineering and simply take a middle course by choosing $\alpha = 0.5$ and $\beta = 0.1$.

Note that, in case we solved the Lagrangean relaxation $\mathrm{LRR}\left(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}\right)$, we did not compute a value z that we could use to determine θ. Therefore, given the computed optimal solution $x \in \mathcal{F}^{\mathrm{NOM}}$, we use $\underline{z}\left(x\right)$ instead of z and then compute θ as above.

5.6.6 Warm Starts

We already noted that solving the continuous relaxations of ROB (Z) constitutes the main computational load for some instances. Therefore, we introduced the smaller clique reformulation ROB $(Z, \mathcal{Q})$ and the Lagrangean relaxation LRR $(Z, \mathcal{Q}, \lambda)$ to reduce this computational burden. Another important component for accelerating the solving of relaxations is warm starting. Remember from Section 2.4.2 that we can use the simplex basis computed for a node in the branch and bound tree as a starting point for a simplex method when solving the problems of the child nodes.

If we considered the Lagrangean relaxation LRR $(Z, \mathcal{Q}, \lambda)$ for the parent node and now want to solve LRR $(Z', \mathcal{Q}, \lambda)$ with $Z' \subset Z$, then both problems only differ in the objective coefficients. We know from Remark 2 that we can use the basis of the parent problem to warm start the primal simplex method in phase II. Unfortunately, we cannot always start in phase II when using optimality-cuts, since we simultaneously manipulate the objective coefficients and add constraints in this case. When adding the optimality-cuts with respect to $\underline{z}'$ and $\overline{z}'$ to the formulation, we also add their slack variables to the basis of the parent problem, which yields a basis for the new problem. If the solution to the parent problem already meets the new cuts, then the basis remains primal feasible and we can start the primal simplex method in phase II. Otherwise, we start in phase I with a basis that is hopefully close to primal feasibility. Note that, in order to preserve the basis of the parent problem, we need to keep the old optimality-cuts if their slack variables are non-basic. This has no affect on the correctness of the formulation since the new optimality-cuts dominate the old ones. If the slack variables of the old optimality-cuts are basic, then we can remove them according to Remark 2 in order to reduce the size of the formulation.

Assume that we considered the continuous relaxation of ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$ and now solve the relaxation of ROB $(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ for $Z' \subset Z$. If we use formulations $\mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ and $\mathcal{F}(Z', \mathcal{Q}, \underline{z}', \overline{z}')$, then the constraints of the latter formulation dominate the ones of the former. In order to obtain a dual feasible basis for the new problem, we keep the constraints with non-basic slack variables from the old problem and add the new constraints with slack variables set to basic. Doing so, we can warm start the dual simplex in phase II. However, formulation $\mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ contains two robustness constraints $p_Q + z \geq \sum_{i \in Q} (\hat{c}_i - \underline{z})^+ x_i + \underline{z}$ and $p_Q \geq \sum_{i \in Q} (\hat{c}_i - \overline{z})^+ x_i$ for each $Q \in \mathcal{Q}$, and is thus less favored compared to the substituted clique formulation $\mathcal{F}^{\mathrm{S}}(Z, \mathcal{Q}, \underline{z}, \overline{z})$. In the latter, we get rid of the second constraint $p_Q \geq \sum_{i \in Q} (\hat{c}_i - \overline{z})^+ x_i$ by substituting $p_Q = p'_Q + \sum_{i \in Q} (\hat{c}_i - \overline{z})^+ x_i$. By doing so, we move $(\hat{c}_i - \overline{z})^+ x_i$ to the objective function for all $i \in [n]$, which leads to an increase of the objective coefficients. We know from Remark 1 that this does preserve dual feasibility as long as x_i is non-basic. However, changing the objective coefficients of basic variables does not preserve dual feasibility in general. Moreover, the substitution also implies a change in the other robustness constraints, which are transformed from $p_Q + z \geq \sum_{i \in Q} (\hat{c}_i - \underline{z})^+ x_i + \underline{z}$ to $p'_Q + z \geq \sum_{i \in Q} (\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+ x_i + \underline{z}$. While this poses no problem for the newly added robustness constraints using bounds $\underline{z}', \overline{z}'$, the ones inherited from the parent problem need

to remain unchanged to preserve dual feasibility. Thus, we can only apply the substitution for non-basic variables x_i that are not contained in a robustness constraint that is inherited from the parent problem. We obtain the new formulation and the dual feasible basis by consecutively performing the following steps, all of which are admissible according to Remark 1.

1. Remove old robustness constraints with basic slack variables.
2. Apply substitution $p_Q = p'_Q + \sum_{i \in Q \cap N} (\hat{c}_i - \overline{z})^+ x_i$ for all $Q \in \mathcal{Q}$, where $N \subseteq [n]$ contains the indices of variables x_i that are non-basic and do not appear in any remaining robustness constraint.
3. Add new robustness constraints respecting the partially substituted variables p'_Q.

Note that, when considering a sequence of subproblems with $Z \supset Z' \supset Z''$, then we also perform a sequence of substitutions. Thus, if x_i is substituted when moving from Z to Z' but not from Z' to Z'', then we still have x_i substituted for p with respect to $\overline{z}'$ when considering Z''. Furthermore, note that we do not apply the substitution $z = z' + \underline{z}$ performed in Section 5.1, as this almost always implies a change in an inherited robustness constraint.

Finally, assume that we considered the Lagrangean relaxation LRR $(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z})$ for the parent node and now solve the relaxation of ROB $(Z', \mathcal{Q}, \underline{z}', \overline{z}')$ for $Z' \subset Z$. In this case, we add new constraints and have a change in the objective function due to the removal of the Lagrangean terms $\sum_{Q \in \mathcal{Q}} \lambda_Q \sum_{i \in Q} (\min\{\hat{c}_i, \overline{z}\} - \underline{z})^+ x_i$. Hence, we cannot guarantee to preserve primal or dual feasibility of the simplex basis. However, we can still build a reasonable basis to warm start in phase I. Since we alter the objective coefficients in any case, we apply substitution to all variables p_Q, and thus have at most one robustness constraint for each $Q \in \mathcal{Q}$. If there exists any $i \in Q$ with $\min\{\hat{c}_i, \overline{z}'\} - \underline{z}' > 0$, that is, the corresponding robustness constraint is not void, then we add the constraint and set p_Q to basic. Since p_Q appears only in its corresponding robustness constraint, this already yields a basis for the new problem. Hence, we set z to non-basic.

Altogether, this yields a warm starting strategy that enables us to often start our simplex methods in phase II despite the extensive changes we apply after each branching step. This warm starting strategy is the last component of our branch and bound algorithm, which we summarize in the following section.

5.6.7 Summary and Implementation

In this section, we summarize the components of our branch and bound approach and merge them into one algorithm, as described in Algorithm 7. We also discuss some details regarding the implementation of the algorithm, which is written in Java and uses Gurobi [52] as an MILP and LP solver. We refer to the problems ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$ and LRR $(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z})$ in order to simplifying notation in the description of the algorithm. In practice, we actually solve

$\text{ROB}^{\text{S}}(Z, \mathcal{Q}, \underline{z}, \overline{z})$ as mixed-integer subproblems. When considering relaxations, we use the problems described in the last section for warm starting.

Algorithm 7 starts with the preparation for the branch and bound by computing a conflict graph and clique partition (line 1), which are then used to compute the set of possible optimal values $\mathcal{Z}$ (line 2). Afterwards, the set of active nodes $\mathcal{N}$ is initialized with the root node $\mathcal{Z}$. We mark it with $\text{typ}(\mathcal{Z}) = \text{L}$, indicating that we start with the Lagrangean relaxation corresponding to $\mathcal{Z}$ (line 3). Afterwards, we initialize the primal and dual bounds (line 4), as well as the set $\mathcal{Z}^*$ of already considered values $z \in \mathcal{Z}$ for mixed-integer subproblems $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ with $z \in Z$ (line 5). Note that we manage the whole branching tree outside of Gurobi, as it does not provide all *callbacks* to perform the necessary branching and node-selection [52].

After the initialization, our algorithm starts processing the nodes Z within the set of active nodes $\mathcal{N}$ until no node remains, and thus the problem is solved to optimality (line 6). In accordance with Section 5.6.4, we choose a node among those having the lowest dual bound $\underline{v}(Z)$ (line 7). Afterwards, we check whether $\text{typ}(Z) = \text{I}$ holds, that is $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ is marked to be solved as a mixed-integer problem (line 8). If so, we try to prune all remaining values z' in active nodes (lines 9 and 10) in order to reduce Z as much as possible and allow for a choice of tighter bounds $\underline{c}, \overline{c}$ for the optimality-cuts (line 11). We then compute the estimators $\delta_Z(z')$ for the remaining values in $[\underline{c}, \overline{c}]$ (line 12), which we need for our termination strategy of $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ and also for updating the dual bounds $\underline{v}(z')$. We then construct the problem $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ and pass it to the solver (line 13).

When constructing the robust subproblem in practice, we have to avoid some pitfalls regarding numerical issues. Since the deviations $\hat{c}_i$, and thus the values $z \in \mathcal{Z}$, can be arbitrarily close to each other, it is possible that our subproblem contains robustness constraints $p_Q + z \geq \sum_{i \in Q} (\hat{c}_i - \underline{z})^+ x_i + \underline{z}$ and $p_Q \geq \sum_{i \in Q} (\hat{c}_i - \overline{z})^+ x_i$ for which the coefficients on the right-hand side are very small. Such constraints may not only be troublesome for the solver's performance but are also irrelevant in practice: Gurobi considers per default all constraints that are violated by less than the *feasibility tolerance* 10^{-6} as satisfied [52]. Hence, we only add the constraints if at least one coefficient on the right-hand side is grater than 10^{-6}.

Once the subproblem is passed to the solver, we monitor the solution process via callbacks. Using these, the solver allows us to access the current best solution of the subproblem and improve it every time a new incumbent is found, as described in Section 5.6.2 (line 14). Furthermore, we can query the current primal and dual bound of the subproblem in order to decide whether it can be terminated (line 15). After the subproblem is solved or terminated, we remove Z from the set of active nodes, add the values in Z to the set of already considered values $\mathcal{Z}^*$ (line 16), and update dual bounds using the estimators $\delta_Z(z')$ (lines 17 and 18).

If we do not solve $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ directly as an MILP, then we remove Z from the set of active nodes, as it will either be pruned or branched (line 20). In order to obtain a formulation that is as strong as possible, we try to prune all values $z \in Z$ using their individual dual bounds $\underline{v}(z)$ (line 21). Afterwards, we check whether $\text{typ}(Z) = \text{L}$ holds, that is, we solve

Algorithmus 7 : The Branch and Bound Algorithm

Input : An instance of ROB

Output : An optimal solution (x^*, p^*, z^*) of value $\overline{v}$

Compute conflict graph and clique partition $\mathcal{Q}$ as in Section 5.3.2
Compute possible optimal values $\mathcal{Z} \subseteq \{\hat{c}_0, \ldots, \hat{c}_n\}$ as in Algorithm 5
Initialize set of active nodes $\mathcal{N} = \{\mathcal{Z}\}$ with $\mathrm{typ}(\mathcal{Z}) = \mathrm{L}$
Set dual bounds $\underline{v}(\mathcal{Z}) = \underline{v}(z) = -\infty$ for all $z \in \mathcal{Z}$, and primal bound $\overline{v} = \infty$
Initialize set $\mathcal{Z}^* = \emptyset$ of already considered values for robust subproblems
while $\mathcal{N} \neq \emptyset$ **do**
    Choose $Z \in \arg\min \{\underline{v}(Z') | Z' \in \mathcal{N}\}$
    if $\mathrm{typ}(Z) = \mathrm{I}$ **then**
        Let $\mathcal{Z}' = \bigcup_{Z' \in \mathcal{N}} Z'$ be the set of remaining values
        Prune all $z' \in \mathcal{Z}'$ with $\mathrm{tol}(\underline{v}(z'), \overline{v}) = 1$
        Choose $\underline{c}, \overline{c}$, as in Section 5.6.1.2
        Compute estimators $\delta_Z(z')$ for all $z' \in \mathcal{Z}' \cap [\underline{c}, \overline{c}]$ as in Algorithm 6
        Solve ROB$(Z, \mathcal{Q}, \underline{c}, \overline{c})$ with the following **callbacks**
            Update $\overline{v}$ and (x^*, p^*, z^*) for all solutions found as in Section 5.6.2
            Terminate ROB$(Z, \mathcal{Q}, \underline{c}, \overline{c})$ as in Section 5.6.3
        Remove $\mathcal{N} \leftarrow \mathcal{N} \setminus \{Z\}$ and add $\mathcal{Z}^* \leftarrow \mathcal{Z}^* \cup Z$
        Update $\underline{v}(z') \leftarrow \max\{\underline{v}(z'), \underline{v}(\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})) - \delta_Z(z')\}$ for all $z' \in \mathcal{Z}' \cap [\underline{c}, \overline{c}]$
        Update $\underline{v}(Z') \leftarrow \max\{\underline{v}(Z'), \min\{\underline{v}(z') | z' \in Z'\}\}$ for all $Z' \in \mathcal{N}$
    else
        Remove $\mathcal{N} \leftarrow \mathcal{N} \setminus \{Z\}$
        Prune all $z \in Z$ with $\mathrm{tol}(\underline{v}(z), \overline{v}) = 1$
        if $\mathrm{typ}(Z) = \mathrm{L}$ **then**
            Compute optimal solution x to LRR$(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z})$ with λ as in Section 5.4
            Set $\underline{v} = v(\text{LRR}(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}))$ and $(x, p, z) = (x, p', \underline{z}(x))$ with $p'_i = (\hat{c}_i - \underline{z}(x))^+ x_i$
        else
            Compute optimal solution $(x, p, z) \in \mathcal{F}(Z, \mathcal{Q}, \underline{z}, \overline{z})$
            Set $\underline{v} = v((x, p, z))$
        if (x, p, z) *is integer-feasible* **then**
            Potentially update $\overline{v}$ with $v((x, p, z))$ and (x^*, p^*, z^*) with (x, p, z)
        else if $\mathrm{tol}(\underline{v}, \overline{v}) = 0$ **then**
            if $\mathrm{tol}(\underline{v}, (x, p', \underline{z}(x))) = 1$ *with* $p'_i = (\hat{c}_i - \underline{z}(x))^+ x_i$ **then**
                Set $\underline{v}(Z) \leftarrow \max\{\underline{v}, \min\{\underline{v}(z) | z \in Z\}\}$
                Set $\mathrm{typ}(Z) = \mathrm{I}$ and reinsert $\mathcal{N} \leftarrow \mathcal{N} \cup \{Z\}$
            else
                Branch Z into Z_1, Z_2 as in Section 5.6.5
                Set $\underline{v}(Z_i) \leftarrow \max\{\underline{v}, \min\{\underline{v}(z) | z \in Z_i\}\}$ for $i = 1, 2$
                Insert $\mathcal{N} \leftarrow \mathcal{N} \cup \{Z_1, Z_2\}$
                Set $\mathrm{typ}(Z_1) = \mathrm{typ}(Z_2) \in \{\mathrm{L}, \mathrm{R}\}$ as in Section 5.6.5
    Prune all $Z' \in \mathcal{N}$ with $\mathrm{tol}(\underline{v}(Z'), \overline{v}) = 1$
return (x^*, p^*, z^*)

the Lagrangean relaxation (line 22). If so, we compute an optimal solution x (line 23), store its objective value for the Lagrangean relaxation $\underline{v} = v\left(\text{LRR}\left(Z, \mathcal{Q}, \lambda, \underline{z}, \overline{z}\right)\right)$, and compute the corresponding bilinear solution $(x, p, z) = \left(x, p', \underline{z}\left(x\right)\right)$ (line 24). The latter is important, since we require the corresponding solution to ROB in case x is integer-feasible. If we do not solve the Lagrangean relaxation, then we solve the continuous relaxation of $\text{ROB}\left(Z, \mathcal{Q}, \underline{z}, \overline{z}\right)$ and let (x, p, z) be an optimal solution (line 26). As before, we let $\underline{v} = v\left(\left(x, p, z\right)\right)$ be the optimal objective value of the relaxation.

If the computed continuous solution (x, p, z) is integer-feasible (line 28), then we update the primal bound $\overline{v} = v\left(\left(x, p, z\right)\right)$ and current best solution $(x^*, p^*, z^*) = (x, p, z)$ in case we have $\overline{v} > v\left(\left(x, p, z\right)\right)$ (line 29). If the solution is not integer-feasible, then we check whether Z can be pruned using the new dual bound $\underline{v}$ (line 30). If this is not the case, then we decide whether Z should be branched further (line 31). If we decide to solve $\text{ROB}\left(Z, \mathcal{Q}, \underline{c}, \overline{c}\right)$ directly as an MILP, then we store the new dual bound of Z (line 32), reinsert it into $\mathcal{N}$, and mark $\text{typ}\left(Z\right) = \text{I}$ (line 33). Otherwise, we branch Z into subsets Z_1, Z_2 (line 35), compute dual bounds for both child nodes (line 36), insert them into the set of active nodes (line 37), and mark them to be solved as a Lagrangean relaxation $\text{LRR}\left(Z_i, \mathcal{Q}, \lambda, \underline{z}_i, \overline{z}_i\right)$ or the continuous relaxation of $\text{ROB}\left(Z_i, \mathcal{Q}, \underline{z}_i, \overline{z}_i\right)$ (line 38).

After Z is processed, either by solving the robust subproblem or a relaxation, we check whether the potentially obtained new primal and dual bounds allow for a pruning of some active nodes (line 39). If any active nodes remain, we continue with choosing the next node, otherwise we report the optimal solution (x^*, p^*, z^*).

Obviously, it will not always be possible to solve ROB to optimality within a given time limit. Hence, in practice, we also keep track of a dual bound $\underline{v}\left(\text{ROB}\right)$ in order to evaluate the quality of the best solution found. We do this by initializing $\underline{v}\left(\text{ROB}\right) = \infty$ and updating it every time a node Z is pruned, using the corresponding dual bound, i.e., $\underline{v}\left(\text{ROB}\right) \leftarrow \min\left\{\underline{v}\left(\text{ROB}\right), \underline{v}\left(Z\right)\right\}$. After the algorithm is terminated, we again update $\underline{v}\left(\text{ROB}\right) \leftarrow \min\left\{\underline{v}\left(\text{ROB}\right), \underline{v}\left(Z\right) | Z \in \mathcal{N}\right\}$ for all remaining active nodes. By doing so, we make sure that the dual bound $\underline{v}\left(\text{ROB}\right)$ is equal to the minimum dual bound $\underline{v}\left(Z\right)$ of all leaves Z of our branching tree.

The summary of our branch and bound algorithm closes the theoretical part of this chapter. In the next section, we perform an extensive computational study to evaluate the performance of our approach.

5.7 Computational Study

In this section, we use the robustified MIPLIB instances generated in Section 4.6.3 to experimentally evaluate different components of our branch and bound algorithm. We then compare the branch and bound to other approaches from the literature and investigate how these can be improved using our theoretical results.

All experiments are implemented in Java 11 and performed on a single core of a Linux machine with an Intel® Core™ i7-5930K CPU @ 3.50GHz with 2 GB RAM reserved for each calculation. We use Gurobi version 9.5.0 [52] in single thread mode and all other settings at default to solve LPs and MILPs. Furthermore, we use a time limit of 3,600 seconds for each algorithm and instance.

All implemented algorithms [46] and generated test instances [48] are freely available online.

5.7.1 Impact of Components of the Branch and Bound Algorithm

We start by evaluating the different components described in the previous sections in order to get a better understanding for our branch and bound approach. To this end, we individually adjust or disable components, leading to the following variants of our branch and bound algorithm.

BnB	is our branch and bound approach described in Algorithm 7.
BnB-noLagr	never uses Lagrangean relaxations from Section 5.4.
BnB-alwaysLagr	always uses Lagrangean relaxations instead of continuous relaxations of ROB $(Z, \mathcal{Q}, \underline{z}, \overline{z})$.
BnB-noClique	does not compute the conflict graph and clique partition from Section 5.3.
BnB-noFilter	does not filter $\mathcal{Z}$ as in Section 5.5, i.e., $\mathcal{Z} = \{\hat{c}_0, \dots, \hat{c}_n\}$.
BnB-noEstimator	does not use estimators $\delta_Z(z')$ from Section 5.6.1.1.
BnB-noCutLP	does not use optimality-cuts from Section 5.6.1.2 when solving continuous relaxations.
BnB-noCutMIP	does not use optimality-cuts when solving mixed-integer robust subproblems.
BnB-noCut	does not use optimality-cuts at all.
BnB-noPrimal	does not improve primal bounds, as in Section 5.6.2.
BnB-noTermination	does not terminate mixed-integer robust subproblems prematurely, as in Section 5.6.3.
BnB-noWarmstart	does not use warm starting of continuous relaxations, as in Section 5.6.6.

We use the above variants to solve the 804 robust instances constructed in Section 4.6.3 within a time limit of 3,600 seconds, including preprocessing, construction of subproblems,

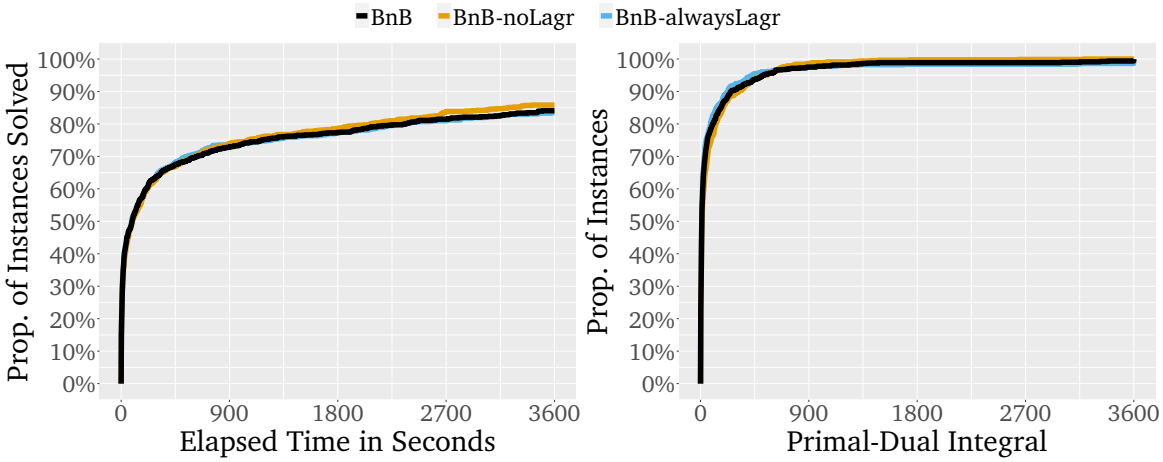


Figure 5.1. Cumulative distribution of computation times and primal-dual integrals for variants of BnB differing in the use of Lagrangean relaxations.

etc. In the following, we evaluate the computational results on an aggregate level as described in Section 4.6.2. Detailed results for each instance and algorithm are provided in [47].

We first compare the different variants of using or not using Lagrangean relaxations. The left graphic in Figure 5.1 shows for BnB, BnB-noLagr, and BnB-alwaysLagr the proportion of instances that were solved within a given number of seconds. The right graphic shows the proportion of instances for which the primal-dual integral is below a specific value after 3,600 seconds. The curves of all three variants are close together for both the computation time and the primal-dual integral. Never using Lagrangean relaxations seems to be slightly better for harder instances, since BnB-noLagr is able to solve 85.8% of all instances within 3,600 seconds, while BnB solves 84.1% and BnB-alwaysLagr solves 83.5%. A detailed look at single instances reveals that we sometimes struggle to solve the Lagrangean relaxations with the warm started primal simplex. For example, BnB-noLagr solves 11 out of 12 instances based on the nominal problem rail01 to optimality, while BnB-alwaysLagr solves none because it spends all its time trying to solve few relaxations. We observe that computation times for relaxations are especially high when the basic solution of the parent problem does not fulfill the optimality-cuts and we thus start at a primal infeasible basis.

Despite these severe issues for some instances, using Lagrangean relaxations overall speeds up the computation, which translates to lower computation times and primal-dual integrals in the shifted geometric mean. Table 5.2 shows that primal-dual integrals of BnB-alwaysLagr are even 19.2% lower compared to BnB-noLagr. This is because we close the optimality gap much faster when considering easier relaxations in the beginning. Given the trade-off between instances solved and primal-dual integrals, we cannot conclude whether applying Lagrangean relaxations is beneficial or not. We note this inconclusiveness for now and will return to this topic later.

Next, we evaluate the impact of cliques and conflict graphs, the filtering of $\mathcal{Z}$, and estimators $\delta_{\mathcal{Z}}(z')$. Note that disabling some component can also have an effect on others. For example, disabling cliques not only prevents us from using the clique reformulation $\mathrm{ROB}(\mathcal{Q})$ but

Table 5.2. Computational results for different variants of BnB. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	BnB	BnB-noLagr	BnB-alwaysLagr	BnB-noClique	BnB-noFilter	BnB-noEstimator
timeout	128	114	133	149	128	168
time	85.94	86.02	85.38	106.79	87.64	123.78
P-D integral	15.82	18.16	14.68	20.36	16.20	23.89

	BnB-noCutLP	BnB-noCutMIP	BnB-noCut	BnB-noPrimal	BnB-noTermination	BnB-noWarmstart
timeout	100	106	83	126	138	122
time	73.62	76.38	67.79	86.51	91.49	91.40
P-D integral	12.40	14.93	12.02	16.07	15.59	20.34

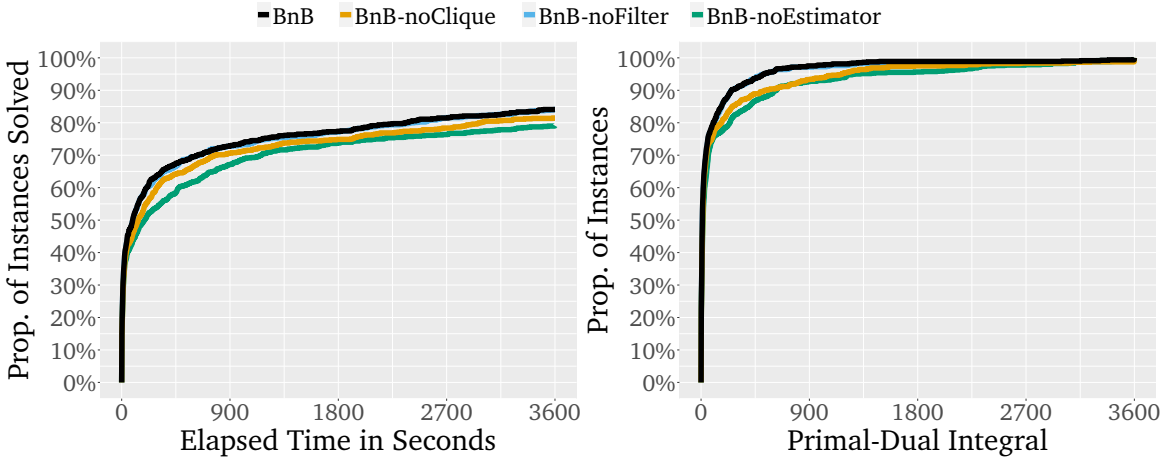


Figure 5.2. Cumulative distribution of computation times and primal-dual integrals for variants of BnB without using cliques, filtering or estimators.

also worsens the filtering of $\mathcal{Z}$ and the estimators $\delta_Z(z')$. Disabling estimators allows us to terminate robust subproblems $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ more aggressively, as raising the dual bound $\underline{v}(\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c}))$ past the current primal bound $\overline{v}$ is no longer beneficial.

As before, Figure 5.2 shows the cumulative distribution of computation times and primal-dual integrals, while Table 5.2 shows shifted geometric mean values. We see that the curves of BnB and BnB-noFilter in Figure 5.2 are almost identical, which shows that our approach is not sensitive to the size of $\mathcal{Z}$. This is quite intuitive, since filtered values $z' \in \{\hat{c}_0, \dots, \hat{c}_n\} \setminus \mathcal{Z}$ are usually close to another value $z \in \mathcal{Z}$ and are therefore implicitly considered within the same subproblem $\text{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ or pruned via estimators $\delta_Z(z')$. Moreover, values of z that are far from being optimal are pruned easily due to their high relaxation-based dual bound. Therefore, filtering only yields a considerable benefit when we are able to discard values z' that are far from other $z \in \mathcal{Z}$ and correspond to (nearly) optimal solutions (x, p, z). While this does not seem to be relevant for our test instances, filtering has at least no structural negative impact and improves the mean values of computation times and primal-dual integrals slightly, as shown in Table 5.2.

The influence of cliques and estimators on the performance is much greater than that of filtering. When not using cliques and conflict graphs, we observe 16.4% more timeouts, an increase of the computation times by 24.3% and of the primal-dual integrals by 28.7%

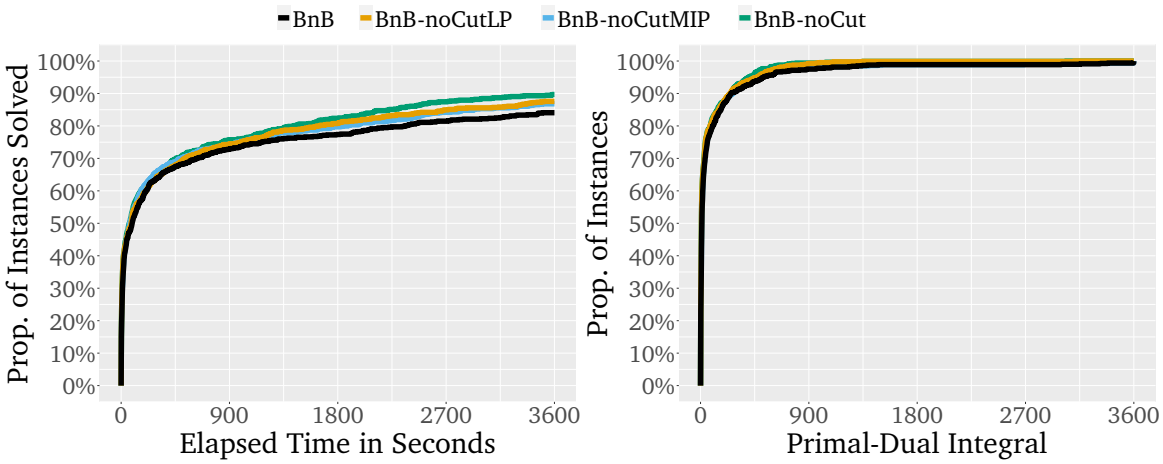


Figure 5.3. Cumulative distribution of computation times and primal-dual integrals for variants of BnB differing in the use of optimality-cuts.

compared to BnB. When disabling estimators, the decline is even more substantial with 31.3% more timeouts, 44.0% higher computation times and 51.0% higher primal-dual integrals. The curves in Figure 5.2 confirm the domination of BnB over BnB-noClique and BnB-noEstimator, as we see that BnB always solves more instances within the same time and below a specific primal-dual integral.

Figure 5.3 shows that disabling optimality-cuts yields a considerable performance improvement, both for relaxed and mixed-integer problems. This might not be surprising for BnB-noCutLP after our discussion on the use of Lagrangean relaxations. We observe much higher computation times for solving relaxations when optimality-cuts are added, since we sometimes have to warm start the simplex method in phase I when applying Lagrangean relaxation with optimality-cuts. As a result, BnB spends 10.30 seconds in the shifted geometric mean for solving relaxations, while BnB-noCutLP only spends 6.40 seconds. This is despite the fact that BnB-noCutLP considers 14.2% more relaxations than BnB on average. The speed-up in solving relaxations translates to BnB-noCutLP having 21.9% fewer timeouts, 14.3% lower computation times, and 21.6% lower primal-dual integrals, as shown in Table 5.2. In particular, BnB-noCutLP is able to solve 9 out of 12 of the already mentioned rail01 instances, of which BnB could not solve any.

When disabling optimality-cuts for mixed-integer subproblems, we have 17.2% fewer timeouts, 11.1% lower computation times, and 5.6% lower primal-dual integrals. This improved performance of BnB-noCutMIP is surprising, as there is no obvious practical interference with other components of our branch and bound. Although optimality-cuts can in theory prevent us from finding good primal bounds early and allow using estimators $\delta_Z(z')$ only for $z' \in [\underline{c}, \overline{c}]$, our careful choice of bounds $\underline{c}, \overline{c}$ for optimality-cuts ensures that these issues do not matter in practice (cf. Section 5.6.1.2). In fact, the primal bound computed by BnB while solving the very first mixed-integer subproblem $\mathrm{ROB}(Z, \mathcal{Q}, \underline{c}, \overline{c})$ is for 98.1% of all instances at most 1% higher than the best primal bound computed by any variant of our branch and bound algorithm. Moreover, the average number of mixed-integer subproblems

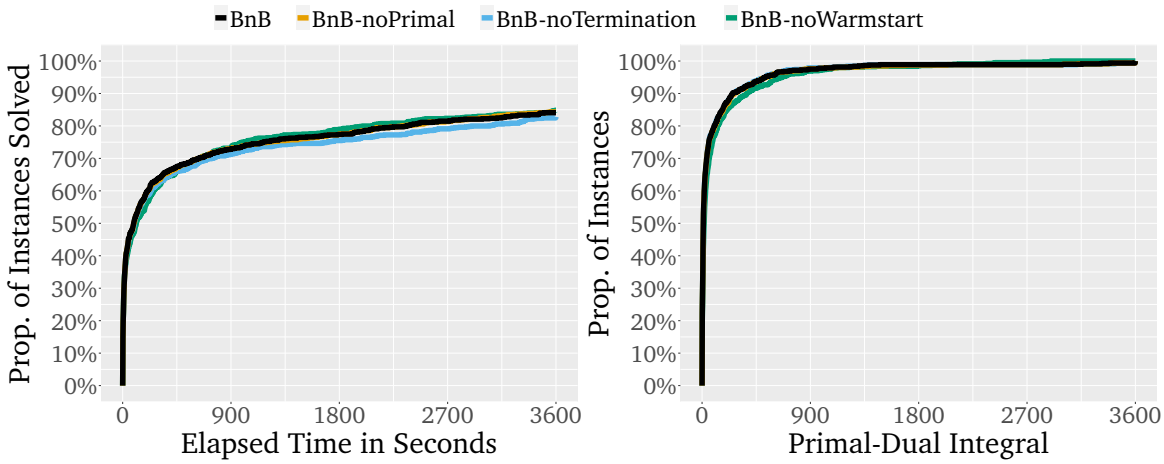


Figure 5.4. Cumulative distribution of computation times and primal-dual integrals for variants of BnB without improving primal bounds, terminating subproblems or using warm starts.

considered by BnB-noCutMIP is 17.2% higher than for BnB, indicating that pruning fewer nodes via estimators is no practical issue. However, despite considering fewer mixed-integer subproblems, BnB spends 17.6% more computation time on these.

When not using optimality-cuts at all, we obtain an even better performance. BnB-noCut has 35.2% fewer timeouts, 21.1% lower computation times, and 24.0% lower primal-dual integrals compared to BnB. The overall negative impact of optimality-cuts indicates that the interference with our warm starting strategy might not be the only reason why BnB-noCutLP outperforms BnB. We will cover this topic again after evaluating the impact of warm starting itself.

Figure 5.4 and Table 5.2 show that improving primal bounds by computing optimal values z for incumbent solutions x has nearly no effect on the performance of our branch and bound approach. This confirms that our branching and node selection strategies already guide us towards subproblems in which we find good solutions early. In fact, the primal bound computed by BnB-noPrimal while solving the first mixed-integer subproblem is still for 93.4% of all instances at most 1% higher than the best primal bound computed by any variant of our branch and bound algorithm.

The premature termination of mixed-integer subproblems has a positive impact on the computation time and the number of instances solved. BnB-noTermination has 7.8% more timeouts and 6.5% higher computation times. Note that the primal-dual integral is not changed significantly, since we tend to terminate subproblems prematurely when the primal-dual gap is already relatively low.

As already discussed, warm starting leads to notoriously bad results for some instances. BnB-noWarmstart solves 9 out of 12 rail01 instances, just like BnB-noCutLP. Accordingly, BnB-noWarmstart has 4.7% fewer timeouts than BnB. However, the computation times increase by 6.4% and the primal-dual integrals even by 28.6%. Thus, the overall effect of warm starting

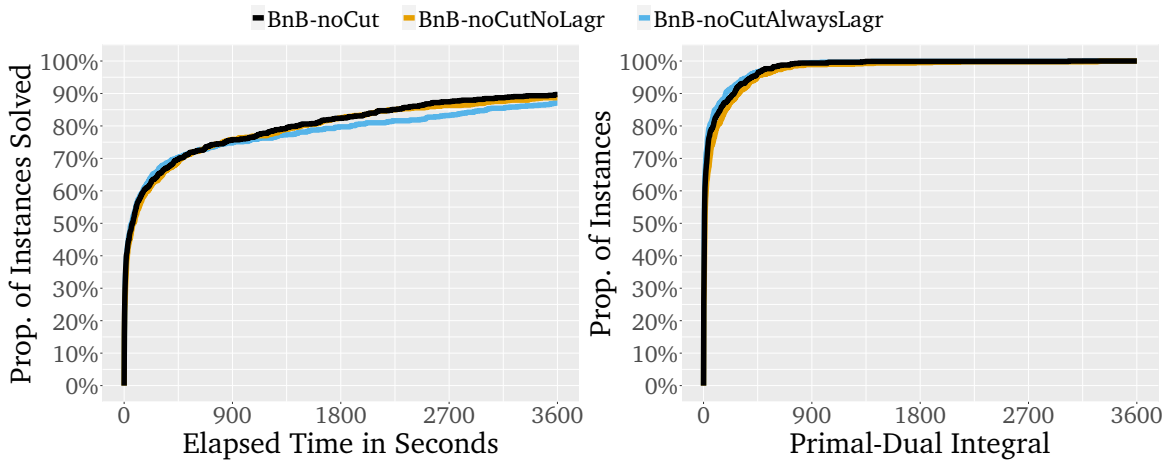


Figure 5.5. Cumulative distribution of computation times and primal-dual integrals for variants of BnB without optimality-cuts differing in the use of Lagrangean relaxations.

is positive, especially at the very beginning of the computation time, where the impact on the primal-dual integral is large due to the high primal-dual gap.

BnB-noCut is clearly the best performing variant of our branch and bound approach so far. However, the effect of using Lagrangean relaxations was inconclusive, and we therefore compare the three variants of using Lagrangean relaxations without optimality-cuts. We also evaluate how optimality-cuts for relaxed problems perform when warm starting is disabled. Finally, we test a variant that will give an interesting outlook on the capabilities of our branch and bound approach for robust optimization with uncertain constraints. To this end, we consider the following variants.

BnB-noCutNoLagr	never uses optimality-cuts or Lagrangean relaxations.
BnB-noCutAlwaysLagr	never uses optimality-cuts but always Lagrangean relaxations.
BnB-noCutLPnoWarm	does not use optimality-cuts when solving continuous relaxations and does not use warm start.
BnBCons-noCut	does not use estimators, does not improve primal bounds, and never uses optimality-cuts.

Table 5.4 and Figure 5.5 show that BnB-noCut is still the best variant regarding the number of instances solved and computation time. Not using Lagrangean relaxations yields 9.6% more timeouts, 11.4% higher computation times, and 41.9% higher primal-dual integrals. Always using Lagrangean relaxations still yields a speed-up in the beginning, resulting in 12.1% lower primal-dual integrals. However, BnB-noCutAlwaysLagr also yields 25.3% more timeouts and 4.4% higher computation times. We therefore stick with BnB-noCut as our variant of choice for the remainder of this study.

By comparing BnB-noWarmstart with BnB-noCutLPnoWarm in Table 5.4 and Figure 5.6, we see that optimality-cuts are also hindering when warm starting is disabled. We conclude that

Table 5.4. Computational results for different variants of BnB. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	BnB-noCut	BnB-noCutNoLagr	BnB-noCutAlwaysLagr	BnB-noWarmstart	BnB-noCutLPnoWarm	BnBCons-noCut
timeout	83	91	104	122	103	146
time	67.79	75.55	70.79	91.40	83.31	101.24
P-D integral	12.02	17.06	10.57	20.34	17.99	19.49

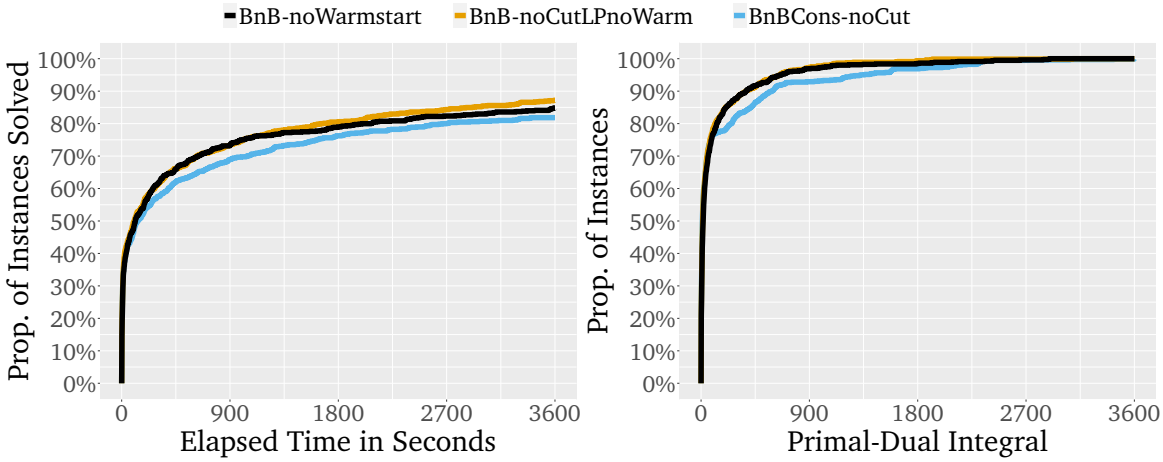


Figure 5.6. Cumulative distribution of computation times and primal-dual integrals for BnB with and without optimality-cuts when warm starting is disabled as well as a variant only using components that can be applied for uncertain constraints.

optimality-cuts are overall not beneficial in the current version of our branch and bound algorithm. However, we will see later that they can enhance other algorithms, which indicates that there is potential in using optimality-cuts.

The additional variant BnBCons-noCut is interesting, since it only relies on results that are also generalizable to uncertain constraints with budgeted uncertainty. In this setting, the j-th row $\sum_{i\in[n]} a_{ji}x_i \leq b_j$ of the constraint matrix $Ax \leq b$ becomes $\sum_{i\in[n]} (a_{ji}x_i + p_{ji}) + \Gamma_j z_j \leq b_j$ with additional robustness constraints $z_j + p_{ji} \geq \hat{a}_{ji}x_i$ for deviations $\hat{a}_{ji}$ and a constraint specific uncertainty budget Γ_j [23]. Since the additional constraints have the same structure as for the uncertain objective function, we can branch on the variables z_j, use clique reformulations, filter possible values for z_j, etc. Only estimators and the improvement of primal bounds cannot be generalized, since these rely on the fact that a feasible solution for a fixed z has a corresponding feasible solution for a different z'. This does not apply when fixing z_j to different values. When disabling the non-generalizable components together with optimality-cuts, we obtain a variant that still solves 81.8% of all instances, and thus outperforms all approaches from literature, as we will see in the next section. The performance of this variant suggests that our approach and the theoretical results in this thesis are also relevant for robust optimization with uncertain constraints.

5.7.2 Comparing Algorithms from the Literature

We now evaluate our branch and bound approach by comparing BnB-noCut with the following algorithms from the literature.

DEF	solves the MILP over the default formulation $\mathcal{F}^{\text{ROB}}$.
SCE	starts with NOM and then separates constraints corresponding to the objective coefficients c' of scenarios from the budgeted uncertainty set, as described by Bertsimas et al. [20].
BS	solves nominal subproblems NOS (z) for all $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$, as proposed by Bertsimas and Sim [22].
RECsepCons	separates violated recycled constraints, as in Section 4.4.1.
SUB	solves ROB and separates submodular-cuts dominating the above scenario cuts of SCE, as proposed by Joung and Park [57].
DnC	solves nominal subproblems NOS (z) but uses Lemma 29 to avoid some $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$, as proposed by Hansknecht et al. [53].
RP1,...,**RP4**	solve the corresponding reformulations of Atamtürk [9].

The approaches DEF, SCE, and BS are widely known and studied, and can thus be considered as the current state-of-the-art approaches. In contrast, SUB has so far been considered for knapsack problems [57] and DnC for robust shortest path problems [53], but none was evaluated for general robust optimization problems. To the best of our knowledge, we also present the first study that evaluates the reformulations RP1,...,RP4 on a broad set of instances. Before we analyze computational results of the above approaches, we give a brief description of SUB, DnC, RP2, and RP3, as we have to adapt them for our purposes. Furthermore, knowledge about SUB and DnC will be relevant for later discussion.

Description of SUB For the cutting plane approach SUB of Joung and Park [57], we consider the inner maximization problem

$$\rho(x) = \max_{\substack{S \cup \{t\} \subseteq [n]: \\ |S| \leq \lfloor \Gamma \rfloor, t \notin S}} \left((\Gamma - \lfloor \Gamma \rfloor) \hat{c}_t x_t + \sum_{i \in S} \hat{c}_i x_i \right)$$

of NLR as a submodular function. For $T \subseteq [n]$, let $f(T) = \rho(x)$ with $x_i = 1$ if $i \in T$ and $x_i = 0$ otherwise. Then f is a submodular set-function and defines a so-called *polymatroid*

$$\Pi_f = \left\{ \pi \in \mathbb{R}^n \middle| \sum_{i \in T} \pi_i \leq f(T) \ \forall T \subseteq [n] \right\}.$$

We have $\rho(x) \geq \pi^\top x$ for all $x \in \{0,1\}^n$ and $\pi \in \Pi_f$ by definition of Π_f. Remember from Section 3.1 that $\rho(x) = \Gamma z + \sum_{i \in [n]} p_i$ holds for optimal p, z with $(x, p, z) \in \mathcal{F}^{\text{ROB}}$. Thus, the

inequality $\Gamma z + \sum_{i \in [n]} p_i \geq \pi^\top x$, which we call a *submodular-cut*, is a valid inequality for $\mathcal{C}^{\text{ROB}}$ for all $\pi \in \Pi_f$. In our implementation of SUB, we start with solving ROB and separate submodular-cuts with maximum violation in the root node of the branching tree. We do this by solving $\max \left\{ \tilde{x}^\top \pi \middle| \pi \in \Pi_f \right\}$ for some fractional $\tilde{x} \in \mathcal{F}^{\text{NOM}}$. Joung and Park [57] show that this separation can be done efficiently based on a result of Edmonds [40] stating that optimization problems over a polymatroid can be solved using a greedy algorithm.

Description of DnC The divide and conquer approach of Hansknecht et al. [53] is similar to BS by Bertsimas and Sim [22] in the sense that it solves nominal subproblems NOS (z) for specific values $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$. However, non-optimal values of z are discarded by using Lemma 29 in order to reduce the number of subproblems to be solved.

Let $\{z_1, \dots, z_k\} = \{\hat{c}_0, \dots, \hat{c}_n\}$ be the set of distinct deviations. We start by solving NOS (z_1) and NOS (z_k). Afterwards, we try to prune values $z \in \{z_2, \dots, z_{k-1}\}$ by comparing the primal bound $\overline{v} = \min \{v(\text{NOS}(z_1)), v(\text{NOS}(z_k))\}$ with the dual bound $\underline{v}(z) = (z_k - z)\Gamma + v(\text{NOS}(z_k))$ obtained from Lemma 29. Of all remaining values $Z \subseteq \{z_2, \dots, z_{k-1}\}$, we select the median $z' \in Z$ and solve the nominal subproblem NOS (z'). Afterwards, we potentially update the primal bound with $v(\text{NOS}(z'))$ and split Z into $Z_1 = \{z \in Z | z < z'\}$ and $Z_2 = \{z \in Z | z > z'\}$. We then proceed recursively with Z_1 and Z_2 until all z are either pruned or considered for a nominal subproblem. Note that the dual bounds of Lemma 29 provide us with a global dual bound on v (ROB), which we can use to compute the primal-dual integral for DnC.

Given some yet to be considered sets $Z_1, \dots, Z_\ell \subseteq \{z_1, \dots, z_k\}$, DnC chooses the next set to be processed such that good solutions are found quickly. For each Z_j, there are surrounding values z^* that have already been considered for a nominal subproblem NOS (z^*). We use their corresponding objective values $v(\text{NOS}(z^*))$ as an indicator for the objective values corresponding to Z_j. Let $\underline{z}_j^*$ be the largest considered value that is smaller than $\min(Z_j)$ and $\overline{z}_j^*$ be the smallest considered value that is greater than $\max(Z_j)$. Then Z_j is chosen such that $\min \left\{ v\left(\text{NOS}\left(\underline{z}_j^*\right)\right), v\left(\text{NOS}\left(\overline{z}_j^*\right)\right) \right\}$ is minimal.

Description of RP2 and RP3 We slightly enhance RP2 and RP3 in our implementation compared to Atamtürk's description [9]. Reformulation RP2 consists of an exponential number of valid inequalities

$$\sum_{j \in [k]} p_{i_j} + z \geq \sum_{j \in [k]} \left(\hat{c}_{i_j} - \hat{c}_{i_{j-1}} \right) x_{i_j} \tag{5.11}$$

for each ordered set $\{i_1, \dots, i_k\} \subseteq [n]$ with $0 = \hat{c}_{i_0} \leq \hat{c}_{i_1} \leq \dots \leq \hat{c}_{i_k}$. These inequalities are not directly added to the formulation but can be separated in $\mathcal{O}(n^2)$ by searching for paths with negative weight in an acyclic directed graph. Assuming that $\hat{c}_1 \leq \dots \leq \hat{c}_n$ holds, this graph consists of nodes $V = [n+1]_0$ and arcs $A = \{(i,j) | i < j \in [n+1]_0\}$. The arc weights are chosen such that the weight of each path from 0 to $n+1$ equals the violation of the

Table 5.6. Computational results for different approaches from the literature. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	BnB-noCut	DEF	SCE	BS	RECsepCons	SUB	DnC
timeout	83	373	547	480	309	352	302
time	67.79	251.99	710.85	901.20	193.71	233.60	414.42
P-D integral	12.02	41.39	131.67	901.20	31.97	38.29	94.14

	RP1	RP2	RP3	RP4
timeout/memory error	652	382	652	680
time	1758.92	290.66	1716.20	2607.05
P-D integral	1089.27	49.35	1348.13	2253.52
memory error	406	0	492	480

corresponding inequality (5.11), where $\{i_1, \dots, i_k\} \subseteq [n]$ are the nodes visited on the path. Atamtürk [9] shows that the inequalities (5.11) with $\hat{c}_{i_0} < \hat{c}_{i_1} < \cdots < \hat{c}_{i_k}$ describe the convex hull of

$$\left\{(x, p, z) \in \{0,1\}^n \times \mathbb{R}^{n+1}_{\geq 0} \middle| p_i + z \geq \hat{c}_i x_i \forall i \in [n]\right\},$$

while inequalities with $\hat{c}_{i_{j-1}} = \hat{c}_{i_j}$ for some $j \in [k]$ are dominated. Therefore, we modify the above graph by deleting all arcs (i, j) with $\hat{c}_i = \hat{c}_j$, so that all paths in the modified graph correspond to non-dominated inequalities.

Reformulation RP3 emerges from RP2 by formulating the problem of finding the shortest path in the above graph as an LP. If the objective value of this LP is non-negative, then all inequalities (5.11) are satisfied. We can incorporate this requirement into ROB by bounding the LP value with zero. We obtain a linear problem by dualizing the LP and adding the resulting $n+2$ additional variables and $\mathcal{O}(n^2)$ constraints into our model. Our implementation omits some of these constraints by defining the LP on the reduced graph constructed for the separation problem of RP2.

Evaluation of Computational Results We now asses the performance of the approaches above experimentally. Just like in the previous section, we try to solve our 804 robust instances within a time limit of 3,600 seconds. As before, Table 5.6 shows for all approaches the number of timeouts and shifted geometric means of computation times and primal-dual integrals. Detailed results per instance and algorithm are provided in [47]. We want to mention that Gurobi sometimes terminates long after 3,600 seconds when it gets stuck in a subroutine. This is especially the case for the large formulation RP4. For the sake of a fair comparison, we always take the minimum of 3,600 and the measured computation time when computing the shifted geometric mean. The same applies for the primal-dual integral, which we also cap at 3,600.

Note that the implementation of DEF evaluated here is not the same as in the computational study from the previous chapter for technical reasons. The implementation from the previous chapter is based on that of the recycling of inequalities. It therefore scales deviations $\hat{c}$ and objective coefficients of p and z as described in Section 4.6.1. Interestingly, the scaled

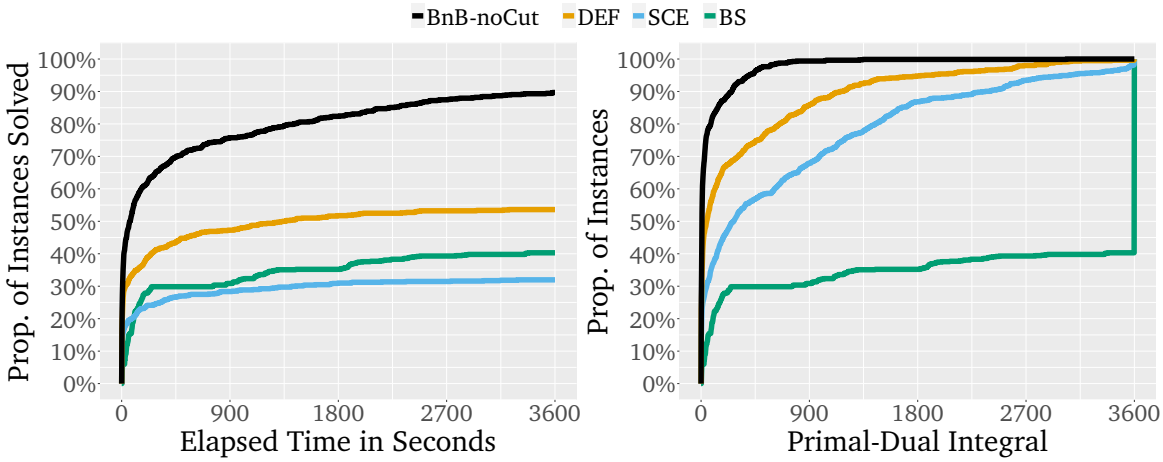


Figure 5.7. Cumulative distribution of computation times and primal-dual integrals for BnB-noCut compared to DEF, SCE, and BS.

implementation performs better with 6.7% fewer timeouts, 10.1% lower computation times, and 14.6% lower primal-dual integrals (cf. Table 4.4). This raises the question of whether scaling also improves other approaches. However, scaling is currently only used for recycling in the current implementation and we therefore leave this question for future research.

Figure 5.7 shows the performance of DEF, SCE, and BS in comparison with BnB-noCut. We see that SCE is clearly inferior to DEF in every performance metric. Interestingly, this is in contrast to the findings of Bertsimas et al. [20], who observed no clear winner between DEF and SCE for robust problems with uncertain constraints. BS is even slower than SCE in the shifted geometric mean but solves more instances, as can be seen in Table 5.6. This is because BS makes "slow but steady" progress and eventually solves all nominal subproblems, while SCE gets stuck for harder instances. However, BS still solves fewer instances than DEF within one hour, which supports our claim from Chapter 3 that solving all nominal subproblems NOS (z) for $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$ is not practical. Instead, BS should rather be seen as a heuristic in which we solve nominal subproblems for specific $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$ in order to quickly compute any solution. Consistent with this heuristic perspective, BS computes no meaningful dual bound until all nominal subproblems are solved. Therefore, the primal-dual integral always equals the computation time.

While DEF is best among the state-of-the art approaches, BnB-noCut outperforms it by far. Table 5.6 shows that BnB-noCut has 77.7% fewer timeouts, 73.1% lower computation times and 71.0% lower primal-dual integrals. Looking at the trend in the left graphic of Figure 5.7, it is natural to assume that the difference between BnB-noCut and DEF would be even larger if the time limit was higher than 3,600 seconds. DEF solves 416 instances within 1,800 seconds and 431 instances within 3,600 seconds. That is, DEF only solves 15 instances within the last 1,800 seconds, which equals 3.9% of the 388 instances that were not solved after 1,800 seconds. In contrast, BnB-noCut solves 662 instances within 1,800 seconds and 721 instances within 3,600 seconds. Thus, BnB-noCut solves within the last 1,800 seconds 41.5% of the 142 instances that were not solved in the first 1,800 seconds. We therefore conclude

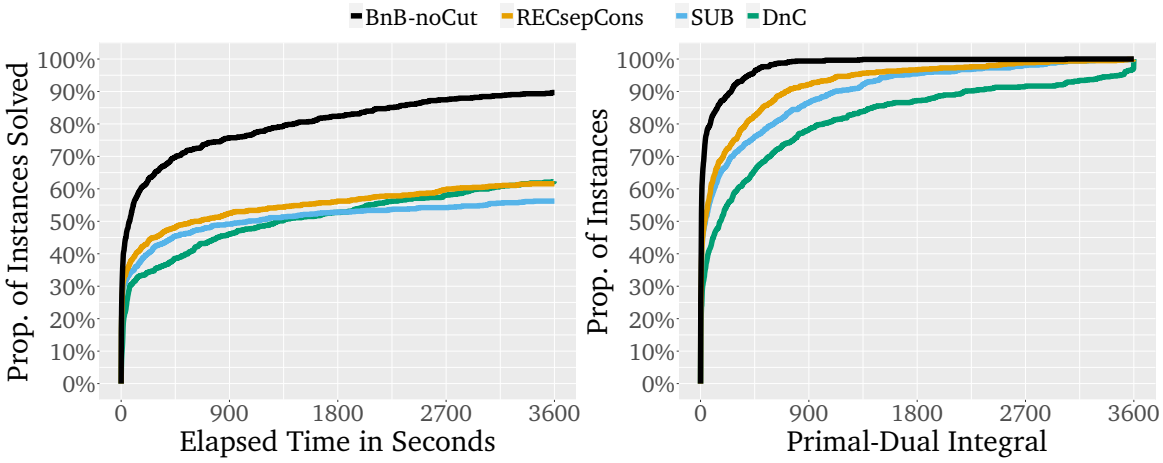


Figure 5.8. Cumulative distribution of computation times and primal-dual integrals for BnB-noCut compared to RECsepCons, SUB, and DnC.

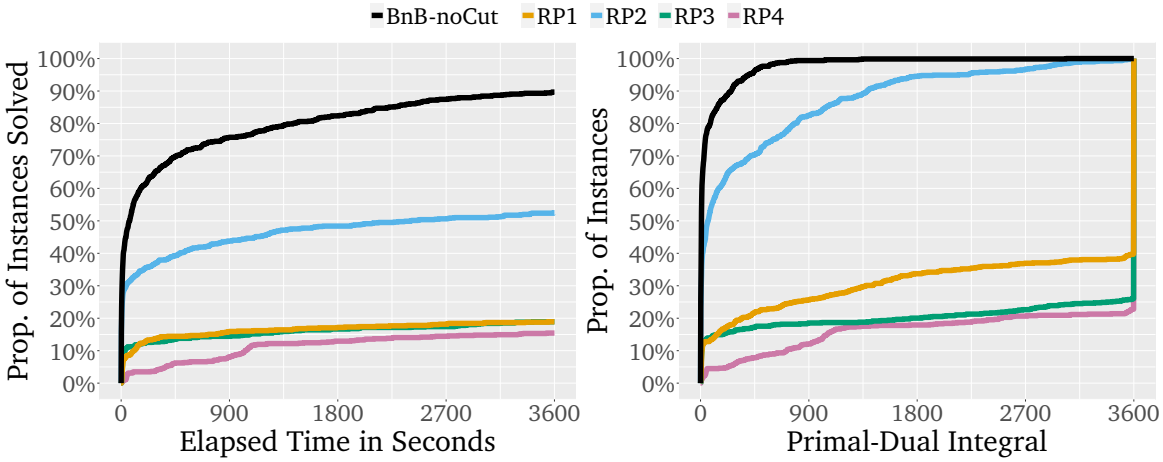


Figure 5.9. Cumulative distribution of computation times and primal-dual integrals for BnB-noCut compared to Atamtürk's reformulations.

that our branch and bound approach makes "fast and steady" progress compared to the other approaches.

Figure 5.8 shows RECsepCons, SUB, and DnC in comparison with our branch and bound. Although these are more sophisticated than the three approaches above, none comes close to BnB-noCut. Our recycling from last chapter proves to be faster than SUB and DnC. Table 5.6 shows that it even has the lowest shifted geometric mean in computation time and primal-dual integral over all approaches except for BnB-noCut. DnC has relatively high computation times and primal-dual integrals but the fewest timeouts after BnB-noCut. Just like the closely related BS approach, DnC makes steady progress in solving nominal subproblems, while other approaches get stuck for harder instances. Note that DnC strictly dominates BS by design, which results in 37.1% fewer timeouts and 54.0% lower computation times. While SUB is the inferior cutting plane approach compared to RECsepCons, it at least provides an improvement over DEF with 5.6% fewer timeouts, 7.3% lower computation times, and 7.5% lower primal-dual integrals.

Figure 5.9 shows that the cutting plane approach RP2 is the only practicable of Atamtürk's four reformulations. RP1, RP3, and RP4 are simply too large for most practical problems. Table 5.6 shows that RP1 exceeds the memory limit for 50.5% of all instances, RP3 for 61.2%, and RP4 for 59.7%. In these cases, we set the computation time and primal-dual integral to the maximum of 3,600. Even if the models can be built obeying the memory limit, they are often still too large for Gurobi to solve them. This results in RP1, RP3, and RP4 having by far the worst performance metrics across all approaches. RP2 does not cause memory issues, but it still provides no practical improvement over DEF, with 2.4% more timeouts, 15.3% higher computation times, and 19.2% higher primal-dual integrals.

5.7.3 Improving Algorithms from the Literature

Our experiments in the last section clearly show that our branch and bound approach outperforms all state-of-the art approaches from the literature. However, we do not only want to establish a new state-of-the art approach in this thesis but also show that our theoretical results can be used to enhance algorithms from the literature.

Note that all approaches tested in the last section can be improved in some way. For BS, we can reduce the number of nominal subproblems to be solved from $|\{\hat{c}_0, \dots, \hat{c}_n\}|$ to $|\mathcal{Z}|$ for some filtered set $\mathcal{Z} \subseteq \{\hat{c}_0, \dots, \hat{c}_n\}$ of possible optimal values for z. All approaches solving a reformulation of ROB can instead work on the equivalent but stronger problem ROB $(\mathcal{Z})$, in which the deviations are capped at $\max(\mathcal{Z})$ by moving $(\hat{c}_i - \max(\mathcal{Z}))^+$ directly into the objective function. However, we will not evaluate these improvements for all approaches. We do not consider an improved version of BS, as DnC is strictly stronger. Likewise, we will also not consider SCE, RP2, and RP3, as they were outperformed by DEF. We will, however, evaluate an improved version of RP1 and RP4, as their size can be reduced easily using filtering. This leaves us with the following approaches.

DEF+	solves the MILP ROB $(\mathcal{Z}, \mathcal{Q})$ using filtering and clique partitions.
SUB+	separates submodular-cuts for ROB $(\mathcal{Z}, \mathcal{Q})$.
RECsepCons+	uses recycled inequalities for ROB $(\mathcal{Z})$.
DnC+	uses multiple components of our branch and bound approach, as described below.
RP1+, RP4+	reduces the size of the formulation based on filtered $\mathcal{Z}$.

The improvements applied to DEF, SUB, and RECsepCons are straightforward, since we simply use stronger formulations. In the case of DEF and SUB, these formulations are also smaller, as we aggregate the variables p and robustness constraints $p_i + z \geq \hat{c}_i x_i$ using cliques. Remember that we cannot trivially use recycling for the clique reformulation ROB $(\mathcal{Z}, \mathcal{Q})$, as the aggregation of variables p is in conflict with the recycling of valid inequalities.

Table 5.7. Computational results for improved approaches from the literature. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	BnB-noCut	DEF	DEF+	SUB	SUB+	RECsepCons	RECsepCons+
timeout	83	373	328	352	332	309	310
time	67.79	251.99	191.80	233.60	188.31	193.71	189.74
P-D integral	12.02	41.39	30.95	38.29	29.83	31.97	31.43

	DnC	DNC+	RP1	RP1+	RP4	RP4+
timeout/memory error	302	118	652	626	680	660
time	414.42	74.30	1758.92	1367.54	2607.05	2101.48
P-D integral	94.14	17.11	1089.27	715.77	2253.52	1701.29
memory error	-	-	406	313	480	439

The basic framework of DnC described in the last section remains unchanged. That is, DnC+ also divides the set of possible values for z into subsets Z, prunes non-optimal choices with respect to individual dual bounds $\underline{v}(z)$, and chooses the median z' of the remaining values within Z for solving the nominal subproblem NOS (z'). However, instead of considering all values $z \in \{\hat{c}_0, \dots, \hat{c}_n\}$, we restrict ourselves to the filtered set $\mathcal{Z}$. Instead of using Lemma 29 for computing dual bounds $\underline{v}(z)$, we use the stronger estimators from Theorem 30. DnC+ also uses optimality-cuts from Section 5.6.1.2 with $\underline{c} = \min(Z)$ and $\overline{c} = \max(Z)$ when considering the subset $Z \subseteq \mathcal{Z}$. Note that this is in line with the reasoning for the careful choice of $\underline{c}, \overline{c}$ at the end of Section 5.6.1.2, as we already considered the closest values $\underline{z}^*, \overline{z}^* \in \mathcal{Z}$ around Z for nominal subproblems. Lastly, we improve primal bounds from incumbent solutions as in Section 5.6.2 and apply the premature termination to nominal subproblems as in Section 5.6.3.

The size of RP1 and RP4 can be reduced significantly. Both introduce a variable $\omega_{ik} \in \{0, 1\}$ for each nominal variable x_i with $i \in [n]$ and deviation $\hat{c}_k$ with $k \in [n]_0$. Here, choosing $\omega_{ik} = 1$ corresponds to choosing $x_i = 1$ and $z = \hat{c}_k$. It is easy to see that it is sufficient to use $\omega \in \{0, 1\}^{n \times |\mathcal{Z}|}$ instead of $\omega \in \{0, 1\}^{n \times |\{\hat{c}_0, \dots, \hat{c}_n\}|}$. Reformulation RP4 can be reduced even more, since it not only uses $|\{\hat{c}_0, \dots, \hat{c}_n\}|$ copies of x in its original form but also as many copies of the constraint matrix $Ax \leq b$. Therefore, we can omit $m\,(|\{\hat{c}_0, \dots, \hat{c}_n\}| - |\mathcal{Z}|)$ constraints for RP4.

Figure 5.10 shows that DEF+ and SUB+ cannot compete with BnB-noCut but perform better than their original version. DEF+ has 12.1% fewer timeouts, 23.9% lower computation times, and 25.2% lower primal-dual integrals compared to DEF, as shown in Table 5.7. The difference between SUB+ and SUB is slightly smaller, but we still observe 5.7% fewer timeouts, 19.4% lower computation times, and 22.1% lower primal-dual integrals. Interestingly, there is no big difference between DEF+ and SUB+. This is because the submodular-cuts are less interesting when using the already strengthened formulation ROB $(\mathcal{Z}, \mathcal{Q})$.

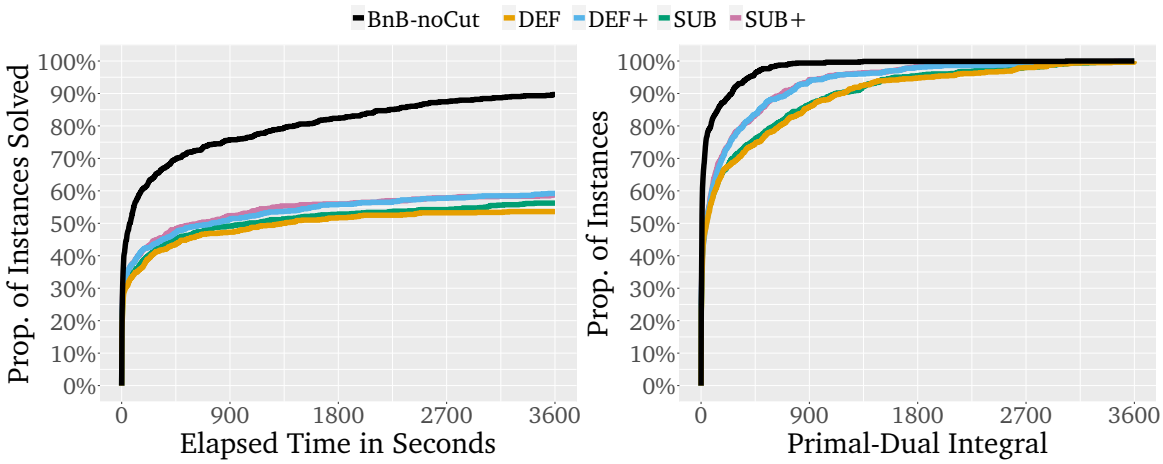


Figure 5.10. Cumulative distribution of computation times and primal-dual integrals for improved versions DEF+ and SUB+ compared to their original versions and BnB-noCut.

The question arises whether submodular-cuts can also be strengthened using conflicts between nominal variables. Remember from the previous section that the submodular-cuts $\Gamma z + \sum_{i\in[n]} p_i \geq \pi^\top x$ are defined via the polymatroid

$$\Pi_f = \left\{ \pi \in \mathbb{R}^n \middle| \sum_{i\in T} \pi_i \leq f(T) \;\; \forall T \subseteq [n] \right\},$$

where f corresponds to the inner maximization problem

$$\rho(x) = \max_{\substack{S\cup\{t\}\subseteq[n]:\\ |S|\leq\lfloor\Gamma\rfloor, t\notin S}} \left((\Gamma - \lfloor\Gamma\rfloor)\,\hat{c}_t x_t + \sum_{i\in S} \hat{c}_i x_i \right)$$

of NLR. This ensures that $\rho(x) \geq \pi^\top x$ holds for all $x \in \{0,1\}^n$ and $\pi \in \Pi_f$. Note that it is sufficient if $\sum_{i\in T} \pi_i \leq f(T)$ only holds for subsets $T \subseteq [n]$ corresponding to feasible solutions. That is, for $T \subseteq [n]$ whose characteristic vector x_T is in $\mathcal{C}^{\text{NOM}}$. Thus, we can enlarge the set of valid submodular-cuts to

$$\Pi'_f = \left\{ \pi \in \mathbb{R}^n \middle| \sum_{i\in T} \pi_i \leq f(T) \;\; \forall T \subseteq [n] : x_T \in \mathcal{C}^{\text{NOM}} \right\}.$$

If we were able to heuristically optimize over Π'_f, e.g., by using conflict graphs and clique partitions, then we would obtain stronger cuts.

Submodular-cuts are also interesting for our branch and bound approach. Remember from Section 5.4 that our Lagrangean relaxations of ROB correspond to convex combinations of scenarios in the budgeted uncertainty set. Instead of combining these scenarios, we might also apply Lagrangean relaxation to a dominating submodular-cut, and thus obtain better dual bounds. However, we currently base our Lagrangean relaxation on ROB $(\mathcal{Q})$ and the above comparison between DEF+ and SUB+ does not give hope that relaxing submodular-cuts yields better results. Combining submodular-cuts with cliques, however, might yield stronger

Table 5.8. Computational results for for BnB-noCut and DnC+ with instances classified with respect to their nominal complexity. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	all		nominal easy		nominal hard	
	BnB-noCut	DnC+	BnB-noCut	DnC+	BnB-noCut	DnC+
timeout	83	118	7	7	76	111
time	67.79	74.30	9.23	9.04	436.31	531.22
P-D integral	12.02	17.11	3.21	3.24	37.96	73.14

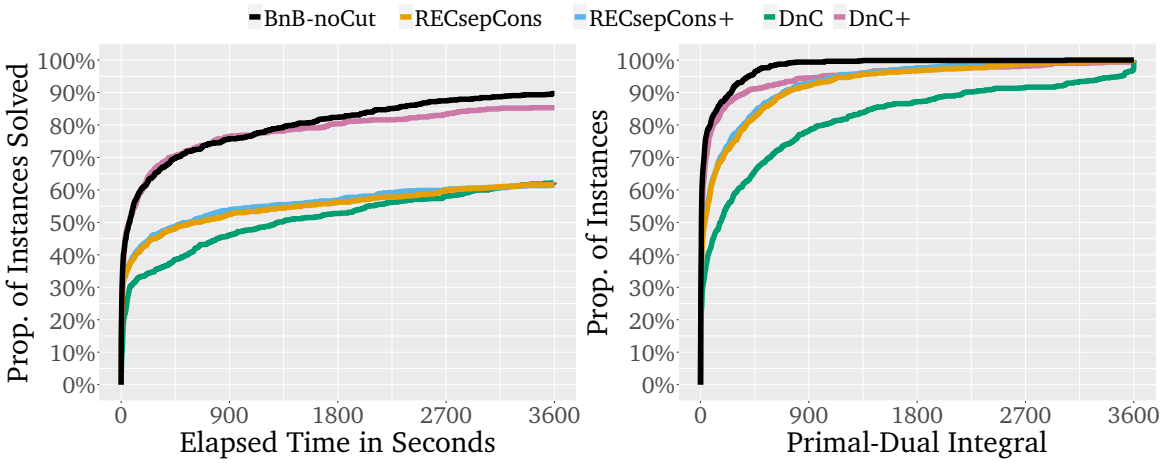


Figure 5.11. Cumulative distribution of computation times and primal-dual integrals for improved versions of RECsepCons and DnC compared to their original versions and BnB-noCut.

Lagrangean relaxations, and thus an improvement of our branch and bound. We leave the investigation of how to optimize over Π'_f and how to incorporate this into our branch and bound approach for future research.

Figure 5.11 shows that DnC+ surpasses DnC by far and is almost as strong as BnB-noCut. Table 5.7 reveals that the many improvements applied to DnC+ lead to 60.9% fewer timeouts, 82.1% lower computation times, and 81.8% lower primal-dual integrals compared to DnC. The distribution of computation time in the left of Figure 5.11 shows that DnC+ even solves more instances than BnB-noCut in the first 900 seconds, although the aggregated performance metrics in Table 5.7 are better for our branch and bound approach.

To get a better understanding of when DnC+ performs better, we split our test set into two halves. We classify the instances into *nominal easy* and *nominal hard* based on the computation time needed to solve the underlying nominal problem from the MIPLIB. All instances for which the nominal computation time is below the median are classified as nominal easy, while the rest is nominal hard. Table 5.8 shows that the performance of BnB-noCut and DnC+ is similar for the nominal easy instances, while BnB-noCut clearly performs better for hard instances with 31.5% fewer timeouts, 17.9% lower computation times, and 48.1% lower primal-dual integrals. This confirms that our strategy of solving relaxations to reduce the number of mixed-integer subproblems especially pays off when the

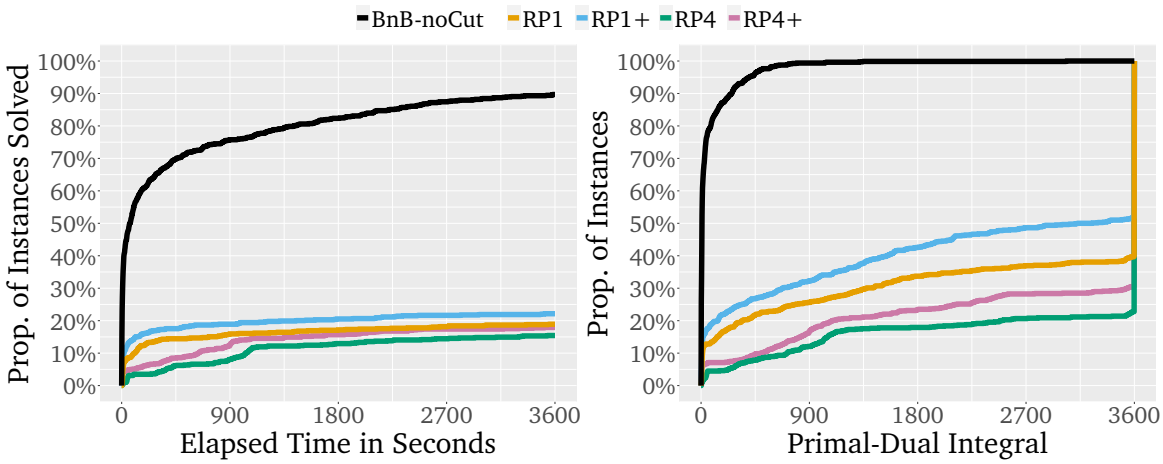


Figure 5.12. Cumulative distribution of computation times and primal-dual integrals for improved versions of RP1 and RP4 compared to their original versions and BnB-noCut.

nominal problem itself is hard to solve. However, DnC+ is a strong alternative when the nominal problem is easily solvable.

In contrast to the great improvement of DnC+, the performance of our recycling approach does not change significantly when applying it to ROB $(\mathcal{Z})$. It appears that the strengthening of the formulation implied by capping the deviations at $\max(\mathcal{Z})$ is negligible in this case. In contrast to DEF+ and SUB+, we are missing the strengthening via cliques and the reduced size of the formulation that we have for ROB $(\mathcal{Q})$. For future research, it would be interesting to investigate whether recycling and cliques can be combined in an efficient way.

We see in Figure 5.12 that the smaller formulations based on the filtering of $\mathcal{Z}$ lead to more instances solved by RP1+ and RP4+. Table 5.7 shows that RP1+ runs out of memory for 22.9% fewer instances. For RP4+, this is the case for 8.5%. Nevertheless, both approaches are still not suitable for instances of practical size.

We close this section with an evaluation of the improvements applied to DnC+. We do so by disabling the components individually, analogously to Section 5.7.1. This yields the following variants of DnC+.

DnC+noClique	does not compute the conflict graph and clique partition from Section 5.3.
DnC+noFilter	does not filter $\mathcal{Z}$ as in Section 5.5, i.e., $\mathcal{Z} = \{\hat{c}_0, \dots, \hat{c}_n\}$.
DnC+HRSestimator	uses the weaker estimators of Lemma 29 from Hansknecht et al. [53].
DnC+noCut	does not use optimality-cuts from Section 5.6.1.2.
DnC+noPrimal	does not improve primal bounds as in Section 5.6.2.
DnC+noTermination	does not terminate nominal subproblems prematurely as in Section 5.6.3.

Table 5.9. Computational results for different variants of DnC+. Computation times and primal-dual integrals are shifted geometric means with shifting parameter $s = 1$.

	DnC+	DnC+noClique	DnC+noFilter	DnC+HRSestimator	DnC+noCut	DnC+noPrimal	DnC+noTermination
timeout	118	120	120	155	163	130	154
time	74.30	78.44	80.32	119.42	129.77	86.92	112.08
P-D integral	17.11	18.46	19.08	25.94	24.26	24.30	24.27

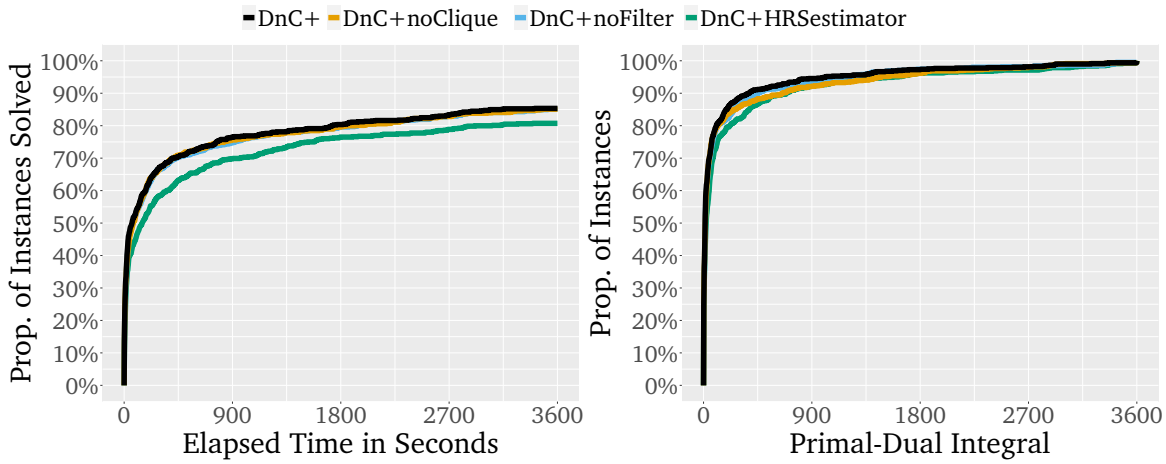


Figure 5.13. Cumulative distribution of computation times and primal-dual integrals for variants of DnC+ without using cliques, filtering or improved estimators.

Figure 5.13 and Table 5.9 show that disabling the filtering of $\mathcal{Z}$ and the computation of cliques lead to a slight degradation in performance. Just like for BnB, filtering is not the most important component of DnC+, as similar values of z are pruned by using estimators. Still, disabling filtering is worse for DnC+ (8.1% higher computation times) than it is for BnB (1.9% higher computation times). This is because BnB also considers similar values within the same robust subproblem ROB (Z). In contrast, disabling cliques is less problematic for DnC (5.6% higher computation times) than it is for BnB (24.3% higher computation times). This is because DnC+ only uses cliques for improving the filtering and the estimators of Theorem 30, but BnB also relies on the strengthened clique reformulation.

It is not surprising to see that the improved estimators from Theorem 30 enhance the divide and conquer substantially. Reverting to the weaker estimators leads to 31.4% more timeouts, 60.7% higher computation times, and 51.6% higher primal-dual integrals. These differences are even higher than the differences between the default variant of BnB and the variant where we don't use estimators at all. This is because DnC completely relies on using estimators for pruning possible values for z.

Figure 5.14 shows that the improvement of primal bounds is important for DnC+, especially with respect to the primal-dual integral. In contrast to BnB, which is selective in solving mixed-integer subproblems, DnC solves nominal subproblems NOS (z) for unpromising values of z in the beginning. Therefore, the improvement of solutions by computing optimal values for z yields better primal bounds early on, which is important for the primal-dual integral. We overall observe 10.2% more timeouts, 17.0% higher computation times, and 42.0% higher primal-dual integrals when comparing DnC+noPrimal with DnC+. For the same reasons,

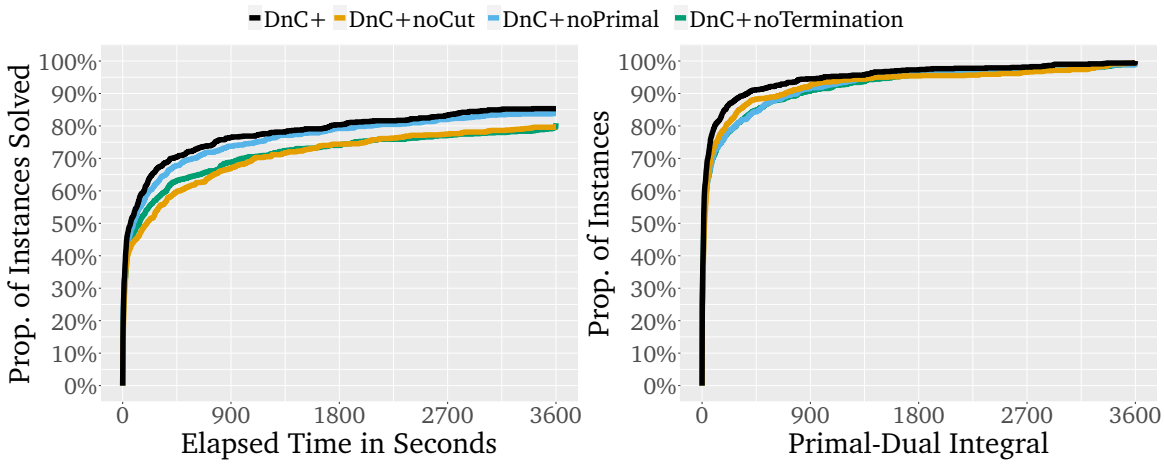


Figure 5.14. Cumulative distribution of computation times and primal-dual integrals for variants of DnC+ without using optimality-cuts, improving primal bounds or terminating subproblems.

terminating nominal subproblems also improves the performance substantially. Nominal subproblems for non-optimal values z can be terminated especially fast when using the improved bound. This leads to DnC+noTermination having 30.5% more timeouts, 50.8% higher computation times, and 41.8% higher primal-dual integrals compared to DnC+.

Surprisingly, disabling optimality-cuts has the largest negative impact on DnC+, although they were hindering for our branch and bound in the current implementation. Not using optimality-cuts leads to 38.1% more timeouts, 74.7% higher computation times, and 41.8% higher primal-dual integrals. This shows that there definitely is potential in using optimality-cuts and raises hope that they can also be a helpful addition to our branch and bound approach with further engineering.

5.8 Conclusion

The preceding computational study not only rounds out the chapter on the branch and bound algorithm but also the whole part of this thesis on robust optimization with budgeted uncertainty. Contrary to the famous result of Bertsimas and Sim [22], stating that the theoretical complexity of ROB is the same as that of NOM, we observed that solving ROB can be a major challenge in practice, even if NOM is easy to solve. We showed that the algorithm of Bertsimas and Sim leading to their theoretical result is too inefficient for solving problems of practical size to optimality. In fact, solving ROB directly as an MILP yielded lower computational times, although the standard formulation $\mathcal{F}^{\text{ROB}}$ has proven to be quite weak. We identified that the variable z within formulation $\mathcal{F}^{\text{ROB}}$ is critical in this regard and addressed this issue by proposing the strong bilinear formulation $\mathcal{F}^{\text{BIL}}$.

While the bilinearity is not desirable, the formulation turned out to be an invaluable foundation for our approaches for solving ROB. In Chapter 4, we derived and extensively analyzed

recycled inequalities, which are facet-defining in surprisingly many cases. In this chapter, we introduced strong linear relaxations of $\mathcal{F}^{\mathrm{BIL}}$ for the case where z is bounded. Building upon these, we proposed a branch and bound algorithm in which we obtain bounds on z via branching. We additionally characterized optimal solutions of ROB and proved further structural results which enabled us to improve our branch and bound approach by computing stronger primal and dual bounds.

We conducted two comprehensive computational studies to test the practicability of our approaches. The first study in Section 4.6 revealed that the easily applicable recycled inequalities substantially improve the standard formulation and often lead to a better running time. While our branch and bound algorithm is much more complex, the second computational study showed that its performance is unparalleled by any state-of-the-art approach. To the best of our knowledge, our study is also the first to compare many sophisticated algorithms from the literature on a broad set of test instanced. Moreover, most of the algorithms from the literature can be substantially improved by using the structural insights gained in this chapter. This is especially true for the divide and conquer approach by Hansknecht et al. [53], whose improved version is a strong alternative to the branch and bound when the underlying nominal problem is easy to solve.

We hope that the provided evidence of the computational price of robustness as well as our progress towards lowering this price motivate further research on making robust optimization applicable. The practical contribution of this thesis, namely the provision of many implemented algorithms [46] and generation of benchmark instances [48], should be helpful in this regard. The broad applicability of our results for enhancing existing approaches indicates that our theoretical contribution can also play an important role for future algorithms. For the branch and bound approach itself, there are countless possibilities to adjust and extend the different components to achieve further performance improvements. In particular, an effective use of optimality-cuts, who have proven to be strong for the divide and conquer approach, would be interesting. Furthermore, the combination of submodular-cuts with cliques in the conflict graph should be investigated for obtaining stronger Lagrangean relaxations.

Both of our approaches should be extensively tested for instances with uncertain constraints. While the recycling is directly applicable, the branch and bound algorithm needs to be adjusted for such problems. However, we already reasoned that most theoretical results can be generalized to this case and showed that the branch and bound performs well relying only on these general results.

Part II

Fair Planning of the Out-of-Hours Service for Pharmacies

6 Introduction

Establishing a functional healthcare system is one of the most crucial obligations of the German welfare state [94]. An essential component of such a system is the supply of pharmaceuticals, which are primarily provided by pharmacies. To ensure the supply at any time of the day, all pharmacies are obliged by law to be open 24/7 [7, § 23]. However, the German state governments mandate the chambers of pharmacists to daily exempt a part of the pharmacies from this obligation outside the regular opening hours. In doing so, the chambers of pharmacists must ensure that the supply of pharmaceuticals is guaranteed by the remaining pharmacies [7, § 23]. Pharmacies that are not exempted are called *out-of-hours pharmacies* and the continuous provision of pharmaceuticals is called the *out-of-hours service*.

The out-of-hours service is described by the Federal Union of German Associations of Pharmacists (ABDA) as "one of the most important obligations to the common good that pharmacies fulfill" [1]. Thus, on the one hand, pharmacists consider the out-of-hours service as an expression of their professional ethos. On the other hand, the obligatory service poses an economically unattractive burden. This is because a highly qualified pharmacist must be present during the whole 24-hour shift while the customer demand is lower outside the regular opening hours. Therefore, it is crucial to strike a balance between an appropriate supply of pharmaceuticals and an acceptable burden for pharmacies when planning the out-of-hours service.

In 2022, the 18,086 pharmacies across Germany worked approximately 430,000 *out-of-hours shifts*, equating to an average of nearly 24 shifts per pharmacy [1]. This average number is likely to increase in the future if there is no systematic change in the planning of the out-of-hours service, since the number of pharmacies in Germany has been in constant decline in recent years (-15.7% from 2015 to 2022 [1]). This development is especially problematic in rural areas, where an already low density of pharmacies results in a higher number of shifts for each pharmacy. For example, a pharmacy in the Bavarian village of Weitnau worked 302 shifts in 2022, and thus needed to be open year-round except for nine weeks of exemption.

Such an extreme difference compared to the average number of shifts is sometimes inevitable due to the local situation of rural pharmacies. However, some planning approaches that are currently in use also favor large variations in the number of shifts, even among nearby pharmacies: It is common practice for the German chambers of pharmacists to divide the planning area into districts, typically based on administrative borders, in which the out-of-hours service is organized locally as a rotation of the resident pharmacies. The districting considerably impacts the burden of individual pharmacies, since being part of a district with few pharmacies results in many shifts. This is especially unsatisfactory if neighboring

pharmacies are in different districts that differ significantly in terms of the implied burden. For example, a pharmacy in the Bavarian village of Zell worked 30 shifts in 2022, while a pharmacy just 3 km away in the city of Würzburg worked only 17 shifts. Justifying such differences on the basis of administrative boundaries alone is difficult for pharmacies with similar location and can lead to frustration among pharmacists who may feel treated unfairly due to their allocation to an unfavorable district.

In addition to a lack of fairness, the missing synchronization between districts can lead to neighboring pharmacies in different districts having an out-of-hours shift on the same day. This results in an oversupply for the corresponding area and unnecessary shifts. Conversely, we can also have an undersupply if the out-of-hours pharmacies of neighboring districts are located far apart from another. These issues clearly show that a centralized approach considering all pharmacies together would be preferable over the district-based planning.

Three out of 17 chambers of pharmacists in Germany already replaced the district-based planning with a centralized planning. The Chamber of Pharmacists Westphalia-Lippe started this transition in 2012, followed by North Rhine in 2014, and Schleswig-Holstein in 2015. The chambers of Hessen and Rhineland-Palatinate switch to a centralized planning in 2024. This trend shows that there is a growing political willingness to restructure the planning of the out-of-hours service, with the driving force being the high burden on the pharmacies [26, 54]. However, the relief for pharmacies should not come at the expense of an avoidable worsening of the coverage of residents with out-of-hours pharmacies. Stated otherwise, a deterioration in coverage cannot be justified by a reduction in the burden of pharmacies if there exists a better plan that is acceptable for all pharmacies and maintains the quality of coverage. This raises the necessity for an optimization-based planning, which guarantees that we obtain the best plan possible.

In this thesis, we consider the planning of the out-of-hours service in the area of our practice partner, the Chamber of Pharmacists North Rhine. Their transition to a centralized planning in 2014 already resulted in a reduction of the total number of shifts of more than 20% [8]. However, since their planning is not based on mathematical optimization, there is still potential regarding fairness, efficiency, and compliance with legal constraints. We will explore this potential in the following chapters by proposing mathematical models and algorithms for an optimized planning of the out-of-hours service. We place a special focus on the computation of fair plans that relieve the burden on pharmacies as far as possible and at the same time achieve a predefined quality of coverage.

6.1 Related Work

To the best of our knowledge, a centralized planning of the out-of-hours service that is not district-based has not yet been considered in the operations research literature. The problem of assigning shifts while retaining districts is known as the *Pharmacy Duty Scheduling*

Problem (PDS). The PDS has been studied for the planning of shifts in Turkey [35, 68, 63, 80], where the setting is similar to Germany in the sense that chambers of pharmacists historically organize the out-of-hours service decentrally in small local districts. The PDS consists in assigning exactly one shift to one pharmacy on each day in each district. The objective is to minimize the sum of the distances between aggregated customer nodes and their nearest out-of-hours pharmacy. The problem is described as a multi duty variation of the p-median problem and has been solved using variable neighborhood search [63], tabu search [80], branch and price [35], and swarm intelligence algorithms [68].

The PDS differs considerably from the problem proposed in this thesis. Most importantly, the number of shifts assigned to each pharmacy is almost fixed: All pharmacies within the same district are assigned an equal number of shifts except for differences of ± 1 if the number of days in the time horizon is not divisible by the number of pharmacies within the district. Thus, fairness in terms of an even distribution of shifts among pharmacies is achieved automatically within districts and neglected across districts. This is in contrast to our problem, where the number of shifts assigned to a pharmacy is not predefined and achieving global fairness is a major concern.

Another difference is that the PDS yields no guarantee regarding the maximum distance residents have to travel to their nearest out-of-hours pharmacy. Furthermore, there are no work regulations with respect to periods of rest between two shifts of the same pharmacy. Both are an integral part of the problem considered in this thesis and are defining for its complexity, with the computation of any feasible solution being already $\mathcal{NP}$-complete. In contrast, the PDS is $\mathcal{NP}$-complete but computing a solution is trivial [80]. Solving each problem therefore represents a structurally different task.

Typical problems requiring the assignment of shifts for providing coverage of demand, while complying with work regulations, are rostering problems. In the literature, rostering problems are, for example, extensively studied for hospital staff like nurses or physicians [28, 34]. The planning of the out-of-hours service sets itself apart from other rostering problems due to the inhomogeneity of pharmacies: A pharmacy's capability to cover an area depends on its location, which can differ considerably from pharmacy to pharmacy. In particular, usually only few pharmacies can cover rural areas, which results in many shifts being assigned to these pharmacies. This impacts the fairness of the plans: While typical rostering problems require the workload to be distributed as evenly as possible [41], the number of shifts assigned to pharmacies can vary significantly. Hence, we need to carefully model fairness for the planning of the out-of-hours service.

Modeling fairness in an optimization problem is generally not trivial. This is especially true if the most straightforward approach, namely pursuing an equal workload for all, is not suitable. Since there is no universal approach that fits all applications, many different fairness concepts have been developed and applied to optimization problems. Karsu and Morton [60] give an overview on how fairness concerns have been addressed in different optimization problems. Chen and Hooker [97] provide a guide on how to formulate different fairness

concepts in optimization models. A recurring subject here is the trade-off between fairness and efficiency. The loss in efficiency is, for example, analyzed by Bertsimas et al. [21] for resource allocation problems. In our case, a fair out-of-hours plan might assign many shifts more than the most efficient plan. We will revisit this topic again in Section 8.1, where we discuss fairness concepts that ensure a high level of fairness while maintaining efficiency for the planning of the out-of-hours service.

6.2 Contribution and Outline

In Chapter 7, we introduce a model for the planning of the out-of-hours service for pharmacies, which arises from a collaboration with the Chamber of Pharmacists North Rhine. We will show that it is $\mathcal{NP}$-hard to decide whether there exists a feasible out-of-hours plan, that is, it is not trivial to compute any feasible plan. We state an MILP formulation for the problem and propose MILP-based approaches for constructing out-of-hours plans in reasonable time. First, we introduce a reformulation via aggregation of mathematically equivalent pharmacies. Aggregation can yield more tractable models but is often an inexact simplification resulting in the loss of optimality after disaggregation. In our case, however, the aggregation is exact and preserves optimality. In addition to the aggregation, we propose a rolling horizon approach that applies a time-based decomposition of the planning problem into smaller subproblems. The rolling horizon approach is a heuristic that offers no approximation guarantee, as we fix shifts in subproblems without considering the whole time horizon. We partially compensate for this with a tailored extension in which we free a subset of the formerly fixed variables before each iteration. The freed variables are chosen such that the impact of suboptimal decisions made in earlier subproblems is reduced but the complexity of subsequent subproblems does not increase. We test our approaches extensively on a real-world instance with 2291 pharmacies provided by the Chamber of Pharmacists North Rhine. We will see that the aggregation and rolling horizon approaches enable us to compute nearly optimal out-of-hours plans in short time. Afterwards, we conclude the chapter with a discussion of the constructed plans, which assign roughly 10% fewer shifts than the real plan while simultaneously maintaining a higher compliance with planning regulations.

The approaches developed in Chapter 7 yield efficient out-of-hours plans but they provide no sufficient control regarding an even distribution of shifts among pharmacies, which results in unfair out-of-hours plans. We therefore focus in Chapter 8 on the computation of fair plans. For this, we consider the related concepts of lexicographic and min-max fairness, which can be seen as the strictest fairness concepts. As computing optimal fair plans directly is not practical, we first consider min-max fairness for a relaxation of our planning problem. We generalize and prove several statements from the literature on min-max fairness, which we then use to compute min-max fair solutions to the relaxed problem within seconds. Afterwards, we show how the min-max fair relaxed solution can be used as an orientation for computing fair out-of-hours plans. We show for our real-world instance that we can compute efficient

plans that almost match the min-max fair solutions, and are thus almost maximally fair. We furthermore show that the min-max fair solutions are invaluable within a decision support environment for analyzing and customizing the planning model. To demonstrate this, we apply small changes to the model, based on observations from the min-max fair solutions, that considerable improve the quality of the resulting out-of-hours plan.

The results from Chapter 7 have been published in Operations Research for Health Care [32] and were also partially described in [71]. A publication of the results in Chapter 8 is in preparation. All described results have been produced by myself together with my supervisors Christina Büsing and Arie M.C.A. Koster.

7 Planning the Out-of-Hours Service for Pharmacies

In this chapter, we introduce our model for the planning of the out-of-hours service for pharmacies. We afterwards develop algorithms for computing out-of-hours plans and conduct a case study in which we study the performance of our algorithms and the practicability of the computed plans.

7.1 Problem Definition and Notation

For the planning of the out-of-hours service, we consider a set of pharmacies $\mathcal{P}$ and a time horizon $[T]$, consisting of $T \in \mathbb{Z}_{>0}$ days to be planned. We identify an *out-of-hours plan* with a mapping $F : [T] \to 2^{\mathcal{P}}$, where $F(t) \subseteq \mathcal{P}$ is the set of out-of-hours pharmacies on day $t \in [T]$. Furthermore, we denote for a plan F with $S_F(p) = \{t \in [T] \mid p \in F(t)\}$ the set of days on which pharmacy $p \in \mathcal{P}$ is assigned a shift.

An out-of-hours plan provides an appropriate coverage of residents if it fulfills several criteria that are defined in the regulations of the responsible chamber of pharmacists. These regulations are established in dialogue with the state government, and thus provide the framework for the chamber of pharmacist to exercise their mandate of planning the out-of-hours service. In consultation with the Chamber of Pharmacists North Rhine, an out-of-hours plan has to fulfill the following requirements regarding the coverage of residents and also the satisfaction of pharmacists.

- *Covering municipality centers:* There is at least one out-of-hours pharmacy in the vicinity of each municipality center. The maximum permissible distance depends on the municipality.
- *Meeting demand in cities:* Multiple out-of-hours pharmacies are needed to meet the demand of residents in larger cities. The number of pharmacies needed is city-specific.
- *Conflict distances:* A minimum distance between out-of-hours pharmacies on the same day ensures that they are geographically dispersed.
- *Periods of rest:* Periods of rest prevent that pharmacies are assigned shifts on consecutive days.
- *Municipality-balancing:* An even distribution of shifts within municipalities increases acceptance of the plan among pharmacists.

- *Minimum number of shifts:* Each pharmacy participates with a minimum number of shifts.

Under the above requirements, which we define in the next paragraphs in more detail, we aim to construct a plan that is efficient in the sense that we assign as few shifts as possible. Accordingly, we search for a plan $F : [T] \to 2^{\mathcal{P}}$ that minimizes $\sum_{t\in[T]} |F(t)|$. We call this problem the *Out-of-Hours Planning Problem* (OHP).

Covering Municipality Centers In order to guarantee a comprehensive supply of pharmaceuticals, we need to ensure that residents do not have to travel too far to the next out-of-hours pharmacy. For this, we consider the centers of municipalities in the planning area as reference points for the location of residents. Based on the regulations of the Chamber of Pharmacists North Rhine, we require for every day and every municipality that there is an out-of-hours pharmacy within a given distance of the municipality center [86]. Let $\mathcal{M}$ be the set of municipalities to be covered. We define distances $\delta : (\mathcal{P} \cup \mathcal{M})^2 \to \mathbb{R}_{\geq 0}$ between all pairs of municipalities and pharmacies, representing the distances between these locations in the road network. For all municipalities $m \in \mathcal{M}$, we denote with $\delta^{\text{cov}}(m) \in \mathbb{R}_{\geq 0}$ the *cover radius,* that is the maximum permissible distance from the municipality to the nearest out-of-hours pharmacy. We say that $p \in \mathcal{P}$ can *cover* $m \in \mathcal{M}$ if $\delta(m,p) \leq \delta^{\text{cov}}(m)$ holds and define $C(m) = \{p \in \mathcal{P} \mid \delta(m,p) \leq \delta^{\text{cov}}(m)\}$ as the set of pharmacies that can cover m (highlighted in Figure 7.1). Likewise, we denote with $C(p) = \{m \in \mathcal{M} \mid \delta(m,p) \leq \delta^{\text{cov}}(m)\}$ the set of municipalities that can be covered by $p \in \mathcal{P}$. We say that a municipality is *covered* in a plan $F : [T] \to 2^{\mathcal{P}}$ on day $t \in [T]$ if at least one pharmacy $p \in C(m)$ is assigned a shift on day t, i.e., $C(m) \cap F(t) \neq \emptyset$. In a feasible out-of-hours plan F, every municipality has to be covered every day, i.e., $C(m) \cap F(t) \neq \emptyset$ for all $m \in \mathcal{M}$ and $t \in [T]$. Note that a pharmacy does not necessarily have to be within the boundaries of a municipality in order to cover it. Furthermore, the cover radius may vary for different municipalities according to the number of residents and the density of pharmacies in the area.

Meeting Demand in Cities For municipalities representing larger cities, a single out-of-hours pharmacy cannot guarantee a sufficient supply of pharmaceuticals. Therefore, we require a minimum number of out-of-hours pharmacies that are located within the boundaries of these municipalities. In contrast to the covering of municipality centers, the demand of larger cities cannot be met by pharmacies that are located outside the municipalities. This is because there are usually sufficiently many pharmacies in cities and residents are less willing to leave the city according to the Chamber of Pharmacists North Rhine. We denote with $M(p) \in \mathcal{M}$ the municipality in which a pharmacy $p \in \mathcal{P}$ is located. Accordingly, $P(m) = \{p \in \mathcal{P} | M(p) = m\}$ denotes the set of all pharmacies within the boundaries of a municipality $m \in \mathcal{M}$ (highlighted in Figure 7.1). We define for every municipality $m \in \mathcal{M}$ and day $t \in [T]$ a *minimum demand* for out-of-hours pharmacies $d(m,t) \in \mathbb{Z}_{\geq 0}$. This minimum demand is always zero for municipalities with few residents and pharmacies. In cities, it may vary depending on the considered day, since we tend to need more out-of-hours pharmacies on Sundays and holidays

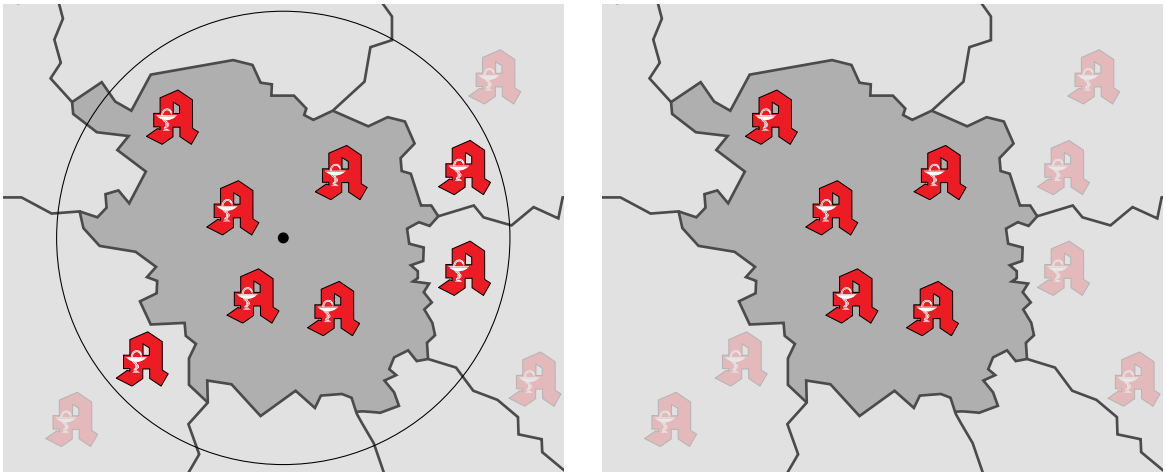


Figure 7.1. Different types of coverage. **Left:** pharmacies covering the center of a municipality. For simplicity, the cover radius is shown as the crow flies. **Right:** pharmacies fulfillung the demand of larger cities (right).

when other pharmacies are regularly closed all day. A feasible out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ assigns for all municipalities $m \in \mathcal{M}$ and all days $t \in [T]$ at least $d(m,t)$ shifts to pharmacies within the municipality, i.e., $|P(m) \cap F(t)| \geq d(m,t)$ for all $m \in \mathcal{M}$ and $t \in [T]$.

Conflict Distances Assigning shifts to geographically close pharmacies on the same day has limited benefit for residents and is economically unfavorable for pharmacists who must share customers. We already tend to avoid assigning shifts to neighboring pharmacies by minimizing the total number of shifts, as close pharmacies often cover the same municipalities. Nevertheless, it is possible that our coverage model is indifferent between choosing two neighboring pharmacies or two pharmacies that are farther apart. This is especially in larger cities, where we assign several shifts in order to meet the demand. To ensure that out-of-hours pharmacies are spread out, we prohibit shifts on the same day for pharmacies that are too close to each other (see Figure 7.2). To this end, we introduce for each pharmacy $p \in \mathcal{P}$ the *conflict distance* $\delta^{\text{con}}(p) \in \mathbb{R}_{\geq 0}$ and say that two pharmacies $p \neq p' \in \mathcal{P}$ are in *conflict* if $\delta(p,p') \leq \min\{\delta^{\text{con}}(p), \delta^{\text{con}}(p')\}$ holds. We denote the set of conflicting pharmacy pairs with $\mathcal{C} = \left\{\{p,p'\} \in \binom{\mathcal{P}}{2} \mid \delta(p,p') \leq \min\{\delta^{\text{con}}(p), \delta^{\text{con}}(p')\}\right\}$. In a feasible out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$, we cannot assign a shift to two conflicting pharmacies on the same day, that is, we require $\{p,p'\} \not\subseteq F(t)$ for all conflicting pairs $\{p,p'\} \in \mathcal{C}$ and days $t \in [T]$. Stated otherwise, $F(t)$ is an independent set in the graph $(\mathcal{P}, \mathcal{C})$ for all days $t \in [T]$. Note that the conflict distance is defined with respect to the pharmacies, as we can attain larger distances between out-of-hours pharmacies in rural areas compared to cities. In practice, we will choose the conflict distance with respect to the municipality, that is, we have $\delta^{\text{con}}(p) = \delta^{\text{con}}(p')$ if $M(p) = M(p')$ holds.

Periods of Rest A pharmacy should not be assigned shifts on consecutive days or days close to each other due to the high workload. We therefore introduce for all pharmacies $p \in \mathcal{P}$

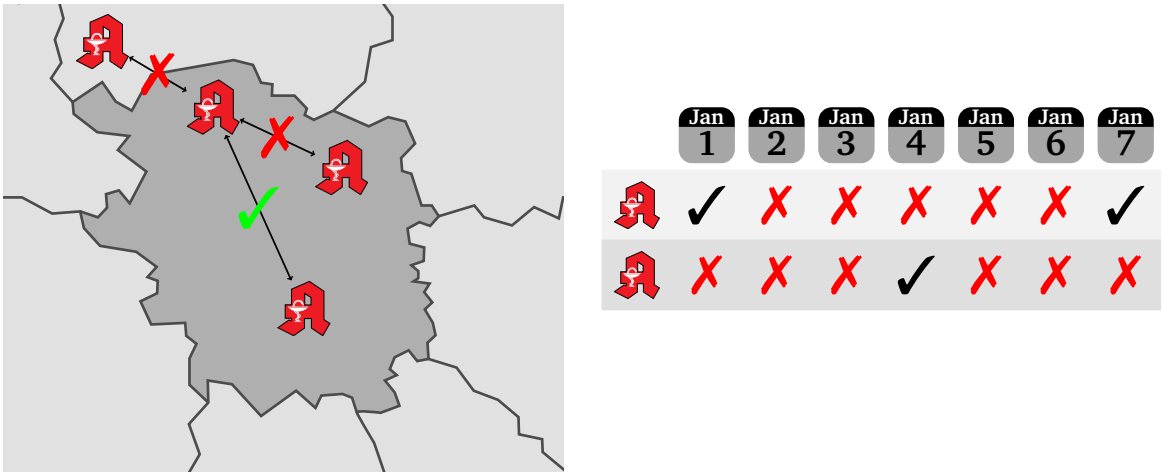


Figure 7.2. Restrictions on the assignment of shifts. **Left:** minimum distance between two out-of-hours pharmacies on the same day. **Right:** period of rest between shifts of the same pharmacy.

the *period of rest* $r(p) \in \mathbb{Z}_{\geq 0}$, that is, the minimum number of days between two shifts of pharmacy p (see Figure 7.2). A feasible out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ has to guarantee these periods of rest, i.e., we have $p \notin F(t) \cap F(t')$ for all pharmacies $p \in \mathcal{P}$ and days $t \neq t' \in [T]$ with $|t - t'| \leq r(p)$. Just like the conflict distance, we will define the period of rest with respect to the municipalities, as we can have longer periods of rest in cities where the pharmacy density is higher compared to rural areas.

Municipality-Balancing The general principle of equality obliges us to assign shifts in a fair manner, as pointed out in the legal commentary on the pharmacy operating regulations [83]. The definition of a fair plan strongly depends on the personal viewpoint of each pharmacist and is in general difficult to measure and not trivial to incorporate into optimization problems. We will focus on a global fair planning that balances the burden of all pharmacies $p \in \mathcal{P}$ in the next chapter. Here, we only consider the balancing of shifts within the same municipality. According to the chamber of pharmacists of the area North Rhine, pharmacists expect that other pharmacies within the same municipality are assigned a similar number of shifts. We therefore introduce the *balancing coefficient* $b \in \mathbb{Z}_{\geq 0}$, which bounds the difference in the number of shifts that two pharmacies $p, p' \in P(m)$ belonging to the same municipality $m \in \mathcal{M}$ can be assigned. We say that an out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ is *municipality-balanced* if $||S_F(p)| - |S_F(p')|| \leq b$ holds for all pairs $p, p' \in \mathcal{P}$ with $M(p) = M(p')$.

Minimum Number of Shifts The authorities designing an out-of-hours plan are legally obliged to assign out-of hours shifts to all pharmacies [83]. Therefore, we have to ensure that all pharmacies participate appropriately in the service. We will see in the case study in Section 7.4 that municipality-balancing is not sufficient to guarantee that all pharmacies are assigned shifts. In fact, there can be whole municipalities in which no pharmacy is assigned any shift.

We therefore introduce a minimum number of shifts $\underline{s} \in \mathbb{Z}_{\geq 0}$ that all pharmacies should be assigned. A feasible plan $F : [T] \to 2^{\mathcal{P}}$ then fulfills $|S_F(p)| \geq \underline{s}$ for all $p \in \mathcal{P}$.

We incorporate the above criteria in an MILP for the OHP. For this, we use binary decision variables $x_{pt} \in \{0,1\}$ to indicate whether or not pharmacy $p \in \mathcal{P}$ is assigned a shift on day $t \in [T]$. Accordingly, $x \in \{0,1\}^{\mathcal{P} \times [T]}$ implies an out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ with $F(t) = \{p \in \mathcal{P} \mid x_{pt} = 1\}$ for $t \in [T]$. To model the municipality-balancing in an efficient way, we introduce two auxiliary variables $\underline{y}_m, \bar{y}_m \in \mathbb{R}_{\geq 0}$ for each municipality $m \in \mathcal{M}$ that bound the minimum (maximum) number of shifts assigned to any pharmacy $p \in P(m)$ within the municipality. The proposed MILP formulation reads

$$\min \sum_{p \in \mathcal{P}} \sum_{t \in [T]} x_{pt} \tag{OHP.a}$$

$$\text{s.t.} \sum_{p \in C(m)} x_{pt} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [T] \tag{OHP.b}$$

$$\sum_{p \in P(m)} x_{pt} \geq d(m,t) \qquad \forall m \in \mathcal{M}, t \in [T] \tag{OHP.c}$$

$$x_{pt} + x_{p't} \leq 1 \qquad \forall \{p,p'\} \in \mathcal{C}, t \in [T] \tag{OHP.d}$$

$$\sum_{t'=t}^{t+r(p)} x_{pt'} \leq 1 \qquad \forall p \in \mathcal{P}, t \in [T - r(p)] \tag{OHP.e}$$

$$\underline{y}_{M(p)} \leq \sum_{t \in [T]} x_{pt} \leq \bar{y}_{M(p)} \qquad \forall p \in \mathcal{P} \tag{OHP.f}$$

$$\bar{y}_m - \underline{y}_m \leq b \qquad \forall m \in \mathcal{M} \tag{OHP.g}$$

$$\underline{y}_m \geq \underline{s} \qquad \forall m \in \mathcal{M} \tag{OHP.h}$$

$$x \in \{0,1\}^{\mathcal{P} \times [T]}, \underline{y}, \bar{y} \in \mathbb{R}^{\mathcal{M}}_{\geq 0}. \tag{OHP.i}$$

The modeling of the objective function (OHP.a), the covering of municipalities (OHP.b), the demand in cities (OHP.c), and the conflicts between pharmacies (OHP.d) is straightforward. For the period of rest, we do not use pairwise constraints $x_{pt} + x_{pt'} \leq 1$ for days $t, t' \in [T]$ with $|t - t'| \leq r(p)$ but constraints (OHP.e) which allow at most one shift in an interval of $r(p) + 1$ days. The interval constraints dominate the pairwise constraints, and thus yield a stronger formulation. Moreover, we only have to add $T - r(p)$ interval constraints instead of $(T - r(p))\, r(p)$ pairwise constraints for each pharmacy $p \in \mathcal{P}$. Constraints (OHP.f) ensure that the auxiliary variables $\underline{y}_m, \bar{y}_m$ bound the number of shifts of all pharmacies $p \in P(m)$ for $m \in \mathcal{M}$. This ensures together with constraints (OHP.g) that the plan is municipality-balanced. Using the $2|\mathcal{M}|$ auxiliary variables $\underline{y}, \bar{y}$ enables us to model the municipality-balancing with $2|\mathcal{P}| + |\mathcal{M}|$ constraints. In comparison, a pairwise modeling with $x_p - x_{p'} \leq b$ for all $m \in \mathcal{M}$ and $p, p' \in P(m)$ would instead require $\sum_{m \in \mathcal{M}} |P(m)|\,(|P(m)| - 1)$ constraints. Finally, we also use the lower bound variables $\underline{y}_m$ in constraints (OHP.h) to ensure that every pharmacy is assigned at least the minimum number of shifts. In the following, we will often neglect the auxiliary variables $\underline{y}, \bar{y}$ for simplicity and say that $x \in \{0,1\}^{\mathcal{P} \times [T]}$ is a solution to the MILP above.

The above formulation for the OHP surely becomes large for real-world instances when planning the out-of-hours service of a whole year for thousands of pharmacies. We will see in our case study in Section 7.4 that solving the MILP directly as above is therefore not practical. However, we will propose approaches in Section 7.3 that enable us to solve the OHP in reasonable time. Before doing so, we consider the theoretical complexity of the OHP in the next section.

7.2 Complexity of the OHP

The OHP is a combination of the classical $\mathcal{NP}$-hard set cover and independent set problem together with some balancing constraints. It is therefore not surprising that the OHP itself is also hard to solve. The proposition below shows that it is hard to compute *any* solution to the OHP, even if several constraints are neglected.

Proposition 34. *The problem of deciding whether there exists a solution to an OHP instance is strongly $\mathcal{NP}$-complete. This holds already for periods of rest $r \equiv 1$ and without considering demands in cities, conflicts, municipality-balancing, and minimum numbers of shifts, i.e., $d \equiv 0$, $\delta^{con} \equiv 0$, $b = T$ and $\underline{s} = 0$.*

Proof. The OHP is in $\mathcal{NP}$, since we only have to verify a polynomial number of linear constraints in order to decide whether a solution is feasible. We prove that it is $\mathcal{NP}$-hard to decide whether there exists any solution to OHP via a reduction from the k *Disjoint Set Covers Problem* (k-DSC). For the k-DSC, we are given a finite basic set $\mathcal{I}$, a family of subsets $\mathcal{J} \subseteq 2^{\mathcal{I}}$ and a positive integer $k \in \mathbb{Z}_{\geq 2}$. The problem then asks whether there exists a partition of $\mathcal{J}$ into k disjoint subsets that all cover $\mathcal{I}$, i.e., $\mathcal{J} = \mathcal{J}_1 \uplus \cdots \uplus \mathcal{J}_k$ with $\mathcal{I} = \bigcup_{J \in \mathcal{J}_i} J$ for all $i \in [k]$. The k-DSC is strongly $\mathcal{NP}$-complete for all $k \in \mathbb{Z}_{\geq 2}$ [33].

Let $(\mathcal{I}, \mathcal{J}, k)$ be a k-DSC instance. We construct a corresponding OHP instance as follows. Since municipalities are covered by pharmacies and elements in $\mathcal{I}$ are covered by subsets in $\mathcal{J}$, we identify the municipalities $\mathcal{M}$ with the basic elements $\mathcal{I}$ and the pharmacies $\mathcal{P}$ with the family of subsets $\mathcal{J}$. Accordingly, we introduce one municipality m_i for each element $i \in \mathcal{I}$ and one pharmacy p_J for each set $J \in \mathcal{J}$. We then define metric distances between pharmacies and municipalities

$$\begin{aligned}
\delta(p, p') &= 1, \text{ for } p, p' \in \mathcal{P}, \\
\delta(m, m') &= 1, \text{ for } m, m' \in \mathcal{P}, \\
\delta(p_J, m_i) = \delta(m_i, p_J) &= 1, \text{ for } p_J \in \mathcal{P}, m_i \in \mathcal{M} \text{ with } i \in J, \\
\delta(p_J, m_i) = \delta(m_i, p_J) &= 2, \text{ for } p_J \in \mathcal{P}, m_i \in \mathcal{M} \text{ with } i \notin J
\end{aligned}$$

and set the cover radii $\delta^{\text{cov}}(m) = 1$ for all $m \in \mathcal{M}$. This yields $C(m_i) = \{p_J \in \mathcal{P} \mid i \in J\}$ for all $m_i \in \mathcal{M}$. Thus, pharmacy $p_J \in \mathcal{P}$ can cover municipality $m_i \in \mathcal{M}$ if the corresponding

subset $J \in \mathcal{J}$ contains the element $i \in \mathcal{I}$. Furthermore, we define the time horizon $[T] = [k]$ and the periods of rest $r \equiv k - 1$. The remaining parameters are defined so that the corresponding constraints are not relevant for the constructed OHP instance, i.e., $d \equiv 0$, $\delta^{\text{con}} \equiv 0$, $b = T$, and $\underline{s} = 0$.

Let $\{\mathcal{J}_1, \cdots, \mathcal{J}_k\}$ be a feasible solution to the k-DSC instance. Then we define an out-of-hours plan by $F(t) = \{p_J \in \mathcal{P} \mid J \in \mathcal{J}_t\}$ for $t \in [T] = [k]$. Every municipality is by construction covered on every day, since each $\mathcal{J}_t$ is a cover for $\mathcal{I}$. Furthermore, the period of rest constraints are fulfilled, since each pharmacy is assigned exactly one shift.

Conversely, we construct a solution for the k-DSC instance from a feasible out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ by defining $\mathcal{J}_t = \{J \in \mathcal{J} \mid p_J \in F(t)\}$ for all $t \in [k-1]$ and $\mathcal{J}_k = \mathcal{J} \setminus \left(\bigcup_{t \in \{[k-1]\}} \mathcal{J}_t\right)$. We have $\mathcal{J} = \mathcal{J}_1 \uplus \cdots \uplus \mathcal{J}_k$, since each pharmacy is assigned at most one shift due to the periods of rest. Additionally, we have $\mathcal{I} = \bigcup_{J \in \mathcal{J}_t} J$ by construction for all $t \in [k]$, since each municipality is covered on every day.

We conclude that there exists a feasible solution to k-DSC if and only if there exists a feasible out-of-hours plan for the constructed OHP instance. This shows that the OHP is $\mathcal{NP}$-hard, as the construction of the OHP instance is polynomial in the input size of the k-DSC. Moreover, all parameters of the OHP instance are polynomial in the input size, which shows that the OHP is $\mathcal{NP}$-hard in the strong sense [45, Section 4.2]. This especially holds for $r \equiv 1$, since the k-DSC is already $\mathcal{NP}$-hard for $k = 2$. □

The theoretical complexity of the OHP rules out the possibility for designing polynomial algorithms, unless $\mathcal{P} = \mathcal{NP}$. However, $\mathcal{NP}$-completeness does not offer much insight into the tractability of specific instances, especially since the construction of the distances in the proof is not close to real OHP instances. In the following section, we propose several MILP-based approaches for solving the OHP, which will provide nearly optimal solutions in our case study in Section 7.4.

7.3 Solution Approaches for the OHP

Solving the OHP directly as an MILP is not practical when planning the out-of-hours service of a whole year for many pharmacies. We therefore propose a smaller reformulation and a heuristic approach in the subsequent sections.

7.3.1 Aggregating Equivalent Pharmacies

Consider a set of neighboring pharmacies that are identical regarding their coverage, their period of rest, and their conflicts. Our model is indifferent between these pharmacies in the sense that we can swap all their shifts in an out-of-hours plan without loosing feasibility.

Such symmetries in the set of solutions should be broken, as otherwise they can reduce the performance of MILP solvers drastically [74]. This is because symmetries potentially lead to the exploration of many unnecessary nodes in the branch and bound tree that emerge from each other through permutation. In our case, however, symmetric structures are actually helpful, as we can use them in order to reduce the size of our MILP formulation.

Formally, we say that two pharmacies $p_1, p_2 \in \mathcal{P}$ are *equivalent* and write $p_1 \sim p_2$ if we have $M(p_1) = M(p_2)$, $C(p_1) = C(p_2)$, $r(p_1) = r(p_2)$, and $N[p_1] = N[p_2]$, where $N[p]$ is the closed neighborhood of $p \in \mathcal{P}$ in the graph of conflicts $(\mathcal{P}, \mathcal{C})$. We will aggregate sets of equivalent pharmacies in order to break symmetries and reduce the size of our MILP formulation. To this end, we call a subset of pharmacies $p^s = \{p_1, \ldots, p_k\} \subseteq \mathcal{P}$ a *superpharmacy* if we have $p \sim p'$ for all pairs $p, p' \in p^s$. Note that a single pharmacy $p \in \mathcal{P}$ already defines a superpharmacy $\{p\}$. Therefore, we can partition the set of pharmacies $\mathcal{P}$ into a *set of superpharmacies* $\mathcal{P}^s$ with $\mathcal{P} = \biguplus_{p^s \in \mathcal{P}^s} p^s$.

It is irrelevant for the coverage and conflicts to which pharmacy of a superpharmacy we assign a shift. Moreover, since all pharmacies within a superpharmacy are in conflict with each other, we can assign them at most one shift per day in total. This allows us to decompose the planning into two steps. In the first step, we compute a plan in which we consider each superpharmacy $p^s \in \mathcal{P}^s$ as a single artificial pharmacy. This yields a smaller formulation with fewer variables and breaks all symmetries regarding pharmacies within the same superpharmacy. In the second step, we then distribute the shifts of each superpharmacy $p^s \in \mathcal{P}^s$ evenly among the aggregated pharmacies $p \in p^s$. We will show later that this can be done without violating period of rest and municipality-balancing constraints.

Before doing so, we update our notation for the superpharmacy-based planning. First, we say that a superpharmacy $p^s \in \mathcal{P}^s$ can cover a municipality $m \in \mathcal{M}$ if the pharmacies aggregated within p^s can cover m, i.e., $p^s \subseteq C(m)$. Accordingly, we define the set of superpharmacies that can cover m as $C^s(m) = \{p^s \in \mathcal{P}^s \mid p^s \subseteq C(m)\}$. Second, we say that a superpharmacy $p^s \in \mathcal{P}^s$ lies within a municipality $m \in \mathcal{M}$ if the pharmacies aggregated within p^s lie within m, i.e., $p^s \subseteq P(m)$. We denote the municipality in which $p^s \in \mathcal{P}^s$ lies with $M(p^s)$ and define the set of superpharmacies that lie within m as $P^s(m) = \{p^s \in \mathcal{P}^s \mid p^s \subseteq P(m)\}$. Third, a pair of superpharmacies $p_1^s, p_2^s \in \mathcal{P}^s$ are in conflict if their aggregated pharmacies are in conflict, i.e., $\{p_1, p_2\} \in \mathcal{C}$ for $p_1 \in p_1^s$ and $p_2 \in p_2^s$. Accordingly, we define the set of superpharmacy conflicts $\mathcal{C}^s = \left\{\{p_1^s, p_2^s\} \in \binom{\mathcal{P}^s}{2} \mid \{p_1, p_2\} \in \mathcal{C} \text{ for } p_1 \in p_1^s, p_2 \in p_2^s\right\}$. Finally, we define the period of rest $r(p^s)$ of $p^s \in \mathcal{P}^s$ to be equal to the period of rest $r(p)$ of the aggregated pharmacies $p \in p^s$. Note that all of the above is well-defined due to the equivalency of pharmacies within a superpharmacy.

We reformulate the OHP for a given partition $\mathcal{P}^{\mathrm{s}}$ of $\mathcal{P}$ into superpharmacies by using decision variables $x^{\mathrm{s}} \in \{0,1\}^{\mathcal{P}^{\mathrm{s}} \times [T]}$. These indicate for every superpharmacy $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ and day $t \in [T]$ whether some pharmacy $p \in p^{\mathrm{s}}$ is assigned a shift on that day. The resulting problem reads

$$\min \sum_{p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \tag{SOHP.a}$$

$$\text{s.t.} \sum_{p^{\mathrm{s}} \in C^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [T] \tag{SOHP.b}$$

$$\sum_{p^{\mathrm{s}} \in P^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \geq d\,(m,t) \qquad \forall m \in \mathcal{M}, t \in [T] \tag{SOHP.c}$$

$$x^{\mathrm{s}}_{p^{\mathrm{s}}_1 t} + x^{\mathrm{s}}_{p^{\mathrm{s}}_2 t} \leq 1 \qquad \forall \{p^{\mathrm{s}}_1, p^{\mathrm{s}}_2\} \in \mathcal{C}^{\mathrm{s}}, t \in [T] \tag{SOHP.d}$$

$$\sum_{t'=t}^{t+r(p^{\mathrm{s}})} x^{\mathrm{s}}_{p^{\mathrm{s}} t'} \leq |p^{\mathrm{s}}| \qquad \forall p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}, t \in [T - r\,(p^{\mathrm{s}})] \tag{SOHP.e}$$

$$\underline{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})} \leq \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \leq \bar{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})} \qquad \forall p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}} \tag{SOHP.f}$$

$$\bar{y}^{\mathrm{s}}_m - \underline{y}^{\mathrm{s}}_m \leq b \qquad \forall m \in \mathcal{M} \tag{SOHP.g}$$

$$\underline{y}^{\mathrm{s}}_m \geq \underline{s} \qquad \forall m \in \mathcal{M} \tag{SOHP.h}$$

$$x^{\mathrm{s}} \in \{0,1\}^{\mathcal{P}^{\mathrm{s}} \times [T]}, \underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}} \in \mathbb{Z}^{\mathcal{M}}_{\geq 0}. \tag{SOHP.i}$$

The objective function (SOHP.a) and the coverage constraints (SOHP.b) and (SOHP.c) arise from OHP by substituting $\sum_{p \in p^{\mathrm{s}}} x_{pt} = x^{\mathrm{s}}_{p^{\mathrm{s}} t}$ for all $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ and $t \in [T]$. Constraints (SOHP.d) model conflicts between superpharmacies. Constraints (SOHP.e) allow the assignment of up to $|p^{\mathrm{s}}|$ shifts to a superpharmacy $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ in an interval of $r\,(p^{\mathrm{s}}) + 1$ consecutive days, since each of the $|p^{\mathrm{s}}|$ included pharmacies can be assigned at most one shift within this interval. Constraints (SOHP.f) bound the number of shifts that we assign to pharmacies within a municipality. Since the shifts of superpharmacy $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ will later be evenly distributed among the aggregated pharmacies $p \in p^{\mathrm{s}}$, each of them will be assigned $\left\lfloor \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \right\rfloor$ or $\left\lceil \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \right\rceil$ shifts. Note that we replace the original continuous variables $\underline{y}, \bar{y} \in \mathbb{R}^{\mathcal{M}}_{\geq 0}$ with integer variables $\underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}} \in \mathbb{Z}^{\mathcal{M}}_{\geq 0}$ to ensure that we have

$$\underline{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})} \leq \left\lfloor \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \right\rfloor \leq \left\lceil \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}} t} \right\rceil \leq \bar{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})}.$$

The remaining constraints (SOHP.g) and (SOHP.h) are then the same as in the original MILP but with the new variables $\underline{y}^{\mathrm{s}}$ and $\bar{y}^{\mathrm{s}}$.

The following result shows that SOHP is a valid reformulation and at least as strong as OHP. Furthermore, the proof gives instructions on how to convert a plan for superpharmacies to a plan for real pharmacies.

Proposition 35. *Let $\mathcal{P}^{\mathrm{s}}$ be a partition of $\mathcal{P}$ into superpharmacies. Then SOHP is a reformulation in a different variable space that is at least as strong as OHP.*

Proof. We first show that for every solution to OHP, there exists a corresponding solution to SOHP with the same objective value. Let $\left(x, \underline{y}, \bar{y}\right)$ be a solution to OHP. We define $\left(x^{\mathrm{s}}, \underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}}\right)$ with $x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \sum_{p \in p^{\mathrm{s}}} x_{pt}$ for all $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}},\ t \in [T]$ as well as $\underline{y}^{\mathrm{s}}_m = \min_{p \in P(m)} \left\{\sum_{t \in [T]} x_{pt}\right\}$ and $\bar{y}^{\mathrm{s}}_m = \max_{p \in P(m)} \left\{\sum_{t \in [T]} x_{pt}\right\}$ for all $m \in \mathcal{M}$. Since $\mathcal{P}^{\mathrm{s}}$ is a partition of $\mathcal{P}$, we have

$$\sum_{p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \sum_{p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}} \sum_{p \in p^{\mathrm{s}}} \sum_{t \in [T]} x_{pt} = \sum_{p \in \mathcal{P}} \sum_{t \in [T]} x_{pt},$$

and thus both solutions have the same objective value. To see that $\left(x^{\mathrm{s}}, \underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}}\right)$ is feasible for the SOHP, first note that we have $x^{\mathrm{s}} \in \{0,1\}^{\mathcal{P}^{\mathrm{s}} \times [T]}$, because at most one of the aggregated pharmacies can have a shift on a fixed day due to the conflicts within superpharmacies. The coverage constraints (SOHP.b) and (SOHP.c) are met, as $\sum_{p^{\mathrm{s}} \in C^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \sum_{p \in C(m)} x_{pt} \geq 1$ and $\sum_{p^{\mathrm{s}} \in P^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \sum_{p \in P(m)} x_{pt} \geq d\,(m,t)$ hold by the definition of superpharmacies and the feasibility of $\left(x, \underline{y}, \bar{y}\right)$. The conflict constraints (SOHP.d) are respected, because $x^{\mathrm{s}}_{p^{\mathrm{s}}_1 t} + x^{\mathrm{s}}_{p^{\mathrm{s}}_2 t} > 1$ for some $\{p^{\mathrm{s}}_1, p^{\mathrm{s}}_2\} \in \mathcal{C}^{\mathrm{s}}$ and $t \in [T]$ would imply that there exist two pharmacies $p_1 \in p^{\mathrm{s}}_1$ and $p_2 \in p^{\mathrm{s}}_2$ with $\{p_1, p_2\} \in \mathcal{C}$ and $x_{p_1 t} + x_{p_2 t} = 2$, which is a contradiction to the constraints (OHP.d). The period of rest constraints (SOHP.e) are fulfilled, since we have $r\,(p) = r\,(p^{\mathrm{s}})$, and thus

$$\sum_{t'=t}^{t+r(p^{\mathrm{s}})} x^{\mathrm{s}}_{p^{\mathrm{s}}t'} = \sum_{p \in p^{\mathrm{s}}} \sum_{t'=t}^{t+r(p)} x_{pt'} \leq \sum_{p \in p^{\mathrm{s}}} 1 = |p^{\mathrm{s}}|$$

for all $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ and $t \in [T - r\,(p^{\mathrm{s}})]$. The bounding constraints (SOHP.f) are met, since we have

$$\underline{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})} = \frac{|p^{\mathrm{s}}|}{|p^{\mathrm{s}}|} \min_{p \in P(M(p^{\mathrm{s}}))} \left\{\sum_{t \in [T]} x_{pt}\right\} \leq \frac{1}{|p^{\mathrm{s}}|} \sum_{p \in p^{\mathrm{s}}} \sum_{t \in [T]} x_{pt} = \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t}$$

and analogously $\bar{y}^{\mathrm{s}}_{M(p^{\mathrm{s}})} \geq \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t}$ for all $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$. By definition and due to the constraints (OHP.f), we have $\underline{y}^{\mathrm{s}} \geq \underline{y}$ and $\bar{y}^{\mathrm{s}} \leq \bar{y}$. Then constraints (SOHP.g) and (SOHP.h) follow from constraints (OHP.g) and (OHP.h).

It remains to show that there exists a polynomial-time computable, objective-preserving mapping from the set of (continuous) solutions for SOHP to the set of (continuous) solutions for OHP. We will first consider the mapping for integer solutions and afterwards extend it to continuous solutions in order to show that SOHP is at least as strong as OHP (cf. 2.4.3).

Let $\left(x^{\mathrm{s}}, \underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}}\right)$ be an integer solution to SOHP and let $\{t_1, \ldots, t_\ell\} = \left\{t \in [T] \mid x^{\mathrm{s}}_{p^{\mathrm{s}}t} = 1\right\}$ with $t_1 < \cdots < t_\ell$ be the sorted set of days on which we assign shifts to a superpharmacy $p^{\mathrm{s}} = \{p_1, \ldots, p_k\} \in \mathcal{P}^{\mathrm{s}}$. We distribute the shifts among the aggregated pharmacies in a cyclic way, that is, we assign the days $\{t_1, t_{k+1}, t_{2k+1}, \ldots\} \subseteq \{t_1, \ldots, t_\ell\}$ to pharmacy p_1, the days $\{t_2, t_{k+2}, t_{2k+2}, \ldots\} \subseteq \{t_1, \ldots, t_\ell\}$ to pharmacy p_2, and so on. For pharmacy $p_i \in p^{\mathrm{s}}$, we then have $x_{p_i t} = 1$ for $t \in \{t_i, t_{k+i}, t_{2k+i}, \ldots\} \subseteq \{t_1, \ldots, t_\ell\}$ and $x_{p_i t} = 0$ otherwise. Each shift is thus assigned to exactly one included pharmacy and we obtain $x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \sum_{p \in p^{\mathrm{s}}} x_{pt}$. Just like above, the satisfaction of constraints (OHP.b) and (OHP.c) is equivalent to the satisfaction of

constraints (SOHP.b) and (SOHP.c). The conflict constraints (OHP.d) are met for pharmacies within the same superpharmacy, because we assign at most one shift per day. For pharmacies within different superpharmacies, constraints (SOHP.d) dominate constraints (OHP.d) when replacing $x^{\mathrm{s}}_{p^{\mathrm{s}}t}$ with $\sum_{p \in p^{\mathrm{s}}} x_{pt}$. For the period of rest, assume that some constraint (OHP.e) is violated. Then there exists a pharmacy $p \in p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ and two days $t_i < t_j \in [T]$ with $t_j - t_i \leq r(p)$ and $x_{pt_i} = x_{pt_j} = 1$. This implies that there exist at least $k - 1 = |p^{\mathrm{s}}| - 1$ days t between t_i and t_j with $x^{\mathrm{s}}_{p^{\mathrm{s}}t} = 1$ due to the cyclic assignment. However, we then have $\sum_{t'=t_i}^{t+r(p^{\mathrm{s}})} x^{\mathrm{s}}_{p^{\mathrm{s}}t'} \geq |p^{\mathrm{s}}| + 1$, which contradicts the constraints (SOHP.e). For the municipality-balancing, we simply define $\underline{y} = \underline{y}^{\mathrm{s}}$ and $\bar{y} = \bar{y}^{\mathrm{s}}$ and therefore satisfy the constraints (OHP.g) and (OHP.h). By construction, we have $\sum_{t \in [T]} x_{pt} \in \left\{ \left\lfloor \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \right\rfloor, \left\lceil \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \right\rceil \right\}$ for all pharmacies $p \in p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$. Then in combination with constraints (SOHP.f) and the definition of $\underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}} \in \mathbb{Z}^{\mathcal{M}}_{\geq 0}$, it follows that we satisfy the constraints (OHP.f). This shows, that $\left(x, \underline{y}, \bar{y}\right)$ is feasible for the OHP.

Given a fractional solution $\left(x^{\mathrm{s}}, \underline{y}^{\mathrm{s}}, \bar{y}^{\mathrm{s}}\right)$ of the continuous relaxation of SOHP, we construct a continuous solution $\left(x, \underline{y}, \bar{y}\right)$ for OHP via $x_{pt} = \frac{1}{|p^{\mathrm{s}}|} x^{\mathrm{s}}_{p^{\mathrm{s}}t}$ for all $p \in p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}},\ t \in [T]$ as well as $\underline{y} = \underline{y}^{\mathrm{s}}$ and $\bar{y} = \bar{y}^{\mathrm{s}}$. The satisfaction of constraints (OHP.b), (OHP.c), (OHP.d), (OHP.g), and (OHP.h) are analogous to the reasoning above. The satisfaction of the period of rest constraints (OHP.e) follow from

$$\sum_{t'=t}^{t+r(p)} x_{pt'} = \sum_{t'=t}^{t+r(p)} \frac{1}{|p^{\mathrm{s}}|} x^{\mathrm{s}}_{p^{\mathrm{s}}t} = \frac{1}{|p^{\mathrm{s}}|} \sum_{t'=t}^{t+r(p^{\mathrm{s}})} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \leq \frac{|p^{\mathrm{s}}|}{|p^{\mathrm{s}}|} = 1$$

and the constraints (OHP.f) directly from $\sum_{t \in [T]} x_{pt} = \frac{1}{|p^{\mathrm{s}}|} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t}$. □

The proof shows that we can first compute an optimal solution for SOHP and then obtain an optimal solution for OHP by distributing the shifts of superpharmacies among their aggregated pharmacies in a cyclic way. Note that SOHP is actually stronger for non-trivial cases, as the aggregation in superpharmacies corresponds to a strengthening of the pairwise conflict constraints into clique inequalities. Therefore, SOHP is not only much smaller but also stronger than OHP.

The number of variables and constraints in SOHP decreases for partitions $\mathcal{P}^{\mathrm{s}}$ of $\mathcal{P}$ with fewer superpharmacies. Therefore, we are interested in superpharmacies containing many pharmacies. We call a superpharmacy $p^{\mathrm{s}}_1 \subseteq \mathcal{P}$ *maximal* if there exists no other superpharmacy $p^{\mathrm{s}}_2 \subseteq \mathcal{P}$ with $p^{\mathrm{s}}_1 \subsetneq p^{\mathrm{s}}_2$. Fortunately, there exists a unique partition $\mathcal{P}^{\mathrm{s}}$ of $\mathcal{P}$ into maximal superpharmacies that can be computed easily. For three pharmacies $p_1, p_2, p_3 \in \mathcal{P}$ with $p_1 \sim p_2$ and $p_2 \sim p_3$, it follows from our definition of equivalency that $p_1 \sim p_3$ holds. Hence, for two maximal superpharmacies $p^{\mathrm{s}}_1 \neq p^{\mathrm{s}}_2 \subseteq \mathcal{P}$, it must hold $p^{\mathrm{s}}_1 \cap p^{\mathrm{s}}_2 = \emptyset$, since otherwise $p^{\mathrm{s}}_1 \cup p^{\mathrm{s}}_2$ would be a larger superpharmacy, contradicting the maximality of p^{s}_1 or p^{s}_2. It follows that every pharmacy $p \in \mathcal{P}$ is contained in exactly one maximal superpharmacy, consisting of all pharmacies $p' \in \mathcal{P}$ with $p \sim p'$.

We will see in our case-study in Section 7.4 that the aggregation into superpharmacies reduces the size of our formulation significantly, resulting in planning problems that are much easier to solve. However, SOHP is still too difficult for our large real-world instance. We therefore propose a heuristic for the planning of the out-of-hours service in the subsequent sections. For simplicity, we will consider the original model OHP, but all following concepts can be directly applied for solving SOHP.

7.3.2 A Rolling Horizon Approach

The decision variables $x_{pt}, x_{pt'}$ for different days $t, t' \in [T]$ are solely connected by the periods of rest, municipality-balancing, and minimum number of shifts constraints. Thus, intuitively, the choice of $x_{pt} \in \{0,1\}$ has less impact on the possible choices of $x_{pt'} \in \{0,1\}$ the farther the days $t, t' \in [T]$ are apart. This is because both variables are not connected directly by a period of rest constraint if $|t - t'| > r(p)$. Planning problems with such loose temporal links over distant time periods can often be solved nearly optimally by a so-called *rolling horizon approach*.

The general idea of rolling horizon approaches is to divide a problem with a large time horizon into a sequence of smaller subproblems. In each of these subproblems, one considers an iteratively extending fraction of the time horizon, which eventually equals the whole time horizon. In doing so, one fixes decisions arising from the solutions of the previous subproblems in order to ensure that each subproblem remains tractable. For solving the OHP, we first split the time horizon $[T]$ into $\ell+1$ intervals $[T] = \{1, \dots, t_1\} \uplus \{t_1 + 1, \dots, t_2\} \uplus \dots \uplus \{t_\ell + 1, \dots, T\}$ and denote $t_0 = 0$ and $t_{\ell+1} = T$ in the following for simplicity. We then iteratively determine for $i \in [\ell + 1]$ a (partial) out-of-hours plan $F_i : [t_i] \to 2^{\mathcal{P}}$ that respects the shifts already computed for the days $[t_{i-1}]$ (note that $[t_{i-1}] = \emptyset$ for $i = 1$). The plan computed in the last iteration is then our plan for the whole time horizon $[T] = [t_{\ell+1}]$.

Rolling horizon approaches often incorporate some form of look-ahead, such that the decisions made in the current iteration also take into account restrictions imposed by later time periods. We do this in our approach by including an additional interval into our subproblems. That is, when computing $F_i : [t_i] \to 2^{\mathcal{P}}$ in iteration $i \in [\ell]$, we actually compute a plan for all days in $[t_{i+1}]$ but only fix shifts assigned in $[t_i]$ as long as $i < \ell$. We thus pay attention to the direct connection induced by the periods of rest, which are arguably the most restrictive temporal constraints for our problem.

Let $F_{i-1} : [t_{i-1}] \to 2^{\mathcal{P}}$ be a partial plan from a previous iteration. For computing a plan $F_i : [t_i] \to 2^{\mathcal{P}}$ that respects the already assigned shifts $S_{F_{i-1}}(p) \subseteq [t_{i-1}]$ of pharmacies $p \in \mathcal{P}$

and takes the days of the next interval $\{t_i + 1, \ldots, t_{i+1}\}$ into account, we solve the following MILP

$$\min \sum_{p \in \mathcal{P}} \sum_{t \in [t_{i+1}]} x_{pt} + \sum_{m \in \mathcal{M}} B\beta_m \tag{ROHP.a}$$

$$\text{s.t.} \sum_{p \in C(m)} x_{pt} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [t_{i+1}] \tag{ROHP.b}$$

$$\sum_{p \in P(m)} x_{pt} \geq d\,(m,t) \qquad \forall m \in \mathcal{M}, t \in [t_{i+1}] \tag{ROHP.c}$$

$$x_{pt} + x_{p't} \leq 1 \qquad \forall \{p, p'\} \in \mathcal{C}, t \in [t_{i+1}] \tag{ROHP.d}$$

$$\sum_{t'=t}^{t+r(p)} x_{pt'} \leq 1 \qquad \forall p \in \mathcal{P}, t \in [t_{i+1} - r\,(p^s)] \tag{ROHP.e}$$

$$\underline{y}_{M(p)} \leq \sum_{t \in [t_{i+1}]} x_{pt} \leq \bar{y}_{M(p)} \qquad \forall p \in \mathcal{P} \tag{ROHP.f}$$

$$\bar{y}_m - \underline{y}_m \leq \left\lceil \frac{bt_{i+1}}{T} \right\rceil + \beta_{M(p)} \qquad \forall m \in \mathcal{M} \tag{ROHP.g}$$

$$\underline{y}_m \geq \left\lceil \frac{\underline{s}t_{i+1}}{T} \right\rceil \qquad \forall m \in \mathcal{M} \tag{ROHP.h}$$

$$x_{pt} = 1 \qquad \forall p \in \mathcal{P}, t \in S_{F_{i-1}}\,(p) \tag{ROHP.i}$$

$$x \in \{0,1\}^{\mathcal{P} \times [t_{i+1}]}, \underline{y}, \bar{y} \in \mathbb{R}^{\mathcal{M}}_{\geq 0}, \beta_m \in \mathbb{Z}^{\mathcal{M}}_{\geq 0}. \tag{ROHP.j}$$

Constraints (ROHP.i) fix variables $x_{pt} = 1$ if we already assigned pharmacy $p \in \mathcal{P}$ a shift on day $t \in [t_{i-1}]$. However, we never fix $x_{pt} = 0$, even if we already considered $t \in [t_{i-1}]$ in a previous iteration and did not assign $p \in \mathcal{P}$ a shift on day t. Doing so gives us more freedom to satisfy the municipality-balancing constraints. For the case that we still cannot construct a municipality-balanced plan, we introduce variables $\beta_m \in \mathbb{Z}_{\geq 0}$ for all $m \in \mathcal{M}$ in constraints (ROHP.g) that allow a violation of the municipality-balancing but are penalized by a big constant $B \in \mathbb{Z}_{>0}$ in the objective (ROHP.a). In this way, we ensure that a subproblem will not be infeasible solely due to the municipality-balancing constraints. This is reasonable, since we may be able to balance the out-of-hours plan in the following iterations. Note that we scale the balancing coefficient b in constraints (ROHP.g) with the relative length of the current time horizon $\frac{t_{i+1}}{T}$. This is to ensure that the difference in the number of shifts within the same municipality is not already large in the first partial plans, as this would be restrictive in later iterations. We also scale the minimum number of shifts $\underline{s}$ in constraints (ROHP.h). This prevents us from assigning unnecessary shifts in the first iterations and also encourages an even distribution of shifts over the whole time horizon.

Algorithm 8 summarizes our rolling horizon approach using the above MILP. Note that our algorithm might fail to compute a solution and returns nothing in this case (line 12). This is especially if a partial plan F_i renders the following subproblems infeasible. However, it is also possible that we are not able to compute a solution for a subproblem within a given time limit although there exists one. If this happens before the last iteration, then we proceed with the following iteration using the previous partial plan (line 14). The subsequent problem

Algorithmus 8 : Rolling horizon heuristic for the computation of out-of-hours plans.

Input : An OHP instance and a sequence of days $t_1 < \cdots < t_\ell \in [T]$

Output : An out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$ or $\emptyset$

```
Initialize F_0 : ∅ → 2^P and t_{ℓ+1} = T
for i ∈ [ℓ] do
    Try to compute solution x ∈ {0,1}^{P×[t_{i+1}]} to ROHP
    if found solution then
        if i = ℓ then
            Define F : [T] → 2^P with F(t) = {p ∈ P | x_pt = 1} for all t ∈ [T]
            return F
        else
            Define F_i : [t_i] → 2^P with F_i(t) = {p ∈ P | x_pt = 1} for all t ∈ [t_i]
    else
        if i = ℓ then
            return ∅
        else
            Define F_i : [t_i] → 2^P with F_i(t) = F_{i-1}(t) if t ∈ [t_{i-1}] and F_i(t) = ∅ otherwise
```

might actually be easier to solve despite the extended planning interval and the lack of fixed shifts from the unsolved subproblem. This is because the additional freedom in the planning of the longer period can enable an easier municipality-balancing of the plan. In fact, the existence of a municipality-balanced solution that does not use the violation variables β_m is crucial for the tractability of the subproblems. If such a solution does not exist, then we observe much larger integrality gaps, since optimal continuous solutions tend to not violate the municipality-balancing.

7.3.3 Deletion of Non-Coverage Shifts

A potential problem of Algorithm 8 is that we fix shifts in the partial plans F_i that are not for the coverage but solely for the municipality-balancing and minimum number of shifts constraints. By fixing these *non-coverage shifts*, we neglect the possibility to satisfy the corresponding constraints in a later iteration by assigning shifts that are actually useful for the coverage. As a result, we obtain an inefficient final plan F that assigns more shifts than necessary. To resolve this problem, we extend our approach with an additional step in which we delete non-coverage shifts from the partial plans F_i.

We aim for deleting as many shifts as possible while still satisfying all constraints but the ones for the municipality-balancing and minimum number of shifts. Since we only consider shifts on days $t \in S_{F_i}(p)$ for $p \in \mathcal{P}$, we do not have to consider conflict constraints (ROHP.d) and periods of rest constraints (ROHP.e), as they are always satisfied due to the feasibility of F_i. Therefore, we are only left with the covering constraints and can formulate the deletion of non-coverage shifts as a covering problem in which we want to retain a minimum number of

shifts. Covering problems are $\mathcal{NP}$-hard in general, but the problem of deleting non-coverage shifts remains tractable in our practical case. This is due to the fact that all days can be considered independently, as we have no more linking constraints, and the number of shifts per day is relatively small. In our case study, we will handle the deletion problem in iteration $i \in [\ell - 1]$ by directly solving the following *Non-Coverage Deletion Problem*

$$\min \sum_{p \in \mathcal{P}} \sum_{t \in [t_i]} x_{pt} \tag{NCDEL.a}$$

$$\text{s.t.} \sum_{p \in C(m)} x_{pt} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [t_i] \tag{NCDEL.b}$$

$$\sum_{p \in P(m)} x_{pt} \geq d\,(m,t) \qquad \forall m \in \mathcal{M}, t \in [t_i] \tag{NCDEL.c}$$

$$x_{pt} = 0 \qquad \forall p \in \mathcal{P}, t \in [T] \setminus S_{F_i}\,(p) \tag{NCDEL.d}$$

$$x \in \{0,1\}^{\mathcal{P} \times [t_i]} \tag{NCDEL.e}$$

We solve the above problem after computing F_i in line 9 of Algorithm 8. Afterwards, we update the partial plan with $F_i\,(t) = \{p \in \mathcal{P} | x_{pt} = 1\}$ for all $p \in \mathcal{P}$ for an optimal solution $x \in \{0,1\}^{\mathcal{P} \times [t_i]}$ of NCDEL. The partial plan F_i then only assigns shifts that are necessary for the coverage of residents. However, the partial plan F_{i+1} computed in the next iteration should be relatively easy to balance, since the remaining shifts in F_i were originally planned with the municipality-balancing and minimum number of shifts constraints in mind. In contrast, if we would neglect these constraints completely in the first iterations when solving ROHP, then we would probably not be able to fulfill them later on.

We will see in our caste study in the next section that the deletion of non-coverage shifts not only leads to more efficient plans with a lower number of shifts but also reduces the computation time of the rolling horizon approach. This is because the deletion actually yields more possibilities to achieve municipality-balance in the next iteration. This typically results in more tractable subproblems, as explained in the previous section.

7.4 Case Study

In this section, we study the planning of the out-of-hours service for a real-world instance from the area North Rhine in Germany. We first describe the setting and basic parameters of our test instance. Afterwards, we assess the performance of our solution approaches from the previous section. Finally, we analyze the effect of different input parameters for our test instance and discuss the computed plans regarding their coverage properties and the distribution of out-of-hours shifts.

All tests are implemented in Java 8 and performed on a Linux machine with an Intel® Core™i7-3770 CPU @ 3.4 GHZ with 32 GB RAM. We use CPLEX version 12.6.3 [55] with default settings to solve MILPs.

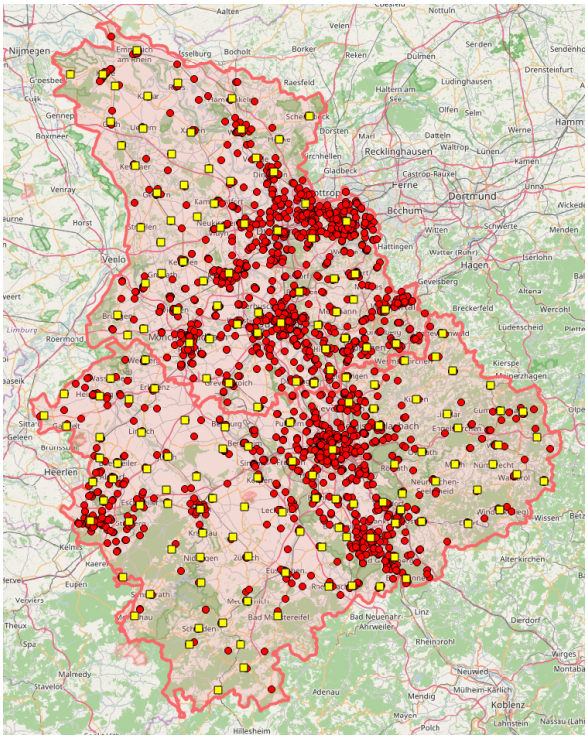


Figure 7.3. Geographical distribution of pharmacies (red dots) and municipalities (yellow squares) in the area North Rhine.

Disclaimer After conducting and publishing the following experiments, we observed that Gurobi version 9.5.0 [52] is capable of solving the instance considered in Section 7.4.3 with an optimality tolerance of 0.15% within 16,071 seconds (4.5 hours). The computed primal bound is at 30,139 shifts and the dual bound at 30,093. For this, we use the superpharmacy formulation SOHP but not the rolling horizon. This constitutes a major improvement over the performance of CPLEX version 12.6.3, which is not able to solve the problem directly for the whole time horizon. Nevertheless, this new observation does not render the rolling horizon algorithm obsolete: While Gurobi requires 15,808 seconds to compute any solution, our approach can compute almost optimal solutions within under 1,000 seconds. This is crucial in a decision support setting, where decision-makers expect fast results when computing plans with different input parameters. Furthermore, the rolling horizon approach will be useful in the next chapter, where we consider harder instances for which a direct approach yields no solution.

7.4.1 Setting of the Case Study

Our case study is based on the planning of the out-of-hours service for the year 2017 in the area of North Rhine in Germany. Accordingly, our time horizon is $[T] = [365]$. The Chamber of Pharmacists North Rhine provided a set of $|\mathcal{M}| = 165$ municipalities and $|\mathcal{P}| = 2291$ pharmacies that existed in October 2016, see Figure 7.3. The spectrum of municipalities ranges from large cities such as Cologne in the mid-east of North Rhine, containing about

Table 7.1. Numbers of municipalities and pharmacies as well as parameters by categories.

	total	large city	medium town	small town	rural municipality
municipalities	165	22	101	34	8
pharmacies	2,291	1,408	799	70	14
cover radius δ^{cov}	-	10 km	15 km	20 km	30 km
period of rest r	-	15 days	7 days	7 days	5 days
conflict distance δ^{con}	-	2 km	4 km	4 km	7 km

one million residents and 244 pharmacies, to rural municipalities in the Eifel region in the south-west, containing a few thousand residents and often only one pharmacy. The distances $\delta : (\mathcal{P} \cup \mathcal{M})^2 \to \mathbb{R}_{\geq 0}$ on the road network and affiliations $M(p) \in \mathcal{M}$ of pharmacies $p \in \mathcal{P}$ are fixed by the geographical properties of the instance.

The demands $d(m,t)$ are set individually by the Chamber of Pharmacists North Rhine for each municipality $m \in \mathcal{M}$ and day $t \in [T]$. Demands are zero year-round for 117 out of the 165 municipalities and go up to 9 shifts per day for the city of Cologne. The cover radii $\delta^{\text{cov}}(m)$ are defined via a classification of the municipalities $m \in \mathcal{M}$ into four categories: large cities, medium-sized towns, small towns, and rural municipalities. The same holds true for the conflict distances $\delta^{\text{con}}(p)$ and periods of rest $r(p)$ of pharmacies $p \in \mathcal{P}$, which are defined implicitly via the category of the corresponding municipality $M(p)$. For each category, the Chamber of Pharmacists North Rhine defines the three parameters based on legal restrictions or experience from prior planning periods. Table 7.1 shows the parameters and the number of municipalities for each category as well as the number of pharmacies within the municipalities belonging to the corresponding category.

The conflict distances range from 2 km for larger cities, where pharmacies are typically close to each other, to 7 km for rural areas. Conversely, the periods of rest range from 15 days in larger cities to 5 days in rural areas, where shifts usually have to be distributed among few pharmacies. The cover radii range from 10 km for larger cities to 30 km for rural municipalities. The Chamber of Pharmacists North Rhine adjusts these cover radii for some municipalities in their planning of the out-of-hours service. On the one hand, radii are lowered for municipalities covered by many pharmacies in order to improve the quality of coverage. On the other hand, radii of some smaller towns in remote areas are extended in order to obtain a plan that satisfies the periods of rest. We decide to retain the default radii for all but three municipalities in remote areas. For these, we increase the cover radii by up to 10%, since otherwise the number of covering pharmacies would be too small for guaranteeing the periods of rest. Overall, we increase the radius for fewer municipalities and to a lesser extent than the Chamber of Pharmacists. This strengthens compliance with the regulations for the out-of-hours service, which the Chamber of Pharmacists is not able to fully achieve, as their planning is not based on mathematical optimization.

The minimum number of shifts per pharmacy $\underline{s} \in \mathbb{Z}_{\geq 0}$ and the balancing coefficient $b \in \mathbb{Z}_{\geq 0}$ are not predefined. We choose $\underline{s} = 10$, which is roughly the minimum number of shifts in the real plan for 2017. Furthermore, we choose $b = 1$, since the Chamber of Pharmacists North Rhine aims to equally split the burden for all pharmacies within one municipality. That is, the difference in the number of shifts assigned to two pharmacies within the same municipality cannot exceed one.

7.4.2 Performance of Solution Approaches

As mentioned before, the original formulation OHP becomes very large for our test instance. In fact, CPLEX is not even able to solve the continuous relaxation in the root node of the branching tree within a time limit of one day. Nevertheless, CPLEX' primal heuristics find a feasible solution that assigns 32,451 shifts. We show in the following that the approaches from Section 7.3 are more reliable and lead to better results in less time.

The aggregation of pharmacies into superpharmacies, as proposed in Section 7.3.1, reduces the size of the MILP considerably. We can partition the set of $|\mathcal{P}| = 2291$ pharmacies into $|\mathcal{P}^s| = 1745$ superpharmacies, thus reducing the number of binary decision variables by almost a quarter. Here, 1,414 pharmacies $p \in \mathcal{P}$ constitute their own superpharmacy $\{p\} \in \mathcal{P}^s$ and 877 pharmacies can be aggregated into 331 superpharmacies of size greater than one. The largest superpharmacy consists of 9 pharmacies. Table 7.2 demonstrates the impact of the aggregation by showing the size of the constraint matrices for both MILPs, OHP and SOHP, after CPLEX performed its preprocessing.

The reduced size of the formulation results in CPLEX being able to solve the continuous relaxation of the SOHP within 42,955 seconds (11.9 hours). From the continuous solution, we deduce that a feasible out-of-hours plan assigns a total of at least 30,078 shifts. This is only 7.3% below the number of shifts in the plan computed for the OHP in 24 hours. Surprisingly, we do not obtain a feasible integer solution within a time limit of one day when solving the SOHP. Although this may suggest that the aggregation of pharmacies is counterproductive for the computation of plans, we will see that the SOHP performs well together with the rolling horizon approach.

In order to test the rolling horizon approach, we first define a penalty parameter $B \in \mathbb{Z}_{\geq 0}$ for the violation of municipality-balancing constraints in ROHP. The penalty should be large enough such that an optimal solution to ROHP always satisfies the municipality-balancing

Table 7.2. Size of constraint matrices after CPLEX' preprocessing.

formulation	rows	columns	non-zero entries
OHP	4,181,984	836,380	22,658,004
SOHP	1,060,723	637,036	14,630,106

constraints if there exists a balanced solution. This can in theory be achieved by setting B to an upper bound on the number of shifts that can be assigned, e.g., $B = |\mathcal{P}|\, T$. Then every solution that violates a municipality-balancing constraint has a larger objective value than any balanced solution. However, setting B that large is undesirable from a numerical point of view. In particular, if all solutions to ROHP are unbalanced, then the objective value would be dominated by the balancing penalty. Thus, almost all solutions with a minimum violation would be within the relative optimality tolerance, which leaves no incentive to minimize the number of shifts after minimizing the violation. We therefore choose $B = 1000$, which is in our experience large enough such that optimal solutions to ROHP are balanced, if possible.

Next, we partition the time horizon $[T]$ into intervals for the rolling horizon approach. We choose a *number of breakpoints* $\ell \leq T - 1$ and split the time horizon into $\ell + 1$ intervals of equal length (except for rounding) by defining breakpoints $t_i = \left\lfloor \frac{iT}{\ell+1} \right\rfloor$. The intervals are then $[T] = \{1, \dots, t_1\} \uplus \{t_1 + 1, \dots, t_2\} \uplus \dots \uplus \{t_\ell + 1, \dots, T\}$. Note that we solve ℓ subproblems ROHP, since the last interval $\{t_\ell + 1, \dots, T\}$ is already considered as a look-ahead in the ℓ-th iteration of our rolling horizon approach. Therefore, choosing $\ell = 1$ corresponds to solving OHP directly as one MILP. We test our rolling horizon approach for numbers of breakpoints $\ell \in [51]$, yielding a minimum interval length of one week for $\ell = 51$.

We set a total time limit of 10,000 seconds for the computation of all subproblems in the rolling horizon approach. This reflects, on the one hand, that the planning of the out-of-hours service is not on an operational level, and therefore does not have to be finished within few seconds. On the other hand, the computation should not require too much time, since our algorithms are intended as a decision support. We expect that decision-makers have to perform several computations during the planning process, all with different input parameters and possibly tailored constraints for special cases among the pharmacies and municipalities.

We split the time limit among the subproblems ROHP to consider as follows: Assume that we are currently in iteration $i \in [\ell]$ and let τ be the total computation time spent for the previous subproblems. Then we set the time limit for the current subproblem to $\frac{10000-\tau}{\ell+1-i}$, that is, we split the remaining time $10000 - \tau$ evenly among the $\ell + 1 - i$ remaining subproblems. Accordingly, the time limit for each subproblem is at least $\frac{10000}{\ell}$ and unspent time in earlier iterations can be used for subsequent iterations. If we are not able to compute a solution for a subproblem ROHP within the time limit, then we fix no shifts in the current iteration and proceed with the next subproblem, as described in Section 7.3.2. In case we obtain a solution but cannot solve the subproblem to optimality, we fix the plan provided by the best solution found. Given the heuristic nature of our algorithm, the fixing of non-optimal plans is not critical. This is especially if we afterwards delete non-coverage shifts, as in Section 7.3.3, and thus improve the efficiency of the plan. For the same reason, we are not interested in proving optimality for ROHP in each iteration. We therefore raise the relative optimality tolerance from the default of 0.01% to 0.1%. This allows for a faster termination of subproblems, and thus saves time for later subproblems that might be harder to solve.

Using the setup above, we compute out-of-hours plans with the following four variants of the rolling horizon approach:

ROL	is the plain rolling horizon approach without aggregation into superpharmacies (cf. Section 7.3.1) and deletion of non-coverage shifts (cf. Section 7.3.3).
ROL-Super	is the rolling horizon approach with aggregation into superpharmacies but without deletion of non-coverage shifts.
ROL-Delete	is the rolling horizon approach with deletion of non-coverage shifts but without aggregation into superpharmacies.
ROL-SuperDelete	is the rolling horizon approach with aggregation into superpharmacies and deletion of non-coverage shifts.

Exact computational results for all four variants and numbers of breakpoints $\ell \in [51]$ are given in Table A.1 in Appendix A. The plots in Figure 7.4 give an overview on the total number of shifts assigned, the sum of municipality-balancing violations $\sum_{m \in \mathcal{M}} \beta_m$, and the computation time required to solve the subproblems ROHP. A gap in the plot of a variant indicates that we did not obtain a solution in the final iteration of the rolling horizon approach for the corresponding number of breakpoints ℓ. Additionally, solutions that violate municipality-balancing constraints are shown with an unfilled circle.

We observe high computation times and relatively high numbers of shifts for small ℓ. This is especially for the two variants without the aggregation into superpharmacies, which both yield no plan at all for four runs each with $\ell \leq 9$. The aggregation into superpharmacies results in easier subproblems, and thus improves the reliability of our approach. Although most plans computed with $\ell \leq 10$ violate municipality-balancing constraints, ROL-Super and ROL-SuperDelete at least return an out-of-hours plan for all $\ell \in [51]$. Apart from a speed-up in computation time, the aggregation yields no structural advantage, since both SOHP and OHP are equivalent. This results in solutions of similar quality once the subproblems ROHP are easy enough to be solved without aggregation: ROL and ROL-Super yield similar results for $\ell \geq 23$; for ROL-Delete and ROL-SuperDelete, this already holds for $\ell \geq 13$.

Even more important than the aggregation into superpharmacies is the deletion of non-coverage shifts. We already mentioned in Section 7.3.3 that fixing fewer shifts leads to more freedom in the municipality-balancing of subsequent plans, and thus results in easier subproblems. In fact, ROL-Delete is often faster than ROL-Super. Moreover, the unfixing of non-coverage shifts allows for more efficient solutions, which is especially important for high numbers of breakpoints ℓ. Enforcing balance in each iteration yields many non-coverage shifts that add up over the course of the rolling horizon approach for ROL and ROL-Super. Deleting these shifts makes ROL-Delete and ROL-SuperDelete less sensitive to the choice of ℓ and generally provides a stable performance.

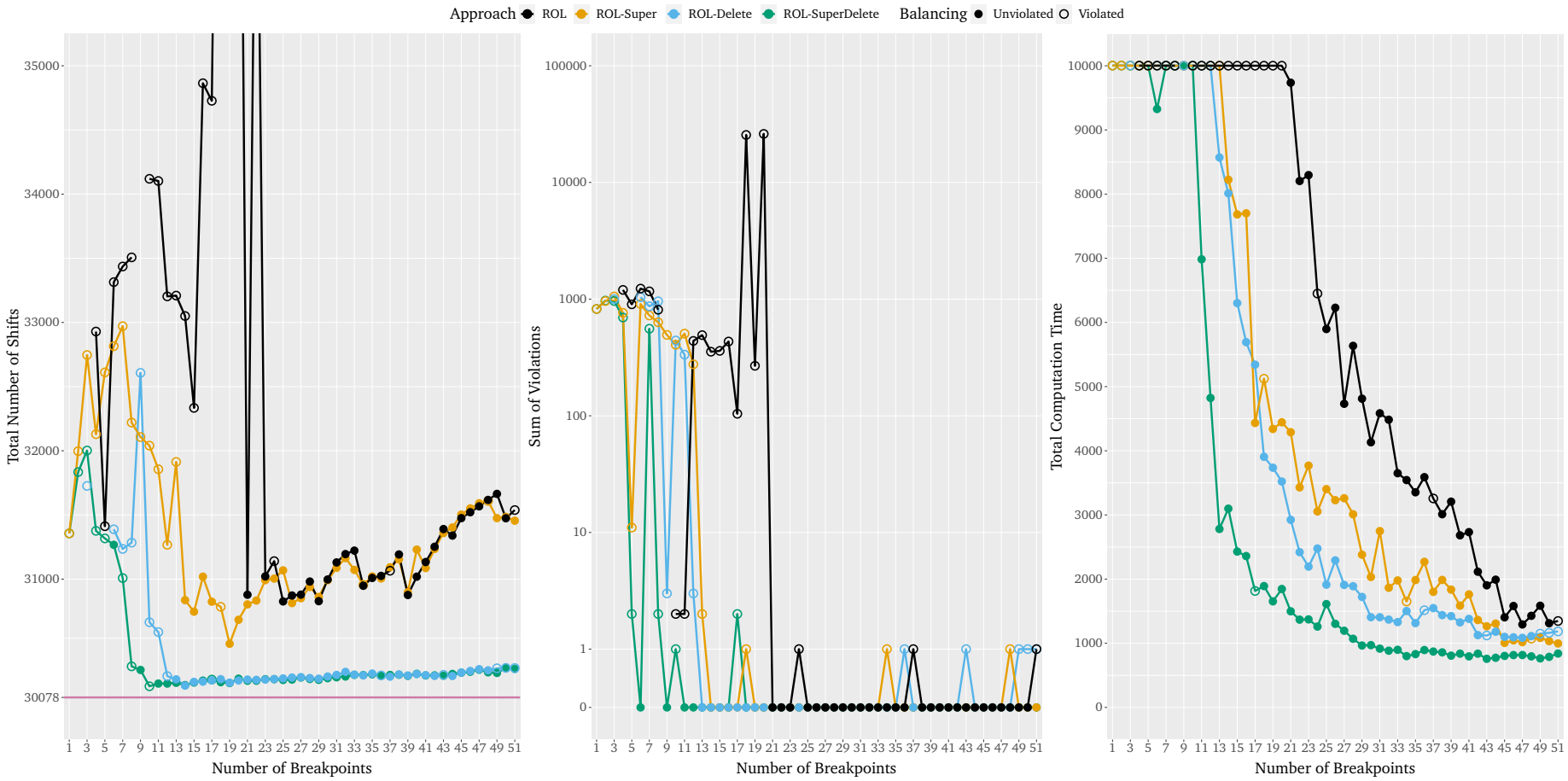


Figure 7.4. Computational results for different variants of the rolling horizon approach using different numbers of breakpoints. Solutions that violate municipality-balancing constraints are displayed as unfilled circles. **Left:** total numbers of shifts assigned compared to the LP lower bound 30,078. **Middle:** sum of violations of municipality-balancing constraints, scaled logarithmically. **Right:** total computation time required to solve subproblems ROHP.

ROL-Delete and ROL-SuperDelete not only compute better plans compared to ROL and ROL-Super but even almost optimal solutions. Both compute their best plan for $\ell = 14$, with 30,170 shifts for ROL-Delete and 30,174 shifts for ROL-SuperDelete. This yields an optimality gap of 0.3% when compared to the lower bound of 30,078 shifts provided by the continuous relaxation. Furthermore, we have a reduction of 10.1% in the number of shifts compared to the real plan of the Chamber of Pharmacists North Rhine in 2017, which assigns 33,574 shifts and has a lower compliance with planning constraints.

We conclude that our test instance is hard to solve with a direct approach but has a fairly low integrality gap of at most 0.3%. This makes the subproblems relatively easy to solve once they are small enough. Our fastest approach ROL-SuperDelete is thus able to compute plans within less than 1,000 seconds for $\ell \geq 29$, while the computation of the best plan for $\ell = 14$ requires 3,098 seconds. Furthermore, the structure of our planning problem with the loose temporal links between distant days allows for computing nearly optimal solutions with our rolling horizon approach.

7.4.3 Influence of Input Parameters

In the following, we study the influence of the minimum number of shifts $\underline{s} \in \mathbb{Z}_{\geq 0}$ and balancing coefficient $b \in \mathbb{Z}_{\geq 0}$. These parameters are particularly interesting for discussion, as they are critical to the fairness of a plan but are not specified by any regulations for the planning of the out-of-hours service. We will first analyze the impact of adjusting $\underline{s}$ and b on the performance of our algorithms and the number of shifts on an aggregate level. In the following section, we will then study specific plans to explore the possibilities and limitations of planning a fair out-of-hours service by adjusting these parameters.

We test the influence of the minimum number of shifts by considering the same setting as in the previous section except that we choose $\underline{s} \in [15]_0$. We solve the resulting problems by using the best performing variant of our rolling horizon approach from the last section, namely ROL-SuperDelete. We use a number of $\ell \in \{11, 23\}$ breakpoints for each instance, yielding $\ell + 1$ intervals with an approximate length of one month or half a month, respectively. ROL-SuperDelete performed well for $\ell \geq 11$ and worse for $\ell \leq 10$ in the previous section, which makes $\ell = 11$ an interesting number of breakpoints. Furthermore, ROL-SuperDelete showed a stable performance with low computation times around $\ell = 23$, and thus should compute reliably good results for the adjusted parameters. We denote with **ROL11-SuperDelete** and **ROL23-SuperDelete** the approach with 11 and 23 breakpoints, respectively. We also compute a lower bound on the number of shifts by solving the continuous relaxation of SOHP. Exact computational results are reported in Table A.2 in Appendix A.

Figure 7.5 gives an overview of the computational results. The left-hand graphic shows that the minimum number of shifts $\underline{s}$ has a large effect on the total number of shifts and that the corresponding constraints (OHP.h) are an actual restriction even for small values of $\underline{s}$. The increase from $\underline{s} = 0$ to $\underline{s} = 1$ in the value of the lower bound shows that an optimal

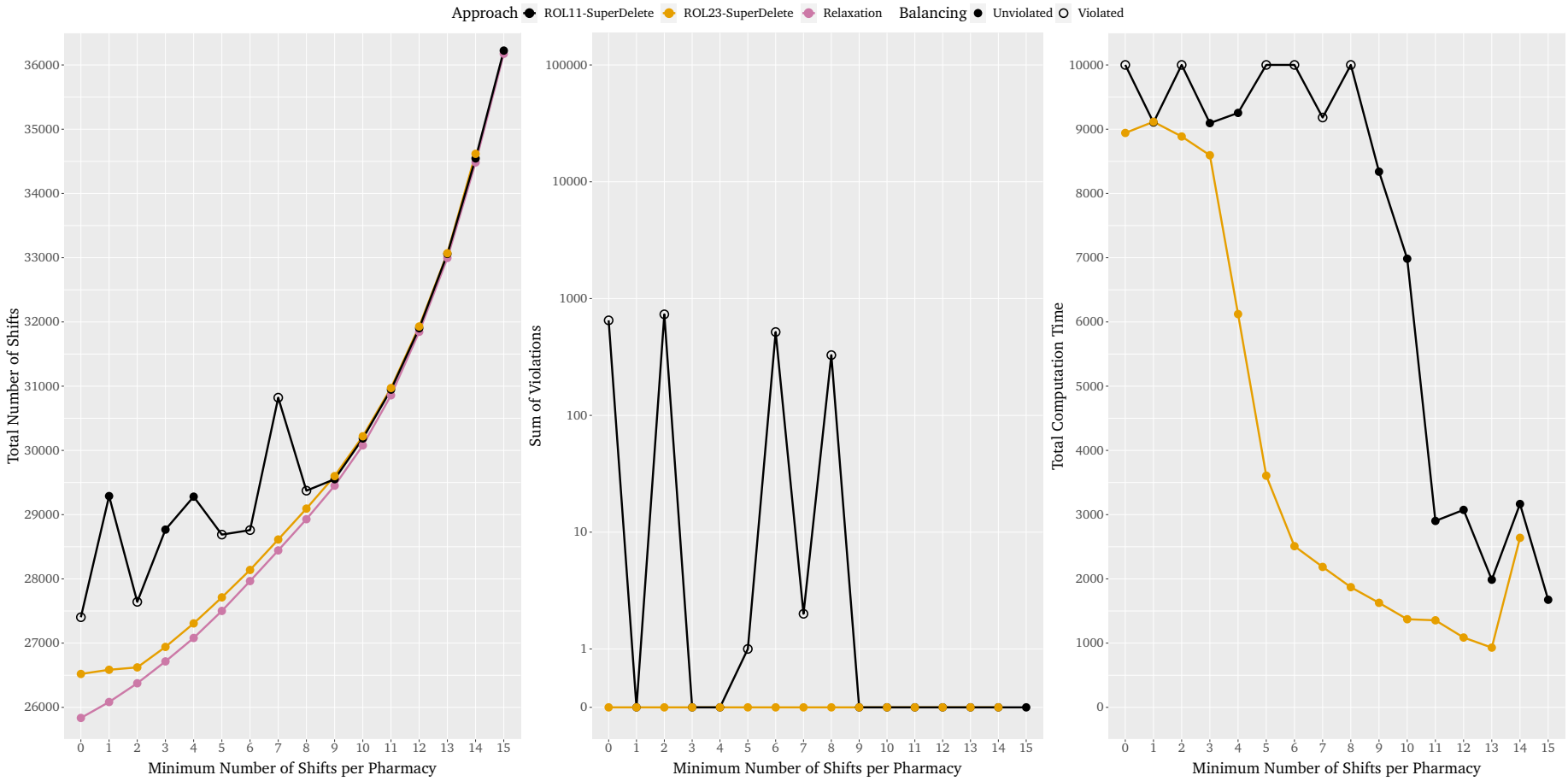


Figure 7.5. Computational results for different minimum numbers of shifts $\underline{s}$. Solutions that violate municipality-balancing constraints are displayed as unfilled circles. **Left:** total numbers of shifts assigned compared to LP lower bounds. **Middle:** sum of violations of municipality-balancing constraints, scaled logarithmically. **Right:** total computation time required to solve subproblems ROHP.

continuous solution assigns less than one shift to some pharmacies. While we cannot verify this for optimal integer solutions, we can deduce that an optimal plan for $\underline{s} \in \{0, 1, 2\}$ assigns fewer than three shifts to some pharmacies, since the best known plans for $\underline{s} \in \{0, 1, 2\}$ assign fewer shifts than the lower bound for $\underline{s} = 3$. As $\underline{s}$ grows, its impact on the number of shifts also increases, since the constraints (OHP.h) become restrictive for more pharmacies. This is demonstrated by the increasing slope of the number of shifts for growing $\underline{s}$.

The minimum number of shifts $\underline{s}$ has not only an effect on the total number of shifts but also on the performance of our algorithms. The right-hand graphic in Figure 7.5 shows that the required computation time of our approach tends to increase for decreasing $\underline{s}$. Moreover, ROL11-SuperDelete is not able to reliably compute good balanced solutions for $\underline{s} \leq 8$. This can most likely be attributed to a higher integrality gap for small $\underline{s}$. Assuming that our computed solutions are almost optimal, we see that the integrality gap is roughly 2.58% for $\underline{s} = 0$ but only 0.14% for $\underline{s} = 15$. This is also supported by the number of fractional variables in the computed optimal continuous solutions. For example, the optimal continuous solution computed within the first iteration of ROL11-SuperDelete for $\underline{s} = 0$ consists of 8,863 integer-infeasible variables, that is, variables that should be integer but take fractional values. In contrast, the solution for $\underline{s} = 15$ only consists of 1,608 integer-infeasible variables. Intuitively speaking, the continuous relaxation has less freedom to assign fewer shifts if the number of shifts is significantly bounded from below.

Although we generally observe lower computation times for higher $\underline{s}$, ROL23-SuperDelete is not able to compute a plan for $\underline{s} = 15$. This is because the small intervals sometimes do not provide enough freedom to meet the highly restrictive minimum number of shifts constraints (ROHP.h), which leads to infeasible subproblems ROHP. CPLEX then fails to find a solution for the following larger subproblem, resulting in a propagation of unsolved problems until the final iteration. This shows the limits of our approach when using a high number of breakpoints ℓ for more restricted instances.

To test a variation in the balancing coefficient, we set the minimum number of shifts back to $\underline{s} = 10$ and choose $b \in \{0, \ldots, 5\}$ to be the maximum permitted difference in the number of shifts within municipalities. We use the same algorithms as above and report exact computational results in Table A.3 in Appendix A, while Figure 7.6 gives an overview.

We see in Figure 7.6 that adjusting the balancing coefficient only has a minor impact on the optimal continuous solutions. Here, the number of shifts range from 30,110 for $b = 0$ to 29,980 for $b = 5$. The same holds for the computed integer solutions for $b \in [5]$, where we achieve optimality gaps of 0.36% and lower. However, there is a notable difference between $b = 1$ and $b = 0$. ROL11-SuperDelete computes a solution with highly violated municipality-balancing constraints, which can therefore not be compared to the continuous solution. The plan computed by ROL23-SuperDelete is almost balanced but assigns 985 shifts more than the lower bound, yielding a gap of 3.17%. It is not surprising that enforcing an equal number of shifts within a municipality results in a significant increase in the total number of shifts. This is already due to rounding effects. For example, if two pharmacies

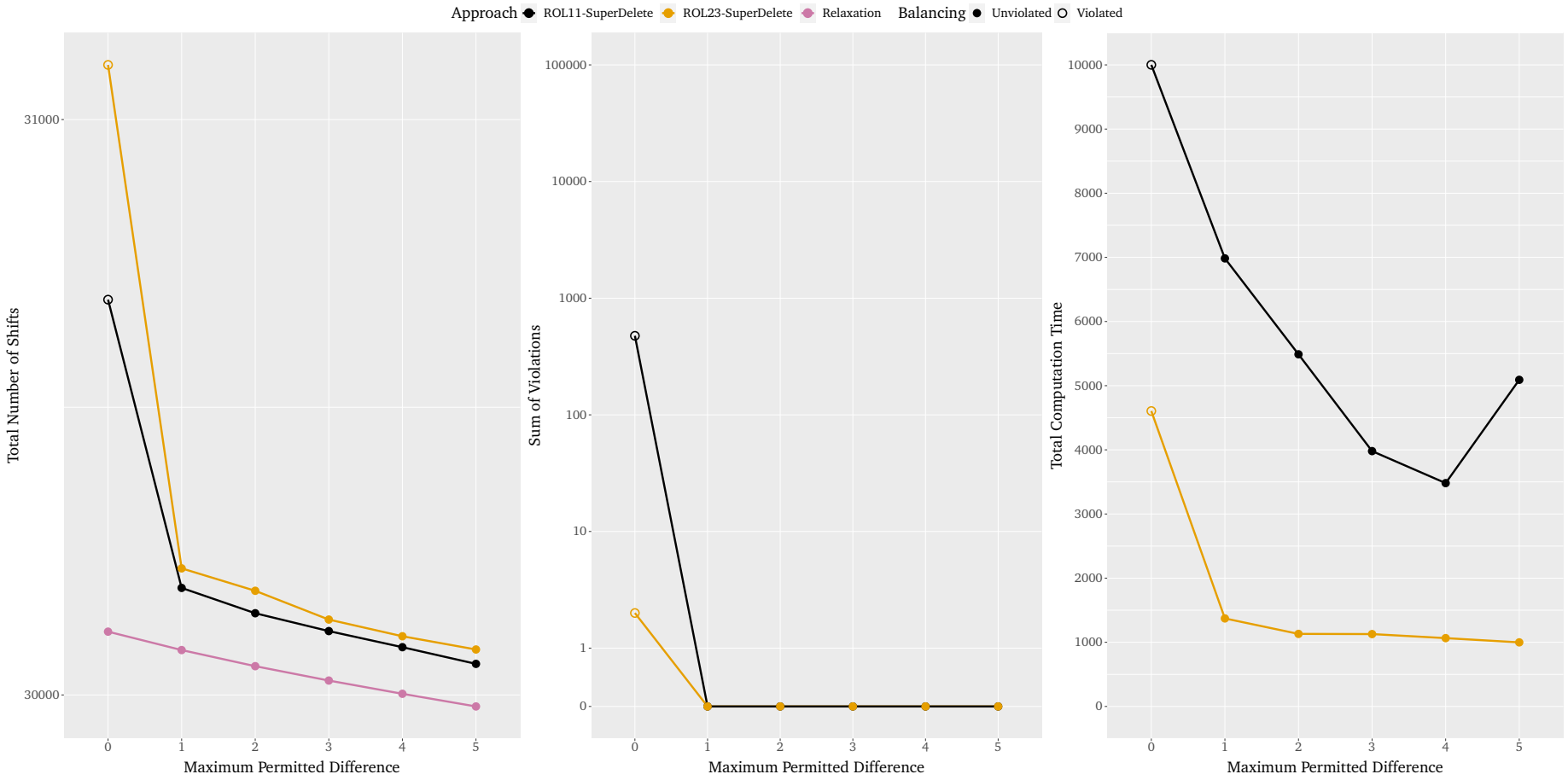


Figure 7.6. Computational results for different balancing coefficients b. Solutions that violate municipality-balancing constraints are displayed as unfilled circles. **Left:** total numbers of shifts assigned compared to LP lower bounds. **Middle:** sum of violations of municipality-balancing constraints, scaled logarithmically. **Right:** total computation time required to solve subproblems ROHP.

need to be assigned 21 shifts in total, then both will receive 11. This is also the reason for the larger integrality gap, as a continuous solution can assign 10.5 shifts each.

The trend in the computation time of ROL11-SuperDelete suggests that the subproblems ROHP become easier to solve for high b. However, we see nearly no effect on the computation time of ROL23-SuperDelete. We thus conclude that the cost of municipality-balancing is small for both the total number of shifts and the computation time, provided that we do not choose $b = 0$.

7.4.4 Discussion of Out-of-Hours Plans

We have seen in the last sections that we are able to compute plans that assign fewer shifts and have a higher compliance with regulations than the real plan used by the Chamber of Pharmacists North Rhine. However, this does not necessarily imply that our plans are suitable in practice. The main concerns here are fairness in terms of the distribution of shifts among pharmacies and coverage in terms of the mean travel distance to the nearest out-of-hours pharmacy. In the following, we analyze selected plans computed in Section 7.4.3 with respect to these two criteria.

We start with an evaluation of the plans on a global level and later consider fairness aspects locally. We already noted in the previous section that a variation of the balancing coefficient $b \in [5]$ has nearly no effect on the global level. Furthermore, choosing $b = 0$ is not reasonable from a practical point of view, as the number of additional shifts compared to $b = 1$ is disproportionate to the enhanced balancing. Hence, for our evaluation on the global level, we consider a fixed balancing coefficient $b = 1$ together with a varying minimum number of shifts $\underline{s} \in \{0, 5, 10, 15\}$. For each value of $\underline{s}$, we consider the plan with the best objective value computed by ROL11-SuperDelete and ROL23-SuperDelete in Section 7.4.3.

First, we analyze the distribution of out-of-hours shifts among pharmacies. As a reminder, Table 7.4 shows the total number of shifts assigned to all pharmacies. The left-hand plot in Figure 7.7 gives a more detailed view by showing the cumulative distribution of the number of shifts. The plan for $\underline{s} = 0$ assigns no shifts at all to 6.4% of all pharmacies and fewer than five shifts to 18.8% of all pharmacies. This is consistent with the observation from the previous section that the minimum number of shifts constraints (OHP.h) are an

Table 7.4. Total number of shifts assigned and the mean distance from the center of the municipalities to the nearest out-of-hours pharmacies over the course of the year in meters. For each value $\underline{s}$, we consider the solution with the best objective value computed in the previous section.

	$\underline{s} = 0$	$\underline{s} = 5$	$\underline{s} = 10$	$\underline{s} = 15$	real plan 2017
total number of shifts	26,520	27,712	30,186	36,224	33,574
mean distance to nearest pharmacy	8,705	8,456	8,030	7,559	7,204

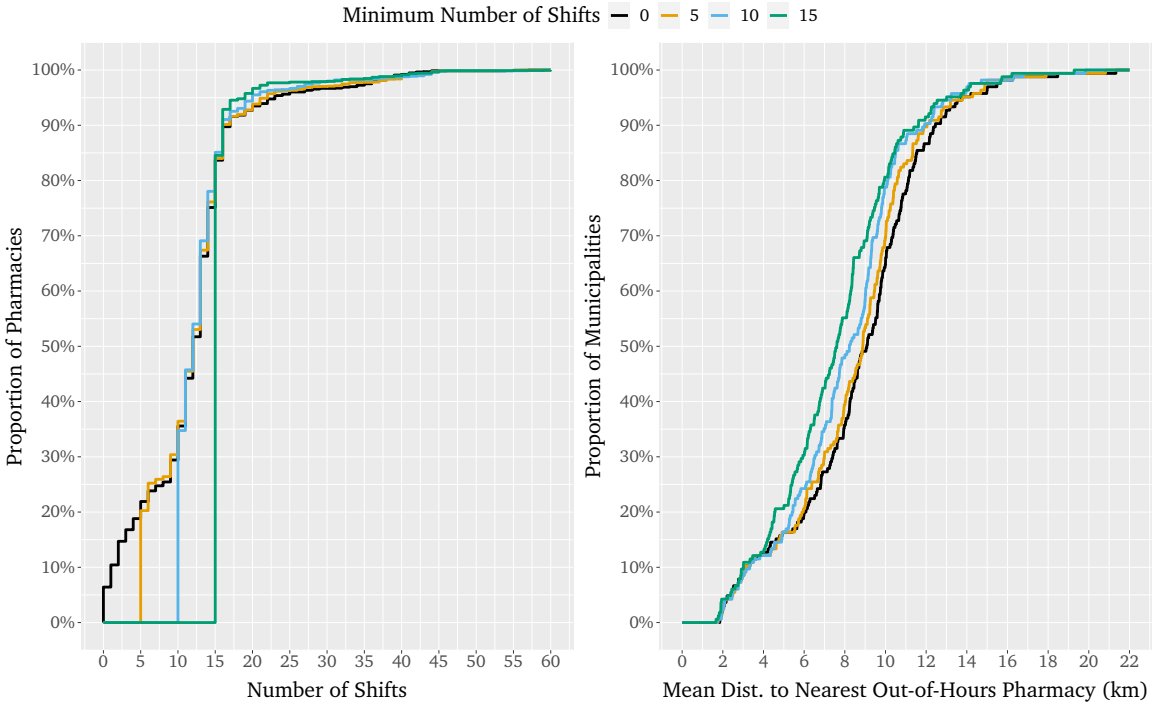


Figure 7.7. Computational results for different minimum numbers of shifts $\underline{s}$. **Left:** cumulative distribution of numbers of shifts assigned to pharmacies. **Right:** cumulative distribution of mean distance from municipality centers to their nearest out-of-hours pharmacy.

actual restriction for the planning. We will see later that most pharmacies without any shifts are located in the vicinity of larger cities. Municipalities bordering cities are often covered year-round by pharmacies within the city, which need to be assigned shifts to meet the city's demand constraints (OHP.c). Then pharmacies outside the city are assigned not more than the minimum number of shifts for the sake of efficiency. Note that we observe this effect although the plans meet the municipality-balancing constraints with $b = 1$. Hence, we have whole municipalities in which the included pharmacies are assigned almost no shifts in the plan for $\underline{s} = 0$.

For the reason above, it is not surprising that the introduction of a minimum number of shifts $\underline{s} > 0$ results in many pharmacies being assigned exactly $\underline{s}$ out-of-hours shifts. For $\underline{s} = 5$, we assign 20.3% of the pharmacies exactly $\underline{s}$ shifts. For $\underline{s} = 10$, this proportion increases to 34.7% and for $\underline{s} = 15$ even to 84.5%. Enforcing a minimum of $\underline{s} = 15$ shifts for all pharmacies reduces the efficiency of the plan, with 36.6% more shifts compared to $\underline{s} = 0$, but it also promotes fairness. While 6.5% of the pharmacies are assigned more than 20 shifts for $\underline{s} = 0$, this proportion is nearly halved for $\underline{s} = 15$ to only 3.3% of all pharmacies. This shows that the shifts can be distributed more evenly in order to relieve pharmacies with relatively many shifts. Unfortunately, raising the minimum number of shifts does not relieve the pharmacies with the highest burden: The number of pharmacies with at least 40 shifts is similar for all plans. We therefore require other tools if we want to reduce this peak in the number of shifts.

The necessity to develop tools for relieving highly burdened pharmacies can be deduced from the distribution of shifts in the real plan of 2017. While a general comparison between our plans and the real plan is not appropriate due to the adjusted cover radii used by the Chamber of Pharmacists North Rhine, the real plan indicates that there is a political will to limit the maximum number of shifts: The cover radii in rural areas are extended such that the number of shifts assigned to any pharmacy is at most 40. We will see later that this can be achieved without extending the cover radii.

In order to evaluate the quality of coverage of a municipality $m \in \mathcal{M}$, we consider the mean distance that residents need to travel from the center of the municipality to the nearest out-of-hours pharmacy over the whole time horizon, i.e., $\overline{\delta}_F(m) = \sum_{t \in [T]} \frac{\min\{\delta(m,p) | p \in F(t)\}}{T}$ for an out-of-hours plan $F : [T] \to 2^{\mathcal{P}}$. We argued in Section 4.6.2 to aggregate values with the shifted geometric mean instead of the arithmetic mean, since the latter is mainly influenced by large values and almost neglects differences in smaller values. We now argue that emphasizing longer distances with the arithmetic mean is reasonable, since the difference between 10 km and 20 km is more relevant than the difference between 1 km and 2 km in emergency situations.

Table 7.4 shows the global quality of coverage by stating the mean of $\overline{\delta}_F(m)$ over all municipalities $m \in \mathcal{M}$. We see that a higher minimum number of shifts leads to a better coverage, with a reduction of 13.2% in the mean distance for $\underline{s} = 15$ compared to $\underline{s} = 0$. However, a comparison with the real plan shows that the additional shifts implied by $\underline{s} = 15$ are not distributed efficiently, as we observe longer mean distances despite assigning more shifts. This effect is due to the indifference of our model with respect to the day to which we assign additional non-coverage shifts. Given this indifference, we might assign multiple non-coverage shifts to pharmacies within the same area on the same day instead of distributing them over the time horizon. We will return to this problem in Section 8.6.2.2, showing that the mean distance can be reduced without increasing the number of shifts.

The right-hand plot in Figure 7.7 shows the impact of the minimum number of shifts on the distribution of the mean distances $\overline{\delta}_F(m)$ for all municipalities $m \in \mathcal{M}$. The improvement resulting from an increase of $\underline{s}$ is primarily observable for municipalities with a mean distance of more than 4 km. This is because municipalities with a lower mean distance are usually larger cities with a positive demand $d > 0$ that requires an assignment of several shifts to pharmacies within the cities. Thus, while the additional shifts are not assigned efficiently, they at least benefit the municipalities with a lower quality of coverage. For example, the maximum mean distance $\max\left\{\overline{\delta}_F(m) \middle| m \in \mathcal{M}\right\}$ is 21.3 km in the plan for $\underline{s} = 0$ and 19.8 km in the plan for $\underline{s} = 10$.

Figure 7.8 gives an explanation for the improvement in the maximum mean distance by showing the geographic distribution of out-of-hours shifts assigned in the plans for $\underline{s} \in \{0, 10\}$. The municipality of Dahlem, which has the highest mean distance, is highlighted with a circle in the very south of the planning area. Dahlem is not only a rural municipality but also located at the border of the planning area, and is thus surrounded by few pharmacies that

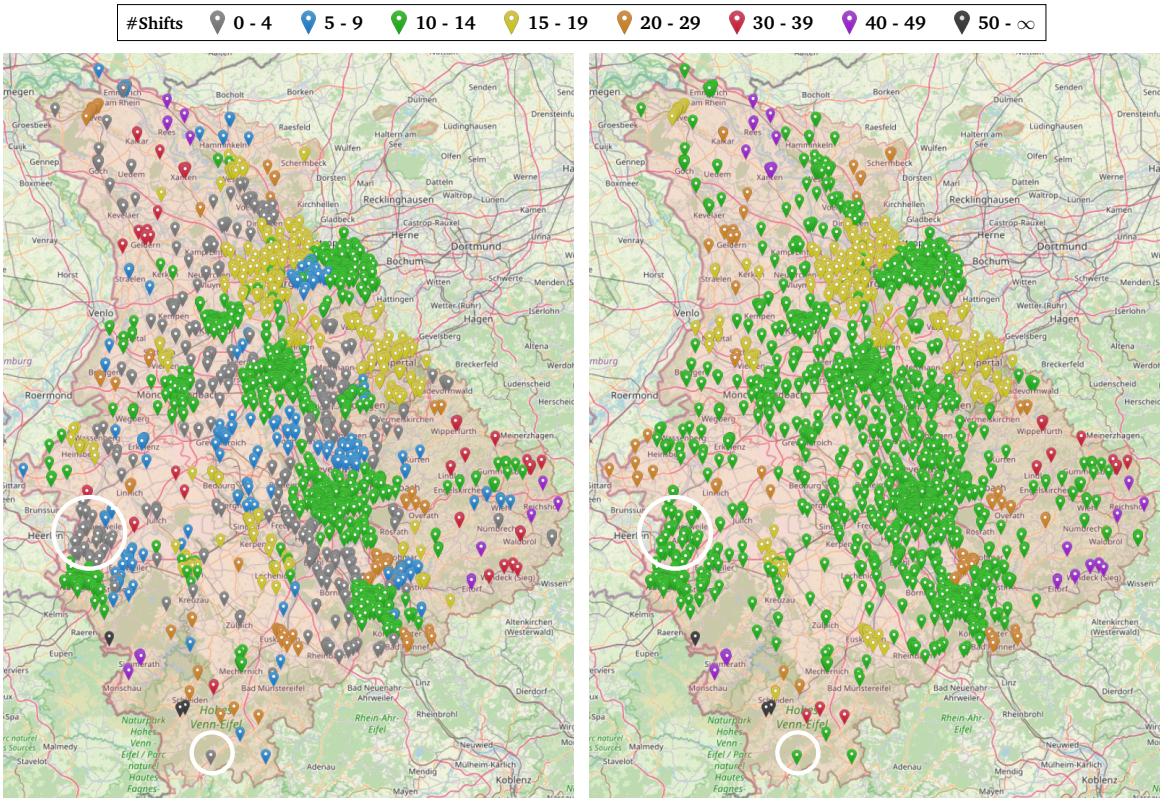


Figure 7.8. Geographical distribution of shifts. **Left:** plan for $\underline{s} = 0$. **Right:** plan for $\underline{s} = 10$.

are subject to our planning. Moreover, for the sake of efficiency, the pharmacy in Dahlem is assigned only three shifts in the plan for $\underline{s} = 0$, as it can cover few municipalities due to its proximity to the border. Hence, requiring a minimum number of shifts yields a significant reduction of the mean distance to the nearest out-of-hours pharmacy.

We already noted above that pharmacies in the vicinity of cities are often assigned few shifts, as their municipalities are already covered by pharmacies within the city. This can, for example, be observed in the municipalities north of the city of Aachen, which are highlighted in Figure 7.8 in the west of the planning area. Here, we see many pharmacies being assigned fewer than five shifts in the plan for $\underline{s} = 0$. This not only results in a worse quality of coverage for the residents but also in many shifts for some of the surrounding pharmacies. To the north and east of the highlighted pharmacies, there are several pharmacies to which we assign 35 to 39 shifts. In the plan for $\underline{s} = 10$, these previously highly burdened pharmacies are assigned fewer than 15 shifts. A minimum number of shifts thus can prevent a concentration of many shifts on some pharmacies.

The pharmacies with few shifts in the north of Aachen are in contrast to the highly burdened pharmacies in the south of Aachen. Here, we assign up to 60 shifts per pharmacy in order to cover the rural municipality of Monschau. As discussed above, these pharmacies are unaffected by an increase of the minimum number of shifts, since all pharmacies covering Monschau are already assigned at least ten shifts in the plan for $\underline{s} = 0$. Nevertheless, it is possible to achieve an assignment of at most 40 shifts per pharmacy by redistributing the

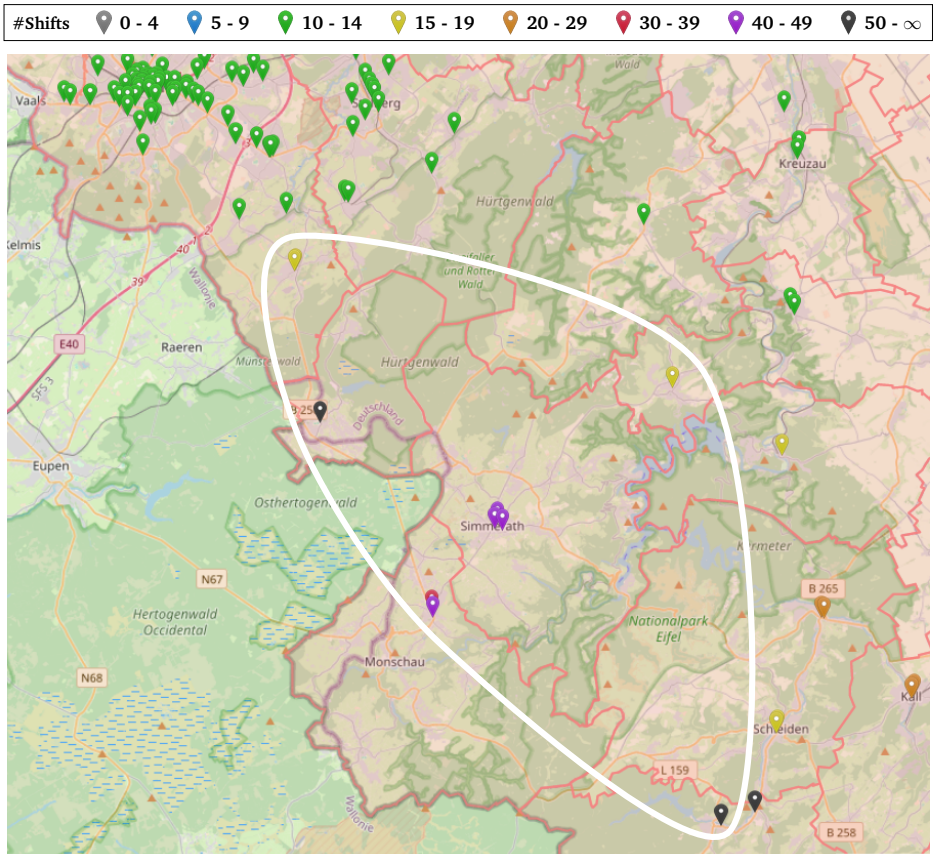


Figure 7.9. Distribution of shifts to pharmacies around Monschau for the plan with $b = 5$.

shifts more evenly. Unfortunately, an even distribution is not optimal for the OHP due to the municipality-balancing. To see this, consider Figure 7.9, in which the pharmacies covering Monschau are highlighted. The map shows the assignment of shifts to pharmacies for the plan with minimum number of shifts $\underline{s} = 10$ and balancing coefficient $b = 5$, that is, we have a rather loose municipality-balancing with a permitted difference of five shifts. The pharmacies in the northwest and northeast covering Monschau are assigned 16 shifts. This is more than in the plan with $\underline{s} = 10$ and $b = 1$ but not enough to substantially relieve the other pharmacies. However, assigning more shifts to them would necessitate assigning additional shifts to all pharmacies within their municipalities, which is not efficient with respect to the total number of shifts.

We conclude that, despite its importance for local fairness, municipality-balancing can promote a high difference in the number of shifts across municipalities. Moreover, Figure 7.9 shows that even the highly burdened pharmacies that are not restricted by municipality-balancing are assigned a range of 39 to 60 shifts. This highlights that our model has no incentive to achieve a globally fair plan. Regardless of the increased efficiency, our plans are therefore not yet suitable for practice, as only a fair plan would be accepted by pharmacists. We will resolve this issue in the next chapter by developing new tools for computing plans that are much fairer and at the same time similarly efficient.

7.5 Conclusion

In this chapter, we introduced the optimization problem OHP in order to compute efficient plans for the out-of-hours service for pharmacies. We proved that finding a feasible solution to the OHP is strongly $\mathcal{NP}$-complete in theory, but we also proposed MILP-based approaches that are capable of computing good solutions in practice. In our aggregation approach, we identify pharmacies that are equivalent within our model and combine them into one artificial superpharmacy. The aggregation significantly reduces the size of our MILP formulation and furthermore breaks symmetries arising from permuting out-of-hours shifts of equivalent pharmacies. We showed that the aggregation is exact, as shifts assigned to a superpharmacy can later be distributed among the aggregated pharmacies without losing optimality. In our rolling horizon approach, we decompose the OHP into multiple tractable subproblems by sequentially computing plans for an iteratively extending fraction of the time horizon. The rolling horizon approach is a heuristic that offers no approximation guarantee, as we fix shifts for a fraction of the time horizon without considering later days. However, the approach fits the structure of the OHP, where decisions taken at the beginning of the time horizon only have a minor impact on decisions at the end of the time horizon. Together with the aggregation of equivalent pharmacies, the rolling horizon approach enables us to compute nearly optimal plans in short computation time for the real-world instance considered in our case study.

The plans computed in our case study for the out-of-hours service in the area North Rhine show the potential of an optimization-based planning: In comparison with the real plan used by the Chamber of Pharmacists North Rhine, our plans assign roughly 10% fewer shifts while maintaining a higher compliance with regulations for the out-of-hours service. Decision-makers may choose to benefit from the increased efficiency of an optimization-based planning by actually reducing the number of shifts, and thus relieving pharmacies. Alternatively, they may decide to apply stricter planning constraints in order to improve the coverage of residents to an extent that the number of shifts assigned is the same as before. Here, the reliability of our approach allows for analyzing the implications of adjusting parameters and applying custom constraints.

In our case study, we analyzed the implications of adjusting the parameters for the balancing within municipalities and the minimum number of shifts that needs to be assigned to each pharmacy. We observed that a strong municipality-balancing can have a negative impact on the global fairness of the plan by promoting inter-municipal differences in the number of shifts. A higher minimum number of shifts can mitigate this effect to some extend, but it is not sufficient for ensuring a fair assignment of shifts among the pharmacies with the highest burden. We address this problem in the next chapter by proposing an approach that integrates fairness directly into the planning of the out-of-hours service.

8 Integrating Fairness into the Planning of the Out-of-Hours Service

We have seen in the last chapter that we can efficiently construct out-of-hours plans that assign relatively few shifts and are balanced within municipalities. However, we also observed variations in the number of shifts of pharmacies in different municipalities, even if the pharmacies are in similar locations and almost equal in terms of the coverage of municipalities. In this chapter, we focus on a globally fair planning that reduces inter-municipal differences while maintaining efficiency regarding the total number of shifts.

8.1 Fairness Concepts for the Planning of the Out-of-Hours Service

Deciding whether an out-of-hours plan is fair is not trivial, as the notion of fairness is subjective and influenced by individual perspectives. It is therefore crucial to clearly define fairness criteria that are widely accepted among pharmacists. Such criteria should be intuitive to understand in order to raise transparency, and thus acceptance, of the planning process. In particular, fairness criteria should not be tailored individually for each pharmacy but should be universally applicable for the whole planning.

Probably the simplest fairness criterion is to require a *fully egalitarian plan* in which all pharmacies are assigned the same number of shifts. Applying this criterion seems tempting, as we would place an equal burden on all pharmacies. However, the equal treatment of all pharmacies comes at the cost of many shifts that would otherwise be unnecessary. This is because the number of shifts of all pharmacies is determined by few pharmacies that need to be assigned many shifts in order to cover a municipality. For the instance from our case study in Section 7.4, where one municipality can be covered by only 10 pharmacies, we would need to assign $\left\lceil \frac{365}{10} \right\rceil = 37$ shifts to each pharmacy. If we neglect that this is not even possible due to the period of rest in cities, then this amounts to 84,767 shifts for the 2,291 pharmacies in the area North Rhine. In contrast, our plans computed in the last chapter assign roughly 30,000 shifts and the real plan of the Chamber of Pharmacists North Rhine assigns roughly 33,000 shifts.

One can argue that equality possesses an inherent value that justifies lower efficiency. However, the large gap between the fully egalitarian plan and the real plan shows that the pursuit of equality must in practice not come at the expense of efficiency at all cost. This implies that the concept of fairness we seek for planning the out-of-hours service does not equal equality. Hence, approaches that minimize an inequality measure, such as the maximum difference in the number of shifts, are not suitable for our cause. Although inequality measures can be combined with efficiency aspects in order to obtain a trade-off between equality and efficiency [97], we argue that these approaches are also not suitable. This is because the effect of the trade-off is not trivial to understand, which reduces transparency for pharmacists. Moreover, defining the trade-off requires a difficult-to-establish consensus on when efficiency takes precedence over equality.

Although we discard equality as the goal for our planning, it is still an intuitive principle of fairness that similar entities should be treated similarly. This implies that pharmacies covering the same municipalities should be assigned the same number of shifts (± 1). We are thus left with the question to what extend pharmacies that cover different municipalities can be treated differently. If some pharmacies cover nearly the same municipalities, then their burden should be similar in some way. In particular, we should not assign an exceptionally high number of shifts to a pharmacy when it is also possible to redistribute some shifts to less burdened pharmacies. This is reflected by the intuitive goal of relieving the pharmacies with the highest burden as much as possible. Minimizing the maximum burden is a popular approach for incorporating fairness in an optimization model [60, 97]. This is not only because this goal can be easily incorporated in the objective function of optimization problems but also because it coincides with the *difference principle* that was philosophically justified by Rawls [85]. Rawls bases this principle on a thought experiment in which a just society is designed by entities that are behind a "veil of ignorance". This veil conceals all personal characteristics in order to ensure an unbiased decision process. Rawls argues that such entities would design a society in favor of the least advantaged entity, as they may find themselves in a disadvantaged position once the veil is lifted. For the planning of the out-of-hours service, this implies that pharmacists would agree to relieve pharmacies in disadvantaged locations as much as possible if they would be unaware of their own location.

Rawls' difference principle should in practice be extended, because it otherwise can result in solutions that are not Pareto-optimal. The principle only gives an incentive to reduce the burden of the worst-off entity but neglects all others. Therefore, even the fully egalitarian out-of-hours plan discussed above would be optimal with respect to the difference principle. A natural and popular extension is *lexicographic fairness*. For this, we not only minimize the highest burden but afterwards also the second highest burden, then the third highest burden, and so forth. Formally, let $\mathcal{X}$ be a set of solutions and $\phi : \mathcal{X} \to \mathbb{R}^n$ a function reflecting the individual burden of $n \in \mathbb{Z}_{>0}$ entities. We rate solutions by applying a *leximax* comparison. Given two solutions $x, x' \in \mathcal{X}$, we first sort the burden for both solutions individually from maximum to minimum, i.e., $\phi_{i_1}(x) \geq \cdots \geq \phi_{i_n}(x)$ and $\phi_{j_1}(x') \geq \cdots \geq \phi_{j_n}(x')$. Afterwards, we compare the sorted burden vectors $(\phi_{i_1}(x), \ldots, \phi_{i_n}(x))$ and

$(\phi_{j_1}(x'), \ldots, \phi_{j_n}(x'))$ lexicographically. We say that x is *leximax smaller* than x' if there exists an index $k \in [n]$ with $\phi_{i_k}(x) < \phi_{j_k}(x')$ and $\phi_{i_\ell}(x) = \phi_{j_\ell}(x')$ for all $\ell \in [k-1]$. In this case, we write $x \prec_{\text{lex}} x'$. If x is leximax smaller than x' or both are rated equal, i.e., $\phi_{i_k}(x) = \phi_{j_k}(x')$ for all $k \in [n]$, then we write $x \preceq_{\text{lex}} x'$. The problem $\text{leximaxmin}\{\phi(x) | x \in \mathcal{X}\}$ then consists in finding a solution $x \in \mathcal{X}$ such that $x \preceq_{\text{lex}} x'$ holds for all $x' \in \mathcal{X}$. We say that such a solution is a *leximax minimal* solution.

A leximax minimal solution $x \in \mathcal{X}$ is Pareto-optimal, because a dominating solution $x' \in \mathcal{X}$ with $\phi(x') \leq \phi(x)$ and $\phi_i(x') < \phi_i(x)$ for some $i \in [n]$ would be leximax smaller. Therefore, lexicographic fair optimization is theoretically efficient while yielding solutions with a high level of fairness. Karsu and Morton [60] even mention in their review on inequity averse optimization that "lexicographic approaches are very inequality averse and considered by some studies as the "most equitable" solution." For the planning of the out-of-hours service, the lexicographic approach reflects the intuitive concept of assigning more shifts to less burdened pharmacies if this enables us to relieve other pharmacies with a higher number of shifts. As a consequence, pharmacies that are similarly important for the coverage are assigned a similar number of shifts. However, dissimilar pharmacies can differ in their number of shifts, as we do not assign unnecessary shifts solely for the sake of equality.

Although lexicographic fair optimization yields Pareto-optimal solutions, the high level of fairness comes in practice sometimes at the cost of reduced overall efficiency. This can be the case if reducing the burden $\phi_i(x)$ is very "expensive" for some $i \in [n]$. For example, assume that we need to allocate money for medical treatment, and there is one person having an incurable disease that can be slightly alleviated with an expensive therapy. A leximax minimal solution treats this worst-off person, which yields a small benefit while leaving little budget for other treatments [97]. Fortunately, such effects tend to be less relevant for the planning of the out-of-hours service. This is because the removal of a shift of one pharmacy can often be compensated by assigning few additional shifts to other pharmacies. Therefore, relieving highly burdened pharmacies is usually not costly with respect to the total number of shifts. However, efficiency can decline when applying a lexicographic fair planning together with municipality-balancing. For example, assigning one more shift to one of the 244 pharmacies in the city of Cologne can imply the need for assigning 243 additional shifts to retain balance within the city. This highlights once more that there is a general conflict between municipality-balancing and inter-municipal fairness. We will later propose approaches to resolve this conflict and show that we can obtain fair plans based on the widely accepted principles of lexicographic fairness while maintaining efficiency.

8.2 Lexicographic Fair Planning of the Out-of-Hours Service

We incorporate the concept of lexicographic fairness into the planning of the out-of-hours service by defining the burden $\phi_p(x) = \sum_{t\in[T]} x_{pt}$ of pharmacy $p \in \mathcal{P}$ as the total number of shifts in a solution $x \in \{0,1\}^{\mathcal{P}\times[T]}$. The *Lexicographic Fair Out-of-Hours Planning Problem* then reads

$$\text{leximaxmin} \left(\sum_{t\in[T]} x_{pt} \right)_{p\in\mathcal{P}} \tag{LOHP.a}$$

$$\text{s.t.} \sum_{p\in C(m)} x_{pt} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [T] \tag{LOHP.b}$$

$$\sum_{p\in P(m)} x_{pt} \geq d(m,t) \qquad \forall m \in \mathcal{M}, t \in [T] \tag{LOHP.c}$$

$$x_{pt} + x_{p't} \leq 1 \qquad \forall \{p,p'\} \in \mathcal{C}, t \in [T] \tag{LOHP.d}$$

$$\sum_{t'=t}^{t+r(p)} x_{pt'} \leq 1 \qquad \forall p \in \mathcal{P}, t \in [T - r(p)] \tag{LOHP.e}$$

$$\sum_{t\in[T]} x_{pt} \leq b_{pp'} \left(\sum_{t\in[T]} x_{p't} \right) \qquad \forall p, p' \in \mathcal{P} \tag{LOHP.f}$$

$$\sum_{t\in[T]} x_{pt} \geq \underline{s} \qquad \forall p \in \mathcal{P} \tag{LOHP.g}$$

$$x \in \{0,1\}^{\mathcal{P}\times[T]}. \tag{LOHP.h}$$

Note that we still include constraints (LOHP.g), which ensure that every pharmacy is assigned a minimum number of shifts, since the lexicographic planning alone is not able to guarantee that all pharmacies participate in the service. Furthermore, we define more general balancing constraints (LOHP.f), where a *balancing function* $b_{pp'} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ defines for a pair of pharmacies $p, p' \in \mathcal{P}$ the maximum number of shifts that can be assigned to pharmacy p based on the number of shifts assigned to pharmacy p'.

It may seem counter-intuitive to define balancing constraints in a model that ensures fairness via the lexicographic objective. However, when only considering lexicographic fairness, then neighboring pharmacies can differ significantly in the number of shifts if one is critical for covering a rural municipality while the other is not. The constraints are therefore intended to ensure balancing due to political reasons. Admittedly, this political balancing is not ideal, as we would prefer a planning where the coverage model and the lexicographic fairness intrinsically result in a balancing of neighboring pharmacies. The coverage can, however, later be refined to smooth the assignment of shifts for neighboring pharmacies.

We require some technical properties of the balancing functions $b_{pp'} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that are not very restrictive in practice and mostly quite natural. For this, let $X_p = \sum_{t\in[T]} x_{pt}$ in

the following be the number of shifts assigned to a pharmacy $p \in \mathcal{P}$. First, we assume that $b_{pp'}$ is greater than or equal to the identity function, i.e., $b_{pp'}(X_{p'}) \geq X_{p'}$ holds for all $X_{p'} \in \mathbb{R}_{\geq 0}$. This assumption is natural, as the constraints (LOHP.f) should ensure that the difference $X_p - X_{p'}$ is bounded but should not force $X_p < X_{p'}$. Second, we assume that $b_{pp'}$ is non-decreasing, which is intuitive, as a higher number of shifts $X_{p'}$ should not result in a smaller upper bound on X_p. Third, we assume that $b_{pp'}$ is continuous. This requirement is not necessary from a modeling point of view, but we argue that most practical balancing functions are continuous anyway. From a technical point of view, continuity ensures that the set of solutions to the continuous relaxation of the LOHP is closed, and thus compact. Lastly, we assume that $b_{pp'}$ is concave. Then the left-hand side of the constraint $X_p - b_{pp'}(X_{p'}) \leq 0$ is convex for $X_p, X_{p'} \in \mathbb{R}_{\geq 0}$, and thus the set of solutions to the continuous relaxation of LOHP is also convex. The concavity of $b_{pp'}$ is not restrictive in practice, as we are not interested in increasing growth rates of $b_{pp'}(X_{p'})$ for increasing $X_{p'}$. Stated otherwise, the maximum permissible difference $X_p - X_{p'}$ should not increase disproportionately for higher $X_{p'}$. We will later use affine functions for $b_{pp'}$ that have the required properties and are defined with respect to the distance between the pharmacies p, p'.

Computing leximax minimal solutions is in general not trivial due to the non-linear objective function. This is especially true for the LOHP, for which it is even hard to compute any solution. We are not aware of an algorithm that would be tractable for the LOHP but sketch a theoretical approach in the following. One can solve general problems $\operatorname{leximaxmin}\{\phi(x) | x \in \mathcal{X}\}$ with some burden function $\phi : \mathcal{X} \to \mathbb{R}^n$ by iteratively determining the value of the highest burden, then the second highest burden, and so forth. For this, we first compute a solution with a minimum maximal burden, that is, we solve $\min\{w | \exists x \in \mathcal{X} : \phi_i(x) \leq w \ \forall i \in [n]\}$. Let w_1 be the optimal value of this problem. Then we know that for a leximax minimal solution x^*, there exists an index $i_1 \in [n]$ with $\max(\phi(x^*)) = \phi_{i_1}(x^*) = w_1$. We can fix $\phi_{i_1}(x) = w_1$ and compute the second highest burden by solving $\min\{w | \exists x \in \mathcal{X} : \phi_{i_1}(x) = w_1,\ \phi_i(x) \leq w \ \forall i \in [n] \setminus \{i_1\}\}$. The new optimal value w_2 can again be fixed for some $\phi_{i_2}(x)$. Repeating these steps eventually yields the leximax minimal solution x^*. However, we obviously do not know which burden $\phi_{i_j}(x)$ to fix in iteration $j \in [n]$. A remedy for this issue is to enumerate all possible choices $i_j \in [n] \setminus \{i_1, \ldots, i_{j-1}\}$ in an enumeration tree. This essentially results in enumerating all $n!$ possible permutations of the burden $\phi(x)$. However, the approach can be converted into a branch and bound algorithm by pruning nodes in the enumeration tree for which fixing $\left(\phi_{i_1}(x), \ldots, \phi_{i_j}(x)\right) = \left(w_{i_1}, \ldots, w_{i_j}\right)$ is infeasible or $\left(w_{i_1}, \ldots, w_{i_j}, -\infty, \ldots, -\infty\right)$ is worse than $\phi(x')$ for an already computed solution $x' \in \mathcal{X}$.

Note that it is crucial to consider all remaining indices $i_j \in [n] \setminus \{i_1, \ldots, i_{j-1}\}$ for fixing $\phi_{i_j}(x) = w_j$. Contrary to a statement in [97], it is not sufficient to only consider indices $i_j \in [n] \setminus \{i_1, \ldots, i_{j-1}\}$ for which the computed burden $\phi_{i_j}(x)$ in iteration $j \in [n]$ is equal to w_j. This can be seen when considering the set of solutions $\mathcal{X} = \{(2,3), (3,1)\}$ with the burden function $\phi(x) = x$ being the identity. In this case, $(2,3)$ is optimal for the first subproblem, in which we compute $w_1 = 3$. When only considering the possibility of fixing

$\phi_2(x) = 3$, and thus $x_2 = 3$, then we obtain $(2, 3)$ at the end of the algorithm. This is despite that $(3, 1)$ is the unique optimal solution.

While the approach above is not practicable for our purpose, it at least shows the existence of leximax minimal solutions. All subproblems in which we compute w_j have an optimal solution if $\mathcal{X} \subseteq \mathbb{R}^m$ is compact and the burden function ϕ is continuous. This yields the following statement, which was already mentioned, for example, in [87].

Corollary 36. *Let $\mathcal{X} \subseteq \mathbb{R}^m$ be a non-empty, compact set and $\phi : \mathcal{X} \to \mathbb{R}^n$ be a continuous function. Then there exists an optimal solution to* $\operatorname{leximaxmin}\{\phi(x) | x \in \mathcal{X}\}$.

In particular, the corollary shows that there exists a leximax minimal solution to the LOHP if there exists any feasible solution.

Ogryczak and Śliwiński [79] show that leximax minimization problems can also be solved by a more direct approach without enumerating possible permutations. They propose two reformulations of leximax minimization problems in which the ordering of the burden is already included in the feasible solutions. It is then sufficient to solve a classical lexicographic optimization problem. This can be done similarly to the approach above where we compute w_j but without the need to guess which burden $\phi_{i_j}(x)$ to fix. The downside, however, is that the size of the reformulation increases significantly. This increase renders the direct approaches impractical for our purposes, as the original problem for computing any out-of-hours plan is already time consuming.

Due to the lack of tractable approaches for solving the LOHP, we will consider the computation of leximax minimal plans for relaxed problems in the following sections. Remember that continuous solutions to the OHP were close to integer solutions in our case study from Section 7.4. Hence, leximax minimal solutions to a relaxed problem might give a good indication of the number of shifts each pharmacy should be assigned in a fair plan. We will therefore use this indication later as guidance for computing fair out-of-hours plans.

8.3 Min-Max Fairness

In the following, we will consider the concept of *min-max fairness*, which is closely related to lexicographic fairness. The advantage of min-max fairness over lexicographic fairness is that it is easier to handle notation-wise and also provides additional structure. This structure will later be useful for computing and proving fair solutions. We will first consider min-max fairness theoretically and then show how it can be of practical use.

Given a set of solutions $\mathcal{X} \subseteq \mathbb{R}^m$ and a burden function $\phi : \mathcal{X} \to \mathbb{R}^n$, we call $x \in \mathcal{X}$ *min-max fair* if for all $y \in \mathcal{X}$ and $i \in [n]$ with $\phi_i(y) < \phi_i(x)$, there exists an index $j \in [n]$ with $\phi_j(y) > \phi_j(x) \geq \phi_i(x)$. Intuitively speaking, a solution is min-max fair if we cannot decrease burden $\phi_i(x)$ without increasing some already greater or equal burden $\phi_j(x)$. Hence, just

like lexicographic fairness, min-max fairness aims for decreasing the burden of the most disadvantaged entities, possibly at the cost of the more advantaged. When maximizing benefits instead of minimizing burdens, then the analogous concept to min-max fairness is *max-min fairness*. In this case, a solution $x \in \mathcal{X}$ is max-min fair if we cannot increase benefit $\phi_i(x)$ without decreasing some already smaller or equal benefit $\phi_j(x)$. Many results from the literature on max-min fairness translate to min-max fairness and vice versa. In particular, a solution $x \in \mathcal{X}$ is min-max fair for a burden function $\phi : \mathcal{X} \to \mathbb{R}^n$ if and only if it is max-min fair for the benefit function $-\phi$.

Min-max (max-min) fairness is in the literature usually defined directly for a set of solutions $\mathcal{X} \subseteq \mathbb{R}^m$ without considering a burden (benefit) function [17, 84, 87]. That is, $x \in \mathcal{X}$ is min-max (max-min) fair if it is fair with respect to the identity $\phi : \mathcal{X} \to \mathbb{R}^n, x \mapsto x$. We call a set $\mathcal{X}$ *min-max achievable* (*max-min achievable*) if there exists a min-max (max-min) fair solution $x \in \mathcal{X}$. The same can be applied for lexicographic fairness, that is, $x \in \mathcal{X}$ is leximax minimal if it is leximax minimal respect to the identity. We will often omit the function ϕ and only consider fairness with respect to the set of solutions $\mathcal{X}$ to simplify notation. Many results can then be generalized to the case with a non-identity function ϕ by replacing $\mathcal{X}$ with $\phi(\mathcal{X})$. For example, there exists a min-max fair solution in $\mathcal{X}$ with respect to ϕ if $\phi(\mathcal{X})$ is min-max achievable.

Min-max achievability cannot be taken for granted, even if there exists a leximax minimal solution. To see this, consider the set of solutions $\mathcal{X} = \{(1,0), (0,1)\}$. In both solutions $(1,0)$ and $(0,1)$, we can decrease one component by increasing the other, which is not greater or equal to the decreased component. Therefore, neither solution is min-max fair, despite both being lexicographic fair. This shows that min-max fairness and lexicographic fairness are not equal, although both aim to relieve the disadvantaged. The following proposition summarizes some results of Sarkar and Tassiulas [87] and shows that min-max fairness is actually stronger than lexicographic fairness.

Proposition 37 (Sarkar and Tassiulas [87])**.** *If $x \in \mathcal{X}$ is min-max fair, then it is also leximax minimal. Furthermore, if $\mathcal{X}$ is min-max achievable, then there exists exactly one min-max fair (leximax minimal) solution.*

The proposition shows that if $\mathcal{X}$ is min-max achievable, then min-max fairness is equivalent to leximax minimality, as there only exists one fair solution that is both min-max fair and leximax minimal. We can therefore use the characterization of min-max fairness for computing leximax minimal solutions in min-max achievable sets. This raises the question of when min-max fair solutions exist. In our example above with $\mathcal{X} = \{(1,0), (0,1)\}$, the solution $(1,0)$ is not min-max fair, because we can opt for the "fairer" solution $(0,1)$, which decreases the first component without increasing an already larger or equal component. The issue, however, is that the change is not continuous in the sense that the situation is now reversed and we can apply the same argument to show that $(1,0)$ is "fairer" than $(0,1)$. This observation indicates that the discrete nature of $\mathcal{X}$, with $(1,0)$ and $(0,1)$ being *disconnected*, might be

hindering for min-max achievability. We therefore investigate whether *connected* sets are min-max achievable.

A topological space $\mathcal{X}$ is called *path-connected* if there exists a *continuous path* between all $x, y \in \mathcal{X}$, that is, there exists a continuous function $f : [0,1] \to \mathcal{X}$ with $f(0) = x$ and $f(1) = y$ [78, §24]. Now, consider $\mathcal{X} = \{(1,a), (a,1) | a \in [0,1]\}$ as a subspace of $\mathbb{R}^2$ with the standard topology (see [78, §13 and §16]). Observe that $\mathcal{X}$ is a single continuous path from $(1,0)$ via $(1,1)$ to $(0,1)$, and thus path-connected. However, there exists no min-max fair solution in $\mathcal{X}$, since $(1,0)$ and $(0,1)$ are again not min-max fair and all other solutions are dominated. Hence, connected sets are in general not min-max achievable.

Although connectivity is not sufficient for min-max achievability, the idea is a step in the right direction. For this, note that *convexity* is a stronger form of connectivity, with the continuous path between two points being the direct line. When considering $\mathcal{X} = \operatorname{conv}(\{(1,0), (0,1)\})$, then $\left(\frac{1}{2}, \frac{1}{2}\right)$ is a min-max fair solution. The idea here is that if x is fairer than y and y is fairer than x, then we can travel along the direct line from x to y to a point z that strikes a balance between x and y and is fairer than both. More generally, Radunovic and Le Boudec [84] showed for **max-min** fairness that $\phi(\mathcal{X})$ is max-min achievable if $\mathcal{X} \subseteq \mathbb{R}^n$ is compact and convex and $\phi : \mathcal{X} \to \mathbb{R}^n, x \mapsto (\phi_1(x_1), \dots \phi_n(x_n))$ is a component-wise function such that ϕ_i is continuous and strictly increasing for all $i \in [n]$. In particular, this proves that $\mathcal{X}$ itself is max-min achievable if it is compact and convex when considering ϕ as the identity. This also translates to min-max fairness, since $\mathcal{X}$ is min-max achievable if $-\mathcal{X}$ is max-min achievable, which is the case because $-\mathcal{X}$ is compact and convex. We can derive the following result on min-max achievability for the planning of the out-of-hours service.

Proposition 38. *If there exists any solution to the continuous relaxation of LOHP, then there exists a min-max fair continuous solution.*

Proof. Let $\mathcal{X} \subseteq [0,1]^{\mathcal{P} \times [T]}$ be the set of solutions to the continuous relaxation of LOHP. Then $\mathcal{X}$ is compact and convex. Furthermore, the burden functions $\phi_p(x) = \sum_{t \in [T]} x_{pt}$ for $p \in \mathcal{P}$ are linear, and therefore preserve compactness and convexity. Hence, the set of possible burdens $\phi(\mathcal{X})$ is compact and convex, and thus min-max achievable. □

Note that if our burden function was not linear, then the result of Radunovic and Le Boudec [84] would not be applicable due to the limitation to increasing, component-wise functions ϕ. First, our burden functions $\phi_p(x) = \sum_{t \in [T]} x_{pt}$ are not defined component-wise. Second, the result is for max-min fairness, and we thus needed to consider $-\phi$ instead of ϕ, which is not increasing. Although it is not of practical relevance for our particular problem, we give a generalization of the result of Radunovic and Le Boudec for theoretical interest. Here, we additionally consider functions ϕ that are not defined component-wise and replace the need for strictly increasing functions ϕ_i with a more general notion of strict monotony. To this end, we first need to define monotony of functions in higher dimension. Given a convex set $\mathcal{X}$, we call $f : \mathcal{X} \to \mathbb{R}$ *monotone* if for all $x, y \in \mathcal{X}$ with $f(x) \leq f(y)$, we have

$f(x) \leq f(z) \leq f(y)$ for all proper convex combinations $z \in \mathcal{X}$ of x and y. We call f *strictly monotone* if it is monotone and $f(x) < f(y)$ also implies $f(x) < f(z) < f(y)$.

Theorem 39. *Consider a non-empty set of solutions $\mathcal{X} \subseteq \mathbb{R}^m$ and a function $\phi : \mathcal{X} \to \mathbb{R}^n$. If $\mathcal{X}$ is convex and compact and ϕ_i is continuous and strictly monotone for all $i \in [n]$, then $\phi(\mathcal{X})$ is min-max and max-min achievable.*

Proof. Assume that the statement is true for min-max achievability. Then $-\phi(\mathcal{X})$ is also min-max achievable, because $-\phi$ is component-wise continuous and strictly monotone. The statement follows for max-min achievability, since $\phi(\mathcal{X})$ is max-min achievable if and only if $-\phi(\mathcal{X})$ is min-max achievable.

To show that $\phi(\mathcal{X})$ is min-max achievable, let $x \in \mathcal{X}$ be such that $\phi(x)$ is leximax minimal in $\phi(\mathcal{X})$. The existence of x follows from Corollary 36 due to the continuity of ϕ and the compactness of $\mathcal{X}$. Now, assume that $\phi(x)$ is not min-max fair. Then there exist $y \in \mathcal{X}$ and $k \in [n]$ such that $\phi_k(y) < \phi_k(x)$ and there exists no $j \in [n]$ with $\phi_j(y) > \phi_j(x) \geq \phi_k(x)$. In the following, we use the convexity of $\mathcal{X}$ to construct a convex combination $z \in \mathcal{X}$ of x and y such that $\phi(z)$ is leximax smaller than $\phi(x)$. This will contradict the choice of x, and thus prove the statement.

Let $z = (1-\alpha)x + \alpha y$ for an $\alpha \in (0,1)$. Since ϕ_j is monotone with respect to the definition above and z is a convex combination of x and y, we have $\phi_j(z) \leq \phi_j(x)$ for all $j \in [n]$ with $\phi_j(y) \leq \phi_j(x)$. In particular, $\phi_k(z) < \phi_k(x)$ holds due to $\phi_k(y) < \phi_k(x)$ and the strict monotony. Consider an index $j \in [n]$ with $\phi_j(x) \geq \phi_k(x)$. Then we have $\phi_j(y) \leq \phi_j(x)$, and thus $\phi_j(z) \leq \phi_j(x)$, as otherwise $\phi_j(y) > \phi_j(x) \geq \phi_k(x)$ contradicts the choice of y and k. Now, consider an index $j \in [n]$ with $\phi_j(x) < \phi_k(x)$. The continuity of ϕ_j implies that there exists a sufficiently small $\varepsilon_j \in (0,1)$ with $\phi_j((1-\varepsilon_j)x + \varepsilon_j y) < \phi_k(x)$. Moreover, the strict monotony of ϕ_j implies $\phi_j((1-\alpha)x + \alpha y) < \phi_k(x)$ for all $\alpha \in (0, \varepsilon_j]$. Choosing $\alpha = \min\{\varepsilon_j | j \in [n], \phi_j(x) < \phi_k(x)\}$ thus yields in summary

- $\phi_j(z) \leq \phi_j(x)$ for all $j \in [n]$ with $\phi_j(x) \geq \phi_k(x)$,
- $\phi_k(z) < \phi_k(x)$,
- $\phi_j(z) < \phi_k(x)$ for all $j \in [n]$ with $\phi_j(x) < \phi_k(x)$.

Assume without loss of generality that the indices are ordered such that $\phi_1(x) \geq \cdots \geq \phi_n(x)$ and either $\phi_k(x) > \phi_{k+1}(x)$ or $k = n$ holds. We then have $\phi_j(z) \leq \phi_\ell(x)$ for all $\ell \in [k-1]$ and $j \in \{\ell, \ldots, n\}$. Hence, when sorting $\phi_{i_1}(z) \geq \cdots \geq \phi_{i_n}(z)$, then $\phi_{i_\ell}(z) \leq \phi_\ell(x)$ holds for all $\ell \in [k-1]$. Furthermore, we have $\phi_j(z) < \phi_k(x)$ for all $j \in \{k, \ldots, n\}$, and thus $\phi_{i_k}(z) < \phi_k(x)$. This shows that z is leximax smaller than x and completes the proof. □

The theorem above is a proper generalization in the sense that it applies to functions ϕ that are not covered by the result of Radunovic and Le Boudec [84] and whose image $\phi(\mathcal{X})$ is non-convex. For example, consider $\mathcal{X} = [0,1]^2$ and $\phi : \mathcal{X} \to \mathbb{R}^2, x \mapsto (x - y, y^2)$, which fulfill

all criteria of the theorem. The function ϕ is not defined component-wise and the image $\phi(\mathcal{X})$ is non-convex. To see the latter, note that $\phi(1,0) = (1,0)$ and $\phi(1,1) = (0,1)$ but $\left(\frac{1}{2}, \frac{1}{2}\right) \notin \phi(\mathcal{X})$ because then we needed $y = \frac{1}{\sqrt{2}}$ and $x = \frac{1}{2} + y > 1$.

Knowing that a set $\mathcal{X}$ is min-max achievable is crucial, because it enables us to compute fair solutions much more efficiently. Algorithm 9 shows how to compute the unique min-max fair solution $x^* \in \mathcal{X}$ in this case. The general idea of the algorithm is similar to that from last section for computing leximax minimal solutions. That is, we first compute the value $w_1 = \min\{w | \exists x \in \mathcal{X} : x_i \leq w \; \forall i \in [n]\}$, which corresponds to the largest component in x^* (line 4). The min-max fair solution x^* is then in $\mathcal{X}_1' = \{x \in \mathcal{X} | x_i \leq w_1 \; \forall i \in [n]\}$ (line 5). Afterwards, we compute with $S_1 = \{i \in [n] | \exists x \in \mathcal{X}_1' : x_i < w_1\}$ the indices whose corresponding components can be below w_1, given that all other components are at most w_1 (line 6). In fact, S_1 contains exactly the indices $i \in [n]$ with $x_i^* < w_1$. We therefore fix $x_i = w_1$ for all $i \notin S_1$ (line 7) and then proceed in the next iteration by minimizing the remaining components for $i \in S_1$. In contrast to the approach from last section for computing leximax minimal solutions, we do not have to enumerate all possibilities for fixing the values w_k for some component x_i. Thus, instead of enumerating all $n!$ permutations in the worst case, we have to compute at most n values w_k and sets S_k. Here, S_k can be determined by computing $x_i' = \min\{x_i | x \in \mathcal{X}_k'\}$ for all $i \in S_{k-1}$ and checking whether $x_i' < w_k$ holds. Hence, we need to solve at most $|S_{k-1}| \leq n$ subproblems for computing S_k. We will show in the proof of Theorem 40 that there always exists an index $i \in S_{k-1}$ with $x_i' = w_k$, and thus $i \notin S_k$. This implies that we need at most n iterations until we have $S_k = \emptyset$, implying that the number of subproblems to solve within Algorithm 9 is in $\mathcal{O}(n^2)$.

Algorithmus 9 : Computation of min-max fair solutions.

Input : A min-max achievable set $\mathcal{X} \subseteq \mathbb{R}^n$
Output : The unique min-max fair vector in $\mathcal{X}$

1 Initialize $k = 0$, $S_0 = [n]$, and $\mathcal{X}_0 = \mathcal{X}$
2 **while** $S_k \neq \emptyset$ **do**
3 Update $k \leftarrow k + 1$
4 Compute $w_k = \min\{w | \exists x \in \mathcal{X}_{k-1} : x_i \leq w \; \forall i \in S_{k-1}\}$
5 Let $\mathcal{X}_k' = \{x \in \mathcal{X}_{k-1} | x_i \leq w_k \; \forall i \in S_{k-1}\}$
6 Compute $S_k = \{i \in S_{k-1} | \exists x \in \mathcal{X}_k' : x_i < w_k\}$
7 Let $\mathcal{X}_k = \{x \in \mathcal{X}_{k-1} | x_i = w_k \; \forall i \in S_{k-1} \setminus S_k\}$
8 **return** *The single element in* $\mathcal{X}_k$

Radunovic and Le Boudec [84] propose an algorithm for max-min fairness that is similar to Algorithm 9. Unfortunately, their version contains a small but significant mistake in the choice of S_k. The analog for our version would be choosing $S_k = \{i \in S_{k-1} | x_i < w_k \; \forall x \in \mathcal{X}_k'\}$, which implies that x_i is not decreased further if there exists any solution $x \in \mathcal{X}_k'$ with $x_i = w_k$. For example, consider the set of solutions $\mathcal{X} = \left\{x \in [0,2]^2 \middle| x_1 \geq 1\right\}$. Then we have $w_1 = 1$ and $\mathcal{X}_1' = \left\{x \in [0,2]^2 \middle| x_1 = 1, x_2 \leq 1\right\}$ in the first iteration. Hence, $(1,1) \in \mathcal{X}_1'$ holds and defining $S_1 = \{i \in \{1,2\} | x_i < w_1 \; \forall x \in \mathcal{X}_1'\} = \emptyset$ would terminate the algorithm with $\mathcal{X}_1 = \{(1,1)\}$, although the min-max fair solution is $(1,0)$. To overcome this issue, we prove the correctness of our algorithm in the following.

Theorem 40. *Algorithm 9 computes a min-max fair solution if $\mathcal{X}$ is min-max achievable.*

Proof. Let x^* be the unique min-max fair solution in $\mathcal{X}$. To prove that the algorithm is correct, we show that it terminates and that the final set $\mathcal{X}_k$ equals $\{x^*\}$. For this, we prove via induction for $k \in \mathbb{Z}_{\geq 0}$ that $x^* \in \mathcal{X}_k$ holds. During the induction, we will also show that $S_{k-1} \supsetneq S_k$ holds for $k \in \mathbb{Z}_{>0}$. This proves that the algorithm terminates after $\ell \in \mathbb{Z}_{>0}$ iterations with $S_\ell = \emptyset$. We then obtain $\mathcal{X}_\ell = \{x^*\}$, since we have $x^* \in \mathcal{X}_\ell$ and each component x_i is fixed in some iteration $k \in [l]$ by the definition of $\mathcal{X}_k$.

We have $x^* \in \mathcal{X} = \mathcal{X}_0$ after the initialization, which proves the induction hypothesis for $k = 0$. Now, consider an arbitrary iteration $k \in \mathbb{Z}_{>0}$ within the algorithm and assume that the induction hypothesis is true for $k-1$. We then have $x^* \in \mathcal{X}_{k-1}$ and can choose $x = x^*$ in line 4 of the algorithm, which yields $w_k \leq \max\{x_i^* | i \in S_{k-1}\}$. We show equality by proving the following:

Claim Let $x \in \mathcal{X}_{k-1}$ with $\max\{x_i | i \in S_{k-1}\} \leq \max\{x_i^* | i \in S_{k-1}\}$. We then have $x_{i^*} = x_{i^*}^*$ for all $i^* \in \operatorname{argmax}\{x_i^* | i \in S_{k-1}\}$.

For this, let $i^* \in \operatorname{argmax}\{x_i^* | i \in S_{k-1}\}$ and assume there exists a solution $x \in \mathcal{X}_{k-1}$ with $\max\{x_i | i \in S_{k-1}\} \leq x_{i^*}^*$ and $x_{i^*} < x_{i^*}^*$. Since x^* is min-max fair, there exists an index $j \in [n]$ with $x_j > x_j^* \geq x_{i^*}^*$. We then have $j \notin S_{k-1}$ due to $x_i \leq x_{i^*}^*$ for all $i \in S_{k-1}$. However, $x \in \mathcal{X}_{k-1}$ and $j \in [n] \setminus S_{k-1}$ implies that x_j has already been fixed to some $w_{k'}$ in an earlier iteration $k' \in [k-1]$. Together with $x^* \in \mathcal{X}_{k-1}$, this implies $x_j = x_j^*$, which contradicts the choice of j and proves the claim.

The claim implies that $w_k = \max\{x_i^* | i \in S_{k-1}\}$ holds, because otherwise there would exist a solution $x \in \mathcal{X}_{k-1}$ with $\max\{x_i | i \in S_{k-1}\} < \max\{x_i^* | i \in S_{k-1}\}$. Furthermore, we obtain $i^* \notin S_k$ for all $i^* \in S_{k-1}$ with $x_{i^*}^* = w_k$, since the claim implies that there exists no $x \in \mathcal{X}_k'$ with $x_{i^*} < w_k$. Thus, $S_{k-1} \supsetneq S_k$ holds and the algorithm terminates. For all indices $i \in S_{k-1}$ with $x_i^* \neq w_k$, we have $i \in S_k$ due to $x^* \in \mathcal{X}_k$ and $x_i^* < w_k$. Hence, $x_i^* = w_k$ holds for all $i \in S_{k-1} \setminus S_k$, which shows $x^* \in \mathcal{X}_k$ and completes the induction. □

Knowing that there exists a min-max fair solution for the continuous relaxation of LOHP, we can use Algorithm 9 to compute a fair continuous plan. For this, we define $S^0 = \mathcal{P}$ and let $\mathcal{X}^0$ be the set of solutions to the continuous relaxation of LOHP. We then determine

$$
\begin{aligned}
w_k &= \min\{w | \exists x \in \mathcal{X}_{k-1} : \phi_p(x) \leq w \ \forall p \in S_{k-1}\}, \\
\mathcal{X}_k' &= \{x \in \mathcal{X}_{k-1} | \phi_p(x) \leq w_k \ \forall p \in S_{k-1}\}, \\
S_k &= \{i \in S_{k-1} | \exists x \in \mathcal{X}_k' : \phi_p(x) < w_k\}, \\
\mathcal{X}_k &= \{x \in \mathcal{X}_{k-1} | \phi_p(x) = w_k \ \forall p \in S_{k-1} \setminus S_k\}
\end{aligned}
$$

in each iteration $k \in \mathbb{Z}_{>0}$. However, although Algorithm 9 is much more efficient compared to the approach from last section for computing leximax minimal solutions, we still need to solve many linear programs over the course of the iterations. Remember from last chapter that

solving the linear relaxation for the instance in our case study was very time consuming due to the size of the formulation. We therefore refrain from computing min-max fair solutions for the continuous relaxation of LOHP. Instead, we consider another relaxation, for which we can compute min-max fair solutions very efficiently.

8.4 Aggregated Planning and Water-Draining

We are only concerned with the number of shifts of each pharmacy when it comes to fairness. In particular, our burden function is indifferent with respect to the day on which a shift is assigned. We therefore consider a relaxed planning problem in which we aggregate the LOHP over the whole time horizon in order to simplify the problem. We will show that the aggregated problem can be solved efficiently via a *water-draining* algorithm, which is analogous to better known *water-filling* algorithms.

For the aggregated relaxation of the LOHP, we denote with $X_p \in [0, T]$ the total number of shifts that we assign to pharmacy $p \in \mathcal{P}$. We allow the assignment of fractional shifts in order to obtain a convex problem. The continuity of the relaxation only slightly reduces the predictive power of the solutions regarding the burden of pharmacies in a fair plan, as it is less relevant whether a pharmacy is assigned 20.5 or 21 shifts. Aggregating the constraints of LOHP over the time horizon and replacing $\sum_{t\in[T]} x_{pt}$ with X_p yields the *Aggregated Lexicographic Fair Out-of-Hours Planning Problem*

$$
\begin{aligned}
&\text{leximaxmin } (X_p)_{p\in\mathcal{P}} && && \text{(ALOHP.a)}\\
&\text{s.t.} \sum_{p\in C(m)} X_p \geq T && \forall m \in \mathcal{M} && \text{(ALOHP.b)}\\
&\sum_{p\in P(m)} X_p \geq \sum_{t\in[T]} d\,(m,t) && \forall m \in \mathcal{M} && \text{(ALOHP.c)}\\
&X_p \leq \left\lceil \frac{T}{r\,(p)+1} \right\rceil && \forall p \in \mathcal{P} && \text{(ALOHP.d)}\\
&X_p \leq b_{pp'}\,(X_{p'}) && \forall p, p' \in \mathcal{P} && \text{(ALOHP.e)}\\
&X_p \geq \underline{s} && \forall p \in \mathcal{P} && \text{(ALOHP.f)}\\
&X \in [0, T]^{\mathcal{P}}\,. && && \text{(ALOHP.g)}
\end{aligned}
$$

Constraints (ALOHP.b) and (ALOHP.c) follow directly from summing the constraints (LOHP.b) and (LOHP.c) over all days $t \in [T]$ of the time horizon. Constraints (ALOHP.d) reflect the periods of rest and can be derived by covering the time horizon $[T]$ with $\left\lceil \frac{T}{r(p)+1} \right\rceil$ intervals of length $r\,(p)+1$ and summing the corresponding constraints (LOHP.e). Constraints (ALOHP.e) and (ALOHP.f) correspond directly to constraints (LOHP.f) and (LOHP.g). Note that we do not add constraints corresponding to the conflict constraints (LOHP.d), as the upper bound on the number of shifts imposed by conflicts is in practice dominated by the aggregated period of rest constraints (ALOHP.d).

The burden function ϕ for the ALOHP is the identity $\phi(X) = X$. Hence, we neglect ϕ in the following and search for fair solutions directly within the set of solutions. The restrictions on the balancing functions $b_{pp'} : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ requested in Section 8.2 ensure that the set of solutions to ALOHP is convex and compact. Therefore, Theorem 39 implies that there exists a min-max fair solution that can be computed using Algorithm 9 from last section. Doing so is much easier than computing a fair solution to the continuous relaxation of LOHP due to the reduced size of ALOHP. However, we can use a special property of ALOHP to compute fair solutions even more efficiently with a combinatorial algorithm. For this, we generalize a theorem from Radunovic and Le Boudec [84] in the following.

Radunovic and Le Boudec state that **max-min** fair solutions can be computed with a so-called water-filling algorithm if the set of solutions $\mathcal{X}$ has the *free disposal property*. A set $\mathcal{X} \subseteq \mathbb{R}^n$ is said to have the free disposal property if, first, there exists a vector $\underline{x} \in \mathbb{R}^n$ with $\underline{x} \leq x$ for all $x \in \mathcal{X}$ and, second, for all $x' \in \mathbb{R}^n$ and $x \in \mathcal{X}$ with $\underline{x} \leq x' \leq x$, we have $x' \in \mathcal{X}$. Intuitively speaking, each entity $i \in [n]$ can choose to dispose some of the x_i goods that they were assigned until they reach a lower bound of $\underline{x}_i$. Since we are considering the allocation of burden instead of goods, we define the following analog for min-max fairness. We say that a set $\mathcal{X} \subseteq \mathbb{R}^n$ has the *free contribution property* if, first, there exists a vector $\overline{x} \in \mathbb{R}^n$ with $\overline{x} \geq x$ for all $x \in \mathcal{X}$ and, second, for all $x' \in \mathbb{R}^n$ and $x \in \mathcal{X}$ with $x \leq x' \leq \overline{x}$, we have $x' \in \mathcal{X}$. That is, each entity can choose to increase their burden x_i up to a bound $\overline{x}_i$. If $\mathcal{X}$ has the free contribution property, then we call $\overline{x}$ the *contribution bounds*.

Unfortunately, the set of solutions to ALOHP does not have the free contribution property. On the one hand, increasing the number of shifts X_p of pharmacy $p \in \mathcal{P}$ up to the individual upper bound $\left\lceil \frac{T}{r(p)+1} \right\rceil$ is advantageous for the coverage constraints (ALOHP.b) and (ALOHP.c) as well as the minimum number of shifts constraints (ALOHP.f). On the other hand, the balancing constraints (ALOHP.e) can prohibit increasing X_p without increasing $X_{p'}$ if we already have $X_p = b_{pp'}(X_{p'})$. To overcome this problem, we introduce the following weaker property. We say that a set $\mathcal{X} \subseteq \mathbb{R}^n$ has the *joint contribution property* if, first, there exists a vector $\overline{x} \in \mathbb{R}^n$ with $\overline{x} \geq x$ for all $x \in \mathcal{X}$ and, second, for all $w \in \mathbb{R}$ and $x \in \mathcal{X}$, we have $x' \in \mathcal{X}$ with $x'_i = \min\{\max\{x_i, w\}, \overline{x}_i\}$. The difference here compared to the free contribution property is that all x_i smaller than w are simultaneously increased to the level w, while respecting the individual contribution bounds $\overline{x}_i$. We will see later that the set of solutions to ALOHP has the joint contribution property, because the simultaneous increase ensures that the balancing constraints (ALOHP.e) remain fulfilled. Before that, we show how to use the joint contribution property for computing min-max fair solutions.

Given a min-max achievable set $\mathcal{X}$ with the joint contribution property and contribution bounds $\overline{x}$, Algorithm 10 computes the unique min-max fair solution $x^* \in \mathcal{X}$. The algorithm is similar to Algorithm 9 from last section, with the only difference being the computation of w_k. Due to the joint contribution property, there exists a solution $x \in \mathcal{X}_{k-1}$ minimizing $w_k = \min\{w | \exists x \in \mathcal{X}_{k-1} : x_i \leq w \; \forall i \in S_{k-1}\}$ that is uniquely defined by the contribution bounds $\overline{x}$ and the value w_k. For this, we simply set all not yet fixed components x_i for

$i \in S_{k-1}$ to the minimum of w_k and $\overline{x}_i$ (line 4). Hence, finding the minimum w_k reduces to a one-dimensional problem.

Algorithmus 10 : Water-draining for computing min-max fair solutions.

Input : A min-max achievable set $\mathcal{X} \subseteq \mathbb{R}^n$ fulfilling the joint contribution property and its contribution bounds $\overline{x}$

Output : The unique min-max fair vector in $\mathcal{X}$

1 Initialize $k = 0$, $S_0 = [n]$, and $\mathcal{X}_0 = \mathcal{X}$
2 **while** $S_k \neq \emptyset$ **do**
3 Update $k \leftarrow k + 1$
4 Minimize w_k such that $x \in \mathcal{X}_{k-1}$ exists with $x_i = \min\{w_k, \overline{x}_i\}$ for all $i \in S_{k-1}$
5 Let $\mathcal{X}'_k = \{x \in \mathcal{X}_{k-1} | x_i \leq w_k \ \forall i \in S_{k-1}\}$
6 Compute $S_k = \{i \in S_{k-1} | \exists x \in \mathcal{X}'_k : x_i < w_k\}$
7 Let $\mathcal{X}_k = \{x \in \mathcal{X}_{k-1} | x_i = w_k \ \forall i \in S_{k-1} \setminus S_k\}$
8 **return** *The single element in* $\mathcal{X}_k$

Intuitively, the idea of the algorithm is to think of the values x_i as water levels within interconnected water tanks of heights $\overline{x}_i$. The water level in each tank $x_i = \min\{w, \overline{x}_i\}$ is determined by a global level w, which is at first high enough such that all tanks are full. We carefully lower the global water level w by draining water from the system, which yields a simultaneous reduction of the levels in all tanks of height $\overline{x}_i \geq w$. We stop draining once any further reduction would render the corresponding solution infeasible. Afterwards, we close all water tanks that don't allow for a further reduction, thus fixing their water level x_i at the current global level w. In the algorithm, this corresponds to fixing $x_i = w_k$ and removing the specific indices from S_{k-1}, the set of all open tanks. We then proceed lowering the water level w_{k+1} in the remaining tanks S_k, subsequently closing them until no more are open. Given this intuition and the analogy to well-known water-filling algorithms, we call Algorithm 10 a *water-draining* algorithm.

Radunovic and Le Boudec [84] propose a water-filling algorithm for computing max-min fair solutions, which unfortunately has the same issue in the choice of S_k as Algorithm 9. Nevertheless, their statement on the exactness of water-filling algorithms for sets $\mathcal{X}$ with the free disposal property still holds true. This result also translates to min-max fairness when applying water-draining in the case that $\mathcal{X}$ has the free contribution property. We show in the following that the water-draining algorithm is also exact if $\mathcal{X}$ only has the joint contribution property.

Theorem 41. *Algorithm 10 computes a min-max fair solution if $\mathcal{X}$ is min-max achievable and has the joint contribution property.*

Proof. We show that the water-draining algorithm is equivalent to Algorithm 9 by proving via induction that $\mathcal{X}_k$ and $\mathcal{S}_k$ are the same in both algorithms for all $k \in \mathbb{Z}_{\geq 0}$. For this, let $\mathcal{X}_k^{\mathrm{W}}$ and S_k^{W} be the sets computed in the water-draining algorithm and $\mathcal{X}_k^{\mathrm{M}}$ and S_k^{M} be those computed in Algorithm 9.

The induction hypothesis holds for the base case $k = 0$, since we have $\mathcal{X}_0^{\mathrm{W}} = \mathcal{X} = \mathcal{X}_0^{\mathrm{M}}$ and $S_0^{\mathrm{W}} = [n] = S_0^{\mathrm{M}}$. Now, consider an arbitrary iteration $k \in \mathbb{Z}_{>0}$ within the algorithms and assume that the induction hypothesis is true for $k-1$. We show that the value w_k^{W} computed in the water-draining algorithm is equal to the value w_k^{M} computed in Algorithm 9. Then $\mathcal{X}_k^{\mathrm{W}} = \mathcal{X}_k^{\mathrm{M}}$ and $S_k^{\mathrm{W}} = S_k^{\mathrm{M}}$ hold by definition.

Consider the solution $x^{\mathrm{M}} \in \mathcal{X}_{k-1}^{\mathrm{M}}$ computed in line 4 of Algorithm 9 for minimizing w_k^{M}. Let x' be the solution obtained by raising x^{M} to the level w_k^{M}, i.e., $x'_i = \min\left\{\max\left\{x_i^{\mathrm{M}}, w_k^{\mathrm{M}}\right\}, \overline{x}_i\right\}$ for all $i \in [n]$. We then have $x' \in \mathcal{X}$ due to the joint contribution property. We show that $x'_i = x_i^{\mathrm{M}}$ holds for all $i \in [n] \setminus S_{k-1}^{\mathrm{M}}$ in order to prove $x' \in \mathcal{X}_{k-1}^{\mathrm{M}}$. To this end, remember from the proof of Theorem 40 that $w_{k'}^{\mathrm{M}} = \max\left\{x_i^* \middle| i \in S_{k'-1}^{\mathrm{M}}\right\}$ holds for the unique max-min fair solution $x^* \in \mathcal{X}$ and all $k' \in \mathbb{Z}_{>0}$. Accordingly, we have $w_{k'}^{\mathrm{M}} \geq w_k^{\mathrm{M}}$ for all $k' \in [k]$ due to $S_{k'-1}^{\mathrm{M}} \supseteq S_{k-1}^{\mathrm{M}}$. Moreover, for all $i \in [n] \setminus S_{k-1}^{\mathrm{M}}$, there exists a $k' \in [k-1]$ with $x_i^{\mathrm{M}} = w_{k'}^{\mathrm{M}}$. Hence, we have $x_i^{\mathrm{M}} \geq w_k^{\mathrm{M}}$, and thus $x'_i = \min\left\{\max\left\{x_i^{\mathrm{M}}, w_k^{\mathrm{M}}\right\}, \overline{x}_i\right\} = \min\left\{x_i^{\mathrm{M}}, \overline{x}_i\right\} = x_i^{\mathrm{M}}$, which proves $x \in \mathcal{X}_{k-1}^{\mathrm{M}}$. We furthermore have $x'_i = \min\left\{\max\left\{x_i^{\mathrm{M}}, w_k^{\mathrm{M}}\right\}, \overline{x}_i\right\} = \min\left\{w_k^{\mathrm{M}}, \overline{x}_i\right\}$ for all $i \in S_{k-1}^{\mathrm{M}}$ due to $x_i^{\mathrm{M}} \leq w_k^{\mathrm{M}}$ by the choice of x^{M}. Together with the induction hypothesis, we conclude $x' \in \mathcal{X}_{k-1}^{\mathrm{M}} = \mathcal{X}_{k-1}^{\mathrm{W}}$ as well as $x'_i = \min\left\{w_k^{\mathrm{M}}, \overline{x}_i\right\}$ for all $i \in S_{k-1}^{\mathrm{M}} = S_{k-1}^{\mathrm{W}}$. Hence, x' is a valid choice in line 4 of the water-draining algorithm and yields $w_k^{\mathrm{W}} \leq w_k^{\mathrm{M}}$.

We also have $w_k^{\mathrm{W}} \geq w_k^{\mathrm{M}}$. Otherwise, there would exist a solution $x^{\mathrm{W}} \in \mathcal{X}_{k-1}^{\mathrm{W}} = \mathcal{X}_{k-1}^{\mathrm{M}}$ with $x_i^{\mathrm{W}} < w_k^{\mathrm{M}}$ for all $i \in S_{k-1}^{\mathrm{W}} = S_{k-1}^{\mathrm{M}}$. Then x^{W} would also be a valid choice in line 4 of Algorithm 9 with $\max\left\{x_i^{\mathrm{W}} \middle| S_{k-1}^{\mathrm{M}}\right\} < w_k^{\mathrm{M}}$, which contradicts the minimality of w_k^{M}. This completes the induction. □

In order to use the water-draining algorithm for the planning of the out-of-hours service, we first show that the set of solutions to ALOHP has the joint contribution property. We compute contribution bounds $\overline{X} \in [0,T]^{\mathcal{P}}$ on the variables $X \in [0,T]^{\mathcal{P}}$ by using Algorithm 11, which propagates the upper bounds from the aggregated period of rest constraints (ALOHP.d) via the balancing constraints (ALOHP.e).

Algorithmus 11 : Computation of contribution bounds on the number of shifts.

Input : An instance of ALOHP
Output : Contribution bounds $\overline{X} \in [0,T]^{\mathcal{P}}$ on the variables $X \in [0,T]^{\mathcal{P}}$

1 Initialize $\mathcal{P}' = \mathcal{P}$ and $\overline{X}_p = \left\lceil \frac{T}{r(p)+1} \right\rceil$ for all $p \in \mathcal{P}$
2 **while** $\mathcal{P}' \neq \emptyset$ **do**
3 Choose $p' \in \operatorname{argmin}\left\{\overline{X}_p \mid p \in \mathcal{P}'\right\}$ and update $\mathcal{P}' \leftarrow \mathcal{P}' \setminus \{p'\}$
4 Update $\overline{X}_p \leftarrow \min\left\{b_{pp'}\left(\overline{X}_{p'}\right), \overline{X}_p\right\}$ for all $p \in \mathcal{P}'$

We show in the following that Algorithm 11 is exact and that the set of solutions to ALOHP indeed has the joint contribution property. For this, it will be important that the balancing functions $b_{pp'}$ are non-decreasing and that $b_{pp'}(X_p) \geq X_p$ holds for all $p, p' \in \mathcal{P}$, as requested in Section 8.2.

Proposition 42. *The set of solutions to ALOHP has the joint contribution property, with* $\overline{X}$ *computed by Algorithm 11 being the contribution bounds.*

Proof. Let $\mathcal{X} \subseteq [0,T]^{\mathcal{P}}$ be the solution space of ALOHP. After initialization in Algorithm 11, $\overline{X}_p = \left\lceil \frac{T}{r(p)+1} \right\rceil$ is a valid upper bound for all $X \in \mathcal{X}$ and $p \in \mathcal{P}$ due to constraints (ALOHP.d). Furthermore, if $\overline{X}_{p'}$ is a valid upper bound on $X_{p'}$, then $b_{pp'}\left(\overline{X}_{p'}\right)$ is a valid upper bound on X_p due to the constraints (ALOHP.e) and $b_{pp'}$ being non-decreasing. It follows that $\overline{X}_p$ is a valid upper bound at the end of Algorithm 11.

Let $X \in \mathcal{X}$ be an arbitrary solution to ALOHP. It remains to show that $X' \in \mathbb{R}^{\mathcal{P}}$ with $X'_p = \min\left\{\max\left\{X_p, w\right\}, \overline{X}_p\right\}$ is also a solution for all $w \in \mathbb{R}$. The vector X' meets the coverage constraints (ALOHP.b) and (ALOHP.c) as well as the minimum number of shifts constraints (ALOHP.f), since we have $X' \geq X$. Furthermore, we have $X'_p \leq \overline{X}_p \leq \left\lceil \frac{T}{r(p)+1} \right\rceil$, and thus X' meets the aggregated period of rest constraints (ALOHP.d).

Now, assume that there exists a pair of pharmacies $p, p' \in \mathcal{P}$ that violates a balancing constraint (ALOHP.e), i.e., $X'_p > b_{pp'}\left(X'_{p'}\right)$. We have $X'_p = \min\left\{w, \overline{X}_p\right\}$, because otherwise $X'_p = X_p$ and thus

$$X'_p = X_p \leq b_{pp'}\left(X_{p'}\right) \leq b_{pp'}\left(X'_{p'}\right)$$

holds due to $X_{p'} \leq X'_{p'}$. We also have $X'_{p'} = \min\left\{w, \overline{X}_{p'}\right\}$, since otherwise $X_{p'} > w$ and then

$$X'_p \leq w < X_{p'} \leq X'_{p'} \leq b_{pp'}\left(X'_{p'}\right)$$

holds due to the assumption on $b_{pp'}$. Together with $X'_p > b_{pp'}\left(X'_{p'}\right) \geq X'_{p'}$, we obtain $w \geq \min\left\{w, \overline{X}_p\right\} = X'_p > X'_{p'}$, which implies $X'_{p'} = \overline{X}_{p'}$.

Assume that p' was chosen before p in Algorithm 11. Then we would have

$$X'_p \leq \overline{X}_p \leq b_{pp'}\left(\overline{X}_{p'}\right) = b_{pp'}\left(X'_{p'}\right),$$

which contradicts the assumption $X'_p > b_{pp'}\left(X'_{p'}\right)$. Now, assume that p was chosen before p'. If we have $\overline{X}_p \leq \overline{X}_{p'}$, then

$$X'_p \leq \overline{X}_p \leq \overline{X}_{p'} \leq b_{pp'}\left(\overline{X}_{p'}\right) = b_{pp'}\left(X'_{p'}\right)$$

would again contradict our assumption and complete the proof. To show that $\overline{X}_p \leq \overline{X}_{p'}$ holds, we prove that the contribution bounds are non-decreasing with respect to the order in which the pharmacies are chosen in Algorithm 11. For this, let $p_k \in \mathcal{P}$ be the pharmacy chosen in iteration $k \in \mathbb{Z}_{>0}$ of Algorithm 11 and let $\overline{X}^j_{p_k}$ be the upper bound at the start of iteration $j \in [k]$. We then have $\overline{X}^k_{p_k} \leq \overline{X}^k_{p_{k+1}}$ due to the choice of p_k as well as $\overline{X}^k_{p_k} \leq b_{p_{k+1}p_k}\left(\overline{X}^k_{p_k}\right)$ due to the assumption on $b_{p_{k+1}p_k}$. It follows

$$\overline{X}_{p_k} = \overline{X}^k_{p_k} \leq \min\left\{b_{p_{k+1}p_k}\left(\overline{X}^k_{p_k}\right), \overline{X}^k_{p_{k+1}}\right\} = \overline{X}^{k+1}_{p_{k+1}} = \overline{X}_{p_{k+1}}. \qquad \square$$

The proposition shows that we can use the water-draining algorithm together with Algorithm 11 for computing a min-max fair solution to the ALOHP. It only remains to see how the values w_k and the sets S_k are computed during the water-draining algorithm.

We only consider the not yet fixed variables X_p with $p \in S_{k-1}$ for the problem of minimizing w_k. In particular, we treat all variables X_p with $p \notin S_{k-1}$ as constants. We can then neglect all constraints that solely contain variables X_p with $p \notin S_{k-1}$, as these are already fulfilled. We can also neglect the aggregated period of rest constraints (ALOHP.d), because we always have $X_p = \min\left\{w_k, \overline{X}_p\right\} \leq \overline{X}_p \leq \left\lceil \frac{T}{r(p)+1} \right\rceil$. Furthermore, the balancing constraints $X_p \leq b_{pp'}\left(X_{p'}\right)$ can be neglected as long as $p, p' \in S_{k-1}$ holds, because we have seen in the proof of Proposition 42 that $\min\left\{w_k, \overline{X}_p\right\} \leq b_{pp'}\left(\min\left\{w_k, \overline{X}_{p'}\right\}\right)$ holds, and thus

$$X_p = \min\left\{w_k, \overline{X}_p\right\} \leq b_{pp'}\left(\min\left\{w_k, \overline{X}_{p'}\right\}\right) = b_{pp'}\left(X_{p'}\right).$$

If we have $p' \notin S_{k-1}$, then we know from the proof of Theorem 41 that $X_{p'} \geq w_k$ holds. Therefore, we have

$$X_p = \min\left\{w_k, \overline{X}_p\right\} \leq X_{p'} \leq b_{pp'}\left(X_{p'}\right)$$

for all $p \in S_{k-1}$, and thus the balancing constraint can again be neglected. If we otherwise have $p \notin S_{k-1}$ and $p' \in S_{k-1}$, then let $b^{-1}_{pp'}\left(X_p\right)$ be the smallest value such that $b_{pp'}\left(b^{-1}_{pp'}\left(X_p\right)\right) = X_p$. In this case, the balancing constraint reduces to $X_{p'} \geq b^{-1}_{pp'}\left(X_p\right)$, with $b^{-1}_{pp'}\left(X_p\right)$ being constant.

Let $m \in \mathbb{Z}_{>0}$ be the number of non-neglected constraints and index them with $j \in [m]$. It follows from above that each constraint can be written as $\sum_{p \in \mathcal{P}_j} X_p \geq c_j$ for some subset of pharmacies $\mathcal{P}_j \subseteq S_{k-1} \subseteq \mathcal{P}$ and a constant right-hand side $c_j \in \mathbb{R}$. The left-hand side $\sum_{p \in \mathcal{P}_j} X_p = \sum_{p \in \mathcal{P}_j} \min\left\{w_k, \overline{X}_p\right\}$ is a non-decreasing, piece-wise linear function in w_k with at most $|\mathcal{P}_j|$ break points. Therefore, each constraint reduces to $w_k \geq c'_j$, where c'_j can be computed in linear time. Then we simply have $w_k = \max\left\{c'_j \middle| j \in [m]\right\}$.

After computing w_k as above, it is easy to compute S_k. For this, we consider all constraints $j \in [m]$ that imply a tight lower bound $c'_j = w_k$. The variables X_p contributing to these constraints are exactly the variables that cannot be reduced further. Therefore, we have

$$S_{k-1} \setminus S_k = \left\{ p \in \bigcup_{j \in [m]: w_k = c'_j} \mathcal{P}_j \middle| X_p = w_k \right\}.$$

Note that $S_{k-1} \setminus S_k$ is the set of indices whose variables are fixed in theory according to the water-draining algorithm. In practice, all X_p with $p \in \bigcup_{j \in [m]: w_k = c'_j} \mathcal{P}_j$ are fixed to $\min\left\{w_k, \overline{X}_p\right\}$, since none of these variables can be reduced.

We conclude that min-max fair solutions to ALOHP can be computed efficiently using the water-draining algorithm. Algorithm 11 computes the contribution bounds $\overline{X}$ in $\mathcal{O}\left(|\mathcal{P}|^2\right)$. Afterwards, we go through at most $|\mathcal{P}|$ iterations in the water-draining algorithm, because at least one variable is fixed in each iteration. Assuming that $b^{-1}_{pp'}\left(X_p\right)$ can be computed

in constant time, the lower bounds $c'_j \leq w_k$ can each be computed in $\mathcal{O}(|\mathcal{P}_j|)$. Let N be the number of non-zero entries in the constraint matrix. We then have $\sum_{j\in[m]} |\mathcal{P}_j| \leq N$, and can thus compute w_k in $\mathcal{O}(N)$. The set $\bigcup_{j\in[m]:w_k=c'_j} \mathcal{P}_j$, and thus S_k, can then again be computed in $\mathcal{O}(N)$. This amounts to a complexity of $\mathcal{O}(|\mathcal{P}|\, N)$. In practice, the number of steps required is much lower. This is because we usually fix multiple variables in each iteration and also only have to update c'_j if one of the variables X_p with $p \in \mathcal{P}_j$ has been fixed in the iteration before.

8.5 Bringing Fairness into Practice

We discuss in the following how the information contained in the min-max fair solution to ALOHP can be used in practice for computing fair out-of-hours plans. Given the size of our real-world problem, we need to formulate a tractable model for which we can compute solutions in reasonable time. Furthermore, the model should again be easy to understand, just like the lexicographic fairness, in order to enhance transparency.

Danna et. al [38] propose a practical and simple approach for using max-min fair solutions in traffic engineering problems in order to find a compromise between fairness and efficiency. In traffic engineering, commodities are routed in a network by allocating a bandwidth to each commodity. An efficient solution optimizes the throughput by maximizing the total bandwidth and a max-min fair solution ensures that the smaller bandwidths are as large as possible. Given a max-min fair solution, Danna et. al propose to maximize the total bandwidth while bounding the degradation of each individual bandwidth compared to the max-min fair solution. That is, if y_i^* is the bandwidth of commodity $i \in [n]$ in the max-min fair solution, then one adds the constraint $y_i \geq q y_i^*$ for a quotient $q \in [0,1]$ or $y_i \geq y_i^* - a$ for an absolute value $a \in \mathbb{R}_{\geq 0}$ to the problem of computing bandwidths $y \in \mathbb{R}^n_{\geq 0}$. This approach is appealing because the resulting problem remains tractable and transparent, as decision-makers directly control the degradation of fairness. In contrast, when defining a trade-off between fairness and efficiency in the objective, then the impact on the fairness cannot be quantified directly [38].

Applying the above approach to the planning of the out-of-hours service means that we bound the number of shifts $\sum_{t\in[T]} x_{pt}$ of each pharmacy $p \in \mathcal{P}$ with respect to their number of shifts X_p^* in the min-max fair solution to ALOHP. Analogous to above, we could add constraints $\sum_{t\in[T]} x_{pt} \leq q X_p^*$ with $q \geq 1$ or $\sum_{t\in[T]} x_{pt} \leq X_p^* + a$ with $a \geq 0$ for all pharmacies $p \in \mathcal{P}$. However, we argue that the approach should be slightly modified for our purpose. Our case study in the following section will reveal that the min-max fair solution X^* is very efficient. Hence, we do not need to reduce fairness for the sake of efficiency but can aim for a plan in which the numbers of shifts $\sum_{t\in[T]} x_{pt}$ are as close as possible to X_p^* for all $p \in \mathcal{P}$. Therefore, when applying $\sum_{t\in[T]} x_{pt} \leq q X_p^*$, we would choose $q \in \mathbb{R}$ as small as possible such that there still exists a feasible plan. Using a global parameter q for all pharmacies does not seem

appropriate, as it would be dictated by the pharmacy with the maximum quotient $\frac{\sum_{t\in[T]} x_{pt}}{X_p^*}$. For example, assume that we need to assign 12 shifts to a pharmacy $p \in \mathcal{P}$ with $X_p^* = 10$. We then have $q \geq 1.2$, and thus a highly burdened pharmacy $p' \in \mathcal{P}$ with $X_{p'}^* = 40$ can be assigned 48 shifts. This is despite that there may exist a plan that actually assigns only 40 shifts. Analogous concerns can be raised for using constraints $\sum_{t\in[T]} x_{pt} \leq X_p^* + a$.

In order to compute a plan that is close to the min-max fair plan X^*, we use upper bound constraints $\sum_{t\in[T]} x_{pt} \leq \left\lceil X_p^* \right\rceil + \beta_p$. Here, $\beta_p \in \mathbb{Z}_{\geq 0}$ is an individual variable for each $p \in \mathcal{P}$ that allows for assigning more than $\left\lceil X_p^* \right\rceil$ shifts. By penalizing β_p in the objective, we ensure that we only obtain $\sum_{t\in[T]} x_{pt} > \left\lceil X_p^* \right\rceil$ if absolutely necessary. We also apply lower bound constraints $\left\lfloor X_p^* \right\rfloor - \beta_p \leq \sum_{t\in[T]} x_{pt}$ so that no pharmacy is assigned too few shifts. Adding both, the lower and upper bound constraints, might appear redundant at first, as the min-max fairness of X^* implies that we cannot assign p fewer shifts than X_p^* without increasing the number of shifts of other pharmacies $p' \in \mathcal{P}$ above $X_{p'}^*$. However, the rounding of X_p^* in the constraints $\sum_{t\in[T]} x_{pt} \leq \left\lceil X_p^* \right\rceil + \beta_p$ necessitates the addition of lower bound constraints. For example, assume that 100 shifts have to be distributed among eleven pharmacies and we neglect the lower bounds. The fair plan then assigns $\frac{100}{11}$ shifts each, which would allow for assigning $\left\lceil \frac{100}{11} \right\rceil = 10$ shifts to ten pharmacies and zero shifts to the eleventh pharmacy. In contrast, when adding the lower bound constraints, we are inclined to assign nine shifts to ten pharmacies and ten shifts to the eleventh pharmacy. Accordingly, we propose the following *Fair Out-of-Hours Planning Problem*

$$\min \sum_{p\in\mathcal{P}} \sum_{t\in[T]} x_{pt} + \sum_{p\in\mathcal{P}} B\beta_p^2 \tag{FOHP.a}$$

$$\text{s.t.} \sum_{p\in C(m)} x_{pt} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [T] \tag{FOHP.b}$$

$$\sum_{p\in P(m)} x_{pt} \geq d\,(m,t) \qquad \forall m \in \mathcal{M}, t \in [T] \tag{FOHP.c}$$

$$x_{pt} + x_{p't} \leq 1 \qquad \forall \{p,p'\} \in \mathcal{C}, t \in [T] \tag{FOHP.d}$$

$$\sum_{t'=t}^{t+r(p)} x_{pt'} \leq 1 \qquad \forall p \in \mathcal{P}, t \in [T - r\,(p)] \tag{FOHP.e}$$

$$\left\lfloor X_p^* \right\rfloor - \beta_p \leq \sum_{t\in[T]} x_{pt} \leq \left\lceil X_p^* \right\rceil + \beta_p \qquad \forall p \in \mathcal{P} \tag{FOHP.f}$$

$$\sum_{t\in[T]} x_{pt} \geq \underline{s} \qquad \forall p \in \mathcal{P} \tag{FOHP.g}$$

$$x \in \{0,1\}^{\mathcal{P}\times[T]}, \beta \in \mathbb{Z}_{\geq 0}^{\mathcal{P}}. \tag{FOHP.h}$$

We penalize the assignment of more shifts than $\left\lceil X_p^* \right\rceil$ or fewer shifts than $\left\lfloor X_p^* \right\rfloor$ by adding $\sum_{p\in\mathcal{P}} B\beta_p^2$ to the objective function, where $B \in \mathbb{Z}_{>0}$ is a big constant. We choose the variables β_p to be quadratic in order to avoid an uneven increase of β_p among the pharmacies. That is, if we can choose between $\beta_p = \beta_{p'} = 1$ and $\beta_p = 2,\ \beta_{p'} = 0$, then we opt for the former. The non-linear objective can be easily linearized by replacing β_p with a sum $\sum_{i\in[\overline{\beta}_p]} \beta_{pi}$ of binary

variables $\beta_{pi} \in \{0,1\}$ for an upper bound $\overline{\beta}_p \in \mathbb{Z}_{\geq 0}$ on the value of β_p. We then can substitute $\beta_p^2 = \sum_{i \in [\overline{\beta}_p]} (2i-1)\,\beta_{pi}$ under the assumption that $\beta_{p1} \geq \cdots \geq \beta_{p\overline{\beta}_p}$ holds. The latter can be assumed without loss of generality, as $\sum_{i \in [\overline{\beta}_p]} (2i-1)\,\beta_{pi}$ is minimized. In theory, we can choose $\overline{\beta}_p = \max\left\{\left\lceil \frac{T}{r(p)+1} \right\rceil - \left\lceil X_p^* \right\rceil, \left\lfloor X_p^* \right\rfloor - \underline{s}\right\}$, which is an upper bound on the maximum possible distance between $\sum_{t \in [T]} x_{pt}$ and the interval $\left[\left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil\right]$. In practice, this distance will be much smaller, which allows for substituting β_p with few binary variables. We will discuss later how to do this efficiently in our rolling horizon approach from last chapter.

Note that we omit the balancing constraints $\sum_{t \in [T]} x_{pt} \leq b_{pp'}\left(\sum_{t \in [T]} x_{p't}\right)$ in the FOHP. Although balancing is not ensured by the other constraints, it is considered in the computation of X^*, and thus almost implied by the constraints (FOHP.f). To see this, remember that we assumed in Section 8.2 that $b_{pp'}$ is non-decreasing. Together with $X_p^* \leq b_{pp'}\left(X_{p'}^*\right)$, we obtain

$$\begin{aligned}
\sum_{t \in [T]} x_{pt} &\leq \left\lceil X_p^* \right\rceil + \beta_p \\
&\leq \left\lceil b_{pp'}\left(X_{p'}^*\right) \right\rceil + \beta_p \\
&= \left\lceil b_{pp'}\left(\left\lfloor X_{p'}^* \right\rfloor + X_{p'}^* - \left\lfloor X_{p'}^* \right\rfloor\right) \right\rceil + \beta_p \\
&\leq \left\lceil b_{pp'}\left(\sum_{t \in [T]} x_{p't} + \beta_{p'} + X_{p'}^* - \left\lfloor X_{p'}^* \right\rfloor\right) \right\rceil + \beta_p.
\end{aligned}$$

In our case study in the next section, we will consider balancing functions $b_{pp'}(y) = y + a_{pp'}$ for $y \in \mathbb{R}_{\geq 0}$ and an absolute value $a_{pp'} \in \mathbb{Z}_{>0}$. Therefore, we have

$$\begin{aligned}
\left\lceil b_{pp'}\left(\sum_{t \in [T]} x_{p't} + \beta_{p'} + X_{p'}^* - \left\lfloor X_{p'}^* \right\rfloor\right) \right\rceil + \beta_p &= \left\lceil \sum_{t \in [T]} x_{p't} + X_{p'}^* - \left\lfloor X_{p'}^* \right\rfloor + \beta_{p'} + a_{pp'} \right\rceil + \beta_p \\
&= \sum_{t \in [T]} x_{p't} + a_{pp'} + \left\lceil X_{p'}^* - \left\lfloor X_{p'}^* \right\rfloor \right\rceil + \beta_{p'} + \beta_p \\
&\leq b_{pp'}\left(\sum_{t \in [T]} x_{p't}\right) + 1 + \beta_{p'} + \beta_p.
\end{aligned}$$

Hence, the balancing constraints $\sum_{t \in [T]} x_{pt} \leq b_{pp'}\left(\sum_{t \in [T]} x_{p't}\right)$ are violated by at most $1 + \beta_{p'} + \beta_p$, where $\beta_p + \beta_{p'}$ will be zero for most $p, p' \in \mathcal{P}$. Therefore, omitting balancing constraints is not exact but only results in a small potential error. The omission of the up to $|\mathcal{P}^2|$ constraints is thus a pragmatic way to significantly reduce the size of our problem.

We show in the following how the approaches developed in Section 7.3 can be adapted for solving the FOHP. That is, we use the aggregation of equivalent pharmacies into superpharmacies together with the rolling horizon approach. Here, the latter is again enhanced with an intermediate step in which we delete non-coverage shifts, as in Section 7.3.3.

For the aggregation into superpharmacies, we have to redefine equivalence between pharmacies. Just like in Section 7.3.1, equivalent pharmacies must be in the same municipality,

cover the same municipalities, have the same period of rest, and have the same conflicts. We now additionally request that equivalent pharmacies $p \sim p'$ must also be equal with respect to the number of shifts in the min-max fair solution X^* up to rounding, i.e., $\left\lfloor X^*_p \right\rfloor = \left\lfloor X^*_{p'} \right\rfloor$ and $\left\lceil X^*_p \right\rceil = \left\lceil X^*_{p'} \right\rceil$. Then the constraints (FOHP.f) are equal for both pharmacies and we can assign both the same number of shifts. This additional requirement is a proper restriction, as pharmacies p, p' that were equivalent with respect to the old definition might have $X^*_p \neq X^*_{p'}$ if their balancing constraints (ALOHP.e) are different.

After partitioning the set of pharmacies $\mathcal{P}$ into superpharmacies $p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}$ consisting of equivalent pharmacies, we replace $\sum_{p \in p^{\mathrm{s}}} x_{pt}$ in FOHP with variables $x^{\mathrm{s}}_{p^{\mathrm{s}}t}$ in order to obtain

$$\min \sum_{p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}} \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} + \sum_{p \in \mathcal{P}} B\beta_p^2 \tag{FSOHP.a}$$

$$\text{s.t.} \sum_{p^{\mathrm{s}} \in C^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \geq 1 \qquad \forall m \in \mathcal{M}, t \in [T] \tag{FSOHP.b}$$

$$\sum_{p^{\mathrm{s}} \in P^{\mathrm{s}}(m)} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \geq d\,(m,t) \qquad \forall m \in \mathcal{M}, t \in [T] \tag{FSOHP.c}$$

$$x^{\mathrm{s}}_{p^{\mathrm{s}}_1 t} + x^{\mathrm{s}}_{p^{\mathrm{s}}_2 t} \leq 1 \qquad \forall \{p^{\mathrm{s}}_1, p^{\mathrm{s}}_2\} \in \mathcal{C}^{\mathrm{s}}, t \in [T] \tag{FSOHP.d}$$

$$\sum_{t'=t}^{t+r(p^{\mathrm{s}})} x^{\mathrm{s}}_{p^{\mathrm{s}}t'} \leq |p^{\mathrm{s}}| \qquad \forall p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}}, t \in [T - r\,(p^{\mathrm{s}})] \tag{FSOHP.e}$$

$$\sum_{p \in p^{\mathrm{s}}} \left\lfloor X^*_p \right\rfloor - \beta_p \leq \sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \leq \sum_{p \in p^{\mathrm{s}}} \left\lceil X^*_p \right\rceil + \beta_p, \qquad \forall p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}} \tag{FSOHP.f}$$

$$\sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \geq |p^{\mathrm{s}}|\,\underline{s} \qquad \forall p^{\mathrm{s}} \in \mathcal{P}^{\mathrm{s}} \tag{FSOHP.g}$$

$$x^{\mathrm{s}} \in \{0,1\}^{\mathcal{P}^{\mathrm{s}} \times [T]}, \beta \in \mathbb{Z}^{\mathcal{P}}_{\geq 0}. \tag{FSOHP.h}$$

The construction is mostly analogous to that of SOHP from Section 7.3.1. The difference here is that we replace the fairness constraints (FOHP.f) and the minimum number of shifts constraints (FOHP.g) with constraints (FSOHP.f) and (FSOHP.g), which arise from aggregating the original constraints for all $p \in p^{\mathrm{s}}$. The aggregation of the constraints is exact, since we can later distribute the shifts $\sum_{t \in [T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t}$ of each superpharmacy p^{s} evenly (up to rounding) among the aggregated pharmacies $p \in p^{\mathrm{s}}$. This is analogous to the procedure in the proof of Proposition 35 for the original superpharmacy formulation SOHP. Note that we do not substitute $\sum_{p \in p^{\mathrm{s}}} \beta_p = \beta_{p^{\mathrm{s}}}$, since this would yield a difference in the quadratic objective $\left(\sum_{p \in p^{\mathrm{s}}} \beta_p\right)^2 \neq \sum_{p \in p^{\mathrm{s}}} \beta_p^2$. However, when substituting $\beta_p = \sum_{i \in [\overline{\beta}_p]} \beta_{pi}$ for the linearization of the objective, then we can aggregate all binary variables β_{pi} with $p \in p^{\mathrm{s}}$ into a single integer variable $\beta_{p^{\mathrm{s}}i} \in [|p^{\mathrm{s}}|]_0$, i.e., $\beta_{p^{\mathrm{s}}i} = \sum_{p \in p^{\mathrm{s}}} \beta_{pi}$.

For the rolling horizon approach, we consider subproblems of FSOHP on intervals $[t_i] \subseteq [T]$ for a sequence of days $t_1 \leq \cdots \leq t_\ell \leq t_{\ell+1} = T$. The construction of the subproblems is

analog to that in Section 7.3.2. That is, we replace $[T]$ with $[t_i]$ in the formulation of FSOHP and replace the constraints (FSOHP.f) with

$$\sum_{p\in p^{\mathrm{s}}} \left\lfloor \frac{X_p^* t_i}{T} \right\rfloor - \beta_p \leq \sum_{t\in[T]} x^{\mathrm{s}}_{p^{\mathrm{s}}t} \leq \sum_{p\in p^{\mathrm{s}}} \left\lceil \frac{X_p^* t_i}{T} \right\rceil + \beta_p$$

as well as replace constraints (FOHP.g) with $\sum_{t\in[t_i]} x_{pt} \geq \left\lceil \frac{\underline{s} t_i}{T} \right\rceil$.

The rolling horizon approach is particularly suitable for linearizing the quadratic terms β_p^2 in the objective function. We can give tighter bounds on the value of β_p in an optimal solution due to the smaller number of days to be planned. In theory, the optimal value of β_p is at most $\overline{\beta}_p = \max\left\{\left\lceil \frac{t_i}{r(p)+1} \right\rceil - \left\lceil \frac{X_p^* t_i}{T} \right\rceil, \left\lfloor \frac{X_p^* t_i}{T} \right\rfloor - \underline{s}\right\}$ for a subproblem on the interval $[t_i]$. We observe in practice that we can even choose $\overline{\beta}_p = 1$ within the first interval, that is, β_p can be substituted by a single variable $\beta_{p1} \in \{0,1\}$. We then dynamically adjust $\overline{\beta}_p$ based on the value of β_p within the last iteration. That is, if we have $\beta_p = \overline{\beta}_p$, then we increase $\overline{\beta}_p$ by one for the next iteration. Hence, the problem remains tractable in practice despite the quadratic objective function.

We will see in the next section that combining superpharmacies and the rolling horizon approach enables us to compute consistently good solutions to FOHP. We will also see that the resulting out-of-hours plans are very close to the min-max fair solutions to ALOHP, and thus are as fair as we could hope for.

8.6 Case Study

In this section, we apply our fair planning approach to the real-world instance in the area North Rhine from Section 7.4. The parameters for our problem are as before, that is, we consider $|\mathcal{P}| = 2291$ pharmacies, $|\mathcal{M}| = 165$ municipalities, and a time horizon $[T] = [365]$ of one year. The coverage, periods of rest, and conflict distances are again with respect to the size of the municipalities, as stated in Table 7.1. Furthermore, we request a minimum number of shifts $\underline{s} = 10$ for each pharmacy. The only differences to the problem considered in Section 7.4 are the way we balance the number of shifts and that we now incorporate fairness into the planning.

We start our study by considering different balancing approaches and analyzing their impact on the out-of-hours service. Afterwards, we discuss two concepts of refining the coverage of municipalities in order to even the distribution of shifts among pharmacies and to improve the quality of coverage for residents. In our analysis, we first focus on min-max fair solutions to ALOHP, as their particular structure is useful for evaluating changes to the model. In a second step, we then test how the min-max fair solutions translate into practice by using them as input for the FOHP. After discussing the changes to our model, we close our study with a comparison of the resulting plan for the FOHP with the optimal plan computed for the OHP in the last chapter.

All experiments are implemented in Java 11 and performed on a Linux machine with an Intel® Core™ i7-5930K CPU @ 3.50GHz with 32 GB RAM. We use Gurobi version 9.5.0 [52] with default settings to solve MILPs.

8.6.1 Balancing Approaches

We first test the impact of the municipality-balancing from last chapter on the min-max fair solution to ALOHP, that is, we allow a difference of at most one in the number of shifts assigned to pharmacies within the same municipality. For this, we define $b_{pp'}(X_{p'}) = X_{p'} + 1$ if $p, p' \in \mathcal{P}$ are in the same municipality and $b_{pp'}(X_{p'}) = \infty$ otherwise. The ALOHP with municipality-balancing can be solved easily with the water-draining Algorithm 10, which computes the min-max fair solution within 0.1 seconds. This solution assigns a total of $\sum_{p\in\mathcal{P}} X_p \approx 32222$ shifts, which is 2,083 shifts more than in the optimal plan computed for the OHP in the last chapter. The decline in efficiency is due to the interaction between min-max fairness and the municipality-balancing. Assume that $p \in P(m)$ is the only pharmacy within a city $m \in \mathcal{M}$ covering a rural municipality $m' \in \mathcal{M}$ that is covered by few pharmacies $C(m')$. An efficient solution to the OHP would not assign many shifts to p for covering m', because the municipality-balancing would then force us to assign many shifts to all pharmacies within $P(m)$. However, when applying min-max fairness, then we need to assign many shifts to p, and thus to all pharmacies in $P(m)$, as otherwise the other pharmacies $p' \in C(m') \setminus \{p\}$ covering m' would be assigned even more shifts.

The left-hand map in Figure 8.1 shows the assignment of shifts in the min-max fair solution to ALOHP with municipality-balancing. Here, the numbers of shifts X_p are rounded to the nearest integer. We have a closer look at the city of Aachen, which is located in the southwest of the planning area and marked by a white circle. There are 69 pharmacies within Aachen, all of which are assigned 22 or 23 shifts. This is due to two pharmacies that are assigned their maximum number of $\left\lceil \frac{T}{r(p)+1} \right\rceil = \left\lceil \frac{365}{16} \right\rceil = 23$ shifts in order to cover the rural town of Monschau south of Aachen. This assignment is necessary, as all other pharmacies covering Monschau are assigned 40 shifts, and thus should be relieved as much as possible in a fair solution. However, the assignment of at least 22 shifts to all pharmacies in Aachen solely for the sake of balancing is not acceptable. We therefore need to consider other balancing functions $b_{pp'}$.

When applying no balancing at all, i.e., $b_{pp'}(X_{p'}) = \infty$ for all $p, p' \in \mathcal{P}$, then the min-max fair solution to ALOHP can again be computed within 0.1 seconds. The solution assigns $\sum_{p\in\mathcal{P}} X_p \approx 30157$ shifts, which is only 18 shifts more than in the optimal plan to the OHP from last chapter. This proves that min-max fair solutions can be efficient, at least when considering the aggregated problem ALOHP. The middle map in Figure 8.1 shows the assignment of shifts to pharmacies. The pharmacies in Aachen are now mostly assigned eleven shifts, yielding a total reduction of 724 shifts within a single city.

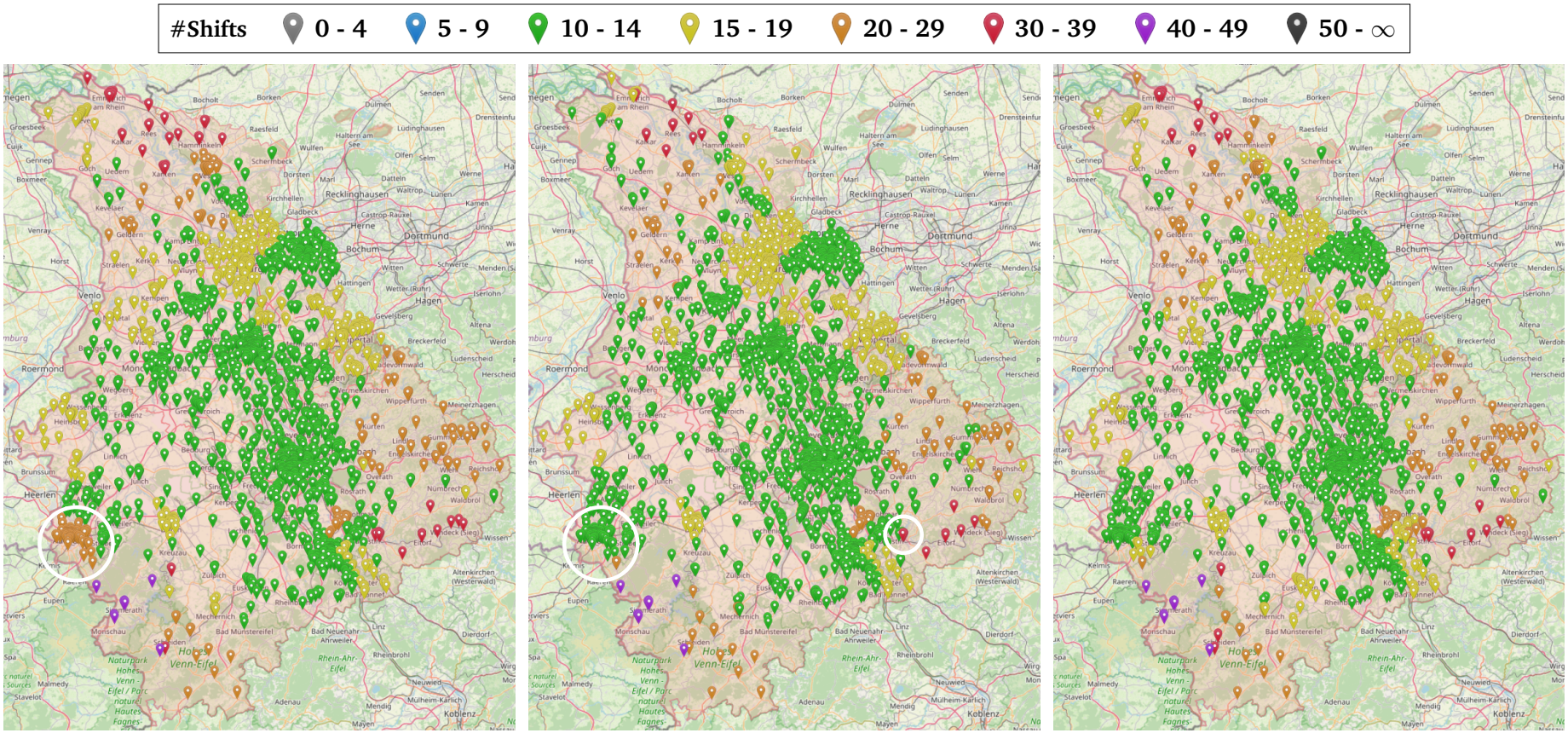


Figure 8.1. Geographical distribution of shifts for min-max fair solutions to ALOHP with different balancing approaches. **Left:** municipality-balancing. **Middle:** no balancing. **Right:** geographic-balancing.

The efficiency of the solution without balancing comes at the cost of unacceptable differences in the number of shifts of neighboring pharmacies. This is most evident in the town of Hennef, located in the southeast of the planning area and marked with a white circle, where one pharmacy is assigned 33 shifts while another pharmacy only 300 meters apart is assigned ten shifts. This difference is due to the rigorous definition of coverage: The highly burdened pharmacy is barely within the cover radius of the municipality of Eitorf, and can thus fully contribute to its coverage, while the other pharmacy is few meters outside the radius and cannot contribute to the coverage at all. We can resolve this issue by applying a small adjustment to the coverage such that either both or none of the pharmacies cover Eitorf. Then balancing is achieved naturally from min-max fairness and does not need to be enforced via constraints. Since this has implications on the quality of coverage, we are faced with a political task in which we seek a compromise between efficiency, balance, and coverage. We will propose an approach for achieving a reasonable compromise in the next section.

In order to identify pairs of pharmacies for which there is a conflict between balance and efficiency, we first need to define functions $b_{pp'}$ that reflect our balancing goals. We request a *geographic-balancing* that is with respect to the distance between pharmacies. Neighboring pharmacies should be assigned a similar number of shifts, while pharmacies located farther apart may have a larger difference. We define $b_{pp'}\left(X_{p'}\right) = X_{p'} + \left\lceil \frac{\delta(p,p')}{1000} \right\rceil$ if $p, p' \in \mathcal{P}$ are within the same municipality; that is, the difference in the number of shifts increases by one for every kilometer that the pharmacies are apart. Since we also want to avoid inter-municipal differences, we define $b_{pp'}\left(X_{p'}\right) = X_{p'} + \left\lceil \frac{\delta(p,p')}{500} \right\rceil$ for all $p, p' \in \mathcal{P}$ in different municipalities. The inter-municipal balancing is weaker, as there is a particular historical emphasis on balance in municipalities.

Solving the ALOHP with geographic-balancing requires 1.6 seconds, which is slower compared to the other balancing approaches. The slow-down is due to the individual balancing constraints for each pair of pharmacies, which lead to more distinct numbers of shifts, and thus to more iterations in the water-draining Algorithm 10. Nevertheless, the computation time is still negligible compared to the time required for computing an actual out-of-hours plan. The computed min-max fair solution assigns $\sum_{p\in\mathcal{P}} X_p \approx 31020$ shifts, which is 1,202 fewer shifts than in the solution with municipality-balancing. Furthermore, we see on the right-hand map in Figure 8.1 that the geographic-balancing yields a relatively homogeneous assignment of shifts to pharmacies.

In order to evaluate how min-max fair solutions translate into practice, we use the solutions computed for the three balancing functions above as input for the FOHP. We attempt to solve each of the three problems in two ways. First, we apply an exact approach by solving the fair superpharmacy problem FSOHP for the whole time horizon. Second, we use the rolling horizon heuristic together with the aggregation into superpharmacies as described in Section 8.5. For the rolling horizon, we define $\ell = 23$ breakpoints for dividing the time horizon into 24 intervals and set a total time limit of 10,000 seconds, which we split among the subproblems as described in Section 7.4.2. For both the exact and the heuristic approach, we define $B = 1000$ as the coefficient for $\sum_{p\in\mathcal{P}} \beta_p^2$ in the objective of FOHP for penalizing the

Table 8.1. Total number of shifts, quadratic fairness violation $\sum_{p\in\mathcal{P}} \beta_p^2$, and computation time in seconds for different balancing approaches.

	municipality-balancing			no balancing			geographic-balancing		
	ALOHP	FSOHP		ALOHP	FSOHP		ALOHP	FSOHP	
		exact	heuristic		exact	heuristic		exact	heuristic
number shifts	32,222	32,131	32118	30,157	30,162	30,178	31,073	31,011	31,024
fairness violation	-	0	20	-	13	16	-	13	23
computation time	0.1	8,350	705	0.1	7,305	660	1.6	5,890	663

violation of $\left\lfloor X_p^* \right\rfloor \leq \sum_{t\in[T]} x_{pt} \leq \left\lceil X_p^* \right\rceil$, where X_p^* is the number of shifts in the corresponding min-max fair solution.

Table 8.1 shows computational results in comparison with the corresponding min-max fair solutions to ALOHP. Interestingly, all three problems are solved exactly in at most 8,350 seconds (2.3 hours), which is roughly half the time required for solving the OHP from Section 7.4 exactly. This decrease in computation time might be attributed to the structure implied by the fairness constraints (FSOHP.f), which almost fix the number of shifts of each pharmacy. This is in line with the observation in Section 7.4.3, where a higher minimum number of shifts $\underline{s}$ resulted in lower computation times. The rolling horizon heuristic is able to compute solutions in under 13 minutes. It therefore computes solutions roughly 10 times faster than the exact approach, where the first feasible plan is in each case computed at the very end of the computation time.

Both the exact and heuristic solutions are very close to the corresponding min-max fair solutions: The observed gap between the number of shifts in the min-max fair solution and the corresponding out-of-hours plans is at most 0.32%. Even more important, the assigned numbers of shifts $\sum_{t\in[T]} x_{pt}$ are very close to X_p^* for all pharmacies $p \in \mathcal{P}$ in all solutions. In the exact solution for the municipality-balancing, we have $\sum_{p\in\mathcal{P}} \beta_p^2 = 0$, and thus $\sum_{t\in[T]} x_{pt} \in \left\{ \left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil \right\}$ for all pharmacies. The highest fairness violation is $\sum_{p\in\mathcal{P}} \beta_p^2 = 23$ in the heuristic solution for the geographic-balancing. Here, we have 19 pharmacies with $\beta_p = 1$ and one pharmacy with $\beta_p = 2$. Conversely, this implies that 2,271 pharmacies have $\sum_{t\in[T]} x_{pt} \in \left\{ \left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil \right\}$.

We conclude that all three balancing approaches yield min-max fair solutions that are very close to an actual out-of-hours plan. Hence, all are suitable for our approach of using the min-max fair solution as an orientation for the actual plan. We will continue with the geographic-balancing, as it additionally strikes a compromise between balance and efficiency.

8.6.2 Refining Coverage

We have seen that min-max fair solutions can be computed fast and serve as good predictors for the number of shifts of each pharmacy in the out-of-hours plans computed for the FOHP. This predictive power is valuable, as min-max fair solutions have a clear structure that makes them relatively easy to analyze. Remember from Section 8.4 that the assignment of X_p^* shifts to pharmacy $p \in \mathcal{P}$ in the water-draining Algorithm 10 can be attributed to constraints of ALOHP that contain X_p and become tight in the iteration in which X_p is fixed. We need to adjust exactly these constraints if we want to relieve p without assigning more shifts to other pharmacies that are already assigned at least X_p^* shifts. We use this insight in the following in order to refine the coverage of municipalities.

8.6.2.1. Smoothing Coverage

We noted above that the differences in the numbers of shifts of neighboring pharmacies in the min-max fair solution without balancing is due to the rigorous definition of coverage. We now want to smooth the coverage in order to obtain a more balanced solution even when not enforcing balancing constraints. For this, we again consider the situation in the town of Hennef, where pharmacies are assigned either 33 or ten shifts in the min-max fair solution without balancing. The map in Figure 8.2 shows the distribution of shifts to pharmacies in the concerned area, wherein Hennef is located in the west. The difference in the numbers of shifts assigned to the pharmacies in Hennef is undesirable: Once geographic-balancing is requested, the pharmacies with ten shifts will be assigned 32 shifts that are mostly not necessary for the coverage.

Thanks to the clear structure of the min-max fair solution, we know that the assignment of the 33 shifts is due to the coverage constraint (ALOHP.b) of the municipality of Eitorf, which is located in the center of the map in Figure 8.2. Relieving the highly burdened pharmacies in Hennef essentially comes down to two possibilities. First, we may assign more shifts to other pharmacies covering Eitorf. Second, we may extend the set of covering pharmacies in order to distribute the shifts among more pharmacies. The first possibility achieves local balance within Hennef at the cost of global unfairness, which is exactly the opposite of what we aim for in this chapter. The second possibility relieves the pharmacies covering Eitorf but results in longer travel distances for the residents in Eitorf. The average travel distance especially increases if we redistribute shifts from pharmacies in the center of Eitorf to pharmacies outside the original cover radius. However, if we only redistribute shifts from pharmacies in Hennef to their neighboring pharmacies, then we achieve balance while maintaining almost the same quality of coverage. We show in the following how to incorporate this idea into ALOHP in order to compute min-max fair solutions that are more balanced without using balancing constraints.

More generally, let X^* be the min-max fair solution to ALOHP without balancing and let $\mathcal{P}' \subseteq \mathcal{P}$ be a set of neighboring pharmacies that differ highly in their assigned number of shifts. We

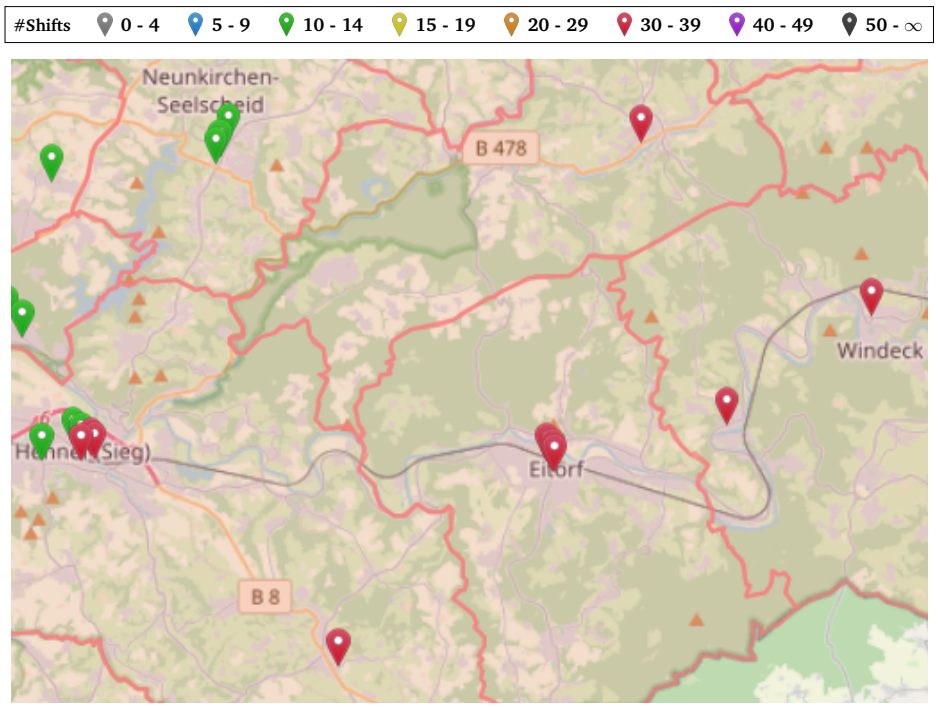


Figure 8.2. Assignment of shifts to pharmacies around Eitorf for the min-max fair solution to ALOHP without balancing.

will define later when pharmacies are considered neighboring. Let $p' \in \operatorname{argmax}\left\{X_p^* \middle| p \in \mathcal{P}'\right\}$ be a pharmacy with the highest number of shifts in $\mathcal{P}'$. Assume that we have $X_p^* < \overline{X}_p$ for all $p \in \mathcal{P}'$ with $X_p^* < X_{p'}^*$, that is, we may assign more shifts to pharmacies in $\mathcal{P}'$ that are not assigned the highest number of shifts. Additionally, assume that the assignment of the $X_{p'}^*$ shifts to p' can be attributed to the coverage constraint $\sum_{p \in C(m')} X_p \geq T$ of a municipality $m' \in \mathcal{M}$ with $p' \in C(m')$. This is the case if $X_{p'}^*$ could not be reduced further due to the coverage constraint becoming tight in the corresponding iteration of the water-draining Algorithm 10. Then we have $p \notin C(m')$ for all $p \in \mathcal{P}'$ with $X_p^* < X_{p'}^*$, as we would otherwise assign more shifts to these pharmacies in a min-max fair solution. Hence, if we extend the set of pharmacies covering m' to be $C(m') \cup \mathcal{P}'$, then we can assign shifts to pharmacies in $\mathcal{P}' \setminus C(m')$ so that the coverage constraint for m' is not tight any longer. We can then reduce the burden on p' if there exists no other tight constraint that restricts $X_{p'}$. If there exist more coverage constraints $\sum_{p \in C(m)} X_p \geq T$ of other municipalities $m \in \mathcal{M}$ restricting $X_{p'}$, then we can also extend their sets of covering pharmacies $C(m)$. If there exists a tight constraint of a different type, then we cannot achieve balance for the pharmacies in $\mathcal{P}'$ by adjusting the coverage. In the following, we assume that there exists no tight constraint of a different type and we thus can actually relieve p'.

If we solved ALOHP again with $\sum_{p \in C(m') \cup \mathcal{P}'} X_p \geq T$ instead of $\sum_{p \in C(m')} X_p \geq T$, then we would not only redistribute shifts of pharmacies within $\mathcal{P}'$ but potentially also from

pharmacies in $C(m') \setminus \mathcal{P}'$ to pharmacies in $\mathcal{P}' \setminus C(m')$. To prevent this, we instead replace $\sum_{p \in C(m')} X_p \geq T$ with

$$\min \left\{ \sum_{p \in \mathcal{P}'} X_p, \sum_{p \in C(m') \cap \mathcal{P}'} X_p^* \right\} + \sum_{p \in C(m') \setminus \mathcal{P}'} X_p \geq T,$$

that is, all pharmacies in $\mathcal{P}'$ can contribute to the coverage of m' but they cannot contribute more shifts than the pharmacies in $C(m') \cap \mathcal{P}'$ contributed in the previous solution X^*. Then the pharmacies within $C(m') \setminus \mathcal{P}'$ need to be assigned at least the same number of shifts as before. In fact, they are assigned exactly the same number of $T - \sum_{p \in C(m') \cap \mathcal{P}'} X_p^*$ shifts in the new min-max fair solution, as the new solution would otherwise be less fair than X^*, which is still feasible for the adjusted ALOHP. Simultaneously, we can distribute the $\sum_{p \in C(m') \cap \mathcal{P}'} X_p^*$ shifts previously assigned to the pharmacies in $C(m') \cap \mathcal{P}'$ more evenly among the pharmacies in $\mathcal{P}'$. Note that we may assign more than $\sum_{p \in C(m') \cap \mathcal{P}'} X_p^*$ shifts to the pharmacies in $\mathcal{P}'$ in order to meet other constraints.

The above adjustment to the coverage constraint $\sum_{p \in C(m')} X_p \geq T$ can be applied multiple times in case we not only want to balance the number of shifts of a single set of pharmacies $\mathcal{P}'$. Let $\mathcal{P}_1, \ldots, \mathcal{P}_\ell \subseteq \mathcal{P}$ be disjoint sets of pharmacies which we want to allow to cover m'. We then define the *smoothed coverage constraint*

$$\sum_{i \in [\ell]} \min \left\{ \sum_{p \in \mathcal{P}_i} X_p, \sum_{p \in C(m') \cap \mathcal{P}_i} X_p^* \right\} + \sum_{p \in C(m') \setminus \bigcup_{i \in [\ell]} \mathcal{P}_i} X_p \geq T \tag{8.1}$$

for replacing the original coverage constraint $\sum_{p \in C(m')} X_p \geq T$ in ALOHP. We call the adjusted problem, with potentially multiple smoothed coverage constraints, the *smoothed ALOHP*.

The set of vectors $X \in [0, T]^{\mathcal{P}}$ fulfilling the smoothed coverage constraints (8.1) is convex and compact. It follows that the set of solutions to the smoothed ALOHP is also convex and compact, and thus remains min-max achievable. Furthermore, the set of solutions still has the joint contribution property, as increasing X never leads to a violation of constraints (8.1). Hence, we can solve the smoothed ALOHP with the water-draining Algorithm 10. Computing w_k in the k-th iteration of the algorithm is again easy: Replacing X_p with $X_p = \min \left\{ w_k, \overline{X}_p \right\}$ in the smoothed coverage constraints (8.1) yields a piece-wise linear function in w_k with at most $\ell + \left| C(m') \cup \bigcup_{i \in [\ell]} \mathcal{P}_i \right|$ breakpoints. Thus, the minimum value w_k fulfilling the constraint can be computed in linear time.

After determining the smoothed coverage constraints (8.1), we add the geographic-balancing constraints to the smoothed ALOHP and compute a min-max fair solution X'. We use X' as input for the FOHP and replace the original coverage constraints $\sum_{p \in C(m')} x_{pt} \geq 1$ with $\sum_{p \in C'(m')} x_{pt} \geq 1$ for all days $t \in [T]$, with $C'(m') = C(m') \cup \bigcup_{i \in [\ell]} \mathcal{P}_i$. By doing so, we risk that m' is covered more often by pharmacies in $C'(m') \setminus C(m')$ than in the min-max fair solution X'. However, we can expect in practice that most pharmacies $p \in \mathcal{P}$ will be

assigned at least $\left\lfloor X'_p \right\rfloor$ shifts in the solution to FOHP. In order to ensure that these shifts are not on the same day, we introduce conflicts between all pharmacies in $C(m')$, that is, we add $\binom{C(m')}{2}$ to the set of conflicts $\mathcal{C}$. We do this for all municipalities $m' \in \mathcal{M}$ for which the coverage constraint is smoothed. Then m' is covered by pharmacies within $C(m')$ on roughly as many days in the actual out-of-hours plan as in the min-max fair solution X'. The additional conflicts pose in our experience no problem in practice. To see this, note that we only smooth the coverage of municipalities $m' \in \mathcal{M}$ with $\sum_{p \in C(m')} X^*_p = T$. Now, consider a solution to the FOHP with the original coverage. When assigning $\sum_{t \in [T]} x_{pt} \in \left\{ \left\lfloor X^*_p \right\rfloor, \left\lceil X^*_p \right\rceil \right\}$ shifts to each pharmacy $p \in C(m')$, then there can be at most $\sum_{p \in C(m')} \left\lceil X^*_p \right\rceil - X^*_p$ days with more than one shift. A fair out-of-hours plan therefore tends to avoid assigning two shifts on the same day, even when conflicts are not enforced.

After discussing how the coverage can be smoothed with limited impact on the coverage quality, we need to identify cases in which the coverage of a municipality should be smoothed and determine the pharmacies that should be included in the coverage. Since this is again a political question, we only consider cases that most likely need to be addressed, that is, when neighboring pharmacies differ highly in their number of shifts. For this, let X^* again be the min-max fair solution to ALOHP without balancing. Here, we consider two pharmacies $p, p' \in \mathcal{P}$ as candidates for smoothing the coverage if their geographic-balancing constraint $X^*_{p'} \leq b_{p'p}\left(X^*_p\right)$ is violated by at least 50%, i.e., $X^*_{p'} \geq \frac{3}{2} b_{p'p}\left(X^*_p\right)$. Assume that the assignment of the $X^*_{p'}$ shifts to p' can be attributed to the coverage constraint (ALOHP.b) of a municipality $m' \in \mathcal{M}$. If $X^*_p < \overline{X}_p$ holds, then p is not within $C(m')$, and can thus be considered for an extended coverage of m' in order to relieve p'. However, we only let p cover m' if the distance from p' to p is at most 10% the cover radius of m', i.e., $\delta(p', p) \leq \frac{\delta^{\text{cov}}(m')}{10}$. In this case, p and p' are geographically close and the distance from m' to p is at most 10% higher than the cover radius. We then consider p and p' together for the smoothed coverage of m'.

Note that there might be multiple pairs of pharmacies $p \in \mathcal{P} \setminus C(m')$ and $p' \in C(m')$ that we want to consider together for the coverage of m'. We use these pairs to build the disjoint sets $\mathcal{P}_1, \ldots, \mathcal{P}_\ell \subseteq \mathcal{P}$ that define the smoothed coverage constraints (8.1). For this, we define a bipartite graph $G = (\mathcal{P}, E)$ with the set of pharmacies $\mathcal{P}$ being the nodes and an edge $\{p, p'\} \in E$ between two pharmacies $p \in \mathcal{P} \setminus C(m')$, $p' \in C(m')$ if p and p' are considered together for the coverage of m'. We then let $\mathcal{P}_1, \ldots, \mathcal{P}_\ell$ be the connected components within G that contain at least two pharmacies. The pharmacies in the sets $\mathcal{P}_i$ are not only geographically close to each other but are also close to the border of the cover radius. This is because each pharmacy $p \in \mathcal{P} \setminus C(m')$ outside the cover radius is at most $\frac{\delta^{\text{cov}}(m')}{10}$ away from a pharmacy $p' \in C(m')$ within the cover radius, and vice versa. We therefore maintain a similar quality of coverage when redistributing shifts within the sets $\mathcal{P}_i$.

When applying the above procedure to our real-world instance in the area North Rhine, then we smooth the coverage of five out of the 165 municipalities. The min-max fair solution to the smoothed ALOHP with geographic-balancing assigns a total of $\sum_{p \in \mathcal{P}} X_p \approx 30696$ shifts,

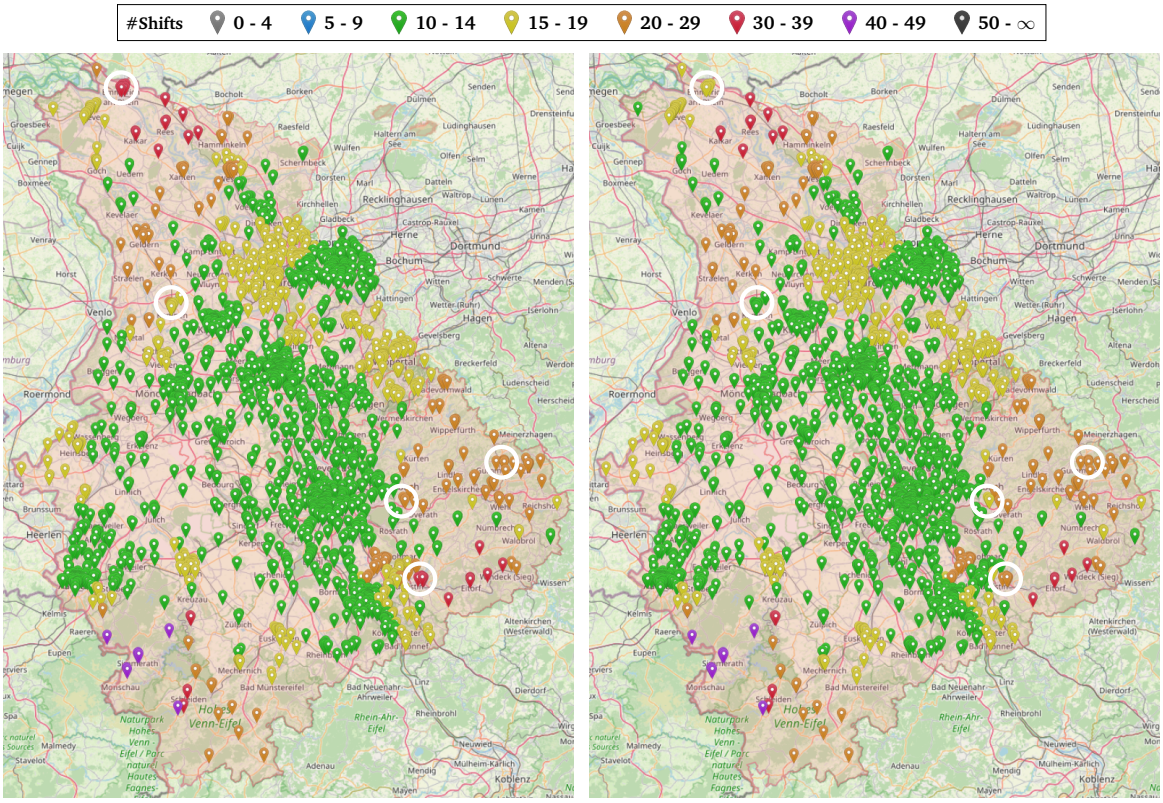


Figure 8.3. Geographical distribution of shifts for min-max fair solutions to ALOHP. **Left:** standard coverage. **Right:** smoothed coverage.

which is 377 fewer shifts than the solution for the non-smoothed problem. This is only a reduction of 1.2%, however, note that our changes to the model are rather conservative and that most pharmacies remain unaffected. In fact, only 98 out of the 2,291 pharmacies are assigned a different number of shifts after smoothing the coverage. This includes 71 pharmacies with fewer shifts and 27 pharmacies that are assigned more shifts to compensate for the decrease of the others. When only considering these 98 pharmacies, then the total number of shifts decreases from 1,843 down to 1,466, which is a reduction of 20.5%.

Figure 8.3 shows the distribution of shifts to pharmacies for both solutions with and without smoothing. Pharmacies that are directly affected by the smoothing of coverage, that is, they are considered together for the coverage of a municipality, are marked with circles. The smoothing proves to be very effective in the municipality of Emmerich, which is located in the very north of the planning area. There are six pharmacies which were previously assigned 31 to 33 shifts, as one of them needed to cover the municipality of Rees to the southeast of Emmerich. After the smoothing, each of these pharmacies is assigned 17 shifts. The already considered pharmacies in Hennef, which are located in the southernmost circle in Figure 8.3, are now assigned 22 or 23 shifts instead of 31 to 33 shifts. This reduction also leads to a decrease in the number of shifts of the pharmacies around Hennef, which were previously assigned many shifts to ensure geographic-balancing.

The reduction of the shifts could be even larger if the pharmacies in Hennef did not need to be balanced with a pharmacy to the southeast which is not considered in the smoothing and

	default coverage	smoothed coverage	extended coverage
Eitorf	8,339	8,438	9,698
Lindlar	9,631	9,618	9,876
Rees	9,017	9,097	10,476
Reichshof	12,233	12,243	13,006
Straelen	9,631	9,708	10,522
average	9,770	9,821	10,716

Table 8.2. Mean distance to the nearest out-of-hours pharmacy over the whole time horizon. Displayed for the five municipalities with smoothed coverage constraints.

is therefore still assigned 33 shifts. The same applies for the pharmacies in Gummersbach, located in the easternmost circle, which can only be relieved by 2 to 3 shifts due to the geographic-balancing with the surrounding pharmacies. Decision-makers might opt for relieving the surrounding pharmacies by applying a more aggressive or specifically tailored smoothing for the respective area. For this study, we continue with the smoothing described above.

The min-max fair solution to the smoothed ALOHP again translates well into practice. An optimal solution to the corresponding FSOHP, which can be computed within 1,505 seconds, assigns a total of 30,631 shifts. This is 65 shifts less than in the min-max fair solution and 380 shifts less than in the optimal out-of-hours plan computed for the default coverage. We obtain $\sum_{p\in\mathcal{P}} \beta_p^2 = 13$, with $\beta_p = 1$ for 13 pharmacies. Hence, the actual plan and the min-max fair solution are again similar in terms of the number of shifts assigned to each pharmacy. Most importantly, the quality of coverage is only slightly worse in the plan computed for the smoothed coverage. Table 8.2 shows the mean distance to the nearest out-of-hours pharmacy for the five municipalities whose coverage constraint is smoothed. The municipality with the largest increase is Eitorf, for which the mean distance rises by 1.2% from 8,339 meters to 8,438 meters. Coincidentally, the mean distance for Lindlar even decreases slightly. The average over the five municipalities increases only by 0.5% from 9,770 meters to 9,821 meters.

To put the quality of coverage into perspective, we compute an additional plan in which we also extend the set of covering pharmacies but don't adjust the coverage as in the smoothed coverage constraints (8.1). Instead, we only replace $\sum_{p\in C(m')} X_p \geq T$ with $\sum_{p\in C'(m')} X_p \geq T$ in the ALOHP and $\sum_{p\in C(m')} x_{pt} \geq 1$ with $\sum_{p\in C'(m')} x_{pt} \geq 1$ for all days $t \in [T]$ in the FOHP. Here, $C'(m')$ is the same extended set of covering pharmacies as used for the smoothed coverage. An optimal plan with the *extended coverage* assigns 542 shifts less than the plan with default coverage. Thus, the reduction in the number of shifts is 41.1% higher compared to the plan for the smoothed coverage. However, the mean distance to the nearest out-of-hours pharmacy increases significantly for the five affected municipalities, as can be seen in Table 8.2. The travel distance is on average 9.7% higher than in the plan for the default

coverage. In particular, we observe an increase of 16.3% for Eitorf. We therefore prefer the smoothed coverage over the extended coverage in the following.

The observations above prove the concept of smoothing the coverage with the use of the min-max fair solution. The min-max fair solution to the ALOHP without balancing indicates precisely which constraints need to be adjusted. The min-max fair solution to the smoothed ALOHP then corresponds to an out-of-hours plan with fewer shifts and a more even distribution at almost no cost with respect to the quality of coverage.

8.6.2.2. Tightening Coverage

Remember from Section 7.4 that some pharmacies are assigned no shifts at all in solutions to the OHP without minimum number of shifts constraints (OHP.h). Assigning shifts to these pharmacies is often not necessary for the coverage, as the local municipalities can already be covered by shifts that need to be assigned to surrounding pharmacies. Once the minimum number of shifts is requested, our model is indifferent of the day to which we assign such non-coverage shifts. We then might assign multiple shifts on the same day to pharmacies within the same area, which is ineffective for reducing the travel distance of residents to their nearest out-of-hours pharmacy. We analyze and resolve this issue, again using min-max fair solutions to ALOHP.

We consider the case of the town of Langenfeld, which is marked in Figure 8.4. Langenfeld is surrounded by several cities with many pharmacies and can therefore be covered by a total of 143 pharmacies. These pharmacies need to be assigned $\frac{365}{143} \approx 2.6$ shifts on average to cover Langenfeld in a solution to ALOHP. Thus, the coverage constraint (ALOHP.b) is clearly redundant when the minimum number of $\underline{s} = 10$ shifts is requested. In fact, the coverage constraint is even redundant when neither balancing nor minimum number of shifts constraints are applied. The left-hand map in Figure 8.4 shows the min-max solution for this setting. All pharmacies within Langenfeld are assigned zero shifts, since the surrounding pharmacies are already assigned a total of 851 shifts to meet the demand within the cities. The coverage of Langenfeld is thus not ensured by nearby pharmacies but by pharmacies that are barely within the cover radius. To counteract this, we reduce the set of covering pharmacies. In this process, we take care that the total number of shifts does not increase.

More generally, let X^* be the min-max fair solution to the ALOHP with geographic-balancing and minimum number of shifts constraints. We only consider non-smoothed coverage constraints for simplicity. The following idea extends analogously to smoothed constraints. Let $m \in \mathcal{M}$ be a municipality with $\sum_{p \in C(m)} X_p^* > T$. Then not all shifts assigned to pharmacies in $C(m)$ are needed for covering m. We might therefore consider *tightening* the coverage constraint by removing pharmacies from $C(m)$. For this, let $p_1 \in \operatorname{argmax} \{\delta(m,p) | p \in C(m)\}$ be a most distant pharmacy covering m. We remove p_1 from $C(m)$ if $\sum_{p \in C(m) \setminus \{p_1\}} X_p^* \geq T$ holds, that is, if X^* is still a feasible solution to ALOHP after the removal. Then X^* is still a min-max fair solution to ALOHP with the adjusted coverage, as the set of feasible solutions is

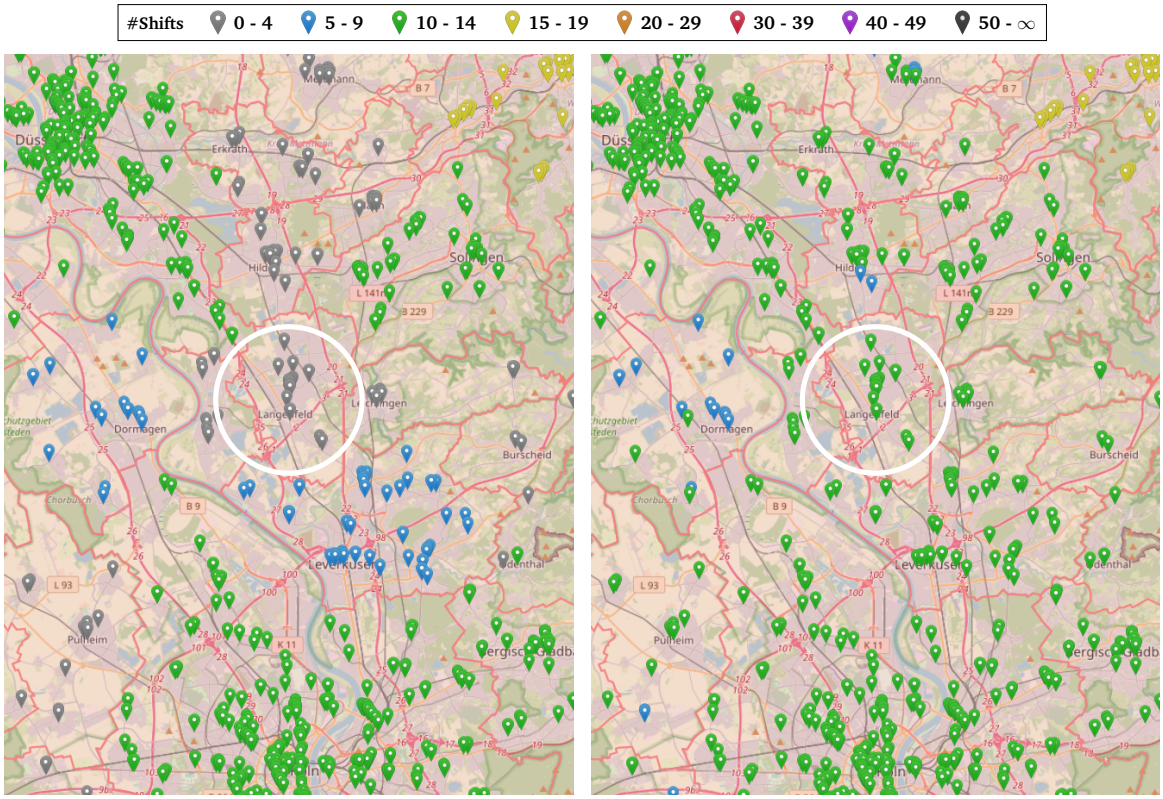


Figure 8.4. Assignment of shifts to pharmacies around Langenfeld for min-max fair solutions to the smoothed ALOHP without balancing and minimum number of shifts constraints. **Left:** without tightened coverage. **Right:** with tightened coverage.

non-increasing when tightening coverage constraints. We repeat this step for the sequence of most distant pharmacies $p_1, \ldots, p_k$ until the next pharmacy p_{k+1} cannot be removed, i.e., until we have $\sum_{p \in C(m) \setminus \{p_1, \ldots, p_{k+1}\}} X^*_p < T$. We can apply this procedure sequence-independently for each municipality $m \in \mathcal{M}$, as the min-max fair solution X^* remains unchanged. In the following, we consider the tightening applied to all municipalities after the smoothing of coverage constraints as in the prior section.

While the tightened coverage has no effect on the min-max fair solution X^*, the impact is significant when considering the ALOHP without geographic-balancing and minimum number of shifts. Before adjusting the coverage, we only assign 26,166 shifts in the min-max fair solution when omitting these constraints. After adjusting the coverage, we assign 30,190 shifts. This almost equals the 30,696 shifts that are assigned in the min-max fair solution to ALOHP with geographic-balancing and minimum number of shifts. After tightening, the town of Langenfeld is covered by 37 pharmacies, which are all assigned almost ten shifts, as can be seen on the right-hand map in Figure 8.4. Hence, the $\underline{s} = 10$ shifts that are assigned for meeting the minimum number of shifts constraints are now also important for the coverage. We are therefore inclined to better distribute these shifts in an out-of-hours plan, such that Langenfeld is covered by nearby pharmacies year-round.

The FOHP with tightened coverage is significantly harder to solve than the previous problems. Before tightening the coverage constraints, the large sets of covering pharmacies provide

Table 8.3. Total number of shifts, quadratic fairness violation $\sum_{p\in\mathcal{P}} \beta_p^2$, mean distance from municipalities to nearest out-of-hours pharmacies, and computation time in seconds for different tightening coefficients.

	tightening coefficient τ							
	1.0		1.1		1.2		1.3	
	exact	heuristic	exact	heuristic	exact	heuristic	exact	heuristic
number shifts	-	31,168	-	30,738	-	30,670	30,629	30,642
fairness violation	-	445	-	51	-	24	13	23
mean distance	-	7,376	-	7,546	-	7,629	7,693	7,701
computation time	100,000	100,000	100,000	46,651	100,000	4,120	72,981	1,081

	1.4		1.5		∞	
	exact	heuristic	exact	heuristic	exact	heuristic
number shifts	30,632	30,643	30,630	30,644	30,631	30,640
fairness violation	13	21	13	21	13	26
mean distance	7,757	7,751	7,794	7,798	8,062	8,024
computation time	14,072	779	2,411	673	1,505	657

much freedom for achieving solutions with $\sum_{t\in[T]} x_{pt} \in \left\{\left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil\right\}$. This freedom no longer exists, now that each coverage constraint is nearly tight in the min-max fair solution X^*. For tight coverage constraints $\sum_{p\in C(m)\setminus\{p_1,\dots,p_k\}} X_p^* = T$, we need an almost perfect rotation of the covering pharmacies $p \in C(m) \setminus \{p_1,\dots,p_k\}$ in order to achieve an out-of-hours plan with $\sum_{t\in[T]} x_{pt} \in \left\{\left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil\right\}$. Unfortunately, synchronizing these rotations across sets of covering pharmacies for different municipalities $m \in \mathcal{M}$ is not always possible. Hence, it is likely that there exists no out-of-hours plan that almost matches the min-max fair solution.

Since the above adjustment is too strong, we consider different levels of tightening coverage constraints. For this, we define a *tightening coefficient* $\tau \geq 1$ that determines the degree of tightening: We only remove pharmacies $p_1,\dots,p_k$ from the set of covering pharmacies $C(m)$ if we have $\sum_{p\in C(m)\setminus\{p_1,\dots,p_k\}} X_p^* \geq \tau T$. Note that we might already have $\sum_{p\in C(m)} X_p^* < \tau T$ for some $m \in \mathcal{M}$ in the case of $\tau > 1$. These constraints are already tight and remain unchanged. Thus, choosing $\tau = 1$ corresponds to the procedure above and $\tau \to \infty$ yields no adjustment at all. We test tightening coefficients $\tau \in \{1.0, 1.1, \dots, 1.5, \infty\}$ for analyzing the effect of our approach. Since we are not able to compute any solution for $\tau \in \{1, 1.1\}$ within 10,000 seconds, we increase the time limit to 100,000 seconds in order to analyze the effect of tightening for small τ.

Table 8.3 shows computational results for solving FSOHP exactly and the rolling horizon heuristic with $\ell = 23$ breakpoints for all $\tau \in \{1.0, 1.1, \dots, 1.5, \infty\}$. We see that the problems become increasingly difficult for smaller τ. In fact, we are not able to compute any solution for $\tau \leq 1.2$ within 100,000 seconds using the exact approach. Interestingly, the computed optimal solutions for $\tau \geq 1.3$ all have the same objective value when neglecting the small

variations in the number of shifts, which are within optimality tolerance of 0.1%. Choosing $\tau = 1.3$ over $\tau = \infty$ thus yields an equally efficient plan while reducing the mean distance by 4.6%. However, the computation time of 72,981 seconds (20.3 hours) of the exact approach is rather high for $\tau = 1.3$.

The rolling horizon approach is more practical, as it computes solutions much faster. Moreover, the heuristic solutions for $\tau \geq 1.3$ are similar to the corresponding optimal solutions in terms of the total number of shifts and the fairness violation. Even for $\tau \in \{1.1, 1.2\}$, for which the exact approach yields no solution at all, the heuristic solutions are close to the min-max fair solution: For $\tau = 1.1$, we have 47 pharmacies with $\beta_p = 1$ and one pharmacy with $\beta_p = 2$, which implies that 2,243 out of the 2,291 pharmacies are assigned a number of shifts $\sum_{t \in [T]} x_{pt} \in \left\{ \left\lfloor X_p^* \right\rfloor, \left\lceil X_p^* \right\rceil \right\}$. For $\tau = 1$, we observe a larger increase in the total number of shifts and the fairness violation. It is unclear whether this worsening is inevitable due to the tightened constraints or whether it is due to bad decisions taken in the rolling horizon approach, during which most subproblems could not be solved to optimality. The exact approach yields a dual bound on the objective value of 89,702, provided by the optimal continuous solution, which indicates that there might be a plan with roughly 30,702 shifts and $\sum_{p \in \mathcal{P}} \beta_p^2 = 59$. However, the integrality gap for $\tau = 1$ might also be much larger due to the already mentioned impossibility to synchronize the rotations of pharmacies implied by tight coverage constraints.

In the following, we will consider the heuristic solution obtained for $\tau = 1.2$, as it can be computed in reasonable time. In comparison with $\tau = \infty$, the solution provides a reduction of 5.5% in the mean distance together with a negligible increase in the number of shifts and fairness violation. Thus, we achieve an improvement of the quality of coverage while maintaining the same burden on the pharmacists. The improvement is particularly noticeable for towns located between larger cities, such as Langenfeld, where the mean distance is reduced by 24.9%.

The above observations once again show that min-max fair solutions are valuable for understanding and improving the structure of our coverage model. Moreover, we have seen that the rolling horizon approach is still able to construct good out-of-hours plans for tighter coverage constraints. This raises hope that the approach is also effective for other real-world instances with different coverage structures.

8.6.3 Comparing OHP and FOHP

We conclude our case study by comparing the optimal plan computed for the OHP in the last chapter to the plan for the FOHP with geographic-balancing as well as smoothed and tightened coverage. We focus here on efficiency, quality of coverage, and fairness.

The plan for the OHP assigns a total of 30,139 shifts and is thus more efficient than the plan for the FOHP, which assigns 30,670 shifts. However, the increase of 1.8% is surprisingly

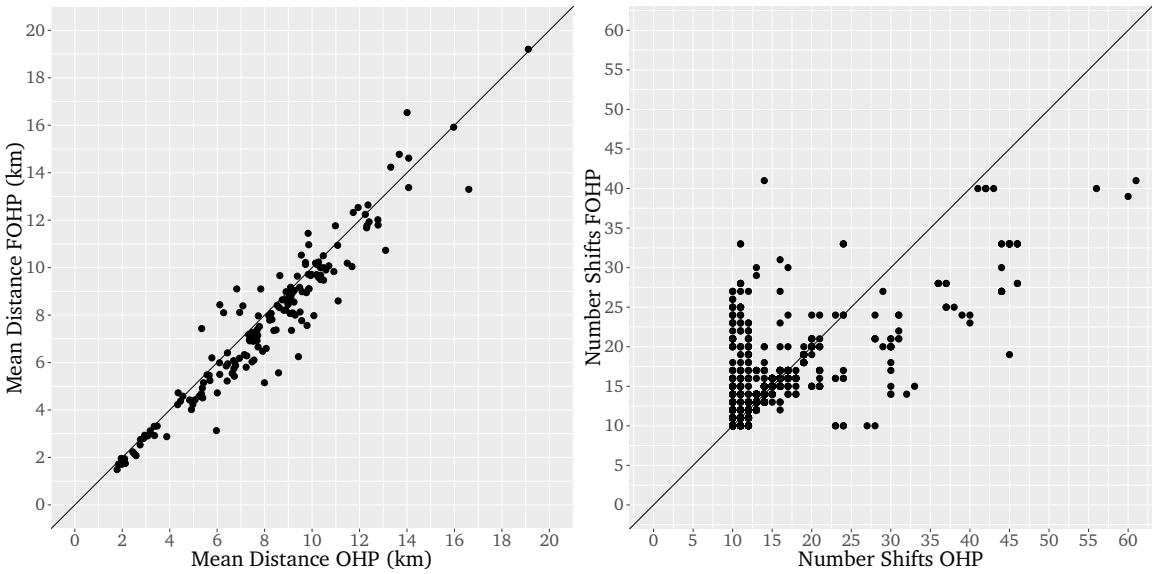


Figure 8.5. Comparison of the plans for OHP and FOHP. **Left:** scatter plot of mean distances from municipality centers to their nearest out-of-hours pharmacy. **Right:** scatter plot of numbers of shifts assigned to pharmacies.

small when bearing in mind that the min-max fairness approach, which is the foundation of the FOHP, aims to reduce the number of shifts of the highly burdened pharmacies at all costs. As already discussed in Section 8.1, highly burdened pharmacies can usually be relieved by a redistribution of shifts without assigning many additional shifts to other pharmacies. The price of fairness thus seems acceptable for the planning of the out-of-hours service. Note that, in order to achieve this level of efficiency for the FOHP, we replaced the municipality-balancing, which is applied for the OHP, with the geographic-balancing. Furthermore, we adjusted the coverage of five municipalities to smooth the assignment of shifts. We thus may have gained the preservation of efficiency at the expense of a potentially weaker local balancing and a potential decline in quality of coverage. However, we will see later that the balancing in the plan for the FOHP is stronger. Additionally, we already showed in Section 8.6.2.1 that the smoothing of coverage has only a negligible impact on the quality of coverage.

The overall quality of coverage is better in the plan for the FOHP. Here, the mean distance to the nearest out-of-hours pharmacy over all municipalities and the whole time horizon is 7,629 meters. The mean distance in the plan for the OHP is 8,049 meters, and thus 5.5% higher. Note that this improvement is not due to the higher number of shifts in the plan for the FOHP but due to the tightening of the coverage. From Table 8.3, we know that the plan for the FOHP without tightening has a mean distance of 8,024, which almost equals that of the plan for the OHP. The left-hand plot in Figure 8.5 shows the mean distance for each municipality in the plans for the OHP and the FOHP. Most points are located below the

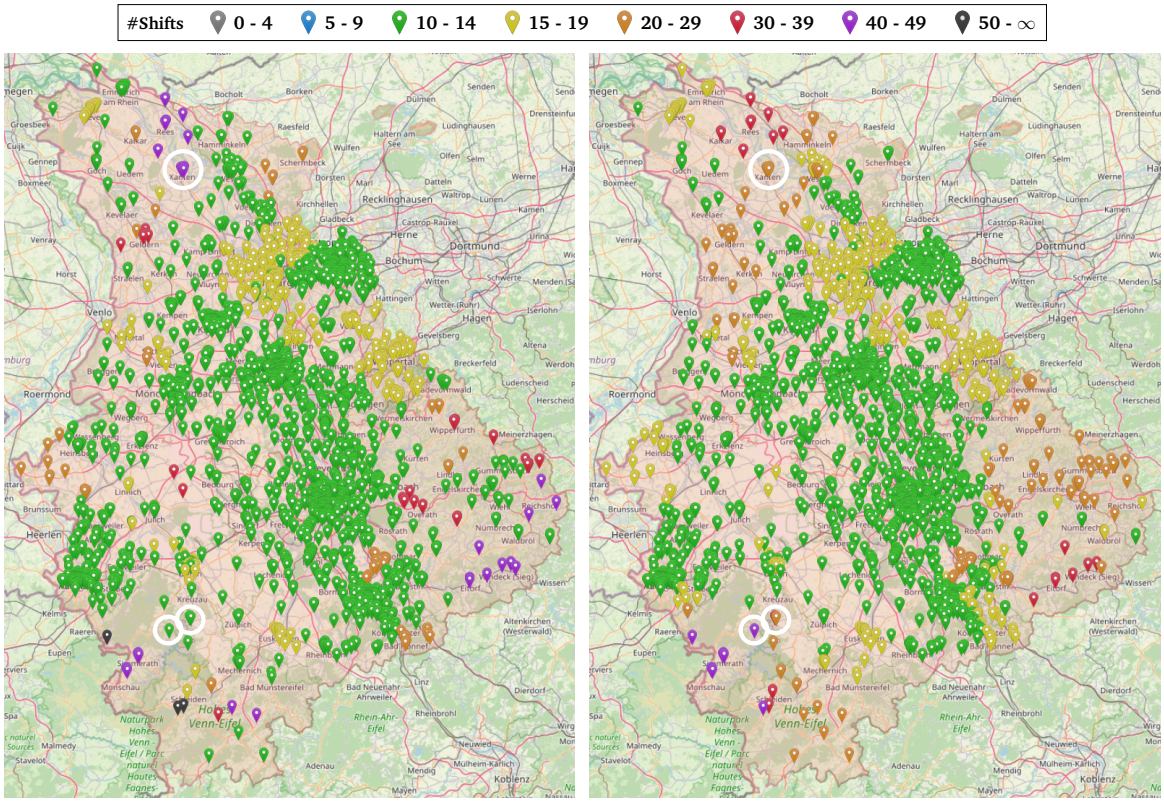


Figure 8.6. Geographical distribution of shifts. **Left:** plan for the OHP. **Right:** plan for the FOHP.

diagonal line, indicating that the distance for the corresponding municipality is shorter in the plan for the FOHP. Municipalities whose mean distance is significantly higher in the plan for the FOHP are those in which pharmacies with many shifts have been relieved.

The right-hand plot in Figure 8.5 shows the number of shifts assigned to each pharmacy in the plans for the OHP and the FOHP. We see that all pharmacies with more than 25 shifts in the plan for the OHP are assigned less shifts in the plan for the FOHP. In particular, the redistribution results in a reduction of the maximum number of shifts from 61 to 41. In return, some pharmacies that are assigned few shifts in the plan for the OHP receive more shifts in the plan for the FOHP in order to relieve other pharmacies. Figure 8.6 shows the geographical aspect of the redistribution. We consider again the case of Monschau, located in the southwest of the planning area, which we already analyzed in Sections 7.4.4 and 8.6.1. In the plan for the OHP, the pharmacies covering Monschau are assigned up to 61 shifts, while they are assigned at most 41 shifts in the plan for the FOHP. This is primarily due to an increase in the number of shifts of a pharmacy in the municipality of Nideggen, which is located in the left of the southwestern circles in Figure 8.6. This pharmacy is now assigned 41 shifts instead of 14 shifts to relieve the other pharmacies covering Monschau. The approach of using the min-max fair solution as an orientation thus resolves a crucial issue of the OHP, which is that there was no incentive to evenly distribute shifts across municipalities.

We began this chapter by quoting Karsu and Morton [60]: “lexicographic approaches are very inequality averse and considered by some studies as the “most equitable” solution.” With our

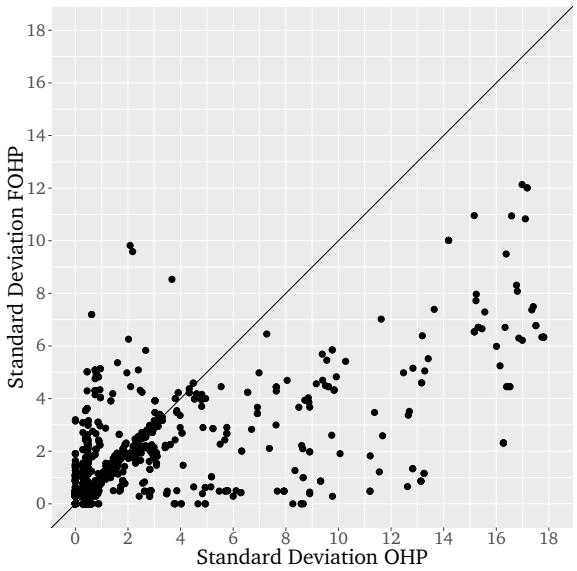


Figure 8.7. Comparison of the standard deviations $\sigma(p)$ of the number of shifts around pharmacies $p \in \mathcal{P}$ for the plans for OHP and FOHP.

plan for the FOHP being close to the min-max fair solution to the ALOHP, and the min-max fair solution being "fairer" than any lexicographically optimal plan, we have successfully constructed an out-of-hours plan that is almost lexicographically optimal. Thus, if we adopt the point of view that lexicographic fair solutions are "most equitable", then our plan can be regarded almost maximally fair. This global fairness is important from a centralized planning perspective, but it is not sufficient for gaining acceptance among pharmacists if the plan is locally unbalanced: If pharmacists have to work many shifts more than their geographically closest competitors, then they may feel treated unfairly. In order to quantify local balance, we compare the number of shifts of each pharmacy with those of the surrounding pharmacies. For this, let $N[p] \in \binom{\mathcal{P}}{11}$ be the set of pharmacies containing p and the ten closest pharmacies $p' \in \mathcal{P} \setminus \{p'\}$ with respect to the distance $\delta(p, p')$. We consider the standard deviation $\sigma(p) = \sqrt{\frac{\sum_{p' \in N[p]} \left(\mu(p) - \sum_{t \in [T]} x_{p't}\right)^2}{|N[p]|}}$ of the number of shifts of pharmacies within $N[p]$ for measuring the balance around p, where $\mu(p) = \frac{\sum_{p' \in N[p]} \sum_{t \in [T]} x_{p't}}{|N[p]|}$ is the mean number of shifts.

Looking at the distribution of shifts in Figure 8.6, the plan for the FOHP appears to be more homogeneous than the plan for the OHP. Indeed, the mean standard deviation $\frac{\sum_{p \in N[p]} \sigma(p)}{|\mathcal{P}|}$ for the former plan is 0.96, while it is 1.32 for the latter. The plan for the FOHP thus yields a better balancing on an aggregated level. Figure 8.7 shows the standard deviation $\sigma(p)$ of each pharmacy in both plans. We see that the maximum standard deviation is roughly 12 in the plan for the FOHP and almost 18 in the plan for the OHP. The largest improvement in balance is achieved for four pharmacies in Xanten, which are marked in Figure 8.6 in the

north of the planning area, whose standard deviation decreases from 16.3 in the plan for the OHP down to 2.3 in the plan for the FOHP. The largest increase in the standard deviation is from 2.1 up to 9.8, attained by two pharmacies in Nideggen, which are located in the right of the southwestern circles in Figure 8.6. This increase results from the more even distribution of shifts in the southwestern Eifel region: The pharmacies in Nideggen are now assigned more shifts to relieve other pharmacies, and thus the border between the highly burdened pharmacies in the Eifel region and the less burdened pharmacies outside the Eifel region now passes through Nideggen. Therefore, the increased standard deviation is not an expression of a worse local balancing but, in fact, a result of a smoother transition between the highly burdened and the less burdened pharmacies.

Whether the balancing in the plan for the FOHP is sufficient needs to be decided by practitioners at the chamber of pharmacists, who are mandated to plan the out-of-hours service. If they decide that some pairs of pharmacies are still too unbalanced, then our approaches offer multiple possibilities to achieve a higher balance: We may apply stronger balancing constraints, smooth the coverage with a negligible deterioration of the quality of coverage, or even adjust the cover radii if we are willing to accept a worse quality of coverage. If practitioners decide that a difference in the number of shifts is acceptable, then they can easily justify the assignment: Due to the orientation towards the min-max fair solution, we know that a pharmacy can only be relieved if we worsen the coverage or assign more shifts to pharmacies with an already higher or equal burden.

8.7 Conclusion

In the preceding chapter, we demonstrated that our solution approaches enable us to compute out-of-hours plans in short time that are efficient with respect to the total number of shifts. However, we observed that the OHP only provides an indirect control regarding an even distribution of shifts among pharmacies, which results in unfair out-of-hours plans. We addressed this issue in the current chapter by integrating fairness directly into our model. We decided that the concept of lexicographic fairness is particularly suitable for achieving an even distribution of shifts, as it relieves the pharmacies with the highest burden as much as possible but does not result in an assignment of many additional shifts.

Since the computation of lexicographic fair plans is not practical, we considered the simpler ALOHP, which arises from the original problem via aggregation over the time horizon. A lexicographic fair solution to the ALOHP reflects the assignment of a number of shifts in an idealized planning, which we use as an orientation for computing fair actual out-of-hours plans. In order to solve the ALOHP, we considered the concept of min-max fairness, which is stricter than lexicographic fairness and provides useful additional structure. We were able to generalize and prove several statements from the literature and showed that we can compute min-max fair solutions to the ALOHP within seconds with a water-draining algorithm.

In our case study for the out-of-hours service in the area North Rhine, we observed that we can compute out-of-hours plans that almost match the min-max fair solutions in terms of the number of assigned shifts for each pharmacy. The computed plans thus have a high level of fairness, as the min-max fair solutions itself can be seen as maximally fair. Furthermore, the orientation towards the min-max fair solution enhances the explainability of our planning to highly burdened pharmacists, as these can only be relieved by relaxing the coverage or assigning more shifts to pharmacies with an already higher or equal burden.

In addition to their value for computing fair out-of-hours plans, min-max fair solutions are also useful for analyzing the impact of adjusting planning constraints in a decision support environment. Here, the min-max fair solutions serve as reliable predictors for the number of shifts assigned, which can be computed within seconds. We demonstrated the potential of this approach by applying some rather conservative but effective changes to the model. First, we smoothed the coverage of rural municipalities with a minor deterioration of the quality of coverage. Second, we tightened the coverage of municipalities with many surrounding pharmacies, resulting in a negligible increase in the number of shifts. Decision-makers might choose to apply stricter adjustments. If they aim to improve the coverage of a certain municipality, then they may reduce the cover radius and directly analyze the effect on the assignment of shifts using the min-max fair solution. Conversely, if they wish to reduce the number of shifts of a pharmacy to a certain level at the expense of a worse quality of coverage, then they can derive from the water-draining algorithm which coverage constraints need to be adjusted to what extend. Hence, we not only computed fairer plans in this chapter but also developed tools to adapt the planning more easily to the needs of decision-makers.

For future work, it would be interesting to extend the fairness concept by considering different types of days. Out-of-hours shifts on Sundays or public holidays are less desirable than shifts on working days, and should thus be distributed evenly among the pharmacies. An easy approach would be to assign each pharmacy a fraction of $\frac{X_p^*}{T}$ of all shifts on Sundays and holidays, where X_p^* is the number of shifts in the min-max fair solution. However, this neglects that the demand for out-of-hours pharmacies in cities is higher on Sundays and holidays, and thus the corresponding constraints (OHP.c) require the assignment of an above-average number of shifts on these days. It therefore remains to be seen whether there exists an elegant way of incorporating different types of days into the concept of using min-max fair solutions.

Bibliography

[1] ABDA - Federal Union of German Associations of Pharmacists. *German Pharmacies: Figures Data Facts*. URL: https://www.abda.de/fileadmin/user_upload/assets/ZDF/ZDF-2023/ABDA_ZDF_2023_Brosch_english.pdf.

[2] T. Achterberg. "Constraint Integer Programming". In: *Ph. D. Thesis, Technische Universitat Berlin* (2007). DOI: 10.14279/depositonce-1634.

[3] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. "Presolve reductions in mixed integer programming". In: *INFORMS Journal on Computing* 32.2 (2020), pp. 473–506. DOI: 10.1287/ijoc.2018.0857.

[4] T. Achterberg, T. Koch, and A. Martin. "Branching rules revisited". In: *Operations Research Letters* 33.1 (2005), pp. 42–54. DOI: 10.1016/j.orl.2004.04.002.

[5] T. Achterberg and R. Wunderling. "Mixed integer programming: Analyzing 12 years of progress". In: *Facets of combinatorial optimization*. Springer, 2013, pp. 449–481. DOI: 10.1007/978-3-642-38189-8_18.

[6] E. Álvarez-Miranda, I. Ljubić, and P. Toth. "A note on the Bertsimas & Sim algorithm for robust combinatorial optimization problems". In: *4OR* 11.4 (2013), pp. 349–360. DOI: 10.1007/s10288-013-0231-6.

[7] *Apothekenbetriebsordnung in der Fassung der Bekanntmachung vom 26. September 1995 (BGBl. I S. 1195), die zuletzt durch Artikel 4a des Gesetzes vom 19. Juli 2023 (BGBl. 2023 I Nr. 197) geändert worden ist*. URL: https://www.gesetze-im-internet.de/apobetro_1987/BJNR005470987.html.

[8] Apothekerkammer Nordrhein. "Kammer im Gespräch". In: Sonderausgabe Herbst 2013 (2013).

[9] A. Atamtürk. "Strong formulations of robust mixed 0–1 programming". In: *Mathematical Programming* 108.2-3 (2006), pp. 235–250. DOI: 10.1007/s10107-006-0709-5.

[10] A. Atamtürk, G. L. Nemhauser, and M. W. Savelsbergh. "Conflict graphs in solving integer programming problems". In: *European Journal of Operational Research* 121.1 (2000), pp. 40–55. DOI: 10.1016/S0377-2217(99)00015-6.

[11] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. "Branching and bounds tightening techniques for non-convex MINLP". In: *Optimization Methods & Software* 24.4-5 (2009), pp. 597–634. DOI: 10.1080/10556780903087124.

[12] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton Series in Applied Mathematics. USA: Princeton university press, 2009. DOI: 10.1515/9781400831050.

[13] A. Ben-Tal and A. Nemirovski. "Robust convex optimization". In: *Mathematics of Operations Research* 23.4 (1998), pp. 769–805. DOI: 10.1287/moor.23.4.769.

[14] A. Ben-Tal and A. Nemirovski. "Robust solutions of linear programming problems contaminated with uncertain data". In: *Mathematical Programming* 88.3 (2000), pp. 411–424. DOI: 10.1007/PL00011380.

[15] A. Ben-Tal and A. Nemirovski. “Robust solutions of uncertain linear programs”. In: *Operations Research Letters* 25.1 (1999), pp. 1–13. DOI: 10.1016/S0167-6377(99)00016-4.

[16] T. Berthold. “Measuring the impact of primal heuristics”. In: *Operations Research Letters* 41.6 (2013), pp. 611–614. DOI: 10.1016/j.orl.2013.08.007.

[17] D. Bertsekas and R. Gallager. *Data networks*. Second. Prentice Hall, 1992. URL: https://web.mit.edu/dimitrib/www/datanets.html.

[18] D. Bertsimas and D. den Hertog. *Robust and Adaptive Optimization*. Dynamic Ideas LLC, 2022. ISBN: 9781733788526. URL: https://books.google.de/books?id=V_RPzwEACAAJ.

[19] D. Bertsimas, D. B. Brown, and C. Caramanis. “Theory and applications of robust optimization”. In: *SIAM Review* 53.3 (2011), pp. 464–501. DOI: 10.1137/080734510.

[20] D. Bertsimas, I. Dunning, and M. Lubin. “Reformulation versus cutting-planes for robust optimization”. In: *Computational Management Science* 13.2 (2016), pp. 195–217. DOI: 10.1007/s10287-015-0236-z.

[21] D. Bertsimas, V. F. Farias, and N. Trichakis. “The price of fairness”. In: *Operations research* 59.1 (2011), pp. 17–31. DOI: https://doi.org/10.1287/opre.1100.0865.

[22] D. Bertsimas and M. Sim. “Robust discrete optimization and network flows”. In: *Mathematical Programming* 98.1-3 (2003), pp. 49–71. DOI: 10.1007/s10107-003-0396-4.

[23] D. Bertsimas and M. Sim. “The price of robustness”. In: *Operations Research* 52.1 (2004), pp. 35–53. DOI: 10.1287/opre.1030.0065.

[24] R. Bixby and E. Rothberg. “Progress in computational mixed integer programming—a look back from the other side of the tipping point”. In: *Annals of Operations Research* 149.1 (2007), pp. 37–41. DOI: 10.1007/s10479-006-0091-y.

[25] K. H. Borgwardt. “The average number of pivot steps required by the simplex-method is polynomial”. In: *Zeitschrift für Operations Research* 26 (1982), pp. 157–177. DOI: https://doi.org/10.1007/BF01917108.

[26] J. Borsch. *Keine Notdienstkreise mehr in Hessen*. Deutsche Apotheker Zeitung. 2023. URL: https://www.deutsche-apotheker-zeitung.de/news/artikel/2023/08/11/keine-notdienstkreise-mehr-in-hessen.

[27] S. S. Brito and H. G. Santos. “Preprocessing and cutting planes with conflict graphs”. In: *Computers & Operations Research* 128 (2021), p. 105176. DOI: 10.1016/j.cor.2020.105176.

[28] E. K. Burke, P. De Causmaecker, G. V. Berghe, and H. Van Landeghem. “The State of the Art of Nurse Rostering”. In: *Journal of scheduling* 7.6 (2004), pp. 441–499. DOI: 10.1023/B:JOSH.0000046076.75950.0b.

[29] C. Büsing, T. Gersing, and A. Koster. “Recycling Inequalities for Robust Combinatorial Optimization with Budget Uncertainty”. In: *Integer Programming and Combinatorial Optimization. IPCO 2023*. Ed. by A. Del Pia and V. Kaibel. Vol. 13904. Lecture Notes in Computer Science. Cham: Springer, 2023, pp. 58–71. DOI: 10.1007/978-3-031-32726-1_5.

[30] C. Büsing, T. Gersing, and A. Koster. “Recycling Inequalities for Robust Combinatorial Optimization with Budgeted Uncertainty”. In: *In Revision at Mathematical Programming Series B, Preprint available at Optimization Online* (2023). URL: https://optimization-online.org/?p=23462.

[31] C. Büsing, T. Gersing, and A. M. Koster. “A branch and bound algorithm for robust binary optimization with budget uncertainty”. In: *Mathematical Programming Computation* (2023). DOI: 10.1007/s12532-022-00232-2.

[32] C. Büsing, T. Gersing, and A. M. Koster. “Planning out-of-hours services for pharmacies”. In: *Operations Research for Health Care* 27 (2020), p. 100277. ISSN: 2211-6923. DOI: https://doi.org/10.1016/j.orhc.2020.100277.

[33] M. Cardei and D.-Z. Du. “Improving Wireless Sensor Network Lifetime through Power Aware Organization”. In: *Wireless Networks* 11.3 (2005), pp. 333–340. DOI: 10.1007/s11276-005-6615-6.

[34] M. W. Carter and S. D. Lapierre. “Scheduling emergency room physicians”. In: *Health care management science* 4.4 (2001), pp. 347–360.

[35] G. Ceyhan and Ö. Özpeynirci. “A branch and price algorithm for the pharmacy duty scheduling problem”. In: *Computers & Operations Research* 72 (2016), pp. 175–182. DOI: 10.1016/j.cor.2016.02.007.

[36] T. Christof and A. Loebel. *POlyhedron Representation Transformation Algorithm (PORTA)*. https://porta.zib.de/. Accessed: 2023-10-24.

[37] M. Conforti, G. Cornuéjols, G. Zambelli, et al. *Integer programming*. Vol. 271. Cham: Springer, 2014. DOI: 10.1007/978-3-319-11008-0.

[38] E. Danna, S. Mandal, and A. Singh. “A practical algorithm for balancing the max-min fairness and throughput objectives in traffic engineering”. In: *2012 Proceedings IEEE INFOCOM*. 2012, pp. 846–854. DOI: 10.1109/INFCOM.2012.6195833.

[39] G. B. Dantzig. “Maximization of a linear function of variables subject to linear inequalities”. In: *Activity analysis of production and allocation* 13 (1951), pp. 339–347.

[40] J. Edmonds. “Submodular Functions, Matroids, and Certain Polyhedra”. In: *Combinatorial Optimization — Eureka, You Shrink!: Papers Dedicated to Jack Edmonds 5th International Workshop Aussois, France, March 5–9, 2001 Revised Papers*. Ed. by M. Jünger, G. Reinelt, and G. Rinaldi. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 11–26. ISBN: 978-3-540-36478-8. DOI: 10.1007/3-540-36478-1_2. URL: https://doi.org/10.1007/3-540-36478-1_2.

[41] A. Ernst, H Jiang, M Krishnamoorthy, and D Sier. “Staff scheduling and rostering: A review of applications, methods and models”. In: *European Journal of Operational Research* 153.1 (2004). Timetabling and Rostering, pp. 3–27. ISSN: 0377-2217. DOI: https://doi.org/10.1016/S0377-2217(03)00095-X.

[42] M. Fischetti, F. Glover, and A. Lodi. “The Feasibility Pump”. In: *Mathematical Programming* 104.1 (2005), pp. 91–104. DOI: 10.1007/s10107-004-0570-3.

[43] M. Fischetti and M. Monaci. “Cutting plane versus compact formulations for uncertain (integer) linear programs”. In: *Mathematical Programming Computation* 4.3 (2012), pp. 239–273. DOI: 10.1007/s12532-012-0039-y.

[44] V. Gabrel, C. Murat, and A. Thiele. “Recent advances in robust optimization: An overview”. In: *European Journal of Operational Research* 235.3 (2014), pp. 471–483. DOI: 10.1016/j.ejor.2013.09.036.

[45] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Mathematical Sciences Series. W. H. Freeman & Co., 1979. URL: https://books.google.de/books?id=fjxGAQAAIAAJ.

[46] T. Gersing. *Algorithms for Robust Binary Optimization*. Dec. 2022. DOI: 10.5281/zenodo.7463371.

[47] T. Gersing. *Computational Results: Algorithms for Robust Combinatorial Optimization with Budgeted Uncertainty*. Jan. 2024. DOI: 10.5281/zenodo.10469044.

[48] T. Gersing, C. Büsing, and A. Koster. *Benchmark Instances for Robust Combinatorial Optimization with Budgeted Uncertainty*. Dec. 2022. DOI: 10.5281/zenodo.7419028.

[49] A. Gleixner, G. Hendel, G. Gamrath, et al. "MIPLIB 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library". In: *Mathematical Programming Computation* (2021). DOI: 10.1007/s12532-020-00194-3.

[50] Z. Gu, G. L. Nemhauser, and M. W. Savelsbergh. "Sequence independent lifting in mixed integer programming". In: *Journal of Combinatorial Optimization* 4 (2000), pp. 109–129. DOI: 10.1023/A:1009841107478.

[51] Gurobi Optimization, LLC. *Advanced user scaling*. https://www.gurobi.com/documentation/9.5/refman/advanced_user_scaling.html. Accessed: 2022-09-27.

[52] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual, Version 9.5*. 2022. URL: http://www.gurobi.com.

[53] C. Hansknecht, A. Richter, and S. Stiller. "Fast robust shortest path computations". In: *18th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2018)*. Vol. 65. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 5:1–5:21. DOI: 10.4230/OASIcs.ATMOS.2018.5.

[54] C. Hohmann-Jeddi. *System für gerechtere Notdienstpläne kommt*. Pharmazeutische Zeitung. 2022. URL: https://www.pharmazeutische-zeitung.de/system-fuer-gerechtere-notdienstplaene-kommt-137095/.

[55] IBM. *IBM ILOG CPLEX Optimization Studio V12.6.3*. URL: https://www.ibm.com/products/ilog-cplex.

[56] D. S. Johnson and M. A. Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*. Vol. 26. USA: American Mathematical Soc., 1996. ISBN: 978-0-8218-6609-2.

[57] S. Joung and S. Park. "Robust Mixed 0-1 Programming and Submodularity". In: *INFORMS Journal on Optimization* 3.2 (2021), pp. 183–199. DOI: 10.1287/ijoo.2019.0042.

[58] N. Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Proceedings of the sixteenth annual ACM symposium on Theory of computing*. 1984, pp. 302–311. DOI: https://doi.org/10.1145/800057.808695.

[59] R. M. Karp. "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9.

[60] Ö. Karsu and A. Morton. "Inequity averse optimization in operational research". In: *European Journal of Operational Research* 245.2 (2015), pp. 343–359. DOI: 10.1016/j.ejor.2015.02.035.

[61] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Berlin Heidelberg, 2004. DOI: 10.1007/978-3-540-24777-7.

[62] L. G. Khachiyan. "A polynomial algorithm in linear programming (english translation)". In: *Soviet Mathematics Doklady*. Vol. 20. 1979, pp. 191–194.

[63] F. Kocatürk and Ö. Özpeynirci. "Variable neighborhood search for the pharmacy duty scheduling problem". In: *Computers & Operations Research* 51 (2014), pp. 218–226. DOI: 10.1016/j.cor.2014.06.001.

[64] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer Berlin Heidelberg, 2018. DOI: 10.1007/978-3-662-56039-6.

[65] A. M. Koster and M. Kutschka. "Network design under demand uncertainties: A case study on the abilene and GEANT network data". In: *Photonic Networks, 12. ITG Symposium*. VDE. 2011, pp. 1–8.

[66] P Kouvelis and G Yu. *Robust discrete optimization and its applications*. USA: Springer, 1997. DOI: 10.1007/978-1-4757-2620-6.

[67] S. Kuhnke, P. Richter, F. Kepp, et al. "Robust optimal aiming strategies in central receiver systems". In: *Renewable Energy* 152 (2020), pp. 198–207. DOI: 10.1016/j.renene.2019.11.118.

[68] F. Kılıç and N. Uncu. "Modified swarm intelligence algorithms for the pharmacy duty scheduling problem". In: *Expert Systems with Applications* 202 (2022), p. 117246. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117246.

[69] A. Land and A. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica: Journal of the Econometric Society* (1960), pp. 497–520. DOI: 10.2307/1910129.

[70] T. Lee and C. Kwon. "A short note on the robust combinatorial optimization problems with cardinality constrained uncertainty". In: *4OR* 12.4 (2014), pp. 373–378. DOI: 10.1007/s10288-014-0270-7.

[71] N. Leithäuser, D. Adelhütte, K. Braun, et al. "Decision-support systems for ambulatory care, including pandemic requirements: using mathematically optimized solutions". In: *BMC Medical Informatics and Decision Making* 22.1 (2022), pp. 1–20. DOI: https://doi.org/10.1186/s12911-022-01866-x.

[72] J. T. Linderoth and M. W. Savelsbergh. "A computational study of search strategies for mixed integer programming". In: *INFORMS Journal on Computing* 11.2 (1999), pp. 173–187. DOI: 10.1287/ijoc.11.2.173.

[73] A. Lodi and G. Zarpellon. "On learning and branching: a survey". In: *Top* 25.2 (2017), pp. 207–236. DOI: 10.1007/s11750-017-0451-6.

[74] F. Margot. "Symmetry in Integer Linear Programming". In: *50 Years of Integer Programming 1958-2008*. Ed. by M. Jünger, T. Liebling, D. Naddef, et al. Springer Berlin Heidelberg, 2009, pp. 647–686. DOI: 10.1007/978-3-540-68279-0_17.

[75] R. R. Meyer. "On the existence of optimal solutions to integer and mixed-integer programming problems". In: *Mathematical Programming* 7 (1974), pp. 223–235. DOI: 10.1007/BF01585518.

[76] M. Monaci and U. Pferschy. "On the robust knapsack problem". In: *SIAM Journal on Optimization* 23.4 (2013), pp. 1956–1982. DOI: 10.1137/120880355.

[77] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell. "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning". In: *Discrete Optimization* 19 (2016), pp. 79–102. DOI: 10.1016/j.disopt.2016.01.005.

[78] J. R. Munkres. *Topology*. Second. Pearson Education, 2000.

[79] W. Ogryczak and T. Śliwiński. "On Direct Methods for Lexicographic Min-Max Optimization". In: *Computational Science and Its Applications - ICCSA 2006*. Ed. by M. Gavrilova, O. Gervasi, V. Kumar, et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 802–811. DOI: https://doi.org/10.1007/11751595_85.

[80] Ö. Özpeynirci and E. Ağlamaz. “Pharmacy duty scheduling problem”. In: *International Transactions in Operational Research* 23.3 (2016), pp. 459–480. DOI: 10.1111/itor.12204.

[81] M. W. Padberg. “On the facial structure of set packing polyhedra”. In: *Mathematical programming* 5.1 (1973), pp. 199–215. DOI: 10.1007/BF01580121.

[82] K. Park and K. Lee. “A note on robust combinatorial optimization problem”. In: *Management Science and Financial Engineering* 13.1 (2007), pp. 115–119.

[83] D. Pfeil, J. Pieck, and H. Blume. *Apothekenbetriebsordnung, Kommentar*. 11. Ergänzungslieferung 2014. Govi Verlag. ISBN: 978-3-7741-0077-0.

[84] B. Radunovic and J.-Y. Le Boudec. “A Unified Framework for Max-Min and Min-Max Fairness With Applications”. In: *IEEE/ACM Transactions on Networking* 15.5 (2007), pp. 1073–1083. DOI: 10.1109/TNET.2007.896231.

[85] J. Rawls. *A Theory of Justice*. Harvard University Press, 1971.

[86] “Richtlinien für die Dienstbereitschaft der öffentlichen Apotheken im Bereich der Apothekerkammer Nordrhein vom 19. Juni 2013”. In: vol. 30. Deutsche Apotheker Zeitung, 2013.

[87] S. Sarkar and L. Tassiulas. “Fair bandwidth allocation for multicasting in networks with discrete feasible set”. In: *IEEE Transactions on Computers* 53.7 (), pp. 785–797. DOI: 10.1109/TC.2004.30.

[88] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[89] H. D. Sherali and W. P. Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*. Vol. 31. New York: Springer, 2013. DOI: 10.1007/978-1-4757-4388-3.

[90] A. L. Soyster. “Convex programming with set-inclusive constraints and applications to inexact linear programming”. In: *Operations Research* 21.5 (1973), pp. 1154–1157. DOI: 10.1287/opre.21.5.1154.

[91] E. Speakman and J. Lee. “On branching-point selection for trilinear monomials in spatial branch-and-bound: the hull relaxation”. In: *Journal of Global Optimization* 72.2 (2018), pp. 129–153. DOI: 10.1007/s10898-018-0620-7.

[92] *The SCIP Optimization Suite 7.0*. ZIB-Report. Zuse Institut Berlin, 2020. URL: https://optimization-online.org/?p=16345.

[93] F. Wilcoxon. “Individual Comparisons by Ranking Methods”. In: *Biometrics* 1.6 (1945), pp. 80–83. DOI: 10.2307/3001968.

[94] Wissenschaftliche Dienste Deutscher Bundestag. “Grundgesetzlicher Anspruch auf gesundheitliche Versorgung”. In: WD 3 -3000 -089/15 (Apr. 2015). URL: https://www.bundestag.de/resource/blob/405508/4dd5bf6452b5b3b824d8de6efdad39dd/wd-3-089-15-pdf-data.pdf.

[95] L. A. Wolsey. “Facets and strong valid inequalities for integer programs”. In: *Operations research* 24.2 (1976), pp. 367–372. DOI: 10.1287/opre.24.2.367.

[96] L. A. Wolsey and G. L. Nemhauser. *Integer and combinatorial optimization*. Vol. 55. John Wiley & Sons, 1999. DOI: 10.1002/9781118627372.

[97] V. Xinying Chen and J. Hooker. “A guide to formulating fairness in an optimization model”. In: *Annals of Operations Research* 326 (2023), pp. 581–619. DOI: https://doi.org/10.1007/s10479-023-05264-y.

[98] E. Zemel. “Lifting the facets of zero–one polytopes”. In: *Mathematical Programming* 15 (1978), pp. 268–277. DOI: 10.1007/BF01609032.

A

Computational Results for Chapter 8

The following tables list the exact computational result from Section 7.4.

Table A.1. Number of shifts assigned, sum of violations of balancing constrains and computation time required for each solution computed by the variants ROL, ROL-Super, ROL-Delete, and ROL-SuperDelete of the rolling horizon approach with numbers of breakpoints $\ell \in [51]$.

	ROL			ROL-Super			ROL-Delete			ROL-SuperDelete		
breakpoints	shifts	violation	time	shifts	violation	time	shifts	violation	time	shifts	violation	time
1	-	-	10,003	31,356	823	10,003	-	-	10,003	31,356	823	10,003
2	-	-	10,003	31,997	962	10,003	-	-	10,003	31,834	971	10,003
3	-	-	10,004	32,746	1,053	10,002	31,726	1,001	10,003	32,002	963	10,002
4	32,929	1,199	10,002	32,129	762	10,002	-	-	10,007	31,376	695	10,001
5	31,412	902	10,002	32,611	11	10,001	-	-	10,005	31,317	2	10,001
6	33,314	1,230	10,002	32,815	908	10,001	31,388	1,038	10,002	31,267	0	9,326
7	33,436	1,165	10,002	32,971	724	10,001	31,235	867	10,002	31,008	557	10,001
8	33,507	810	10,001	32,220	633	10,001	31,284	958	10,001	30,320	2	10,001
9	-	-	10,012	32,107	493	10,001	32,607	3	10,001	30,292	0	10,001
10	34,119	2	10,001	32,040	406	10,001	30,664	442	10,002	30,165	1	10,001
11	34,102	2	10,001	31,856	506	10,001	30,587	334	10,001	30,186	0	6,984
12	33,203	438	10,001	31,267	276	10,001	30,244	3	10,001	30,186	0	4,823
13	33,209	491	10,001	31,913	2	10,001	30,219	0	8,570	30,192	0	2,781
14	33,050	354	10,001	30,836	0	8,224	30,170	0	8,012	30,174	0	3,098
15	32,334	360	10,001	30,746	0	7,683	30,199	0	6,301	30,194	0	2,429
16	34,864	433	10,001	31,019	0	7,701	30,201	0	5,693	30,209	0	2,359
17	34,727	104	10,001	30,824	0	4,433	30,211	0	5,342	30,217	2	1,815
18	40,463	25,524	10,001	30,784	1	5,123	30,220	0	3,906	30,197	0	1,891
19	38,819	268	10,001	30,496	0	4,340	30,193	0	3,736	30,190	0	1,653
20	39,750	26,063	10,001	30,683	0	4,444	30,210	0	3,521	30,224	0	1,846
21	30,878	0	9,736	30,803	0	4,289	30,217	0	2,924	30,208	0	1,498
22	37,668	0	8,204	30,833	0	3,429	30,215	0	2,420	30,208	0	1,368
23	31,021	0	8,296	30,992	0	3,769	30,217	0	2,194	30,220	0	1,372
24	31,140	1	6,450	31,004	0	3,054	30,221	0	2,477	30,220	0	1,260
25	30,826	0	5,897	31,068	0	3,402	30,226	0	1,913	30,214	0	1,611
26	30,872	0	6,229	30,813	0	3,230	30,233	0	2,292	30,217	0	1,303
27	30,879	0	4,732	30,851	0	3,260	30,236	0	1,909	30,233	0	1,195
28	30,981	0	5,635	30,939	0	3,010	30,229	0	1,889	30,222	0	1,070
29	30,828	0	4,812	30,861	0	2,380	30,224	0	1,722	30,215	0	965
30	30,997	0	4,132	30,991	0	2,033	30,241	0	1,406	30,228	0	971
31	31,130	0	4,584	31,088	0	2,748	30,253	0	1,406	30,235	0	916
32	31,195	0	4,484	31,164	0	1,865	30,277	0	1,369	30,241	0	884
33	31,222	0	3,650	31,072	0	1,979	30,255	0	1,330	30,254	0	898
34	30,948	0	3,543	30,956	1	1,651	30,250	0	1,502	30,252	0	800
35	31,009	0	3,353	31,021	0	1,985	30,264	0	1,314	30,258	0	831
36	31,025	0	3,589	31,005	0	2,269	30,249	1	1,515	30,250	0	894
37	31,065	1	3,257	31,092	0	1,801	30,240	0	1,548	30,253	0	871
38	31,192	0	3,011	31,154	0	1,987	30,257	0	1,439	30,255	0	858
39	30,876	0	3,207	30,898	0	1,836	30,249	0	1,423	30,246	0	807
40	31,019	0	2,682	31,230	0	1,586	30,260	0	1,325	30,262	0	838
41	31,133	0	2,732	31,086	0	1,762	30,251	0	1,380	30,249	0	796
42	31,252	0	2,115	31,235	0	1,363	30,246	0	1,127	30,250	0	838
43	31,390	0	1,903	31,359	0	1,265	30,254	1	1,121	30,247	0	758
44	31,339	0	1,991	31,402	0	1,307	30,245	0	1,180	30,261	0	774
45	31,475	0	1,404	31,503	0	1,005	30,272	0	1,100	30,271	0	803
46	31,521	0	1,581	31,551	0	1,045	30,284	0	1,093	30,278	0	816
47	31,567	0	1,293	31,591	0	1,017	30,298	0	1,085	30,294	0	817
48	31,617	0	1,428	31,609	1	1,072	30,289	0	1,112	30,274	0	796
49	31,664	0	1,585	31,475	0	1,084	30,305	1	1,150	30,269	0	764
50	31,474	0	1,312	31,485	0	1,033	30,308	1	1,162	30,307	0	788
51	31,538	1	1,347	31,455	0	997	30,307	1	1,185	30,300	0	840

Table A.2. Number of shifts assigned, sum of violations of balancing constrains and computation time required for different minimum numbers of shifts $\underline{s} \in [15]_0$ using the approaches ROL11-SuperDelete and ROL23-SuperDelete.

		ROL11-SuperDelete			ROL23-SuperDelete		
min. number shifts $\underline{s}$	relaxation	shifts	violation	time	shifts	violation	time
0	25,835	27,403	651	10,001	26,520	0	8,942
1	26,083	29,289	0	9,104	26,585	0	9,116
2	26,375	27,642	733	10,001	26,622	0	8,887
3	26,715	28,767	0	9,095	26,941	0	8,595
4	27,080	29,280	0	9,254	27,307	0	6,122
5	27,501	28,689	1	10,001	27,712	0	3,606
6	27,966	28,759	517	10,001	28,139	0	2,509
7	28,443	30,821	2	9,182	28,613	0	2,186
8	28,929	29,373	328	10,001	29,094	0	1,871
9	29,451	29,552	0	8,339	29,600	0	1,627
10	30,078	30,186	0	6,984	30,220	0	1,372
11	30,858	30,946	0	2,902	30,971	0	1,355
12	31,846	31,902	0	3,074	31,929	0	1,087
13	32,996	33,059	0	1,987	33,069	0	930
14	34,483	34,546	0	3,166	34,617	0	2,639
15	36,174	36,224	0	1,675	-	-	-

Table A.3. Number of shifts assigned, sum of violations of balancing constrains and computation time required for different balancing coefficients $b \in [5]_0$ using the approaches ROL11-SuperDelete and ROL23-SuperDelete.

		ROL11-SuperDelete			ROL23-SuperDelete		
balancing coeff. b	relaxation	shifts	violation	time	shifts	violation	time
0	30,110	30,687	476	10,001	31,095	2	4,605
1	30,078	30,186	0	6,984	30,220	0	1,372
2	30,050	30,142	0	5,489	30,181	0	1,131
3	30,025	30,111	0	3,980	30,131	0	1,128
4	30,002	30,083	0	3,481	30,102	0	1,065
5	29,980	30,054	0	5,091	30,079	0	999

Eidesstattliche Erklärung

Ich, Timo Gersing,

erkläre hiermit, dass diese Dissertation und die darin dargelegten Inhalte die eigenen sind und selbstständig, als Ergebnis der eigenen originären Forschung, generiert wurden.

Hiermit erkläre ich an Eides statt

1. Diese Arbeit wurde vollständig oder größtenteils in der Phase als Doktorand dieser Fakultät und Universität angefertigt;
2. Sofern irgendein Bestandteil dieser Dissertation zuvor für einen akademischen Abschluss oder eine andere Qualifikation an dieser oder einer anderen Institution verwendet wurde, wurde dies klar angezeigt;
3. Wenn immer andere eigene- oder Veröffentlichungen Dritter herangezogen wurden, wurden diese klar benannt;
4. Wenn aus anderen eigenen- oder Veröffentlichungen Dritter zitiert wurde, wurde stets die Quelle hierfür angegeben. Diese Dissertation ist vollständig meine eigene Arbeit, mit der Ausnahme solcher Zitate;
5. Alle wesentlichen Quellen von Unterstützung wurden benannt;
6. Wenn immer ein Teil dieser Dissertation auf der Zusammenarbeit mit anderen basiert, wurde von mir klar gekennzeichnet, was von anderen und was von mir selbst erarbeitet wurde;
7. Teile dieser Arbeit wurden zuvor veröffentlicht und zwar in:

    - C. Büsing et al. “Recycling Inequalities for Robust Combinatorial Optimization with Budget Uncertainty”. In: *Integer Programming and Combinatorial Optimization. IPCO 2023*. Ed. by A. Del Pia and V. Kaibel. Vol. 13904. Lecture Notes in Computer Science. Cham: Springer, 2023, pp. 58–71. DOI: 10.1007/978-3-031-32726-1_5
    - C. Büsing et al. “Recycling Inequalities for Robust Combinatorial Optimization with Budgeted Uncertainty”. In: *In Revision at Mathematical Programming Series B, Preprint available at Optimization Online* (2023). URL: https://optimization-online.org/?p=23462
    - C. Büsing et al. “A branch and bound algorithm for robust binary optimization with budget uncertainty”. In: *Mathematical Programming Computation* (2023). DOI: 10.1007/s12532-022-00232-2

- C. Büsing et al. “Planning out-of-hours services for pharmacies”. In: *Operations Research for Health Care* 27 (2020), p. 100277. ISSN: 2211-6923. DOI: https://doi.org/10.1016/j.orhc.2020.100277
- N. Leithäuser et al. “Decision-support systems for ambulatory care, including pandemic requirements: using mathematically optimized solutions”. In: *BMC Medical Informatics and Decision Making* 22.1 (2022), pp. 1–20. DOI: https://doi.org/10.1186/s12911-022-01866-x

Timo Gersing
Aachen, 23. Mai 2024