

# Algorithms and Complexity Theory for Bilevel Problems with Nonlinear Lower Levels

Vom Fachbereich IV der Universität Trier zur Verleihung des akademischen Grades  
Doktor der Naturwissenschaften (Dr. rer. nat.) genehmigte Dissertation

von

Andreas Horländer

Trier, 2025

Betreuer:	Prof. Dr. Martin Schmidt
1. Berichterstatter:	Prof. Dr. Martin Schmidt
2. Berichterstatterin:	Prof. Dr. Ivana Ljubić
3. Berichterstatter:	Prof. Dr. Volker Schulz
Datum der Disputation:	02.10.2025



# Acknowledgement

I would like to sincerely thank my supervisor, Martin Schmidt, not only for his invaluable support and the numerous tips he shared with me, but also for his patience and the truly pleasant working environment he provided. It is remarkable how he always finds the time to listen and support me in any way he can, despite his full schedule. I am deeply grateful for the knowledge I gained and the development I experienced under his guidance and support.

I would also like to express my deepest gratitude to all of my co-authors, Immanuel Bomze, Jan Kronqvist, Ivana Ljubić, and Martin Schmidt for many enriching discussions, a warm and friendly atmosphere, and the valuable insights and expertise they contributed. Many thanks also to Ivana Ljubić for inviting me to Paris for collaboration.

My sincere thanks go to my thesis committee, Martin Schmidt, Ivana Ljubić, Volker Schulz, and Leonhard Frerick, for their time, availability, and the valuable feedback they provided.

Special thanks are due to the “Alliance of High Performance Computing Rheinland-Pfalz” (AHRP), which provided access to state-of-the-art computational resources for my experiments through the high-performance cluster Elwetritsch at TU Kaiserslautern. I also acknowledge the German Research Foundation (DFG) for the financial support of this research within the Research Training Group 2126 in Algorithmic Optimization (ALOP).

Thanks to all my colleagues at Trier University for the wonderful and unique time we had over the last years. Additionally, I would like to thank Yasmine Beck, Alois Duguet, Henri Lefebvre, Ioana Molan, Lucas Moschen, and Simon Stevens for proofreading parts of my thesis.

I would also like to thank my family and friends for their support.

Finally, I am deeply grateful to my beloved girlfriend, Laura. Her immense support and love over the last decade are something one can only dream of, and so is her patience, especially during the final months of my PhD. This thesis is dedicated to her.



# Curriculum Vitae

## Personal Information

Name Andreas Horländer  
Date of Birth June 27, 1997  
Place of Birth Wiesbaden, Germany

## Academic Career

04/2022 – 05/2025 **Research Assistant in the RTG ALOP**  
Speaker: Prof. Dr. Martin Schmidt

## Education

04/2022 **Master of Science, Applied Mathematics**  
Trier University, Trier, Germany

04/2020 – 04/2022  
03/2020 **Bachelor of Science, Applied Mathematics**  
10/2016 – 03/2020 Trier University, Trier, Germany

03/2016 **Abitur**  
08/2013 – 03/2016 Hofenfels-Gymnasium, Zweibrücken, Germany

07/2013 **Mittlere Reife**  
08/2007 – 07/2013 Mannlich-Realschule, Zweibrücken, Germany



# Abstract

Bilevel problems are optimization problems for which parts of the variables are constrained to be an optimal solution to another nested optimization problem. This structure renders bilevel problems particularly well-suited for modeling hierarchical decision-making processes. They are widely applicable in areas such as energy markets, transportation systems, security planning, and pricing. However, the hierarchical nature of these problems also makes them inherently challenging to solve, both in theory and in practice.

In this thesis, we study different nonlinear problem settings for the nested optimization problem. First, we focus on nonlinear but convex bilevel problems with purely integer variables. We propose a solution algorithm that uses a branch-and-cut framework with tailored cutting planes. We prove correctness and finite termination of the method under suitable assumptions and put it into context of existing literature. Moreover, we provide an extensive numerical study to showcase the applicability of our method and we compare it to the state-of-the-art approach for a less general setting on suitable instances from the literature. Furthermore, we discuss challenges that arise when we try to generalize our approach to the mixed-integer setting.

Next, we study mixed-integer bilevel problems for which the nested problem has a nonconvex and quadratic objective function, linear constraints, and continuous variables. We state and prove a  $\Sigma_2^P$ -hardness result for this problem class and develop a lower and upper bounding scheme to solve these problems. We prove correctness and finite termination of the proposed method under suitable assumptions and test its applicability in a numerical study.

Finally, we consider bilevel problems with continuous variables, where the nested problem has a convex-quadratic objective function and linear constraints. We reformulate them as single-level optimization problems using necessary and sufficient optimality conditions for the nested problem. Then, we explore the family of so-called  $P$ -split reformulations for this single-level problem and test their applicability in a preliminary numerical study.



# Author's Contribution

This dissertation is based on one published article, one article that is currently under review, and one unpublished project. All projects have been carried out in co-authorship. The contribution of the author of this dissertation to these projects is described in the following. All authors are listed in alphabetical order.

- [AH1] A. Horländer, I. Ljubić, and M. Schmidt (2024). “Using Disjunctive Cuts in a Branch-and-Cut Method to Solve Convex Integer Non-linear Bilevel Problems.” Revised and resubmitted. URL: [https://optimization-online.org/wp-content/uploads/2024/03/disjunctive\\_cuts\\_for\\_bilevel.pdf](https://optimization-online.org/wp-content/uploads/2024/03/disjunctive_cuts_for_bilevel.pdf)

The initial idea for this article was proposed by Ivana Ljubić and Martin Schmidt. All authors jointly developed the theoretical results of the article. The author of this thesis wrote most parts of the article, proved all statements, developed all examples, implemented the algorithm, and performed the numerical studies under the supervision of the other authors.

- [AH2] I. Bomze, A. Horländer, and M. Schmidt (2025). “Mixed-Integer Bilevel Optimization with Nonconvex Quadratic Lower-Level Problems: Complexity and a Solution Method.” In: *Journal of Global Optimization* 93, pp. 1–25. DOI: [10.1007/s10898-025-01522-4](https://doi.org/10.1007/s10898-025-01522-4)

The initial idea for this article was proposed by Immanuel Bomze and Martin Schmidt. All authors jointly contributed to the development of the presented method. The author of this thesis wrote most parts of the article, proved all statements, developed all examples, implemented the algorithm, and performed the numerical studies under the supervision of the other authors.

Another project that contributes to this dissertation is presented in Chapter 5. The initial idea for this work was proposed by Jan Kronqvist and Martin Schmidt. All authors jointly contributed to the development of the presented method. The author of this thesis wrote all parts of the work, implemented all models, and performed the numerical studies under the supervision of the other authors.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Mathematical Background</b>	<b>4</b>
2.1	Preliminaries in Bilevel Optimization . . . . .	4
2.2	Computational Complexity of Bilevel Problems . . . . .	7
2.3	Single-Level Reformulations . . . . .	9
2.4	Lower and Upper Bounds . . . . .	12
2.5	Branch-and-Bound . . . . .	14
2.6	Branch-and-Cut . . . . .	23
<b>3</b>	<b>Using Disjunctive Cuts in a Branch-and-Cut Method to Solve Convex Integer Nonlinear Bilevel Problems</b>	<b>33</b>
3.1	Motivation . . . . .	34
3.2	Problem Statement . . . . .	34
3.3	Disjunctive Cuts . . . . .	35
3.4	A Branch-and-Cut Method . . . . .	46
3.5	Comparison to the Literature and Generalization Challenges .	52
3.6	Numerical Results . . . . .	59
<b>4</b>	<b>Mixed-Integer Bilevel Optimization with Nonconvex Quadratic Lower-Level Problems: Complexity and a Solution Method</b>	<b>80</b>
4.1	Motivation . . . . .	80
4.2	Problem Statement . . . . .	82
4.3	$\Sigma_2^P$ -Hardness . . . . .	83
4.4	Solution Approach . . . . .	90
4.5	Numerical Results . . . . .	99
<b>5</b>	<b>P-Splits for Bilevel Optimization</b>	<b>105</b>
5.1	Motivation . . . . .	105
5.2	Problem Statement . . . . .	106
5.3	The Big- $M$ and Convex-Hull Reformulations . . . . .	108
5.4	The $P$ -Split Reformulation . . . . .	111
5.5	Numerical Results . . . . .	116
<b>6</b>	<b>Conclusion and Outlook</b>	<b>122</b>

# Chapter 1

## Introduction

Bilevel optimization is an important area within mathematical optimization that has attracted growing attention in recent years and decades. Bilevel problems model hierarchical settings in which two agents with potentially conflicting interests make decisions that influence each other and, therefore, cannot be considered independently.

The origins of bilevel optimization can be traced back to the seminal work of von Stackelberg (1934) in the context of game theory. The concept was formally introduced to the optimization community by Bracken and McGill (1973), who studied “mathematical programs with optimization problems in the constraints”, a formulation that captures the core structure of bilevel problems. Today, the outer optimization problem is referred to as the upper level, while the embedded problem is called the lower level. Their nested structure renders bilevel problems a powerful tool to model hierarchical decision making processes that occur in various applications such as energy markets (Gabriel et al. 2012; Grimm et al. 2019), transportation (Marcotte 1986), security applications (Tambe 2011; Wang et al. 2016), or pricing (Labbé and Violin 2013). However, they are very hard to solve, both in theory (Jeroslow 1985; Hansen et al. 1992) and in practice (Thürauf et al. 2024). Despite these challenges, the field has made substantial progress, enabling the solution of significantly more complex and larger instances compared to what was possible 10 to 15 years ago. One of the main ways on how to propel the field has been to study bilevel optimization problems that have increasingly complicated lower-level problems, as these often drive the overall computational difficulty.

For bilevel problems with continuous and convex lower levels, there are established approaches that transform the bilevel problem into a single-level problem by replacing its lower level with necessary and sufficient optimality conditions; see, e.g., Dempe and Zemkoho (2020). Both mathematically and algorithmically more challenging classes of bilevel problems are those with integer aspects or nonconvex functions in the lower-level problem. In such

cases, compact necessary and sufficient optimality conditions do not exist in general. As a result, standard reformulations based on Karush–Kuhn–Tucker conditions or strong duality (Zare et al. 2019a) are not applicable.

Bilevel problems having integer aspects in the lower-level problem are frequently solved using some kind of branch-and-cut framework, in which branching is performed on integrality constraints and cutting planes are used to cut off integer-feasible but bilevel-infeasible points. The first approach in this direction has been published by DeNegre and Ralphs (2009) for purely integer linear bilevel problems. Since then, many authors developed further techniques for tackling (mixed-)integer linear bilevel problems such as the watermelon approach in Wang and Xu (2017) or the use of intersection cuts in Fischetti et al. (2017) and Fischetti et al. (2018). Moreover, convex (mixed-)integer nonlinear bilevel problems have been solved, e.g., using outer approximation techniques (Kleinert et al. 2021a) or disjunctive cuts within a branch-and-cut method (Gaar et al. 2023).

There has also been progress on bilevel problems with nonconvex lower levels in the last years and decades. For purely continuous problems, Mitsos et al. (2008) propose a global optimization framework, which is extended to the mixed-integer setting in Mitsos (2010). Furthermore, there is a series of papers on the so-called branch-and-sandwich approach for mixed-integer nonconvex bilevel problems; see, e.g., Kleniati and Adjiman (2014a) and Kleniati and Adjiman (2014b) on purely continuous bilevel problems and the extension to the mixed-integer case in Kleniati and Adjiman (2015). Another method addressing mixed-integer nonconvex bilevel problems is the value-function approach proposed in Lozano and Smith (2017). Finally, lower-level problems can be nonconvex due to an incorporation of stochastic or robust models for addressing uncertainty; see Beck et al. (2023a) for an overview.

In this dissertation, we present our contributions to the development of new methods for solving different classes of bilevel problems with a special focus on hard lower levels. Our work includes the derivation of theoretical properties for the considered problem settings, the design of novel solution approaches, and comprehensive numerical studies that evaluate the effectiveness and applicability of the proposed methods. This thesis is organized as follows. In Chapter 2, we introduce the standard notation and mathematical background that forms the foundation for the subsequent chapters. Chapter 3 focuses on convex integer nonlinear bilevel problems for which we develop tailored disjunctive cuts that we integrate into a branch-and-cut framework to solve instances of this problem class. We prove correctness and finite termination of the presented method under suitable assumptions. Moreover, we provide an extensive numerical study to showcase the applicability of our method and we compare it to the state-of-the-art approach for a less general setting on suitable instances from the literature. Furthermore, we discuss challenges that arise when we try to generalize our approach to the mixed-integer setting. In Chapter 4, we address mixed-integer bilevel problems featuring

nonconvex quadratic lower levels having continuous variables and linear constraints. We present and prove a  $\Sigma_2^P$ -hardness result for this problem class and propose a lower- and upper-bounding scheme to solve these problems. We prove correctness and finite termination of the proposed method under suitable assumptions and test its applicability in a numerical study. In Chapter 5, we study bilevel problems having purely continuous variables and convex-quadratic lower-level problems with linear constraints. We explore the family of so-called  $P$ -split reformulations for the necessary and sufficient optimality conditions of the lower-level problem and test their applicability in a preliminary numerical study. Finally, Chapter 6 concludes this thesis and offers an outlook on potential extensions of the presented results in future research.

## Chapter 2

# Mathematical Background

In this chapter we recall fundamental concepts of bilevel optimization and review existing solution methods. In Section 2.1, which is based on Dempe (2002), we present the formal definition of a bilevel problem and introduce standard notation. Section 2.2 is inspired by the work of Cerulli (2021) and studies the computational hardness of bilevel optimization. Moreover, it provides a foundation for the results in Section 4.3. In Section 2.3, which is based on Beck and Schmidt (2021), we introduce frequently used single-level reformulations that can be applied to specific classes of bilevel problems. Furthermore, we describe in Section 2.4 how upper and lower bounds on the optimal objective function value of a bilevel problem can be computed. We use these concepts in Sections 2.5 and 2.6 to introduce the branch-and-bound and branch-and-cut methods, respectively. These methods will be used in Chapters 3, 4, and 5 of this thesis to tackle the bilevel problems discussed therein.

### 2.1 Preliminaries in Bilevel Optimization

In bilevel optimization one considers optimization problems in which parts of the variables have to be an optimal solution to another nested optimization problem. They model situations of two decision makers with potentially different objectives and constraints. One decision maker, called the *leader*, makes a decision first while anticipating the reaction of the second decision maker, referred to as the *follower*. The follower then acts based on the leader's decision and his own objective and constraints. Throughout this thesis, we use “her” for the leader and “his” for the follower.

*Bilevel problems* are optimization problems of the form

$$\min_{x \in X} F(x, y) \tag{2.1a}$$

$$\text{s.t. } G(x, y) \geq 0, \tag{2.1b}$$

$$y \in S(x), \tag{2.1c}$$

where  $S(x)$  is the set of optimal solutions to the  $x$ -parameterized problem

$$\min_{y \in Y} f(x, y) \quad (2.2a)$$

$$\text{s.t. } g(x, y) \geq 0. \quad (2.2b)$$

The objective functions are given by  $F, f : \mathbb{Q}^{n_x} \times \mathbb{Q}^{n_y} \rightarrow \mathbb{Q}$  and the constraint functions are given by  $G : \mathbb{Q}^{n_x} \times \mathbb{Q}^{n_y} \rightarrow \mathbb{Q}^m$  and  $g : \mathbb{Q}^{n_x} \times \mathbb{Q}^{n_y} \rightarrow \mathbb{Q}^l$ . Throughout this thesis, we abbreviate  $(x^\top, y^\top)^\top$  by  $(x, y)$  for ease of reading. Furthermore, we assume that all functions  $F, G, f$ , and  $g$  are continuous and that  $n_x, n_y, m, l \in \mathbb{N}$ . We define  $n := n_x + n_y$  as well as  $[\iota] := \{1, \dots, \iota\}$  for  $\iota \in \{n_x, n_y, m, l\}$ . We refer to Problem (2.1) as the *upper-level* (or leader's) problem and call  $x$  *upper-level variables* (or leader's decisions). Problem (2.2) is denoted as *lower-level* (or follower's) problem and  $y$  as *lower-level variables* (or follower's decisions/response). The set  $S(x)$  is called the *rational reaction set*. If  $S(x)$  is not a singleton, i.e., the follower's response  $y$  to the leader's decision  $x$  is not unique, then the bilevel problem as stated in (2.1) is ill-posed. To overcome this issue, either the *optimistic* or the *pessimistic* variant of the bilevel problem is considered. In the former, it is assumed that the follower acts in the leader's favor if he is indifferent between multiple decisions. This leads to the problem

$$\min_{x \in X, y} F(x, y) \quad (2.3a)$$

$$\text{s.t. } G(x, y) \geq 0, \quad (2.3b)$$

$$y \in S(x), \quad (2.3c)$$

with  $S(x)$  being the set of optimal solutions to the  $x$ -parameterized lower-level problem (2.2). That is, the leader can choose a follower's response that minimizes her objective function, if multiple responses are available that lead to an equal outcome for the follower, i.e., the same lower-level objective function value. For the pessimistic variant, the leader assumes that the follower will always choose the worst possible solution for her if multiple solutions are available. Throughout this dissertation, we consider the optimistic variant, i.e., problems of the form (2.3).

Upper-level variables appearing in the lower-level constraints are called *linking variables* and upper-level constraints  $G_i(x, y) \geq 0$  for  $i \in [m]$  that explicitly depend on the lower-level variables are called *coupling constraints*. The sets  $X$  and  $Y$  impose bounds and/or integrality conditions on the  $x$  and  $y$  variables, respectively. The set

$$\tilde{\Omega} := \{(x, y) \in X \times Y : G(x, y) \geq 0, g(x, y) \geq 0\} \quad (2.4)$$

is called *shared constraint set* and its projection to the  $x$ -space is denoted with

$$\tilde{\Omega}_x := \left\{ x \in X : \exists y \text{ with } (x, y) \in \tilde{\Omega} \right\}. \quad (2.5)$$

A point  $(\hat{x}, \hat{y})$  is called *bilevel-feasible* if it lies in  $\tilde{\Omega}$  and if  $\hat{y}$  is a globally optimal solution to the  $\hat{x}$ -parameterized lower-level problem (2.2). The set of bilevel-feasible points is called *inducible region* and is given by

$$\mathcal{F} := \{(x, y) \in \tilde{\Omega}, y \in S(x)\}. \quad (2.6)$$

A point  $(x^*, y^*)$  is *bilevel-optimal* if it minimizes the upper-level objective function  $F$  over  $\mathcal{F}$ . The problem of minimizing the upper-level objective function over the shared constraint set, i.e.,

$$\min_{(x, y) \in \tilde{\Omega}} F(x, y), \quad (2.7)$$

is called the *high-point relaxation* (HPR) of Problem (2.3). Its continuous relaxation (C-HPR) is commonly used to compute lower bounds for the optimal objective function value as we will see in Section 2.4. As the name suggests, the C-HPR drops all integrality conditions of Problem (2.7). We denote its feasible region with  $\Omega$ .

In this thesis, we focus mostly on (mixed-)integer bilevel problems. In the literature, the following assumptions are usually made; see, e.g., Fischetti et al. (2018), Gaar et al. (2023), and Kleinert et al. (2021a).

**Assumption 2.1.1.** *The shared constraint set  $\tilde{\Omega}$  is non-empty and compact.*

**Assumption 2.1.2.** *For all  $x \in \tilde{\Omega}_x$ , the feasible region of the  $x$ -parameterized lower-level problem, i.e., the set  $\tilde{\Omega}_l(x) := \{y \in Y : g(x, y) \geq 0\}$ , is bounded.*

**Assumption 2.1.3.** *If we consider mixed-integer bilevel problems, then all linking variables  $x_I$  for  $I \subseteq [n_x]$  are bounded integers.*

These assumptions are due to different observations that have been made for mixed-integer linear bilevel problems. The first assumption ensures that the high-point relaxation has a finite solution. According to Xu and Wang (2014), the unboundedness of the high-point relaxation does not provide sufficient information to conclude whether the bilevel problem is unbounded, infeasible, or admits an optimal solution. Moreover, it has been shown in Rodrigues et al. (2025) that proving unboundedness for continuous linear bilevel problems is strongly NP-hard, while it is  $\Sigma_2^P$ -hard for mixed-integer linear bilevel problems. However, if one considers problem settings that lead to unboundedness of the high-point relaxation, different techniques can be used to handle this situation; see, e.g., Fischetti et al. (2018).

The second assumption states that the  $x$ -parameterized lower-level problem is bounded for every feasible leader's decision  $x$ . If it were unbounded, then the bilevel problem is infeasible; see Xu and Wang (2014).

The third assumption is frequently made if the lower-level problem involves integrality constraints for some of the lower-level variables. It is shown in Moore and Bard (1990) that in this case, the appearance of fractional linking variables can lead to unattainability of solutions because the inducible region may be open.

## 2.2 Computational Complexity of Bilevel Problems

Bilevel problems are inherently hard to solve, both theoretically and computationally. In this section, we focus on the latter by introducing the notion of the polynomial hierarchy. This is a framework that allows to classify decision problems based on their computational complexity.

The base level of the polynomial hierarchy is denoted with  $\Sigma_0^P$  (or P) and contains problems that are solvable in polynomial time, e.g., linear programs (LPs). The next level,  $\Sigma_1^P$  (or NP), includes problems that are solvable in nondeterministic polynomial time, e.g., mixed-integer linear programs (MILPs). For an overview of the standard complexity classes P and NP, we refer to Garey and Johnson (2002) and the references therein. In general, the  $k$ th level of the hierarchy is denoted with  $\Sigma_k^P$  and contains problems that can be solved in nondeterministic polynomial time with access to a constant-time oracle for  $\Sigma_{k-1}^P$  problems; see Meyer and Stockmeyer (1972). This leads to the following inclusion chain; see, e.g., Cerulli (2021):

$$P \subseteq NP \subseteq \Sigma_2^P \subseteq \dots \subseteq \Sigma_k^P \subseteq \Sigma_{k+1}^P \subseteq \dots$$

Whether these inclusions are strict remains an open question. However, it is widely assumed that they are. Otherwise, the polynomial hierarchy would collapse. A problem is said to be  $\Sigma_k^P$ -hard, if every problem in  $\Sigma_k^P$  can be reduced to it in polynomial time. Moreover, a problem is  $\Sigma_k^P$ -complete if it is in  $\Sigma_k^P$  and  $\Sigma_k^P$ -hard.

Any discussion on the complexity of a bilevel problem generally refers to its decision variant. For the ease of presentation, let us consider the decision variant of a bilevel problem without coupling constraints, i.e., the question: “Does there exist a decision for the leader, so that for all possible decisions of the follower, her objective function value is at least as good as some given bound?”; see, e.g., Caprara et al. (2014) and Woeginger (2021). Note that classifications of bilevel problems into complexity classes are often established for instances without coupling constraints; see, e.g., Caprara et al. (2014). Moreover, statements regarding the complexity-theoretic hardness of bilevel problems without coupling constraints remain valid when coupling constraints are added. For linear bilevel problems, it has been shown in Henke et al. (2025a) and Henke et al. (2025b) that the presence of coupling constraints does not increase their hardness either. However, for other classes of bilevel problems, it remains unclear whether the addition of coupling constraints could cause the problem to fall into a higher complexity class.

It has been shown in Hansen et al. (1992) that linear (or LP-LP) bilevel problems, i.e., those that have a linear objective function, linear constraints, and continuous variables in both the upper- and the lower level, are strongly NP-hard in general. For such problems, even checking local optimality for a given point is NP-hard; see Vicente et al. (1994). Moreover, Jeroslow (1985) showed that  $(k + 1)$ -level linear problems are  $\Sigma_k^P$ -hard. Apart from that, the

complexity class  $\Sigma_2^P$  is of special interest for bilevel optimization, since it naturally captures the structure of bilevel problems. In Woeginger (2021) and Stockmeyer (1976), the following definition is made.

**Definition 2.2.1** ( $\Sigma_2^P$ ; see Woeginger (2021) and Stockmeyer (1976)). *A decision problem is contained in the complexity class  $\Sigma_2^P$ , if its YES-instances  $\mathcal{I}$  are characterized by a formula of the form*

$$\exists x \in \mathcal{X} \forall y \in \mathcal{Y} : P(\mathcal{I}, x, y), \quad (2.8)$$

where the predicate  $P(\mathcal{I}, x, y)$  can be evaluated in time polynomially bounded in  $|\mathcal{I}|$ .

In the context of bilevel optimization, the objects  $x \in \mathcal{X}$  and  $y \in \mathcal{Y}$  are leader's and follower's decisions, respectively. The decision variant of a bilevel problem without coupling constraints fits into the framework of (2.8). Given that all potential decisions of the leader and the follower are encoded with polynomially many bits and that the evaluation of the Boolean predicate  $P(\mathcal{I}, x, y)$  can be done in polynomial time, the decision variant of a bilevel problem is contained in  $\Sigma_2^P$ ; see Woeginger (2021).

In a recent paper by Buchheim (2023) it is shown that linear bilevel problems even belong to NP. However, due to their hierarchical structure, many variants of bilevel problems are actually  $\Sigma_2^P$ -hard. This is usually the case if they are build on top of an NP-hard lower-level problem. For instance, in Caprara et al. (2014), three bilevel-variants of knapsack problems are studied, i.e., the variants presented in Dempe and Richter (2000), Mansi et al. (2012), and DeNegre (2011) and it is proven that all of them are  $\Sigma_2^P$ -hard. In Section 4.3, we state a hardness result for a bilevel problem containing a mixed-integer quadratic program (MIQP) in the upper level and an indefinite quadratic program (QP) in the lower level. Therefore, we use the results in Caprara et al. (2014) w.r.t. the Dempe Richter (DeRi) variant.

**Definition 2.2.2** (DeRi bilevel knapsack problem; see Dempe and Richter (2000)). *The Dempe-Richter (DeRi) variant of a bilevel knapsack problem is given by*

$$\begin{aligned} \max_{x \in \mathbb{Z}, y} \quad & c_x x + c_y^\top y \\ \text{s.t.} \quad & x^- \leq x \leq x^+, \\ & y \in \arg \max_{\bar{y}} \left\{ d^\top \bar{y} : d^\top \bar{y} \leq x, \bar{y} \in \{0, 1\}^{n_y} \right\}, \end{aligned} \quad (2.9)$$

with  $0 \neq d \in \mathbb{Z}_+^{n_y}$ ,  $c_y \in \mathbb{Z}_+^{n_y}$ , and  $c_x, x^-, x^+ \in \mathbb{Z}_+$ .

The  $\Sigma_2^P$ -hardness proof for Problem (2.9) is done via a reduction from an instance of SUBSET-SUM-INTERVAL; see Caprara et al. (2014). The latter is proven to be  $\Sigma_2^P$ -complete by Eggermont and Woeginger (2013).

## 2.3 Single-Level Reformulations

A common technique to tackle bilevel-problems is to reformulate them as single-level problems. In general, this can be done by introducing the optimal-value function

$$\Phi(x) := \min_{y \in Y} \{f(x, y) : g(x, y) \geq 0\}. \quad (2.10)$$

With that, one can restate the bilevel problem as

$$\min_{x \in X, y \in Y} F(x, y) \quad (2.11a)$$

$$\text{s.t. } G(x, y) \geq 0, \quad (2.11b)$$

$$f(x, y) \leq \Phi(x), \quad (2.11c)$$

which is denoted as value-function reformulation; see Dempe (2002). Although Problem (2.11) is a single-level problem, it is rarely used directly to solve bilevel problems of the form (2.3). The reason is that the optimal-value function is nonsmooth and computationally intractable in general; see Dempe et al. (2007). However, the value-function reformulation is frequently used indirectly in branch-and-cut approaches to tackle mixed-integer bilevel problems. Here, constraint (2.11c) is omitted, leading to the high-point relaxation (2.7), which is used as a starting point in many branch-and-cut algorithms.

In addition to the value-function reformulation, other single-level reformulations are commonly used to tackle bilevel-problems with convex and continuous lower-level problems. That is, the lower-level variables  $y$  are continuous, the objective function  $f$  is convex in  $y$ , and the constraint functions  $g_i$  are concave in  $y$  for  $i \in [l]$ . Moreover, the lower level has to satisfy a constraint qualification (CQ) for all  $x \in \tilde{\Omega}_x$ , e.g., Slater's CQ. Classic examples for such problems are bilevel problems having an LP or a QP in the lower level. Note that the presence of only linear constraints in the lower level implies the Abadie constraint qualification; see Bazarraa et al. (2006) for further details. For such problems, one can state optimality conditions that are not only necessary but also sufficient. Hence, the lower level problem can be replaced by its optimality conditions. To give more insights about this idea, let us consider problem

$$\min_{x \in \mathbb{R}^{n_x}, y} c_x^\top x + c_y^\top y \quad (2.12a)$$

$$\text{s.t. } Ax + By \geq a, \quad (2.12b)$$

$$x \in \{0, 1\}^{n_x}, \quad (2.12c)$$

$$y \in \arg \min_{\bar{y} \in \mathbb{R}^{n_y}} \left\{ d^\top \bar{y} : Cx + D\bar{y} \geq b \right\} \quad (2.12d)$$

with  $A \in \mathbb{Q}^{m \times n_x}$ ,  $B \in \mathbb{Q}^{m \times n_y}$ ,  $C \in \mathbb{Q}^{l \times n_x}$ ,  $D \in \mathbb{Q}^{l \times n_y}$ ,  $a \in \mathbb{Q}^m$ , and  $b \in \mathbb{Q}^l$ . Furthermore, assume that the shared constraint set

$$\tilde{\Omega} = \{(x, y) \in \{0, 1\}^{n_x} \times \mathbb{R}^{n_y} : Ax + By \geq a, Cx + Dy \geq b\}$$

is bounded; see Assumption 2.1.1. Problem (2.12) is a bilevel problem with an integer linear program (ILP) in the upper and an LP in the lower level. It is strongly NP-hard; see Hansen et al. (1992). We can state the Karush Kuhn Tucker (KKT) conditions for the lower level, i.e.,

$$\begin{aligned} Cx + Dy &\geq b, \\ D^\top \lambda &= d, \quad \lambda \geq 0, \\ \lambda^\top (Cx + Dy - b) &= 0. \end{aligned} \tag{2.13}$$

Note that the KKT conditions are necessary and sufficient for the lower-level LP in (2.12). Hence, we replace (2.12d) with the KKT conditions of the lower level to obtain the single-level reformulation

$$\min_{x, y, \lambda} c_x^\top x + c_y^\top y \tag{2.14a}$$

$$\text{s.t. } Ax + By \geq a, \quad Cx + Dy \geq b, \tag{2.14b}$$

$$D^\top \lambda = d, \quad \lambda \geq 0, \tag{2.14c}$$

$$\lambda^\top (Cx + Dy - b) = 0, \tag{2.14d}$$

$$x \in \{0, 1\}^{n_y} \tag{2.14e}$$

of Problem (2.12). Problem (2.14) is called the *KKT reformulation* of Problem (2.12). It is a mixed-integer nonlinear optimization problem (MINLP) and it is nonconvex due to the present complementarity conditions (2.14d). Moreover, it is NP-hard; see Lee and Leyffer (2011). The single-level reformulation (2.14) is equivalent to the bilevel problem (2.12) in the sense that their globally optimal solutions correspond to each other; see Dempe and Dutta (2012). Note that this does not hold in general for locally optimal solutions or stationary points; see Dempe and Dutta (2012). Due to the complementarity conditions, standard nonlinear optimization algorithms cannot be applied to Problem (2.14) because classic constraint qualifications like the Mangasarian Fromowitz or the linear independence constraint qualification are violated at every feasible point; see, e.g., Ye and Zhu (1995). Therefore, the complementarity conditions are often reformulated using auxiliary binary

variables and sufficiently large big- $M$  constants, i.e.,

$$\begin{aligned}
& \min_{x,y,\lambda} && c_x^\top x + c_y^\top y \\
& \text{s.t.} && Ax + By \geq a, \quad Cx + Dy \geq b, \\
& && D^\top \lambda = d, \quad \lambda \geq 0, \\
& && \lambda_j \leq Mz_j, \quad j \in [l], \\
& && (Cx + Dy - b)_j \leq M(1 - z_j), \quad j \in [l], \\
& && x \in \{0, 1\}^{n_x}, \quad z \in \{0, 1\}^l;
\end{aligned} \tag{2.15}$$

see Fortuny-Amat and McCarl (1981). Problem (2.15) is a MILP that can be solved, e.g., with a branch-and-bound method. It is called the *big- $M$  reformulation* of Problem (2.14). For a sufficiently large constant  $M$ , it is equivalent to the KKT reformulation (2.14) in the sense that their globally optimal solutions correspond to each other. Hence, a globally optimal solution to Problem (2.15) leads to a globally optimal solution to the bilevel problem (2.12). However, the determination of a suitable big- $M$  constant is a challenging task as we need bounds for both the primal and the dual variables. While the former is often available in practice, e.g., via box constraints on the primal variables or boundedness assumptions on the shared constraint set, the latter is typically not provided. Kleinert et al. (2020) showed that even the validation of the correctness of a given big- $M$  is NP-hard in general. Therefore, in practice, often standard values, e.g.,  $10^6$ , are used without verifying their correctness. This may lead to suboptimal solutions; see Pineda and Morales (2019). Moreover, choosing too large big- $M$  values may lead to numerical instabilities in the algorithms that are applied.

Another way to tackle the complementarity conditions in (2.14d) that does not require the use of generic big- $M$  constants is to introduce *special ordered sets of type 1* (SOS1). This set of variables requires that at most one of its elements takes a nonzero value. They have been introduced in Beale and Tomlin (1970) and can be used to rephrase the constraint (2.14d) as

$$s_j = (Cx + Dy - b)_j, \quad \{s_j, \lambda_j\} \text{ is SOS1}, \quad j \in [l].$$

This idea was initially presented in the context of bilevel optimization by Fortuny-Amat and McCarl (1981) and is still used in recent works; see, e.g., Kleinert and Schmidt (2023). The SOS1 can be handled by modern mixed-integer solvers in two different ways. If provably correct bounds are available for the  $s_j$  and  $\lambda_j$ , then Problem (2.14) is reformulated as in (2.15) because an appropriate value for  $M$  can be guaranteed. Otherwise, the solvers branch on the variables  $s_j$  and  $\lambda_j$ . We will use SOS1 in the implementation of the algorithms presented in Chapters 4 and 5. In Kleinert and Schmidt (2023), the big- $M$  approach and the SOS1 approach are compared on LP-LP bilevel instances. It turns out that the big- $M$  approach is faster and could solve more

instances overall. However, the authors also mention that the performance of the big- $M$  approach strongly relies on problem-specific knowledge that allows one to choose a small value for  $M$  that is known to be valid. They emphasise that without this knowledge, the SOS1 approach should be considered.

Another way to reformulate a bilevel problem into a single-level problem is to use a strong-duality theorem for the lower level; see, e.g., Dempe and Mehrlitz (2025). For the lower-level LP in (2.12), the dual problem is given by

$$\max_{\lambda} \quad (b - Cx)^\top \lambda \quad \text{s.t.} \quad D^\top \lambda = d, \lambda \geq 0 \quad (2.16)$$

with dual variables  $\lambda \in \mathbb{R}^l$ . For every leader's decision  $x$ , weak duality states that

$$d^\top y \geq (b - Cx)^\top \lambda$$

holds for every primal and dual feasible pair  $(y, \lambda)$ . Moreover, strong duality ensures that if

$$d^\top y \leq (b - Cx)^\top \lambda$$

holds, then  $(y, \lambda)$  is a pair of optimal solutions to the  $x$ -parameterized lower level. With that, we state the so-called *strong duality reformulation* of Problem (2.12), i.e.,

$$\min_{x, y, \lambda} \quad c_x^\top x + c_y^\top y \quad (2.17a)$$

$$\text{s.t.} \quad Ax + By \geq a, \quad Cx + Dy \geq b, \quad (2.17b)$$

$$D^\top \lambda = d, \quad \lambda \geq 0, \quad (2.17c)$$

$$d^\top y \leq (b - Cx)^\top \lambda, \quad (2.17d)$$

$$x \in \{0, 1\}^{n_x}. \quad (2.17e)$$

It is easy to verify that Constraint (2.17d) in the strong-duality reformulation of Problem (2.12) is equivalent to the KKT complementarity condition (2.14d) in the KKT reformulation (2.14). Hence, both single-level reformulations for Problem (2.12) are equivalent. However, the nonlinearities in (2.14d) and (2.17d) are handled differently; see, e.g., Zare et al. (2019b) for further details.

## 2.4 Lower and Upper Bounds

As discussed in Section 2.2, bilevel problems are intrinsically hard to solve computationally. State-of-the-art methods to solve (mixed-integer) bilevel problems usually rely on the use of bounds for the optimal objective function value. We obtain lower bounds (also called dual bounds) for this value by introducing the notion of a relaxation.

**Definition 2.4.1** (Relaxation). Consider the problem  $\min\{F(x) : x \in \mathcal{F}\}$ . Problem  $\min\{F'(x) : x \in \mathcal{F}'\}$  is called a relaxation of the first problem if  $\mathcal{F} \subseteq \mathcal{F}'$  and  $F'(x) \leq F(x)$  for all  $x \in \mathcal{F}$ .

By definition, a globally optimal solution to a relaxation yields a lower bound for the optimal objective function value of the original minimization problem. In the context of bilevel optimization, the most frequently used relaxation is the high-point relaxation (2.7), which ignores the lower-level objective function. Note that the HPR is a mixed-integer problem, if the sets  $X$  and  $Y$  in Problem (2.3) impose integrality constraints to the upper- or lower-level variables. In this case, the HPR is NP-hard. Hence, it may be computationally too expensive to solve this problem to obtain a lower bound. Therefore, usually the continuous relaxation of the HPR, the C-HPR, is solved to compute an initial, potentially weak, lower bound.

Upper bounds (also called primal bounds) for the optimal objective function value of Problem (2.3) can be obtained by computing bilevel-feasible points. However, checking feasibility of a given point requires the evaluation of the optimal-value function (2.10) of the lower level, which is rarely available in analytical form. Therefore, usually the  $\hat{x}$ -parameterized lower level has to be solved for a given leader's decision  $\hat{x}$  to obtain an optimal follower's response  $\hat{y}$ . If the point  $(\hat{x}, \hat{y})$  satisfies the upper-level constraints, it is bilevel feasible and provides an upper bound. Note that solving the lower-level problem (2.2) can be a challenging task. In fact, the bilevel problems we study in Chapter 3 and 4 in this thesis contain NP-hard lower-level problems. A method to solve (mixed-integer) bilevel problems that strongly relies on the use of lower and upper bounds is the branch-and-bound algorithm. This is also the standard approach in mixed-integer optimization solvers like CPLEX, Gurobi, or SCIP; see IBM Corporation (2023), Gurobi Optimization, LLC (2025), and Achterberg (2009).

If a lower bound  $L$  and an upper bound  $U$  for the optimal objective function value of an optimization problem are at hand, one can compute the (relative) optimality gap as

$$\gamma := \frac{|U - L|}{10^{-10} + |U|}; \quad (2.18)$$

see IBM Corporation (2023). The optimality gap quantifies how far the best-known feasible solution is from the optimal solution. When the gap is zero, i.e., when the lower and upper bounds coincide, the best-known solution is proven to be optimal. Consequently, the optimality gap is used as a key termination criterion for algorithms that rely on bounding techniques, such as branch-and-bound or the approach that we present in Chapter 4 of this thesis. In practice, these methods typically terminate once the gap falls below a user-defined threshold. Therefore, it can also be used to terminate processes early with a feasible point of “good quality” rather than a globally optimal

solution to save computation time. Moreover, the optimality gap serves as a performance indicator for a given method. If a problem instance cannot be solved within a user-defined time limit, the gap provides insight into how effectively the method has performed up to that point.

## 2.5 Branch-and-Bound

Branch-and-bound is an algorithmic framework used to solve optimization problems that have combinatorial aspects. It was first presented by Land and Doig (1960) and is still used in state-of-the-art methods, particularly for addressing (mixed-)integer problems. A correctness result for the branch-and-bound method is given in, e.g., Locatelli and Schoen (2013). In this dissertation, we will use branch-and-bound and the strongly related branch-and-cut method to solve most of the problems that are considered in Chapters 3, 4, and 5.

The introduction of integrality constraints can add significant difficulties to existing problems, e.g., LPs can be solved in polynomial time while MILPs are generally NP-hard. While the branch-and-bound framework is broadly used to tackle integrality constraints, it can also handle nonconvexities that can be decomposed into convex parts, e.g., bilinear terms. Instead of solving the original problem, the method systematically solves a series of easier subproblems, each of which covers a part of the total search space.

Consider the bilevel problem (2.12). We know from Section 2.3 that it is equivalent to the single-level reformulation (2.14) w.r.t. their globally optimal solutions. The latter problem contains integrality and complementarity constraints that we tackle using the branch-and-bound method. The method starts by solving a relaxation of the original problem. A suitable relaxation of Problem (2.14) is given by

$$\begin{aligned}
 \min_{x,y,\lambda} \quad & c_x^\top x + c_y^\top y \\
 \text{s.t.} \quad & Ax + By \geq a, \quad Cx + Dy \geq b, \\
 & D^\top \lambda = d, \quad \lambda \geq 0, \\
 & x \in [0, 1]^{n_x}.
 \end{aligned} \tag{2.19}$$

Problem (2.19) is called the *root-node relaxation* or simply the *root-node* of Problem (2.12). It drops the integrality conditions on the  $x$ -variables and the complementarity conditions (2.14d). Note that Problem (2.19) is an LP, i.e., it is solvable in polynomial time, while the KKT reformulation (2.14) is NP-hard. Instead, if only either the integrality or complementarity constraints were omitted, while keeping the others in the root-node relaxation, the resulting problem would remain NP-hard. This would be detrimental for the branch-and-bound method as each of the subproblems that is solved during the procedure is as hard as the root-node relaxation. Hence, the method

would solve a series of subproblems with each of them being as hard as the original problem.

The first step in a branch-and-bound procedure is to solve the root-node relaxation (2.19). If this problem is infeasible, then Problem (2.12) is infeasible as well because the latter only imposes additional constraints. Otherwise, we obtain a solution  $(x^0, y^0, \lambda^0)$ . Note that Problem (2.19) cannot be unbounded w.r.t.  $x$  and  $y$  since we assumed that the feasible region of the shared constraint set is bounded. The value  $c_x^\top x^0 + c_y^\top y^0$  yields a lower bound for the optimal objective function value of Problem (2.12). Moreover, for every point  $(\hat{x}, \hat{y})$  satisfying the constraints of Problem (2.12), the value  $c_x^\top \hat{x} + c_y^\top \hat{y}$  yields an upper bound for the optimal objective function value of Problem (2.12). Consequently, if the point  $(x^0, y^0, \lambda^0)$  satisfies all integrality or complementarity constraints, then it is a globally optimal solution to Problem (2.12) and we terminate. Otherwise,  $(x^0, y^0, \lambda^0)$  violates at least one of those conditions and we continue with the procedure as follows.

### 2.5.1 Branching

In the branching step, we choose one of the violated constraints. If we choose a violated integrality constraint, i.e., an index  $i \in [n_x]$  with  $x_i^0 \notin \{0, 1\}$ , we create the two subproblems

$$\begin{aligned}
\min_{x,y,\lambda} \quad & c_x^\top x + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \quad Cx + Dy \geq b, \\
& D^\top \lambda = d, \quad \lambda \geq 0, \\
& x \in [0, 1]^{n_y}, \\
& x_i = 0
\end{aligned} \tag{2.20}$$

and

$$\begin{aligned}
\min_{x,y,\lambda} \quad & c_x^\top x + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \quad Cx + Dy \geq b, \\
& D^\top \lambda = d, \quad \lambda \geq 0, \\
& x \in [0, 1]^{n_y}, \\
& x_i = 1.
\end{aligned} \tag{2.21}$$

Both problems (2.20) and (2.21) are LPs. To refer to the branching decision made, we introduce the index sets  $Z \subseteq [n_x]$  and  $O \subseteq [n_x]$  with  $Z \cap O = \emptyset$ . For a given subproblem, the former set stores the indices  $i \in [n_x]$  for which we set  $x_i$  to zero and  $O$  stores the indices for which we set  $x_i$  to one.

Otherwise, if we choose a violated complementarity constraint, i.e., an

index  $k \in [l]$  with  $\lambda_k(Cx + Dy - b)_k > 0$ , we create the subproblems

$$\begin{aligned}
& \min_{x,y,\lambda} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax + By \geq a, \quad Cx + Dy \geq b, \\
& \quad D^\top \lambda = d, \quad \lambda \geq 0, \\
& \quad x \in [0, 1]^{n_x}, \\
& \quad (Cx + Dy - b)_k = 0
\end{aligned} \tag{2.22}$$

and

$$\begin{aligned}
& \min_{x,y,\lambda} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax + By \geq a, \quad Cx + Dy \geq b, \\
& \quad D^\top \lambda = d, \quad \lambda \geq 0, \\
& \quad x \in [0, 1]^{n_x}, \\
& \quad \lambda_k = 0.
\end{aligned} \tag{2.23}$$

Again, we introduce index sets to refer to our branching decisions. We use  $\hat{D} \subseteq [l]$  to store the indices  $k \in [l]$  for which we set  $\lambda_k = 0$  and  $P \subseteq [l]$  contains the indices for which we set  $(Cx + Dy - b)_k = 0$ , i.e.,  $P \cap \hat{D} = \emptyset$ . After creating two subproblems by branching, we choose one of them and repeat the procedure. Listing all of the created subproblems can be done in a tree structure as illustrated in Figure 2.1. This is where some of the notation comes from. The root-node is the starting point of the tree and “branching” is the process of creating new branches. A subproblem in the branch-and-bound tree is called a *node* (problem). Each of them is uniquely labeled with a set  $N := (Z, O, P, \hat{D})$  and can be written as

$$\begin{aligned}
& \min_{x,y,\lambda} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax + By \geq a, \quad Cx + Dy \geq b, \\
& \quad D^\top \lambda = d, \quad \lambda \geq 0, \\
& \quad x \in [0, 1]^{n_x}, \\
& \quad x_i = 0, \quad i \in Z, \\
& \quad x_i = 1, \quad i \in O, \\
& \quad \lambda_k = 0, \quad k \in \hat{D}, \\
& \quad (Cx + Dy - b)_k = 0, \quad k \in P.
\end{aligned} \tag{2.24}$$

Every subproblem that has not been solved yet is declared as an *open node* (problem) and stored in the *set of open nodes*  $\mathcal{N}_o$ . A globally optimal solution to Problem (2.14) can be computed by solving every node problem with  $Z \cup O = [n_x]$  and  $P \cup \hat{D} = [l]$ , i.e., by enumeration. However, the enumeration approach requires solving  $2^{n_x+l}$  many subproblems, which is out

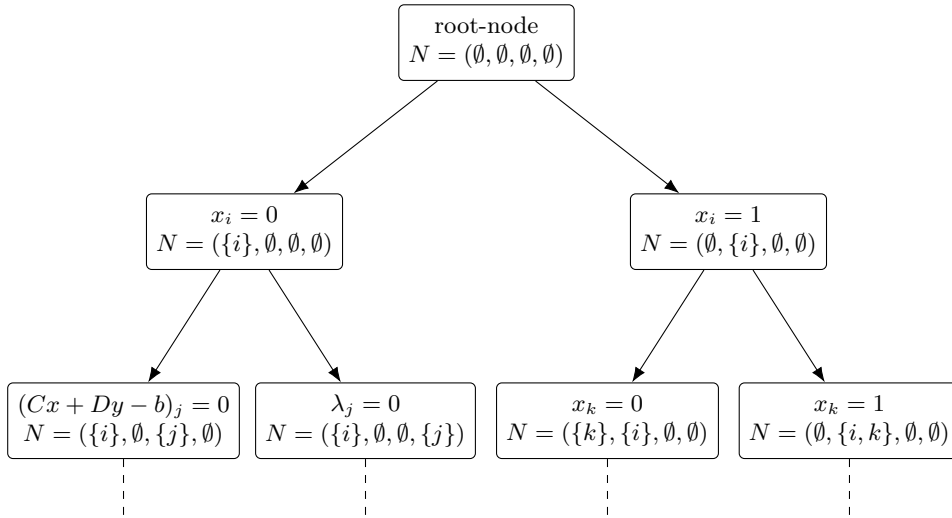


Figure 2.1: The structure of a branch-and-bound tree.

of scope for many applications in practice. Therefore, in branch-and-bound, several techniques are used to reduce the overall number of subproblems by exploiting the tree structure in which the problems are created.

### 2.5.2 Bounding

If a node problem  $N$  is infeasible, then every subsequent node in the branch-and-bound tree is infeasible as well. This is because subsequent nodes are generated by branching, i.e., by adding further constraints to node  $N$ . Hence, we do not perform a branching step at this node, i.e., we do not add subproblems that are based on  $N$  to the set of open nodes  $\mathcal{N}_o$ . This is called *pruning* the node  $N$  and can be visualized by chopping a branch of the tree.

Otherwise, if a globally optimal solution to a node problem  $N$  satisfies every integrality and complementarity constraint, it is feasible for the original problem (2.12). Hence, no further branching is required and we prune the node  $N$ . Moreover, we get an upper bound for the optimal objective function value of Problem (2.12); see Section 2.4. The best known upper bound is called the *incumbent* and will be denoted with  $u$  throughout this thesis. At the start of the branch-and-bound procedure, the incumbent is usually initialized with positive infinity because no upper bound is known yet. In single-level mixed-integer optimization, there are several heuristics to compute integer-feasible points that provide a finite value of  $u$ . They can be used as a warm start for the method. For example, one can round a fractional optimal solution of the continuous relaxation and repair it if it is infeasible; see, e.g., Wallace (2010) and Fischetti et al. (2005). However, in the context of bilevel optimization, this is more sophisticated because a feasible solution

to a bilevel problem has to satisfy lower-level optimality; see Section 2.4. Nonetheless, one might use problem specific knowledge that allows to obtain a finite incumbent in advance.

If a solution  $(x^N, y^N, \lambda^N)$  to a node problem  $N$  satisfies  $c_x^\top x^N + c_y^\top y^N \geq u$ , we prune the node  $N$  because further exploring the subtree rooted in  $N$  cannot yield a better upper bound. Note that even with the introduction of pruning techniques, solving a problem using branch-and-bound may be equivalent to a complete enumeration in the worst case. Fortunately, this is rarely the case in practice. With the branching and pruning rules at hand, we formally state a branch-and-bound procedure; see Algorithm 1.

We prune a node if it is either infeasible (Line 5), if its optimal objective function value exceeds the incumbent value (Line 9), or if its solution satisfies all integrality and complementarity constraints (Line 11). In the latter case we obtain a bilevel-feasible point and use it to improve our incumbent  $u$  if possible; see Lines 8 and 11. The branching step is implemented in Line 13. After finitely many iterations, Algorithm 1 reaches Line 17 and terminates either with a globally optimal solution to the KKT reformulation (2.14) (Line 18) or with a correct indication of infeasibility (Line 20); see Beck and Schmidt (2021).

It remains open how to select a subproblem  $N \in \mathcal{N}_o$  in Line 2 when multiple options are available. The same holds for the choice of the constraint to branch on, if multiple constraints are violated by the node solution; see Line 13. Both subjects are addressed in the following.

### 2.5.3 Node Selection

There are various node selection strategies presented in the literature; see, e.g., Pastor and Corominas (2000), Huang et al. (2021), and the references therein. Two intuitive and easy-to-implement node selection strategies are *depth-first search* and *breadth-first search*. The former, that we also use in the implementation of Algorithm 3 in Section 3, always chooses the deepest unexplored node, i.e., it explores each branch as far as possible before backtracking. Therefore, depth-first search aims to quickly find an incumbent that is used to prune other subtrees. However, it may spend a significant amount of time in an unpromising part of the branch-and-bound tree, potentially reaching nodes that prove to be infeasible. In contrast, breadth-first search first explores every node in one level of the branch-and-bound tree before descending into a deeper level. Therefore, it usually takes more time to find an incumbent. However, if it does, one may prune subtrees early in the tree leading to a large reduction of the overall branch-and-bound nodes. Another node selection strategy that is worth mentioning is the *best-bound search* which is the default approach in state-of-the-art MIP solvers like CPLEX. Here, a node is chosen that yields the smallest lower bound over all unexplored nodes.

---

**Algorithm 1:** A branch-and-bound method to solve the KKT reformulation (2.14).

---

**Input:** An instance of Problem (2.14),  $\mathcal{N} = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$ , and  $u \leftarrow +\infty$ .

**Output:** A globally optimal solution  $(x^*, y^*, \lambda^*)$  to the KKT reformulation (2.14) or a correct indication of infeasibility.

```

1 while  $\mathcal{N} \neq \emptyset$  do
2   Choose any  $N = (Z, O, P, \hat{D}) \in \mathcal{N}$  and set  $\mathcal{N} \leftarrow \mathcal{N} \setminus N$ .
3   Solve node  $N$ , i.e., Problem (2.24) for given  $Z, O, P$ , and  $\hat{D}$ .
4   if Problem  $N$  is infeasible then
5     Go to Line 1.
6   else
7     Let  $(x^N, y^N, \lambda^N)$  denote the solution of node  $N$ .
8     if  $c_x^\top x^N + c_y^\top y^N \geq u$  then
9       Go to Line 1.
10    else if  $(x^N, y^N, \lambda^N)$  is feasible for Problem (2.14) then
11      Set  $(x^*, y^*, \lambda^*) \leftarrow (x^N, y^N, \lambda^N)$  and  $u \leftarrow c_x^\top x^N + c_y^\top y^N$ 
        and go to Line 1.
12    else
13      Choose any  $i \in [n_x]$  with  $x_i \notin \{0, 1\}$  or  $j \in [l]$  with
         $\lambda_j (Cx + Dy - b)_j > 0$  and set
         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(Z \cup \{i\}, O, P, \hat{D}), (Z, O \cup \{i\}, P, \hat{D})\}$  or
         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(Z, O, P \cup \{j\}, \hat{D}), (Z, O, P, \hat{D} \cup \{j\})\}$ ,
        respectively.
14    end
15  end
16 end
17 if  $u < +\infty$  then
18   return A globally optimal solution  $(x^*, y^*, \lambda^*)$  to Problem (2.14).
19 else
20   return “Problem (2.14) is infeasible.”
21 end

```

---

#### 2.5.4 Branching Rules

To solve Problem (2.14), branching can be done on fractional variables or on the disjuncts of a complementarity constraint, i.e., the cases in which either  $(Cx + Dy - b)_i$  or  $\lambda_i$  is enforced to be zero for an  $i \in [l]$ . For the ease of reading, we sometimes refer to the former as “branching on integrality constraints” and to the latter as “branching on complementarity constraints”. A straightforward approach is to branch on the integrality or complementarity constraint that is the *most violated*. Note, that for Problem (2.14), this

will likely prioritize branching on complementarity constraints because the violation of the integrality constraints is bounded by one. Conversely, one can branch on the constraint that is *least violated* to prioritize the integrality conditions.

A more advanced branching rule is the so-called *strong branching*. For each violated integrality or complementarity constraint, one temporarily branches on it and solves the resulting subproblem to estimate the impact of that decision on the objective function value. This is computationally very expensive but it also leads to good branching decisions that can drastically reduce the number of nodes explored during the branch-and-bound search. Another branching rule is *pseudo-cost branching*. This technique tracks previous branching decisions and their impact on the objective function value. It then selects the variable or disjunct expected to perform best based on past outcomes. While it may be less effective early in the tree, its performance improves as more branching information becomes available. Modern solvers as CPLEX or Gurobi often use combinations of strong branching and pseudo-cost branching, i.e., the former is used early in the branch-and-bound tree to obtain good branching decisions and at some point it is switched to the latter that benefits from the branching information. A more detailed discussion regarding branching rules can be found in Achterberg et al. (2005).

### 2.5.5 Branch-and-Bound for Bilevel Problems with Integer Lower Levels

We have seen that the branch-and-bound method is a suitable tool with many possible enhancements to solve MILP-LP bilevel problems. Therefore, a single-level reformulation of the bilevel problem is exploited. Note that the method can straightforwardly be applied to problems of the form (2.12) in which the binary constraints on the upper-level variables are replaced with general integrality constraints  $x_i \in \mathbb{Z}$  for  $i \in [n_x]$ . However, Assumptions 2.1.1 and 2.1.2 have to be satisfied. One can also consider nonlinear functions in both the upper and lower levels, provided that necessary and sufficient conditions for lower-level optimality are available to enable a single-level reformulation of the bilevel problem. Even if no such single-level reformulation is at hand, branch-and-bound methods may still be applicable to solve certain problem classes. We now consider two different ideas proposed in the literature to solve (mixed-)integer linear bilevel problems using branch-and-bound methods.

#### Branch-and-Bound by Moore and Bard (1990)

The work of Moore and Bard (1990) was the first to propose a solution algorithm for mixed-integer linear bilevel problems. Consider Problem (2.12) without coupling constraints, i.e.,  $B = 0$ , and with additional binary con-

straints on the lower-level variables, i.e.,

$$\begin{aligned}
& \min_{x \in \mathbb{R}^{n_x}, y} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax \geq a, \quad x \in \{0, 1\}^{n_x}, \\
& y \in \arg \min_{\bar{y} \in \mathbb{R}^{n_y}} \left\{ d^\top \bar{y} : Cx + D\bar{y} \geq b, \bar{y} \in \{0, 1\}^{n_y} \right\}.
\end{aligned} \tag{2.25}$$

Problem (2.25) is  $\Sigma_2^P$ -hard as shown by Caprara et al. (2014), i.e., adding integrality constraints to the lower-level problem significantly increases the difficulty of the bilevel problem. Since the lower level of Problem (2.29) is an ILP, we cannot state necessary and sufficient optimality conditions. Hence, we cannot apply a single-level reformulation as we did for Problem (2.12). In the work by Moore and Bard (1990), a branch-and-bound procedure is described that can be used to solve Problem (2.25). It solves bilevel problems in each node of the branch-and-bound tree for which the integrality constraints are relaxed in both levels, and it branches on fractional and integer variables. A detailed description of the method as well as a proof of correctness can be found in Moore and Bard (1990) but in this section we will focus only on specific parts of the algorithm.

Consider the continuous counterpart of the bilevel problem (2.25), i.e.,

$$\begin{aligned}
& \min_{x \in \mathbb{R}^{n_x}, y} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax \geq a, \quad x \in [0, 1]^{n_x}, \\
& y \in \arg \min_{\bar{y} \in \mathbb{R}^{n_y}} \left\{ d^\top \bar{y} : Cx + D\bar{y} \geq b, \bar{y} \in [0, 1]^{n_y} \right\}.
\end{aligned} \tag{2.26}$$

Note that Problem (2.26) is not a relaxation of Problem (2.25); see Moore and Bard (1990). Hence, a globally optimal solution to Problem (2.26) does not provide a lower bound on the optimal objective function value of Problem (2.25). This is because relaxing the integrality constraints in the follower's problem enlarges his feasible set, which may enable the follower to select decisions resulting in a higher objective value for the leader. Moreover, if a globally optimal solution to the continuous counterpart (2.26) is feasible for the original problem (2.25), i.e., it satisfies all the integrality constraints, then it does not need to be globally optimal for Problem (2.25); see Moore and Bard (1990). In this scenario, we need to continue branching on integer-feasible variables.

A branch-and-bound method as previously described solves node problems

$N = (Z_x, O_x, Z_y, O_y)$  of the form

$$\begin{aligned}
& \min_{x \in \mathbb{R}^{n_x, y}} c_x^\top x + c_y^\top y \\
& \text{s.t. } Ax \geq a, \quad x \in [0, 1]^{n_x}, \\
& \quad x_i = 0, \quad i \in Z_x, \\
& \quad x_i = 1, \quad i \in O_x, \\
& \quad y \in S_N(x).
\end{aligned} \tag{2.27}$$

Here,  $Z_x, O_x \subseteq [n_x]$ ,  $Z_x \cap O_x = \emptyset$ , and  $S_N(x)$  is the set of optimal solution for the  $x$ -parameterized lower-level problem

$$\begin{aligned}
& \min_{y \in \mathbb{R}^{n_y}} d^\top y \\
& \text{s.t. } Cx + Dy \geq b, \quad y \in [0, 1]^{n_y}, \\
& \quad y_i = 0, \quad i \in Z_y, \\
& \quad y_i = 1, \quad i \in O_y
\end{aligned} \tag{2.28}$$

with  $Z_y, O_y \subseteq [n_y]$  and  $Z_y \cap O_y = \emptyset$ . That is, we have to solve a strongly NP-hard LP-LP bilevel problem in each node.

If a globally optimal solution to a node  $N$  in the branch-and-bound tree is integer feasible, we cannot prune that node. This is because it remains unclear whether additional branching can lead to points that are integer and bilevel feasible for the original problem (2.25) and yield better upper-level objective function values. Note that this is in contrast to the branch-and-bound method applied to the KKT reformulation (2.14) where satisfying all integrality and complementarity conditions allows for node pruning.

If a solution  $(x^N, y^N)$  to Problem (2.27) for  $Z_y = \emptyset$  and  $O_y = \emptyset$  is integer feasible, then the point is bilevel feasible and its upper-level objective function value  $c_x^\top x^N + c_y^\top y^N$  is an upper bound for the optimal objective function value of Problem (2.25). However, if either  $Z_y$  or  $O_y$  are not empty, i.e., the node problem  $N$  contains branching decisions on the lower-level variables, we do not obtain an upper bound; see Moore and Bard (1990). This is because branching on lower-level variables reduces the follower's decision space for various decisions of the leader, leading to bilevel-feasible and potentially integer-feasible solutions to the continuous counterpart (2.27) that are bilevel-infeasible for Problem (2.25). However, an upper bound can still be computed by fixing all leader's decisions  $x$  and solving the  $x$ -parameterized lower-level problem. The optimal follower's response cannot violate the upper-level constraints because they do not depend on  $y$ . Hence, a bilevel-feasible point is computed that provides an upper bound.

In summary, to apply the branch-and-bound method in Moore and Bard (1990) to Problem (2.25), we need to solve a linear bilevel problem in each node. Moreover, we cannot compute an upper bound from a node solution  $(x^N, y^N)$

if the node contains branching decisions on lower-level variables. Finally, we cannot prune a node if its solution satisfies all integrality constraints. Those difficulties combined limit the practical efficiency of the proposed branch-and-bound algorithm when addressing problems of the form (2.25).

### Branch-and-Bound by Fischetti et al. (2018)

In Fischetti et al. (2018), the authors describe a branch-and-bound framework to solve mixed-integer linear bilevel problems. In contrast to Moore and Bard (1990), the upper-level may include coupling constraints and they do not solve a bilevel problem in each node of the branch-and-bound tree. Instead, the node problems are based on the C-HPR of the bilevel problem; see Section 2.1. Moreover, they can compute upper bounds from the solutions of node problems involving branching decisions of the  $y$  variables. In the same article, the authors expand the branch-and-bound method by the use of cutting planes. Based on this work, we discuss the branch-and-cut framework in the following section.

## 2.6 Branch-and-Cut

Branch-and-cut methods combine the branch-and-bound framework with the use of cutting planes to separate infeasible points. The first time branch-and-cut methods have been used in the context of bilevel optimization was in the work of DeNegre and Ralphs (2009) for purely integer linear bilevel problems. Note that Problem (2.25) fits into this framework. Since then, many approaches based on branch-and-cut have been developed along with tailored cutting planes to tackle different classes of bilevel problems. Especially for (mixed-)integer bilevel settings, branch-and-cut methods are state of the art in numerous applications; see Kleinert et al. (2021b).

### 2.6.1 Cutting Planes

To describe branch-and-cut methods, we first introduce the notion of cutting planes. Therefore, we make the following definitions.

**Definition 2.6.1** (Valid Inequality; see DeNegre and Ralphs (2009)). *Let  $\Psi \subset \mathbb{R}^n$  be any set. An inequality defined by  $(\alpha, \beta, \tau) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}$  is called a valid inequality for  $\Psi$  if  $\alpha^\top x + \beta^\top y \leq \tau$  holds for all  $(x, y) \in \Psi$ .*

**Definition 2.6.2** (Cutting Plane). *Let  $\Psi \subset \mathbb{R}^n$  be any set and let  $(\tilde{x}, \tilde{y}) \in \mathbb{R}^n$  be a point with  $(\tilde{x}, \tilde{y}) \notin \Psi$ . A valid inequality for  $\Psi$  that is parameterized by  $(\alpha, \beta, \tau) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}$  is called a cutting plane (or cut) for  $(\tilde{x}, \tilde{y})$  if it is violated by  $(\tilde{x}, \tilde{y})$ , i.e., if  $\alpha^\top \tilde{x} + \beta^\top \tilde{y} > \tau$  holds.*

We make use of the above definitions as follows. Let  $\Theta \subseteq \mathbb{R}^n$  be a set that contains both the point  $(\tilde{x}, \tilde{y})$  and the set  $\Psi$ , i.e.,  $(\tilde{x}, \tilde{y}) \in \Theta$  and  $\Psi \subseteq \Theta$ . Based

on the Definitions 2.6.2 and 2.6.1, a cutting plane satisfies two properties, i.e.,

$$(\tilde{x}, \tilde{y}) \notin \Theta \cap \left\{ (x, y) \in \mathbb{R}^n : \alpha^\top x + \beta^\top y \leq \tau \right\}$$

and

$$\Psi \subseteq \Theta \cap \left\{ (x, y) \in \mathbb{R}^n : \alpha^\top x + \beta^\top y \leq \tau \right\}.$$

Therefore, we can use the cut parameterized by  $(\alpha, \beta, \tau)$  to *separate* the point  $(\tilde{x}, \tilde{y})$  from  $\Theta$  without removing points that lie in  $\Psi$ . Note that the cutting plane can also cut off points that lie in  $\Theta \setminus \Psi$  other than  $(\tilde{x}, \tilde{y})$ , i.e., we may use it to separate multiple points. However, the existence of a cutting plane depends on both the point  $(\tilde{x}, \tilde{y})$  and the set  $\Psi$  and can only be guaranteed if  $(\tilde{x}, \tilde{y}) \notin \text{conv}(\Psi)$ ; see Chapter 3. Here, and for the remainder of this thesis, we denote with  $\text{conv}(\Lambda)$  the convex hull of a given set  $\Lambda$ .

In the context of mixed-integer bilevel optimization, the point  $(\tilde{x}, \tilde{y})$  is a bilevel-infeasible point that satisfies the integrality constraints and the set  $\Theta$  usually corresponds to the feasible region of a node in the branch-and-bound tree or the feasible region of the C-HPR. The set  $\Psi \subseteq \Theta$  is then a convex superset of  $\Theta \cap \mathcal{F}$ , where  $\mathcal{F}$  is the inducible region of the bilevel problem; see (2.6). Different types of cutting planes and their derivation are discussed in Section 2.6.4.

## 2.6.2 Application to Integer Bilevel Problems

Consider the bilevel problem (2.3) where the sets  $X$  and  $Y$  contain binary constraints on the  $x$  and  $y$  variables, respectively, i.e.,

$$\min_{x \in \mathbb{R}^{n_x}, y} F(x, y) \tag{2.29a}$$

$$\text{s.t. } G(x, y) \geq 0, \quad x \in \{0, 1\}^{n_x}, \tag{2.29b}$$

$$y \in \arg \min_{\bar{y} \in \{0, 1\}^{n_y}} \{f(x, \bar{y}) : g(x, \bar{y}) \geq 0\}. \tag{2.29c}$$

Note that Problem (2.25) fits into this setting. To solve Problem (2.29) using branch-and-cut, branching is performed on the integrality constraints and cutting planes are used to separate bilevel-infeasible points. The procedure starts with solving the C-HPR, i.e., the problem

$$\begin{aligned} \min_{x, y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \geq 0, \quad g(x, y) \geq 0, \\ & x \in [0, 1]^{n_x}, \quad y \in [0, 1]^{n_y}. \end{aligned} \tag{2.30}$$

The C-HPR (2.30) is used as the root-node relaxation in the branch-and-bound tree. It drops the lower-level optimality condition and the integrality

constraints on the  $x$  and  $y$  variables. An arbitrary node  $N := (Z_x, O_x, Z_y, O_y)$  in the branch-and-bound tree can be written as

$$\begin{aligned}
& \min_{x,y} F(x, y) \\
& \text{s.t. } G(x, y) \geq 0, \quad g(x, y) \geq 0, \quad H(x, y) \leq 0, \\
& \quad x \in [0, 1]^{n_x}, \quad y \in [0, 1]^{n_y}, \\
& \quad x_i = 0, \quad i \in Z_x, \quad x_i = 1, \quad i \in O_x, \\
& \quad y_i = 0, \quad i \in Z_y, \quad y_i = 1, \quad i \in O_y,
\end{aligned} \tag{2.31}$$

where  $H$  contains the cutting planes that have been added to node  $N$ . If a node solution satisfies all integrality constraints of Problem (2.29), it still can be bilevel-infeasible because the constraints of (2.31) do not address lower-level optimality. In this case, we separate the node solution by using a cutting plane. We define  $\Theta$  as the feasible region of Problem (2.31) and choose  $\Psi$  as a convex superset of  $\Theta \cap \mathcal{F}$ . The computation of a suitable set  $\Psi$  can be a challenging task because  $\mathcal{F}$  is typically not known in algebraic form. We will discuss its derivation in more detail in Chapter 3 of this thesis.

### 2.6.3 Description of the Method

We now discuss the branch-and-cut method in detail. A formal listing of the procedure is given in Algorithm 2.

Starting with the root-node (2.30), we iteratively solve node problems of the form (2.31). If a node problem  $N$  is infeasible, we prune the node in Line 5 of Algorithm 2. Otherwise, we obtain a solution  $(x^N, y^N)$  and check in Line 7 if its objective function value exceeds the incumbent  $u$ , i.e., the best known upper bound. If it does, we prune the node since further branching cannot lead to an improvement of the objective function value. Otherwise, we check if the point  $(x^N, y^N)$  satisfies all integrality constraints; see Line 9. If the node solution violates at least one of the integrality constraints of Problem (2.29), we branch on a fractional variable in Line 10.

If not, we fix the upper-level decisions  $x^N$  and solve the  $x^N$ -parameterized lower-level problem (2.29c) to obtain its globally optimal objective function value  $\Phi(x^N)$  as well as an optimal follower's response  $\hat{y}$ ; see Line 11. In Line 12 we check bilevel feasibility of the node solution. If it is bilevel feasible, we update the incumbent  $u$  in Line 13. Otherwise, we consider the point  $(x^N, \hat{y})$ . Whether this point is feasible for the bilevel problem (2.29) depends on the satisfaction of the upper-level constraints. Moreover, if  $(x^N, \hat{y})$  is bilevel feasible but  $S(x^N)$  is not a singleton, then there may exist other optimal lower-level solutions that yield smaller upper-level objective function values.

---

**Algorithm 2:** A branch-and-cut method for binary bilevel problems.

---

**Input:** An instance of Problem (2.29),  $\mathcal{N} = \{(\emptyset, \emptyset, \emptyset, \emptyset)\}$ , and  $u \leftarrow +\infty$ .

**Output:** A globally optimal solution  $(x^*, y^*)$  to the bilevel problem (2.29) or a correct indication of infeasibility.

```

1 while  $\mathcal{N} \neq \emptyset$  do
2   Choose any  $N = (Z_x, O_x, Z_y, O_y) \in \mathcal{N}$  and set  $\mathcal{N} \leftarrow \mathcal{N} \setminus N$ .
3   Solve node  $N$ , i.e., Problem (2.31) for given  $Z_x, O_x, Z_y$ , and  $O_y$ .
4   if Problem  $N$  is infeasible then
5     | Go to Line 1.
6   Let  $(x^N, y^N)$  denote the solution of node  $N$ .
7   if  $F(x^N, y^N) \geq u$  then
8     | Go to Line 1.
9   if  $(x^N, y^N) \notin \mathbb{Z}^n$  then
10    Choose any  $i \in [n_x]$  with  $x_i \notin \{0, 1\}$  or any  $i \in [n_y]$  with
         $y_i \notin \{0, 1\}$  and set
         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(Z_x \cup \{i\}, O_x, Z_y, O_y), (Z_x, O_x \cup \{i\}, Z_y, O_y)\}$  or
         $\mathcal{N} \leftarrow \mathcal{N} \cup \{(Z_x, O_x, Z_y \cup \{i\}, O_y), (O_x, O_x, Z_y, O_y \cup \{i\})\}$ ,
        respectively. Go to Line 1.
11   Compute  $\hat{y} \in \arg \min_{\bar{y} \in \{0,1\}^{n_y}} \{f(x^N, \bar{y}) : g(x^N, \bar{y}) \geq 0\}$  and  $\Phi(x^N)$ .
12   if  $f(x^N, y^N) = \Phi(x^N)$  then
13     | The point  $(x^N, y^N)$  is bilevel feasible. Set  $(x^*, y^*) \leftarrow (x^N, y^N)$ 
        and  $u \leftarrow F(x^N, y^N)$ . Go to Line 1.
14   else
15     | Solve the refinement problem (2.32).
16     if the refinement problem (2.32) is feasible then
17       | A solution  $(x^N, y^{\text{Ref}})$  to Problem (2.32) is bilevel-feasible.
18       if  $F(x^N, y^{\text{Ref}}) < u$  then
19         | Set  $u \leftarrow F(x^N, y^{\text{Ref}})$  and  $(x^*, y^*) \leftarrow (x^N, y^{\text{Ref}})$ 
20       Add a cutting plane to node  $N$  to separate the
        bilevel-infeasible point  $(x^N, y^N)$  and go to Line 3.
21   end
22 end
23 if  $u < +\infty$  then
24   | return A globally optimal solution  $(x^*, y^*)$  to Problem (2.29).
25 else
26   | return “Problem (2.29) is infeasible.”
27 end

```

---

To address both topics, we solve the so-called *refinement problem*

$$\begin{aligned}
\min_{x,y} \quad & F(x, y) \\
\text{s.t.} \quad & (x, y) \in P, \\
& x_I = x_I^N, \quad I \subseteq [n_x], \\
& f(x^N, y) \leq \Phi(x^N)
\end{aligned} \tag{2.32}$$

in Line 15; see Fischetti et al. (2017) for further details. That is the HPR extended by a temporarily fixing of linking variables  $x_I$  to the values of the node solution as well as the constraint  $f(x^N, y) \leq \Phi(x^N)$ . The latter ensures optimality of the  $x^N$ -parameterized lower-level. If Problem (2.32) is feasible, then every optimal solution to it leads to a solution to the  $x^N$ -parameterized lower-level solution that is best for the leader; see Lines 16–17. If the resulting point improves the incumbent  $u$ , we update it; see Lines 18–19. Otherwise, the leader’s decision  $x^N$  does not lead to a bilevel-feasible point. Finally, in Line 20, we derive a cutting plane to separate the bilevel-infeasible node solution  $(x^N, y^N)$  and re-solve the node.

When the set  $\mathcal{N}_o$  of open nodes becomes empty in some iteration, we terminate the algorithm in Line 23. If we found at least one bilevel-feasible point during the procedure, we terminate with a globally optimal solution of Problem (2.29); see Line 24. Otherwise, we identify the bilevel problem as infeasible; see Line 26. We can state the correctness of Algorithm 2 as follows.

**Theorem 2.6.3** (see, e.g., DeNegre and Ralphs (2009) for ILP-ILP bilevel problems). *Assume that we can solve every node problem (2.31) to global optimality or prove its infeasibility in a finite number of steps. Moreover, assume that for every time we are in Line 20 of Algorithm 2, we can compute a cutting plane as defined in Definition 2.6.2 in finitely many steps. Then, the method terminates in a finite number of iterations with either a globally optimal solution to the bilevel problem (2.29) or with a correct indication of infeasibility.*

*Proof.* Note that the Problem (2.29) is purely integer and all variables are bounded. Therefore, the shared constraint set  $\tilde{\Omega}$  consists of only finitely many points that we need to examine to obtain a globally optimal solution of Problem (2.29), if any exists. Hence, even in the worst case, we only need to construct finitely many branch-and-bound nodes. In Line 11 of the algorithm, we can solve  $x^N$ -parameterized lower level (2.29c) in a finite number of steps because all variables are binary. The same argument applies to the refinement problem (2.32) that we solve in Line 15. Moreover, we cannot get stuck in the Lines 3–20 of Algorithm 2 for a node  $N$  in the branch-and-bound tree. This is because a cutting plane as defined in 2.6.2 and applied to the setting in (2.29) separates at least one integer-feasible point in  $\tilde{\Omega}$ , namely the bilevel-infeasible

node solution  $(x^N, y^N)$ . Hence, by branching and by cutting, we can examine every integer-feasible point in a node  $N$  in finitely many iterations. With the assumption that the cut generation can be done in finitely many steps, the remainder follows from the correctness of the branch-and-bound method that is shown, e.g., in Beck and Schmidt (2021).  $\square$

Note that Algorithm 2 can also be used to solve more general (mixed-) integer bilevel problems of the form (2.3) under tailored conditions. This has already been done for several problem settings; see, e.g., Fischetti et al. (2018) or Gaar et al. (2023) and is still part of current research. However, the performance of branch-and-cut methods strongly relies on the efficiency of the computed cutting planes. As we will see in Chapter 3, computing potentially deep cuts can be a challenging task.

#### 2.6.4 Examples of Cutting Planes in the Literature

The development of tailored cutting planes that can be applied in branch-and-cut algorithms to solve (mixed-)integer bilevel problems has been the focus of several studies over the past decade; see, e.g., Caprara et al. (2016), Fischetti et al. (2018), or Gaar et al. (2023). An overview of existing cutting planes can be found in Tahernejad and Ralphs (2020), which builds the foundation of this section. We now present several cutting planes that can be used in Line 20 of Algorithm 2, applicable both to integer linear bilevel problems as well as more general problem settings. The computation of cutting planes is frequently done by solving a cut generation problem that depends on the feasible region of the current node in the branch-and-bound tree; see, e.g., Fischetti et al. (2018) or Gaar et al. (2023). In this case, the cuts are added locally in the branch-and-bound tree, i.e., they are only used in the subtree that is rooted in the node where the cut was invoked. This prevents the cutting planes from excluding bilevel-feasible points that lie outside the feasible region of the node problem.

Let us revise Problem (2.29). Arguably the fastest way to compute cutting planes to separate integer-feasible but bilevel-infeasible points  $(\tilde{x}, \tilde{y})$  is to use so-called *no-good* cuts. They specifically exploit the binary conditions on the  $x$  and  $y$  variables and can be written as

$$\sum_{i \in [n_x]: \tilde{x}_i = 0} x_i + \sum_{i \in [n_x]: \tilde{x}_i = 1} (1 - x_i) + \sum_{i \in [n_y]: \tilde{y}_i = 0} y_i + \sum_{i \in [n_y]: \tilde{y}_i = 1} (1 - y_i) \geq 1.$$

No-good cuts are computationally very cheap and easy to implement. Moreover, they can be used to tackle bilevel problems with general continuous objective and constraint functions in both the upper and the lower level as long as all upper- and lower-level variables are assumed to be binary. However, they are also very shallow as they only separate one single point at a time.

Another type of cutting planes are *integer no-good cuts*. As the name suggests, they are a straightforward generalization of the no-good cuts to the integer case. For integer variables  $x_i$  and  $y_j$  with bounds  $x_i^- \leq x_i \leq x_i^+$  and  $y_j^- \leq y_j \leq y_j^+$ , we represent them with binary vectors  $v^i$  and  $w^j$  first,

$$x_i = -2^{s_{x,i}} v_{s_{x,i}}^i + \sum_{k=0}^{s_{x,i}-1} 2^k v_k^i, \quad y_j = -2^{s_{y,j}} w_{s_{y,j}}^j + \sum_{k=0}^{s_{y,j}-1} 2^k w_k^j$$

with

$$s_{x,i} := \lceil \log_2 (\max \{|x_i^-|, |x_i^+|\}) + 1 \rceil, \quad s_{y,j} := \lceil \log_2 (\max \{|y_j^-|, |y_j^+|\}) + 1 \rceil$$

for  $i \in [n_x]$  and  $j \in [n_y]$ . Then, we apply a no-good cut on the auxiliary binary variables  $v$  and  $w$ . As for the no-good cuts, the advantages of integer no-good cuts are their cheap computation and easy implementation. Moreover, they can be used for a broad class of problems, e.g., we will use them in Chapters 3 and 4 of this thesis. However, using the binary expansion lifts the variable space from  $\mathbb{R}^n$  to  $\mathbb{R}^s$  with  $s = \sum_{i=1}^{n_x} s_{x,i} + \sum_{i=1}^{n_y} s_{y,i}$ , which can quickly lead to memory issues for large-sized instances along with numerical problems. If one considers integer linear bilevel problems, integer no-good cuts can be derived without the use of a binary expansion for the variables  $x$  and  $y$ . Instead, one uses the set of active constraints of a given point  $(\tilde{x}, \tilde{y})$  to separate it; see DeNegre and Ralphs (2009) for further details.

It is known that (integer) no-good cuts perform bad in branch-and-cut methods in general because usually many of them are needed to solve problems to optimality. Therefore, other cutting planes have been developed that aim to cut deeper into the feasible region of the node problems, i.e., to remove as many integer-feasible points as possible. Among these, intersection cuts used by Fischetti et al. (2018) have become a state-of-the-art technique in mixed-integer linear bilevel optimization. They are computed using bilevel-free sets, i.e., sets that do not contain bilevel-feasible points, that are intersected with the feasible regions of the node problems. We introduce these sets in Section 3.3 of this dissertation.

Bilevel-free sets are also used in the work of Gaar et al. (2023), where the authors consider convex nonlinear integer bilevel problems with second-order cone constraints in the upper level and a convex quadratic objective function and linear constraints in the lower level. They use *disjunctive cuts* to separate bilevel-infeasible points and solve a second-order cone programming (SOCP) problem to compute them. In Chapter 3 of this thesis, we will generalize their approach to general convex integer nonlinear bilevel problems.

### 2.6.5 Branch-and-Cut for ILP-ILP Bilevel Problems: An Example

Algorithm 2 is presented for binary bilevel problems. Nevertheless, it can be directly applied to general integer bilevel problems as well. To illustrate this,

consider the following ILP-ILP bilevel problem

$$\begin{aligned}
& \min_{x \in \mathbb{Z}} && x - y \\
& \text{s.t.} && 7x - 2y \geq -2, \quad x - y \geq -2, \quad -3x - y \geq -8, \\
& && y \in \arg \min_{\bar{y} \in \mathbb{Z}} \{-x + \bar{y} \geq -2, x + 2\bar{y} \geq 2\}.
\end{aligned} \tag{2.33}$$

We start the branch-and-cut algorithm by solving the C-HPR of Problem (2.33) as the root-node, i.e., the problem

$$\begin{aligned}
& \min_{x, y \in \mathbb{R}} && x - y \\
& \text{s.t.} && 7x - 2y \geq -2, \quad x - y \geq -2, \quad -3x - y \geq -8, \\
& && -x + y \geq -2, \quad x + 2y \geq 2.
\end{aligned} \tag{2.34}$$

We compute the optimal solution  $(x^0, y^0) = (3/2, 7/2)$  which is fractional in both  $x$  and  $y$ . For this example, let us assume that we branch on fractional  $x$  variables if possible and that we do depth-first search. The resulting branch-and-bound tree is given in Figure 2.2. Branching on  $x$  creates two subproblems based on the C-HPR. One adds the additional constraint  $x \leq 1$  and the other one adds  $x \geq 2$ ; see Figure 2.3. We continue by solving the first of the two subproblems, i.e., node 1, and compute its optimal solution  $(x^1, y^1) = (1, 3)$ . The point is integer feasible and, hence, we compute the optimal solution to the  $x^1$ -parameterized lower-level problem, which is  $\hat{y} = 1$ . Since the point  $(1, 1)$  satisfies the upper-level constraints, it is bilevel feasible and we use it to set our incumbent  $u$  to its upper-level objective function value, i.e.,  $u = 0$ . The node solution  $(x^1, y^1) = (1, 3)$  itself is not bilevel feasible and we separate it using a cutting plane, e.g.,  $y \leq 2$ ; see Figure 2.4. We re-optimize the node problem with the cutting plane added and compute the solution  $(\tilde{x}^1, \tilde{y}^1) = (2/7, 2)$ .

We branch on  $x$  again, creating two subproblems that are based on node 1. The first one adds the constraint  $x \leq 0$  and has the solution  $(x^2, y^2) = (0, 1)$ ; see Figure 2.4. We prove the point to be bilevel feasible and update the incumbent to  $u = -1$ . The second subproblem adds the constraint  $x \leq 1$  which fixes  $x = 1$  and has the solution  $(x^3, y^3) = (1, 2)$ ; see Figure 2.4. The point yields an upper-level objective function value that is equal to the incumbent  $u$ . Consequently, we prune the node. Finally, we solve the remaining node problem, i.e., the C-HPR with the additional constraint  $x \geq 2$ , and obtain the integer-feasible solution  $(x^4, y^4) = (2, 2)$  with an optimal objective function value of zero. Since our incumbent  $u$  has a lower value, we prune the node. After that, the list of open nodes is empty and we have a finite incumbent  $u$ . As a result, the branch-and-cut algorithm terminates with the globally optimal solution to the bilevel problem  $(x^*, y^*) = (0, 1)$ .

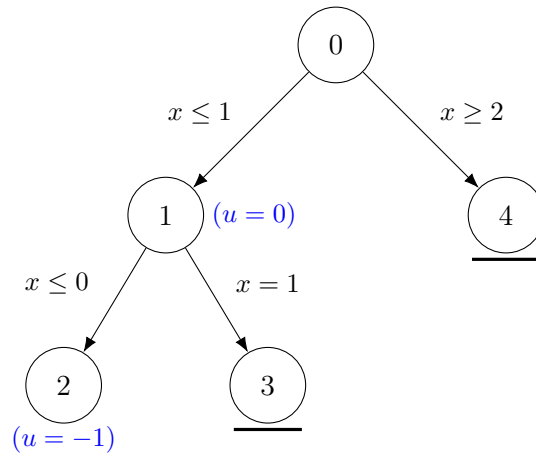


Figure 2.2: A possible branch-and-bound tree for Problem (2.33).

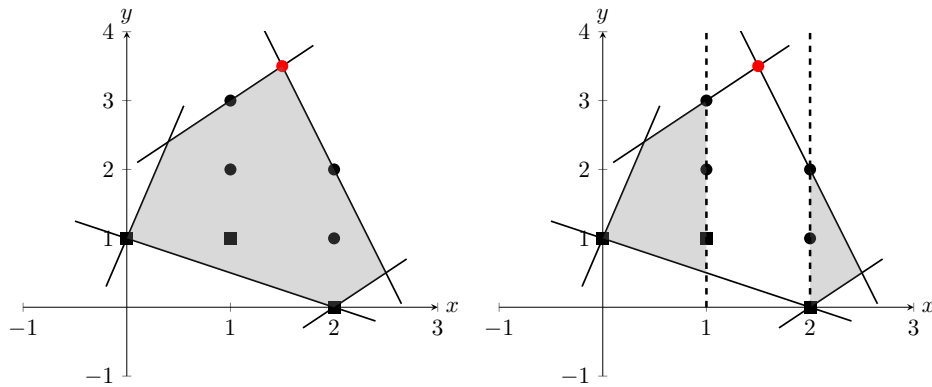


Figure 2.3: Left: The shared constraint set of Problem (2.33) is depicted in gray. The point  $(3/2, 7/2)$  in red is the bilevel-infeasible solution to the C-HPR. The points  $(0, 1)$ ,  $(1, 1)$ , and  $(2, 0)$  are bilevel feasible. Right: Partition of the feasible region of the C-HPR after branching is performed.

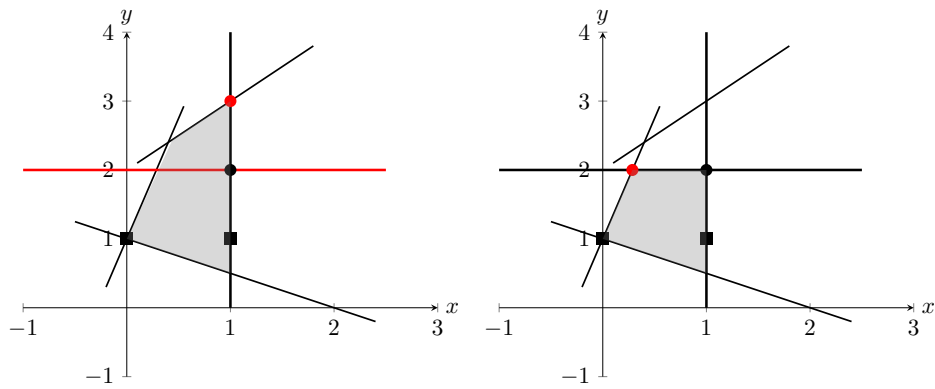


Figure 2.4: Left: The feasible region of node 1. The integer-feasible but bilevel-infeasible point  $(1, 3)$  in red is the solution to node 1. We separate it using the cutting plane  $y \leq 2$  and re-optimize the node. Right: The feasible region of node 1 after the cut is added. Its solution  $(2/7, 2)$  in red is fractional. After another branching step, we compute the bilevel-optimal point  $(0, 1)$  in node 2.

## Chapter 3

# Using Disjunctive Cuts in a Branch-and-Cut Method to Solve Convex Integer Nonlinear Bilevel Problems

This chapter presents the theoretical and numerical results from [AH1], along with further numerical studies not included in [AH1]. These additional results are discussed in Section 3.6.3.

This chapter is organized as follows. In Section 3.1, we put the content of this chapter into the context of this thesis. We describe the problem setting in Section 3.2. Then, in Section 3.3, we discuss disjunctive cuts in the context of our setup and give an in-depth discussion of how to compute them. Next, we describe the overall branch-and-cut framework including the use of disjunctive cuts in Section 3.4. Moreover, we prove the correctness of the method and discuss further techniques that can be applied to it. Afterward, in Section 3.5, we present comparisons of the cuts presented in Section 3.3 with the most prominent ones in the related literature. Further, we discuss the challenges that arise when we try to generalize the proposed method to the mixed-integer case. Section 3.6 contains the numerical results. We compare the presented branch-and-cut method with a branch-and-cut algorithm based on classic integer no-good cuts to illustrate the potential benefits from using disjunctive cuts. Furthermore, we test additional algorithmic techniques that can be applied to the method. Finally, we compare the proposed method to a state-of-the-art approach from the literature that is designed to solve integer bilevel problems with second-order cone constraints in the upper and a quadratic program in the lower level.

### 3.1 Motivation

We have seen in Section 2.3 that for bilevel problems with convex and continuous lower levels one can derive single-level reformulations using compact optimality certificates like KKT conditions or strong duality. This is no longer possible if the lower level includes integrality constraints. Hence, several branch-and-cut methods have been developed to address (M)ILP-(M)ILP bilevel problems; see, e.g., DeNegre and Ralphs (2009), Wang and Xu (2017), or Fischetti et al. (2018). The next step to further generalize bilevel problems w.r.t. their lower level is to consider convex integer nonlinear lower-level problems, which we address in this chapter.

The idea of this work has been motivated by the article of Gaar et al. (2023), where the authors consider convex, purely integer, and nonlinear bilevel problems that have SOCP constraints in the upper level and a convex quadratic objective function and linear constraints in the lower level. The authors developed problem-specific SOCP-based disjunctive cuts to solve such problems using a branch-and-cut framework. Based on their work, we develop a solution algorithm for bilevel problems with convex integer nonlinear problems in both levels. In the considered class of problems, the nonlinearities can appear in the objective function as well as in the constraints of both levels but we restrict ourselves to convex nonlinearities. This is a generalization of the problem setting studied in Gaar et al. (2023) as (i) we allow nonlinearities in the lower-level constraints, (ii) we do not restrict ourselves to second-order cone constraints, and (iii) we allow for jointly convex functions; as opposed to Gaar et al. (2023), where the nonlinear objective function of the lower level is quadratic in the lower-level variables.

### 3.2 Problem Statement

We consider the problem

$$\min_{x \in \mathbb{Z}^{n_x}, y} F(x, y) \tag{3.1a}$$

$$\text{s.t. } G(x, y) \leq 0, \tag{3.1b}$$

$$y \in \arg \min_{\bar{y} \in \mathbb{Z}^{n_y}} \{f(x, \bar{y}) : g(x, \bar{y}) \leq 0\}, \tag{3.1c}$$

where  $n := n_x + n_y$  and  $F, f : \mathbb{Q}^n \rightarrow \mathbb{Q}$ ,  $G : \mathbb{Q}^n \rightarrow \mathbb{Q}^m$ , as well as  $g : \mathbb{Q}^n \rightarrow \mathbb{Q}^l$  are continuous and jointly convex functions. Note that we change the sense of the upper- and lower-level constraints from “ $\geq$ ” to “ $\leq$ ” compared to Problem (2.3) to avoid distinguishing between convex objective functions and concave constraint functions. Moreover, the constraints in (3.1b) and (3.1c) may also contain bounds on the  $x$ - and  $y$ -variables. Without loss of generality, we assume that  $F$  is linear, because we can consider the epigraph reformulation

of Problem (3.1) otherwise. As discussed in Section 2.1, we consider the optimistic version of the given bilevel optimization problem.

The value-function reformulation of Problem (3.1) reads

$$\min_{x \in \mathbb{Z}^{n_x}, y \in \mathbb{Z}^{n_y}} F(x, y) \quad (3.2a)$$

$$\text{s.t. } G(x, y) \leq 0, \quad (3.2b)$$

$$g(x, y) \leq 0, \quad (3.2c)$$

$$f(x, y) \leq \Phi(x) \quad (3.2d)$$

with

$$\Phi(x) := \min_{y \in \mathbb{Z}^{n_y}} \{f(x, y) : g(x, y) \leq 0\}. \quad (3.3)$$

Dropping Condition (3.2d) leads to the HPR, i.e., the convex integer nonlinear problem (INLP)

$$\min_{(x, y) \in \tilde{\Omega}} F(x, y) \quad (3.4)$$

with  $\tilde{\Omega} := \{(x, y) \in \mathbb{Z}^n : G(x, y) \leq 0, g(x, y) \leq 0\}$ . The set  $\tilde{\Omega}$  is the shared constraint set and its projection onto the  $x$ -space is denoted with

$$\tilde{\Omega}_x := \left\{ x \in \mathbb{Z}^{n_x} : \exists y \text{ with } (x, y) \in \tilde{\Omega} \right\}.$$

Further removing the integrality constraints on the variables  $x$  and  $y$  leads to the C-HPR, i.e., the convex optimization problem

$$\min_{(x, y) \in \Omega} F(x, y) \quad (3.5)$$

with  $\Omega := \{(x, y) \in \mathbb{R}^n : G(x, y) \leq 0, g(x, y) \leq 0\}$ .

Similar to Fischetti et al. (2018), Tahernejad and Ralphs (2020), and Gaar et al. (2023), we make Assumptions 2.1.1 and 2.1.2, i.e., we assume boundedness of the HPR and the  $x$ -parameterized lower level. Note that if Assumption 2.1.1 holds, the C-HPR (3.5) cannot be unbounded due to the convexity of the problem. Hence, this implies the boundedness of the set  $\Omega$ . Moreover, Problem (3.1) satisfies Assumption 2.1.3 by construction.

### 3.3 Disjunctive Cuts

We tackle Problem (3.1) using a branch-and-cut method, where branching is performed on the integrality constraints and cutting planes are used to separate bilevel-infeasible points; see Section 2.6 for further details. In this section, we show how to derive cutting planes in the form of disjunctive cuts. Therefore, we make use of the following result taken from Fischetti et al. (2018) and Xu and Wang (2014), which also applies to our setup.

**Theorem 3.3.1.** *For any  $\hat{y} \in \mathbb{Z}^{n_y}$ , the set*

$$S(\hat{y}) := \{(x, y) \in \mathbb{R}^n : g(x, \hat{y}) \leq 0, f(x, y) > f(x, \hat{y})\} \quad (3.6)$$

*does not contain any bilevel-feasible point.*

The set  $S(\hat{y})$  in (3.6) is commonly denoted as the *bilevel-free set*; see Fischetti et al. (2018) or Gaar et al. (2023). When the objective function is convex but nonlinear, the set  $S(\hat{y})$  is not convex, which prevents us from using the standard theory of intersection cuts previously exploited in Fischetti et al. (2017) and Fischetti et al. (2018). Gaar et al. (2023) overcome this difficulty by using a disjunctive interpretation of the bilevel-free set. However, their approach only works under the assumption that the functions  $g_i$  are affine. In this chapter, we handle a more general case in which  $g_i$  and  $f$  are jointly convex, a setting to which the existing theoretical frameworks for deriving intersection and disjunctive cuts cannot be applied.

We later use the interior of  $S(\hat{y})$  and its complement to derive valid inequalities to cut off bilevel-infeasible points. In Section 3.3.4, we show how to compute bilevel-free sets, i.e., how to choose  $\hat{y}$ . Using the above theorem, for a given  $\hat{y} \in \mathbb{Z}^{n_y}$ , we now define the disjuncts

$$\mathcal{D}_i(\hat{y}) := \{(x, y) \in \mathbb{R}^n : g_i(x, \hat{y}) \geq 0\}$$

for  $i \in I := [l]$  and

$$\mathcal{D}_0(\hat{y}) := \{(x, y) \in \mathbb{R}^n : f(x, y) \leq f(x, \hat{y})\}.$$

The sets  $\mathcal{D}_i(\hat{y})$  for  $i \in I_0 := \{0, \dots, l\}$  are, in general, nonconvex and unbounded. Furthermore, each of the non-redundant disjuncts has an empty intersection with the interior of the bilevel-free set  $S(\hat{y})$  and we have

$$\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) = \text{int}(S(\hat{y}))^c, \quad (3.7)$$

where for an arbitrary set  $\Lambda$ , we denote  $\Lambda^c$  as the complement of  $\Lambda$ . In the following, we use the disjunction  $\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})$  intersected with the integer lattice to derive disjunctive cuts.

### 3.3.1 Computing Disjunctive Cuts

We now give a formal definition of disjunctive cuts in our context; see Section 2.6.1 for a general definition of cutting planes.

**Definition 3.3.2** (Disjunctive Cut). *Let  $\Theta \subseteq \Omega$  be non-empty and convex and let  $(\tilde{x}, \tilde{y}) \in \Theta$  be a bilevel-infeasible point. Moreover, let  $S(\hat{y})$  with  $\hat{y} \in \mathbb{Z}^{n_y}$  be any bilevel-free set and let  $\mathcal{D}_i(\hat{y})$  be the associated disjuncts for  $i \in I_0$ . Furthermore, let*

$$\alpha^\top x + \beta^\top y - \tau = 0 \quad (3.8)$$

be a hyperplane that is parameterized by  $\alpha \in \mathbb{Q}^{n_x}$ ,  $\beta \in \mathbb{Q}^{n_y}$ , and  $\tau \in \mathbb{Q}$ . Suppose that the following two statements hold:

- (i) The bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  is strictly on one side of the hyperplane, i.e.,  $\alpha^\top \tilde{x} + \beta^\top \tilde{y} - \tau > 0$ .
- (ii) Every point

$$(x', \bar{y}) \in \Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\}$$

is on the other side of the hyperplane, i.e.,  $\alpha^\top x' + \beta^\top \bar{y} - \tau \leq 0$ .

Then, the inequality  $\alpha^\top x + \beta^\top y - \tau \leq 0$  is called a disjunctive cut, which separates  $(\tilde{x}, \tilde{y})$  from all bilevel-feasible points inside  $\Theta$ .

In this chapter, the set  $\Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\}$  takes the role of  $\Psi$  in Definition 2.6.2. Note that the point  $(\tilde{x}, \tilde{y})$  in Definition 3.3.2 does not need to be integer feasible. Whenever we talk about a bilevel-infeasible point that is already integer feasible, we explicitly mention this in what follows.

**Remark 3.3.3.** A disjunctive cut as stated in Definition 3.3.2 separates a bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  from all other integer-feasible points that are inside  $\Theta$  but outside of  $\text{int}(S(\hat{y}))$  for a given  $\hat{y} \in \mathbb{Z}^{n_y}$ . If  $(\tilde{x}, \tilde{y})$  is integer feasible and outside or on the boundary of the bilevel-free set  $S(\hat{y})$ , then it would also be included in  $\Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right)$ . Hence, we have to exclude that point in (ii) of Definition 3.3.2 to ensure the existence of the respective disjunctive cut. In the following, we show how to compute the cut.

To compute a disjunctive cut, we solve the cut-generating problem

$$\begin{aligned} \max_{\alpha, \beta, \tau} \quad & \alpha^\top \tilde{x} + \beta^\top \tilde{y} - \tau && \text{(CGP)} \\ \text{s.t.} \quad & \alpha^\top x + \beta^\top y - \tau \leq 0 \quad \text{for all } (x, y) \in \Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\}, \\ & \|\alpha, \beta, \tau\|_1 \leq 1. \end{aligned}$$

The norm constraint ensures that (CGP) is bounded. We use the  $\ell_1$ -norm as it is commonly done in the literature; see, e.g., Fischetti et al. (2011) and Lodi et al. (2023). Note that (CGP) cannot be infeasible because the point  $(\alpha, \beta, \tau) = (0, 0, 0) \in \mathbb{Q}^{n_x} \times \mathbb{Q}^{n_y} \times \mathbb{Q}$  is feasible for (CGP). If a solution  $(\alpha, \beta, \tau)$  to (CGP) yields a strictly positive objective function value, then the resulting hyperplane  $\alpha^\top x + \beta^\top y - \tau$  separates the point  $(\tilde{x}, \tilde{y})$  from the set  $\Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\}$ . Overall, we have the following theorem.

**Theorem 3.3.4.** *Suppose that there is a bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  that is an extreme point of  $\Theta$ . Furthermore, let  $S(\hat{y})$  be a bilevel-free set for some  $\hat{y} \in \mathbb{Z}^{n_y}$ . Then, there exists a disjunctive cut that separates  $(\tilde{x}, \tilde{y})$  from  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  and it can be obtained by solving (CGP).*

*Proof.* If the point  $(\tilde{x}, \tilde{y})$  is an extreme point of the convex set  $\Theta$ , it cannot be represented as a proper convex combination of two different points inside  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$ . Therefore, the point is not in the convex hull of  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$ . Hence, we can separate  $(\tilde{x}, \tilde{y})$  from  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  via a linear inequality. Such an inequality is a disjunctive cut according to Definition 3.3.2 and is obtained when solving the (CGP).  $\square$

Note that the proof of Theorem 3.3.4 only relies on the convexity of  $\Theta$  and  $(\tilde{x}, \tilde{y})$  being an extreme point of  $\Theta$ . Contrary to the common assumption in the literature, when intersection or disjunctive cuts are derived from the convex hull of the disjunction, this approach does not require that the point  $(\tilde{x}, \tilde{y})$  is in the interior of the bilevel-free set. This means that we allow the point  $(\tilde{x}, \tilde{y})$  to be on the boundary or even outside of the bilevel-free set  $S(\hat{y})$ . We only use the bilevel-free set  $S(\hat{y})$  to get a description of a region that contains every bilevel-feasible point in  $\Theta$ , which is  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  in our case. In the context of a branch-and-bound algorithm, the set  $\Theta$  may correspond to a subset of the search space determined by the branching decisions together with additional constraints generated during the branching process.

**Remark 3.3.5.** *In the works of Fischetti et al. (2018) and Gaar et al. (2023), cutting planes are used in a branch-and-cut procedure to solve (mixed-)integer bilevel problems. Therefore, bilevel-infeasible points are separated from the convex hull of the disjunction using either intersection or disjunctive cuts. This is also what is standard in the single-level literature. We refer to Conforti et al. (2014) and Balas (2018) for the theoretical foundations of intersection and disjunctive cuts, respectively. However, in this approach, we want to separate  $(\tilde{x}, \tilde{y})$  from a discrete set. Therefore, we do not need to work with the convex hull of  $\mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  because we can obtain an implicit representation of the discrete set as shown in the next section.*

### 3.3.2 Cut Generation

The cut-generating problem (CGP) can be seen as a robust optimization problem with a nonconvex, discrete, and finite uncertainty set; see, e.g., Ben-Tal et al. (2009) and Bertsimas et al. (2011) for further details. Moreover, the problem itself is nonlinear and nonconvex and thus hard to solve. Therefore, we apply an adversarial approach; see, e.g., Gorissen et al. (2015).

For  $k = 0, 1, 2, \dots$ , we solve the relaxed cut-generating problem (RCGP)

$$\begin{aligned} \max_{\alpha, \beta, \tau} \quad & \alpha^\top \tilde{x} + \beta^\top \tilde{y} - \tau \\ \text{s.t.} \quad & \alpha^\top x' + \beta^\top y' - \tau \leq 0 \quad \text{for all } (x', y') \in \mathcal{Z}^k, \\ & \|\alpha, \beta, \tau\|_1 \leq 1, \end{aligned} \quad (\text{RCGP})$$

where  $\mathcal{Z}^k \subseteq \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  is a discrete and finite set. Note that (RCGP) can be reformulated as a linear optimization problem in  $(\alpha, \beta, \tau)$ . Hence, it can be solved in polynomial time. The set  $\mathcal{Z}^0$  can be initialized with the empty set or with bilevel-feasible points that lie in  $\Theta$ , if any are known. After solving (RCGP), we obtain an optimal solution  $(\alpha^k, \beta^k, \tau^k)$  and check if this solution satisfies the constraint

$$(\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \leq 0 \quad \text{for all } (x, y) \in \Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\},$$

i.e., we check if  $\Upsilon(\alpha^k, \beta^k, \tau^k) \leq 0$  holds, where

$$\begin{aligned} \Upsilon(\alpha^k, \beta^k, \tau^k) := & \quad (3.9) \\ \max_{x, y} \left\{ (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k : (x, y) \in \Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\} \right\}. \end{aligned}$$

Problem (3.9) is bounded due to the boundedness of  $\Omega \supseteq \Theta$ . If Problem (3.9) is infeasible, the set  $\Theta$  does not contain bilevel-feasible points as stated in the following lemma.

**Lemma 3.3.6.** *Let  $(\tilde{x}, \tilde{y})$  be a bilevel-infeasible point and let  $S(\hat{y})$  be any bilevel-free set. If Problem (3.9) is infeasible, then the set  $\Theta \subseteq \Omega$  is bilevel-free.*

*Proof.* If Problem (3.9) is infeasible, we have

$$\Theta \cap \mathbb{Z}^n \cap \left( \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \setminus \{(\tilde{x}, \tilde{y})\} = \Theta \cap \mathbb{Z}^n \cap \text{int}(S(\hat{y}))^c \setminus \{(\tilde{x}, \tilde{y})\} = \emptyset;$$

see (3.7) for the first equality. It follows that

$$\Theta \cap \mathbb{Z}^n \subseteq \text{int}(S(\hat{y})) \cup \{(\tilde{x}, \tilde{y})\}$$

holds, i.e., every integer-feasible point inside  $\Theta$  other than  $(\tilde{x}, \tilde{y})$  lies inside a bilevel-free set. Therefore,  $\Theta$  does not contain any bilevel-feasible point.  $\square$

This property can be exploited within a branch-and-cut procedure to prune nodes. Note that the feasibility of Problem (3.9) does not depend on the iteration  $k$ . To obtain  $\Upsilon(\alpha^k, \beta^k, \tau^k)$  we can solve the equivalent problem

$$\max_{x,y,b} (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \quad (3.10a)$$

$$\text{s.t. } (x, y) \in \Theta \cap \mathbb{Z}^n \setminus \{(\tilde{x}, \tilde{y})\}, \quad (3.10b)$$

$$f(x, \hat{y}) - f(x, y) \geq -M_0(1 - b_0), \quad (3.10c)$$

$$g_i(x, \hat{y}) \geq -M_i(1 - b_i), \quad i \in I, \quad (3.10d)$$

$$b_i \in \{0, 1\}, \quad i \in I_0, \quad (3.10e)$$

$$\sum_{i \in I_0} b_i \geq 1, \quad (3.10f)$$

where  $b_i$  are binary variables and  $M_i$  are sufficiently large constants for all  $i \in I_0$ . For each  $i \in I_0$ , Constraints (3.10c) and (3.10d) ensure that  $(x, y)$  has to be in the set  $\mathcal{D}_i(\hat{y})$  if the corresponding binary variable  $b_i$  is one. Otherwise, if  $b_i$  is zero, the constraint is trivially satisfied. Constraint (3.10f) ensures that at least one of the binary variables is one, i.e.,  $(x, y)$  has to be in at least one of the sets  $\mathcal{D}_i(\hat{y})$  for each feasible point  $(x, y, b)$ . Therefore, in Problem (3.10), we optimize over the set  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$ . Note that the implementation of Constraint (3.10b) can be done using an integer no-good cut on the point  $(\tilde{x}, \tilde{y})$ ; see Section 2.6.4 for a discussion. However, this is only required if  $(\tilde{x}, \tilde{y})$  is integer. Otherwise, we have  $(\tilde{x}, \tilde{y}) \notin \Theta \cap \mathbb{Z}$  and, hence, we do not need to involve a no-good cut.

### 3.3.3 Decomposition of (CGP)

Finding suitable big- $M$  values is a challenging task; see; e.g., Kleinert et al. (2020). However, such values are needed to guarantee equivalence of the problems (3.9) and (3.10) and to prevent numerical instabilities; see Section 2.3. To solve Problem (3.10), we can avoid using big- $M$  constraints by decomposing the problem into the  $l + 1$  subproblems

$$\begin{aligned} \max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & (x, y) \in \Theta \cap \mathbb{Z}^n \setminus \{(\tilde{x}, \tilde{y})\}, \\ & f(x, \hat{y}) - f(x, y) \geq 0 \end{aligned} \quad (3.11)$$

and, for every  $i \in I$ ,

$$\begin{aligned} \max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & (x, y) \in \Theta \cap \mathbb{Z}^n \setminus \{(\tilde{x}, \tilde{y})\}, \\ & g_i(x, \hat{y}) \geq 0. \end{aligned} \quad (3.12)$$

In each of the subproblems (3.11) and (3.12) one aims to find a point  $(x, y) \in \Theta \cap \mathbb{Z}^n \cap \mathcal{D}_i(\hat{y}) \setminus \{(\tilde{x}, \tilde{y})\}$ , which yields the largest objective value

for the current hyperplane that is parameterized by  $\alpha^k$ ,  $\beta^k$ , and  $\tau^k$ . Again, we use an integer no-good cut to exclude the point  $(\tilde{x}, \tilde{y})$  from the feasible region of the subproblems (3.11) and (3.12) if it is integer; see Section 2.6.4.

Note that the subproblems (3.11) and (3.12) are generally nonconvex and, thus, hard to solve. They are bounded due to the boundedness of  $\Theta$ . It is possible that some or even all of the subproblems (3.11) and (3.12) are infeasible, which depends on  $\hat{y}$  and  $\Theta$ . Hence, we consider  $I_0^{\text{feas}} \subseteq I_0$  as the index set of the feasible subproblems. If every subproblem appears to be infeasible, i.e.,  $I_0^{\text{feas}} = \emptyset$ , then Problem (3.9) is infeasible as well and, hence,  $\Theta$  is bilevel-free; see Lemma 3.3.6. Otherwise, for each subproblem  $i \in I_0^{\text{feas}}$  we get a solution  $(x^i, y^i)^k$  in every iteration  $k$ . Now, we check if there are solutions  $(x^i, y^i)^k$  with

$$(\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0. \quad (3.13)$$

If this is the case, those solutions are on the same side of the hyperplane induced by  $(\alpha^k, \beta^k, \tau^k)$  as the point  $(\tilde{x}, \tilde{y})$ . Then, we set

$$\mathcal{Z}^{k+1} \leftarrow \mathcal{Z}^k \cup \left\{ (x^i, y^i)^k : (\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0, i \in I_0^{\text{feas}} \right\} \quad (3.14)$$

and repeat the procedure by solving (RCGP) with the updated set  $\mathcal{Z}^{k+1}$ . If, on the other hand, there is no  $(x^i, y^i)^k$  satisfying (3.13) for some  $i \in I_0^{\text{feas}}$ , then  $(\alpha^k, \beta^k, \tau^k)$  strictly separates  $(\tilde{x}, \tilde{y})$  from all points  $(x, y) \in \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  and we have found an appropriate cut.

One advantage of considering the subproblems (3.11) and (3.12) instead of Problem (3.10) is that we neither need big- $M$  constants nor the respective binary vector  $b$ . Moreover, we can solve all subproblems in parallel as they are independent of each other. We now show that computing a disjunctive cut with the described adversarial approach only takes a finite number of iterations.

**Lemma 3.3.7.** *Suppose that there is a bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  that is an extreme point of  $\Theta$ . Furthermore, let  $S(\hat{y})$  be a bilevel-free set for some  $\hat{y} \in \mathbb{Z}^{n_y}$ . Then, computing the disjunctive cut from Theorem 3.3.4 can be done in finitely many steps.*

*Proof.* Solving (RCGP) can be done in polynomial time as it is a linear problem. Since the subproblems (3.11) and (3.12) are purely integer, it takes finitely many steps to solve them. Note that, although these problems are nonconvex in general, we only have to enumerate a finite number of points in the worst case to get a globally optimal solution. This is due to the boundedness of the shared constraint set  $\Omega$ , see Assumption 2.1.1, which implies the boundedness of  $\Theta$ . Furthermore, in every iteration  $k$ , we enlarge the set  $\mathcal{Z}^k$  by at least one point if we have not found a cutting plane yet; see (3.14). Hence, we only need finitely many updates until  $\mathcal{Z}^{k'} = \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$  holds for some  $k' < \infty$ .  $\square$

In general, the subproblems (3.11) and (3.12) are nonconvex INLPs but the following proposition gives a condition under which the problems are convex INLPs.

**Proposition 3.3.8.** *Let  $(\tilde{x}, \tilde{y})$  be an extreme point of  $\Theta$  and let the functions  $f$  and  $g_i$  be jointly convex and linear in  $x$  for all  $i \in I$ . Then, the problems (3.11) and (3.12) are convex INLPs for all  $i \in I$ .*

*Proof.* The set  $\Theta \setminus \{(\tilde{x}, \tilde{y})\}$  is convex since  $\Theta$  is convex and  $(\tilde{x}, \tilde{y})$  is an extreme point of  $\Theta$ . If the functions  $f$  and  $g_i$ ,  $i \in I$ , are jointly convex and linear in  $x$ , then the set of points  $(x, y) \in \mathbb{R}^n$  satisfying the condition

$$f(x, \hat{y}) - f(x, y) \geq 0$$

in Problem (3.11) is convex. Furthermore, the set of points  $x \in \mathbb{R}^{n_x}$  satisfying condition

$$g_i(x, \hat{y}) \geq 0$$

in Problem (3.12) is also convex for all  $i \in I$ . Note that  $\hat{y}$  is a constant in this context.  $\square$

### 3.3.4 Computing Bilevel-Free Sets

So far, we assumed that we are given a bilevel-free set  $S(\hat{y})$  with  $\hat{y} \in \mathbb{Z}^{n_y}$ ; see Definition 3.3.2. In this section, we show how to derive a suitable bilevel-free set for a given integer-feasible but bilevel-infeasible point  $(\tilde{x}, \tilde{y})$ . We discuss two approaches.

#### Improving-Solution Bilevel-Free Set

The idea of Fischetti et al. (2018) for mixed-integer linear bilevel problems consists of using an optimal follower's response for a given  $\tilde{x}$  as  $\hat{y}$  to create a bilevel-free set. We can also use this technique in our setting. However, in Fischetti et al. (2018),  $S(\hat{y})$  is additionally enlarged in a way that a bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  is guaranteed to be in the interior of this set; see Theorem 4 in Fischetti et al. (2018). In the description of  $S(\hat{y})$ , the authors replace linear lower-level constraints  $Ax + B\hat{y} \leq c$  with  $Ax + B\hat{y} \leq c + 1$ , under the assumption that all entries of  $A$ ,  $B$ ,  $c$  are integer. Note that this assumption can be satisfied w.l.o.g. by scaling the entries in the respective matrices, if rational input data is considered. For our nonlinear setting, we can impose a similar requirement on the problem, i.e., that the lower-level functions  $f$  and  $g$  should be integer-preserving. That is, they map integer inputs to integer outputs. However, in our nonlinear setting, this assumption cannot be satisfied without loss of generality. As a result, an extension of the bilevel-free set as done in Fischetti et al. (2018) is not possible in our context as the following example shows.

**Example 3.3.9.** Consider the problem

$$\begin{aligned} \min_{x \in \{0,1\}} \quad & x + y \\ \text{s.t.} \quad & y \in \arg \min_{\bar{y} \in \mathbb{Z}} \{2\bar{y}^2 - 9\bar{y} + 10 : 2.5^x \leq \bar{y}, 1 \leq \bar{y} \leq 3\}. \end{aligned} \quad (3.15)$$

The feasible region of the high-point relaxation is illustrated in Figure 3.1. The bilevel-feasible points of Problem (3.15) are given by  $(x, y) \in \{(0, 2), (1, 3)\}$  and the solution to the high-point relaxation is given by  $(x, y) = (0, 1)$ . For  $x = 0$ , the optimal follower's response is  $\hat{y} = 2$  with a lower-level objective function value of zero. Hence, we consider the bilevel-free set

$$S(2) = \{(x, y) : 2.5^x \leq 2, 2y^2 - 9y + 10 > 0\},$$

which is equivalent to

$$S(2) = \{(x, y) : x \in (-\infty, \ln(2)/\ln(2.5)], y \in (-\infty, 2) \cup (2.5, \infty)\}.$$

If we extend this set in the same way as it is done in Theorem 4 in Fischetti et al. (2018), we get

$$S^+(2) = \{(x, y) : 2.5^x \leq 3, 2y^2 - 9y + 10 \geq 0\},$$

which is equivalent to

$$S^+(2) = \{(x, y) : x \in (-\infty, \ln(3)/\ln(2.5)], y \in (-\infty, 2] \cup [2.5, \infty)\}.$$

However,  $S^+(2)$  contains the bilevel-feasible point  $(1, 3)$  in its interior as shown in Figure 3.1, i.e., its interior is not bilevel-free.

Note that without an extension of the bilevel-free set, the generation of intersection cuts or related disjunctive cuts for mixed-integer linear bilevel problems may fail if the point to be separated lies on the boundary of the bilevel-free set; see Fischetti et al. (2018). However, this is not the case for the disjunctive cuts proposed in this work; see the discussion below Theorem 3.3.4. Hence, we do not rely on an extension of a given bilevel-free set to compute disjunctive cuts.

### Improving-Direction Bilevel-Free Set

Another way to derive a suitable bilevel-free set is to find an improving direction  $\Delta y$  for the lower-level objective, and to construct a bilevel-free set based on the point  $(\tilde{x}, \tilde{y})$  and the direction  $\Delta y$ . To this end, we need to solve an auxiliary problem that finds a vector  $\Delta y$  such that  $(\tilde{x}, \tilde{y} + \Delta y)$  is feasible for the lower-level problem and such that  $f(\tilde{x}, \tilde{y} + \Delta y)$  is better than  $f(\tilde{x}, \tilde{y})$ . We generalize the type-1 scoop problem of the watermelon approach presented by Wang and Xu (2017), see also Fischetti et al. (2017), where integer linear

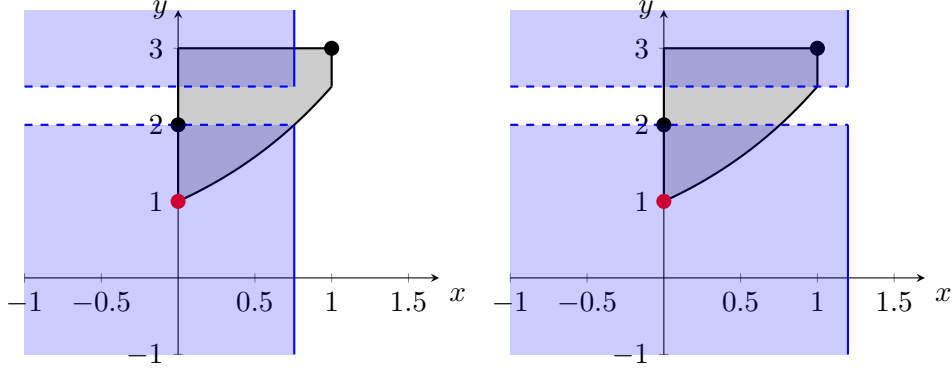


Figure 3.1: The shared constraint set of Problem (3.15) is shown by the gray shaded region. The point  $(0, 1)$  is the bilevel-infeasible solution to the HPR. The points  $(0, 2)$  and  $(1, 3)$  are bilevel feasible. Left: the bilevel-free set  $S(2)$  in blue. Right: the extended bilevel-free set  $S^+(2)$  in blue.

bilevel problems are considered. We construct the bilevel-free set as follows. Starting from  $\tilde{y} \in \mathbb{Z}^{n_y}$ , we try to find a direction  $\Delta y$  such that the bilevel-free set  $S(\tilde{y} + \Delta y)$  contains the bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  as far in its interior as possible to generate a potentially deep cut. Note that  $\tilde{y} + \Delta y$  takes the role of  $\hat{y}$  in the notation previously used. The bilevel-free set of the point  $\tilde{y} + \Delta y$  is given by

$$S(\tilde{y} + \Delta y) := \{(x, y) \in \mathbb{R}^n : g(x, \tilde{y} + \Delta y) \leq 0, f(x, y) > f(x, \tilde{y} + \Delta y)\}$$

and the point  $(\tilde{x}, \tilde{y})$  belongs to  $\text{int}(S(\tilde{y} + \Delta y))$  if it fulfills

$$g(\tilde{x}, \tilde{y} + \Delta y) < 0, \quad f(\tilde{x}, \tilde{y}) > f(\tilde{x}, \tilde{y} + \Delta y).$$

To compute a suitable direction  $\Delta y$ , we solve the convex MINLP

$$\max_{\Delta y \in \mathbb{R}^{n_y}, s \in \mathbb{R}^{l+1}, t \in \mathbb{R}} t \quad (3.16a)$$

$$\text{s.t.} \quad t \leq s_i, \quad i \in I_0, \quad (3.16b)$$

$$g_i(\tilde{x}, \tilde{y} + \Delta y) + s_i \leq 0, \quad i \in I, \quad (3.16c)$$

$$f(\tilde{x}, \tilde{y} + \Delta y) - f(\tilde{x}, \tilde{y}) + s_0 \leq 0, \quad (3.16d)$$

$$\Delta y \in \mathbb{Z}^{n_y}, \quad (3.16e)$$

$$s_i \geq 0, \quad i \in I_0, \quad (3.16f)$$

which is a generalization of the type-1 scoop problem in Wang and Xu (2017) to the nonlinear case. Problem (3.16) is convex in our setting because we assume that  $f$  and  $g_i$  are jointly convex. Hence, the left-hand sides of Constraints (3.16c) and (3.16d) are convex in  $\Delta y$ . These constraints ensure that the point  $(\tilde{x}, \tilde{y})$  is inside the bilevel-free set  $S(\tilde{y} + \Delta y)$ . If we find a solution

to Problem (3.16) such that (3.16c) and (3.16d) are strictly satisfied, i.e., all slack variables  $s_i$  for  $i \in I_0$  are strictly positive, then  $(\tilde{x}, \tilde{y}) \in \text{int}(S(\tilde{y} + \Delta y))$  holds. To this end, we maximize the auxiliary variable  $t$ , which is a lower bound for all slack variables  $s_i$ ,  $i \in I_0$ . Furthermore, we need the integrality constraints on the  $\Delta y$  variables because  $\tilde{y} + \Delta y$  has to be integer for being able to apply Theorem 3.3.1.

Note that we do not need  $(\tilde{x}, \tilde{y})$  to be inside the bilevel-free set  $S(\tilde{y} + \Delta y)$  to create a disjunctive cut; see Theorem 3.3.4. However, the solution value of  $t$  gives a measure of the distance between the bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  and the points inside  $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\tilde{y}))$ . Hence, it also gives a lower bound for the distance between  $(\tilde{x}, \tilde{y})$  and any bilevel-feasible point. Therefore, maximizing  $t$  leads to a bilevel-free set from which potentially deep cuts can be derived. We now derive some properties of Problem (3.16).

**Proposition 3.3.10.** *Let  $(\tilde{x}, \tilde{y}) \in \Theta \subseteq \Omega$  be given. Then, Problem (3.16) is solvable and the solution  $(\Delta y^*, s^*, t^*)$  to Problem (3.16) yields a non-negative optimal objective value  $t^*$ .*

*Proof.* Due to Assumption 2.1.2, the constraints in (3.16c) bound the  $\Delta y$  variables. Hence, the scoop problem cannot be unbounded. Since the functions  $f$  and  $g_i$ ,  $i \in I$ , are continuous, the feasible region of problem (3.16) is closed. If  $(\tilde{x}, \tilde{y})$  is in  $\Theta \subseteq \Omega$ , the point  $(\Delta y, s, t) = (0, 0, 0)$  is always feasible for Problem (3.16) and yields a non-negative objective value, i.e., the feasible region of (3.16) is non-empty. With the Weierstraß theorem we thus get the existence of an optimal solution for the scoop problem.  $\square$

**Proposition 3.3.11.** *Let  $(\tilde{x}, \tilde{y}) \in \Theta \subseteq \Omega$  be an integer-feasible point. Furthermore, let  $(\Delta y^*, s^*, t^*)$  be an optimal solution to Problem (3.16) that is parameterized by  $(\tilde{x}, \tilde{y})$  and suppose that  $t^* > 0$  holds. Then, the point  $(\tilde{x}, \tilde{y})$  is in the interior of the bilevel-free set  $S(\tilde{y} + \Delta y^*)$ .*

*Proof.* If  $t^* > 0$  holds, then every slack variable  $s_i^*$ ,  $i \in I_0$ , is strictly positive as well. Therefore, every inequality constraint of  $S(\tilde{y} + \Delta y^*)$  is strictly satisfied at  $(\tilde{x}, \tilde{y})$ . Hence, the point is in the interior of the bilevel-free set  $S(\tilde{y} + \Delta y^*)$ . Note that we need  $\tilde{y}$  to be integer because a set  $S(\tilde{y} + \Delta y^*)$  with fractional  $\tilde{y} + \Delta y^*$  is not guaranteed to be bilevel-free; see Definition 3.6.  $\square$

We can use Proposition 3.3.11 to detect, in which of the subproblems (3.11) and (3.12) we need to use an integer no-good cut when deriving a disjunctive cut for a given integer-feasible but bilevel-infeasible point  $(\tilde{x}, \tilde{y})$  and a bilevel-free set  $S(\tilde{y} + \Delta y^*)$ . Indeed, let  $\tilde{I} := \{i : s_i^* = 0\}$  be the set of indices for which the slack variables of the scoop problem take a value of zero in the optimal solution. If  $t^* > 0$ , then  $\tilde{I} = \emptyset$  and  $(\tilde{x}, \tilde{y}) \in \text{int}(S(\tilde{y} + \Delta y^*))$  holds. Hence,  $(\tilde{x}, \tilde{y}) \notin \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\tilde{y} + \Delta y^*))$  holds and we do not need to add an integer no-good cut in any of the subproblems (3.11) or (3.12). Otherwise,

if  $t^* = 0$ , then  $\tilde{I} \neq \emptyset$  and  $(\tilde{x}, \tilde{y}) \in \mathcal{D}_i(\tilde{y} + \Delta y^*)$  for every  $i \in \tilde{I}$ . Hence, we add an integer no-good cut to the  $i$ th subproblem for all  $i \in \tilde{I}$ .

**Remark 3.3.12.** *We can also consider a different objective function for the scoop problem (3.16), e.g.,  $\|s\|_1 = \sum_{i=1}^{l+1} s_i$ ; see Wang and Xu (2017) and Fischetti et al. (2017). This corresponds to a different norm to measure how deep the bilevel-infeasible point lies inside the bilevel-free set. Note that if we use the 1-norm in the objective function, then Proposition 3.3.11 no longer holds since a strictly positive value does not imply that all slack variables are strictly positive. However, we can still construct  $\tilde{I} := \{i : s_i^* = 0\}$  and apply an integer no-good cut to the  $i$ th subproblem in (3.11) and (3.12) for all  $i \in \tilde{I}$ . We did numerical tests to determine which objective function for the scoop problem performs best in our approach and the results indicated that it is the one introduced in Problem (3.16); see Section 3.6.3 for further details.*

*In Wang and Xu (2017), the bilevel-free set derived from the solution to the scoop problem is a polyhedron that is separated from the feasible region of the HPR. To avoid nonconvexities, the remaining set is then divided into polyhedral subsets by branching. These subsets are similar to the sets  $\Theta \cap \mathbb{Z}^n \cap \mathcal{D}_i$  for  $i \in I_0$  in our method. However, we use them in a different way. Instead of applying a multi-branching strategy, we rely on disjunctive cuts and use the subsets to compute them; see Section 3.3.3.*

## 3.4 A Branch-and-Cut Method

In this section, we describe how to use the results of Section 3.3 in a branch-and-cut procedure to solve convex integer nonlinear bilevel problems. In Section 3.4.1 we describe how to process a node in the branch-and-bound tree and how we compute disjunctive cuts to separate bilevel-infeasible node solutions. Furthermore, we prove the correctness of our method in Section 3.4.2. Finally, in Section 3.4.3, we discuss further algorithmic techniques that we can add to our method.

### 3.4.1 Description of the Method

A general branch-and-cut framework for integer bilevel problems can be found in Section 2.6. A formal description of our method is given in Algorithm 3. Starting with the continuous HPR (3.5) as the root-node relaxation, we solve in every node  $N$  of the branch-and-bound tree problems of the form

$$\min_{(x,y) \in \Omega_N} F(x, y), \quad (3.17)$$

where  $\Omega_N := \Omega \cap \{(x, y) \in \mathbb{R}^n : A^N x + B^N y \leq a^N\}$ ; see Line 1. The linear constraints  $A^N x + B^N y \leq a^N$  contain cuts that we have already added, branching decisions, and potential other cuts used by the underlying solver.

---

**Algorithm 3:** Processing Node  $N$ .

---

**Input:** A node problem of the form (3.17) and an incumbent value  $u$ .

- 1 Solve node problem (3.17).
- 2 **if** Problem (3.17) is infeasible **then**
- 3 | Fathom the current node.
- 4 Let  $(x^N, y^N)$  denote the solution to Problem (3.17).
- 5 **if**  $F(x^N, y^N) \geq u$  **then**
- 6 | Fathom the current node.
- 7 **if**  $(x^N, y^N) \notin \mathbb{Z}^n$  **then**
- 8 | Apply integrality branching.
- 9 Determine  $\bar{y} \in \arg \min_{y \in \mathbb{Z}^{n_y}} \{f(x^N, y) : g(x^N, y) \leq 0\}$  and  $\Phi(x^N)$ .
- 10 **if**  $G(x^N, \bar{y}) \leq 0$  **then**
- 11 | The point  $(x^N, \bar{y})$  is bilevel-feasible. Set  $u \leftarrow \min\{u, F(x^N, \bar{y})\}$ .
- 12 **if**  $f(x^N, y^N) > \Phi(x^N)$  **then**
- 13 | Compute  $\hat{y}$  to derive a bilevel-free set  $S(\hat{y})$  as discussed in Section 3.3.4. Set  $\Theta := \Omega_N$  and  $\mathcal{Z}^0 := \emptyset$ .
- 14 **for**  $k = 0, 1, 2, \dots$  **do**
- 15 | Compute a solution  $(\alpha^k, \beta^k, \tau^k)$  to (RCGP).
- 16 | Given  $(\alpha^k, \beta^k, \tau^k)$ , solve the subproblems (3.11) and (3.12) for all  $i \in I_0$ . Let  $I_0^{\text{feas}}$  denote the index set of the feasible subproblems.
- 17 | **if**  $I_0^{\text{feas}} = \emptyset$  **then**
- 18 | | Fathom the current node.
- 19 | **else**
- 20 | | Compute solutions  $(x^i, y^i)^k$  with  $i \in I_0^{\text{feas}}$ . Set  $\mathcal{M} := \{(x^i, y^i)^k : (\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0, i \in I_0^{\text{feas}}\}$ .
- 21 | | **if**  $\mathcal{M} \neq \emptyset$  **then**
- 22 | | | Set  $\mathcal{Z}^{k+1} \leftarrow \mathcal{Z}^k \cup \mathcal{M}$ .
- 23 | | **else**
- 24 | | | Add the cut  $\alpha^k x + \beta^k y - \tau \leq 0$  to Problem (3.17) and go to Line 1.
- 25 | | **end**
- 26 | | **end**
- 27 | **end**
- 28 **else**
- 29 | The node solution  $(x^N, y^N)$  is bilevel feasible. Set  $u \leftarrow \min\{u, F(x^N, y^N)\}$ . Fathom the current node.
- 30 **end**

---

Note that due to Assumption 2.1.1, each node problem is bounded. As stated in Section 3.2 we assume w.l.o.g. that the upper-level objective function  $F$  is linear, because we can solve the epigraph reformulation of Problem (3.1) otherwise.

As usual in branch-and-bound, we fathom a node  $N$  if it is infeasible; see Lines 2 and 3. Otherwise, we denote a solution to Problem (3.17) with  $(x^N, y^N)$ ; see Line 4. Note that, in our setting, we can always obtain a node solution  $(x^N, y^N)$ , which is an extreme point of  $\Omega_N$ . This is because we minimize a linear function over a continuous, convex, and bounded set. If the objective function value of the node solution  $(x^N, y^N)$  exceeds the upper bound  $u$  for the objective function value of the bilevel-optimal solution, we prune node  $N$ ; see Lines 5 and 6. Otherwise, we check if the solution is integer feasible and if it is not, we perform the branching step in Line 8 of Algorithm 3.

If the node solution  $(x^N, y^N)$  is integer feasible, we compute an optimal follower's response  $\bar{y}$  and the corresponding objective value  $\Phi(x^N)$  for the leader's decision  $x^N$ ; see Line 9. Note that the lower-level problem is an integer convex nonlinear problem, i.e., it is NP-hard. If the point  $(x^N, \bar{y})$  satisfies every constraint of the upper level, it is bilevel feasible and, hence, we update the incumbent  $u$  with the minimum of  $u$  and  $F(x^N, \bar{y})$ ; see Lines 10 and 11. In Line 12, we check if the integer-feasible node solution  $(x^N, y^N)$  is bilevel feasible by comparing its lower-level objective function value with the optimal objective function value of the  $x^N$ -parameterized lower level. If the point  $(x^N, y^N)$  is bilevel feasible, we update the incumbent  $u$  and prune the current node; see Line 29. Note that at this point we need to update the incumbent to hedge against the case that  $(x^N, y^N)$  is bilevel feasible but we obtain a  $\bar{y} \neq y^N$  such that  $(x^N, \bar{y})$  violates the upper-level constraints. In this case, we would not update  $u$  in Line 11. This scenario can happen because we do not assume that the lower level has a unique solution for all parameterizations  $x \in \tilde{\Omega}_x$ . Another way to deal with such situations is to perform a refinement step; see Line 15 of Algorithm 2. We will discuss this idea in Section 3.4.3.

If the node solution  $(x^N, y^N)$  is integer feasible but bilevel infeasible, we compute a disjunctive cut separating the node solution  $(x^N, y^N)$  from all bilevel-feasible points inside  $\Omega_N$ . Therefore, we compute a bilevel-free set  $S(\hat{y})$  in Line 13, which can either be done by taking the follower's response, i.e.,  $\hat{y} := \bar{y}$  (see Section 3.3.4), or by solving an auxiliary problem (see Section 3.3.4). In the latter case, we solve the scoop problem (3.16) with  $(\tilde{x}, \tilde{y}) := (x^N, y^N)$  and obtain an optimal solution  $(\Delta y^N, s^N, t^N)$ . The bilevel-free set is then constructed using the point  $\hat{y} := y^N + \Delta y^N$ .

After determining  $S(\hat{y})$ , we compute a disjunctive cut by using the adversarial approach as discussed in Section 3.3.2. Iteratively, we first solve (RCGP) to obtain a hyperplane separating the node solution  $(x^N, y^N)$  from every point in  $\mathcal{Z}^k$  for the current iteration  $k$ ; see Lines 14 and 15.

Afterward, we check if the computed hyperplane separates  $(x^N, y^N)$  from  $\Omega_N \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})) \setminus \{(\tilde{x}, \tilde{y})\}$ . Therefore, in Line 16, we determine for every set  $\mathcal{D}_i(\hat{y})$ ,  $i \in I_0$ , a point  $(x^i, y^i)^k \in \Omega_N \cap \mathbb{Z}^n \cap \mathcal{D}_i(\hat{y}) \setminus \{(\tilde{x}, \tilde{y})\}$ , which maximizes the value of the left hand-side of the given hyperplane. We do this by solving the subproblems (3.11) and (3.12) with  $\Theta := \Omega_N$ .

The feasibility of these subproblems depends on  $\Omega_N$  and  $\hat{y}$ . We denote the index set of feasible subproblems with  $I_0^{\text{feas}}$ . If each of the subproblems is infeasible, i.e.,  $I_0^{\text{feas}} = \emptyset$ , we prune the current node due to Lemma 3.3.6; see Line 18. This is because  $\Omega_N$  does not contain any bilevel-feasible point. The correctness of this pruning step is shown in Lemma 3.4.1 below. Otherwise, we consider the solutions  $(x^i, y^i)^k$  with  $i \in I_0^{\text{feas}}$  to the feasible subproblems and check if at least one of them lies on the same side of the computed hyperplane as the node solution  $(x^N, y^N)$ ; see Line 21. If this is the case, we extend  $\mathcal{Z}^k$  by those points in Line 22. Otherwise, we add the hyperplane as a local constraint to the node problem (3.17), i.e., it only affects the nodes of the subtree rooted in node  $N$ . Note that adding the disjunctive cut as a global constraint to the model is not valid because we may cut off bilevel-feasible points, which are not in the feasible region of the current node.

### 3.4.2 Correctness of the Method

The following lemma makes a statement about the correctness of the pruning step in Line 18.

**Lemma 3.4.1.** *Let  $\Omega_N$  be the feasible region of node  $N$  and let  $S(\hat{y})$  be any bilevel-free set for  $\hat{y} \in \mathbb{Z}^{n_y}$ . Furthermore, let  $(x^N, y^N) \in \Omega_N$  be a bilevel-infeasible point. If the subproblems (3.11) and (3.12) with  $\Theta := \Omega_N$  and  $(\tilde{x}, \tilde{y}) := (x^N, y^N)$  are infeasible for all  $i \in I$ , then we can prune node  $N$ .*

*Proof.* Let  $\Theta := \Omega_N$  and  $(\tilde{x}, \tilde{y}) := (x^N, y^N)$ . If the subproblems (3.11) and (3.12) are infeasible for all  $i \in I$  and any  $\hat{y} \in \mathbb{Z}^n$ , Problem (3.9) is infeasible as well. Due to Lemma 3.3.6, we have  $\Omega_N \cap \mathbb{Z}^n \subseteq \text{int}(S(\hat{y})) \cup \{(x^N, y^N)\}$ . Therefore,  $\Omega_N$  is bilevel-free and we can prune node  $N$ .  $\square$

With the above lemma we finally show the correctness of Algorithm 3.

**Theorem 3.4.2.** *Suppose that the Assumptions 2.1.1 and 2.1.2 hold. Then, the branch-and-cut procedure based on the node processing in Algorithm 3 terminates after a finite number of steps with a globally optimal solution to the bilevel problem (3.1) or with a correct indication of infeasibility.*

*Proof.* The assumptions ensure a finite number of solutions to the HPR, to the  $x$ -parameterized lower-level problem (3.1c) for every parameterization  $x \in \tilde{\Omega}_x$ , and, therefore, also to the bilevel problem (3.1). Let  $N$  be an arbitrary node in the branch-and-bound tree. The set  $\Omega_N \cap \mathbb{Z}^n$  is finite due to Assumption 2.1.1. Since every node problem (3.17) is convex, the node

solution  $(x^N, y^N)$  we obtain in Line 4 of Algorithm 3 is globally optimal for Problem (3.17). Let  $\mathcal{B}^N = \{(x, y) \in \mathbb{Z}^n : f(x, y) \leq \Phi(x)\} \cap \Omega_N$  be the set of bilevel-feasible points in the feasible region of node  $N$ . If we get a node solution  $(x^N, y^N) \in \mathbb{Z}^n$  that is bilevel infeasible, i.e.,  $(x^N, y^N) \notin \mathcal{B}^N$ , Algorithm 3 either adds a constraint to  $\Omega_N$ , which excludes at least this point from the finite set  $\Omega_N \cap \mathbb{Z}^n$ , see Line 24, or prunes the node  $N$ ; see Line 18. Note that the pruning step is correct due to Lemma 3.4.1.

From Lemma 3.3.7, we know that computing a disjunctive cut only takes finitely many steps. Furthermore, it only takes finitely many cuts to separate every integer-feasible but bilevel-infeasible point in  $\Omega_N$ . A complete evaluation of the branch-and-bound tree also takes finitely many steps. Therefore, if  $\bigcup_{N \in \mathcal{N}_e} \mathcal{B}^N$  is not empty, the algorithm terminates after a finite number of steps with a bilevel-optimal solution

$$(x^*, y^*) \in \arg \min_{x, y} \left\{ F(x, y) : (x, y) \in \bigcup_{N \in \mathcal{N}_e} \mathcal{B}^N \right\}.$$

Here,  $\mathcal{N}_e$  denotes the set of explored nodes in the branch-and-bound tree. Otherwise, if  $\bigcup_{N \in \mathcal{N}_e} \mathcal{B}^N$  is empty, we need only finitely many steps to exclude every integer-feasible point in  $\Omega$ , which results in a correct indication of infeasibility.  $\square$

**Remark 3.4.3.** *Throughout this chapter, we assume that the upper-level functions  $F$  and  $G$  as well as the lower-level functions  $f$  and  $g$  are jointly convex. This assumption is needed because the existence of a disjunctive cut computed in Lines 14–27 of Algorithm 3 can only be guaranteed, if the feasible region  $\Omega_N$  of a node problem  $N$  is convex and the node solution  $(x^N, y^N)$  is an extreme point of  $\Omega_N$ ; see Theorem 3.3.4. However, both conditions are independent of the lower-level objective function. Consequently, Algorithm 3 can also be applied to bilevel problems of the form (3.1) in which the functions  $F$ ,  $G$ , and  $g$  are jointly convex but the lower-level objective function  $f$  is nonconvex. Moreover, one could even consider problems with nonconvexities in the lower-level constraints that are shifted to the lower-level objective function using sufficiently large penalty terms. Note that such settings lead to nonconvexity of the lower level (3.1c), the scoop problem (3.16), and Problem (3.11). As a result, this frameworks introduce additional computational challenges. Addressing these challenges is beyond the scope of this dissertation but provides a promising direction for future work.*

### 3.4.3 Further Algorithmic Techniques

Now, we discuss further algorithmic techniques, that can be added to Algorithm 3 to enhance our method. The corresponding numerical studies are given in Section 3.6.3.

### Sibling Node Pruning

Line 18 of Algorithm 3 contains a pruning technique based on Lemma 3.3.6. We check the feasibility of the subproblems (3.11) and (3.12) that are parameterized by  $\hat{y}$  and prune the node if all subproblems are infeasible, i.e., if  $\Omega_N \cap \mathbb{Z}^n \subseteq \text{int}(S(\hat{y})) \cup \{x^N, y^N\}$  holds. As stated in Lemma 3.4.1, one can use any  $\hat{y}$  to construct the bilevel-free set  $S(\hat{y})$ . Based on this, if a node  $N$  is pruned due to Lemma 3.3.6 in Line 18 of Algorithm 3, we can check if  $\Omega_{\tilde{N}} \cap \mathbb{Z}^n \subseteq \text{int}(S(\hat{y})) \cup \{x^N, y^N\}$  holds. Here,  $\Omega_{\tilde{N}}$  is the feasible region of the sibling node  $\tilde{N}$  of node  $N$ . We do this by checking the feasibility of the subproblems (3.11) and (3.12) with  $\Theta := \Omega_{\tilde{N}}$  and the same  $\hat{y}$ . If we verify that the sibling node  $\tilde{N}$  of  $N$  is also contained in  $S(\hat{y})$ , we prune node  $\tilde{N}$ . We only apply this technique at the sibling node because its feasible region is the one which is most likely to be contained in the same bilevel-free set as the feasible region of node  $N$ . That is because the sets  $\Omega_{\tilde{N}}$  and  $\Omega_N$  only differ in the bounds of one variable, i.e., their difference is as small as possible. This approach might be useful in scenarios in which the sibling node  $\tilde{N}$  can be pruned and has a fractional solution. Then, we prune the node instead of branching on an integrality condition and creating two more subproblems in Line 8 of Algorithm 3.

### Initializing $\mathcal{Z}^0$

In Line 15 of Algorithm 3, we solve the relaxed cut-generating problem (RCGP). In the first iteration  $k = 0$ , we set  $\mathcal{Z}^0 = \emptyset$ ; see Line 13. We can also initialize  $\mathcal{Z}^0$  with all bilevel-feasible points that we have computed in Lines 9–11 and 29 for previous nodes and that lie in  $\Omega_N$ . Note that bilevel-feasible points outside  $\Omega_N$  cannot generally be used, as this often leads to infeasibility in the (RCGP).

### Refinement Procedure

As in Fischetti et al. (2018) and Algorithm 2, we can apply a refinement procedure in Algorithm 3 to obtain potentially better upper bounds for the optimal objective function value. After computing  $\Phi(x^N) \in \mathbb{R}$  in Line 9 of Algorithm 3, we can solve a restricted HPR in which we temporarily fix all linking variables, i.e.,  $x_i = x_i^N$  for  $i \in [n_x]$ , and add the constraint  $f(x^N, y) \leq \Phi(x^N)$ . Solving this restricted HPR leads to an  $x^N$ -parameterized lower-level solution, which minimizes the upper-level objective function and, hence, gives the best upper bound for it. Note that we can make use of this idea because we consider the optimistic version of the bilevel problem; see Section 2.1. If we include the refinement step, we have to solve a convex ILP every time the procedure is called. This can be computationally expensive. In case the solution set of the  $x$ -parameterized lower level is a singleton, which we usually do not know a priori, we do not benefit from using this approach.

### Handling the Subproblems (3.11) and (3.12)

The feasibility of the subproblems (3.11) and (3.12) does not depend on the parameters  $\alpha^k$ ,  $\beta^k$ , and  $\tau^k$ . Hence, if one of those problems is infeasible for  $k = 0$ , we do not solve them again for any iteration  $k \geq 1$ . Moreover, if we obtain a non-positive upper bound for the objective value of one of the subproblems (3.11) and (3.12), we terminate the solution process for this specific subproblem. This is because there is no point  $(x, y)$  that is feasible for the subproblem and that violates the inequality  $\alpha^k y + \beta^k y - \tau^k \leq 0$ .

Furthermore, at each iteration  $k$ , we can optionally decide whether to solve the subproblems (3.11) and (3.12) to global optimality. Alternatively, we may terminate the solution process early if a strictly positive lower bound on the objective function value is found. In the latter case we still obtain a point, that violates the inequality  $\alpha^k y + \beta^k y - \tau^k \leq 0$  but this violation may be small. By doing so, we save time in the optimization process of the subproblems but we may need more iterations  $k$  to find a cutting plane because the change of the hyperplane parameterized by  $(\alpha^k, \beta^k, \tau^k)$  in each iteration is potentially small.

We can also optionally decide if we solve every feasible subproblem in each iteration  $k$  or if we only consider those subproblems that led to a point with positive optimal objective function value in iteration  $k - 1$ . In the latter option, we may avoid solving subproblems multiple times that are not relevant for computing the hyperplane. Let us denote the index set for the feasible subproblems that had a negative globally optimal objective function value in iteration  $k - 1$  with  $I_0^{k, \text{feas}}$ . Then, in iteration  $k'$ , if  $I_0^{k', \text{feas}} = \emptyset$ , we perform a correction step in which we solve every subproblem (3.11) and (3.12) again for  $i \in I_0^{\text{feas}}$  with parameters  $\alpha^{k'}$ ,  $\beta^{k'}$ , and  $\tau^{k'}$ . We do this to check if the hyperplane is still valid for those subproblems that we did not solve again. If this is the case, we have a valid cutting plane and, otherwise, we repeat this procedure.

## 3.5 Comparison to the Literature and Generalization Challenges

In Section 3.5.1, we give insights about the strength of the proposed disjunctive cuts by comparing them to other cutting planes from the literature. The compared cutting planes are embedded in state-of-the-art methods for more specialized problem classes. Hence, we use two illustrative examples of a problem setting that each of the methods can handle and show that the cuts cannot be compared in general. Furthermore, in Section 3.5.2, we discuss the arising difficulties when trying to generalize the proposed method from the purely integer to the mixed-integer setting.

### 3.5.1 Comparison to Other Cuts from the Literature

As mentioned in the introduction, there are other methods that solve bilevel problems using cutting planes that are tailored for more specific problem settings. For example, in the work by Fischetti et al. (2018), intersection cuts are used in a branch-and-cut framework to solve mixed-integer linear bilevel problems. Furthermore, in the work by Gaar et al. (2023), disjunctive cuts are used to tackle integer bilevel problems with SOCP constraints in the upper level and a convex quadratic objective function in the lower level. In this section, we discuss the differences between the disjunctive cuts that we derived in this work and the cutting planes presented in the works of Fischetti et al. (2018) and Gaar et al. (2023) using two illustrative examples. Note that our approach can deal with convex integer nonlinear bilevel problems in which the nonlinearities in both levels can occur both in the objective function and in the constraints, which is in contrast to the other methods. However, we do not consider mixed-integer problems as it is done in Fischetti et al. (2018); see Section 3.5.2 for a discussion. Hence, we compare the strength of the cutting planes of the three different approaches on a setting that all three methods can handle, i.e., integer linear bilevel problems. In Section 3.3.4, we have discussed different methods on how to compute bilevel-free sets. Since the computation of a cutting plane heavily relies on the underlying bilevel-free set in all three methods, we use the same point  $\hat{y}$  to generate the bilevel-free set in the following comparison. Both in Fischetti et al. (2018) and Gaar et al. (2023), the authors propose the idea of taking an optimal follower's response  $\hat{y}$  to a given  $\tilde{x}$  to define the bilevel-free set  $S(\hat{y})$  that is used for the cut generation. This can also be done in our method; see Section 3.3.4. Therefore, we use this technique to compute a bilevel-free set in the following examples and compare the strength of the resulting cuts.

**Example 3.5.1.** *Consider the problem*

$$\begin{aligned} \min_{x \in \mathbb{Z}, y} \quad & x - 4y \\ \text{s.t.} \quad & 6x + 5y \leq 26, \\ & y \in \arg \min_{\bar{y} \in \mathbb{Z}} \{-11x + 5\bar{y} \leq 9, 5x - 10\bar{y} \leq -2\}. \end{aligned} \tag{3.18}$$

*Problem (3.18) is illustrated in Figure 3.2. The solution to the HPR is  $(x^0, y^0) = (1, 4)$  and the optimal follower's response for  $x^0 = 1$  is  $y^0 = 1$ . We get the bilevel-free set*

$$S(1) = \left\{ (x, y) : -\frac{4}{11} \leq x \leq \frac{8}{5}, y > 1 \right\}$$

*that we use to compute our disjunctive cut. With that, the disjuncts are given*

by

$$\mathcal{D}_0 = \{(x, y) \in \mathbb{R}^n : y \leq 1\}, \quad \mathcal{D}_1 = \left\{ (x, y) \in \mathbb{R}^n : x \leq -\frac{4}{11} \right\},$$

and

$$\mathcal{D}_2 = \left\{ (x, y) \in \mathbb{R}^n : x \geq \frac{8}{5} \right\}.$$

The resulting cut computed in Lines 14–24 of Algorithm 3 then reads  $-x + 2y \leq 2$ . Both the methods in Fischetti et al. (2018) and Gaar et al. (2023) enlarge the bilevel-free set to

$$S^+(1) = \left\{ (x, y) : -\frac{5}{11} \leq x \leq \frac{9}{5}, y \geq 1 \right\}.$$

The set  $S^+(1)$  does not contain bilevel-feasible points in its interior but it can have bilevel-feasible points on its boundary; see Fischetti et al. (2018) and Figure 3.2. The respective disjuncts are given by

$$\mathcal{D}_0 = \{(x, y) \in \mathbb{R}^n : y \leq 1\}, \quad \mathcal{D}_1 = \left\{ (x, y) \in \mathbb{R}^n : x \leq -\frac{5}{11} \right\},$$

and

$$\mathcal{D}_2 = \left\{ (x, y) \in \mathbb{R}^n : x \geq \frac{9}{5} \right\}.$$

For both methods from the literature, the computed cutting planes are equal and read  $-33x + 35y \leq 47$ .

It can be seen in Figure 3.2 that the cut we compute cuts deeper than the ones computed by the other methods. Moreover, it is the only one that separates the point  $(1, 2)$  as shown in Figure 3.2. This is despite using  $S(1)$  as bilevel-free set, which is only a subset of  $S^+(1)$  that is used in the other two methods. As discussed in Remark 3.3.5, we separate the bilevel-infeasible point from the convex hull of the integer-feasible points inside the disjuncts  $\mathcal{D}_i$  for  $i \in I_0$ . Therefore, our disjunctive cut may cut into the intersection of  $\Theta$  with the disjuncts  $\mathcal{D}_i$  and separate fractional points that are in  $\Theta \cap \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})$ . Note that in Example 3.5.1,  $\Theta$  can be replaced by the continuous relaxation of the shared constraint set  $\Omega$  as we consider the high-point relaxation of the bilevel problem, i.e., the root-node. We see in Example 3.5.1 that the disjunctive cut intersects with  $\Omega \cap \mathcal{D}_0$  and  $\Omega \cap \mathcal{D}_2$ ; see Figure 3.2. This is in contrast to the other two methods where the bilevel-infeasible point is separated from the convex hull of the disjunction, disallowing the cuts to cut inside  $\Omega \cap \mathcal{D}_i$  for any  $i$ . Hence, even without an extension of the bilevel-free set, our method leads to cuts that can be stronger than the ones presented in Fischetti et al. (2018) or Gaar et al. (2023) in some cases. However, this does not hold in general as the following example shows.

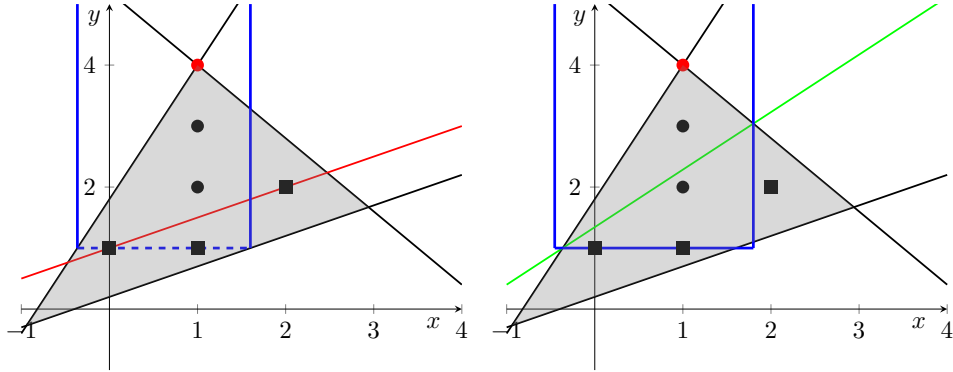


Figure 3.2: The shared constraint set of Problem (3.18) is depicted in gray. The point  $(1, 4)$  is the bilevel-infeasible solution to the HPR. The points  $(0, 1)$ ,  $(1, 1)$ , and  $(2, 2)$  are bilevel feasible. Our disjunctive cut (for  $\hat{y} = 1$ ; in red) is shown in the left figure. The intersection cut of Fischetti et al. (2018) and the disjunctive cut of Gaar et al. (2023) are computed using the extended bilevel-free set  $S^+(1)$ . Both cuts are the same as shown (in green) in the right figure.

**Example 3.5.2.** Consider the problem

$$\begin{aligned}
 \min_{x \in \mathbb{Z}, y} \quad & x - y \\
 \text{s.t.} \quad & 5x + y \leq 17, \quad x + 2y \leq 9 \\
 & y \in \arg \min_{\bar{y} \in \mathbb{Z}} \{ \bar{y} : -3x + \bar{y} \leq 1, \quad x - 2\bar{y} \leq -1 \}.
 \end{aligned} \tag{3.19}$$

Problem (3.19) is illustrated in Figure 3.3. The solution to the HPR is  $(x^0, y^0) = (1, 4)$  and the optimal follower's response for  $x^0 = 1$  is  $y^0 = 1$ . We get the bilevel-free set

$$S(1) = \{(x, y) : 0 \leq x \leq 1, \quad y > 1\}$$

that we use to compute our disjunctive cut. The resulting cut then reads  $y \leq 3$ . For the methods in Fischetti et al. (2018) and Gaar et al. (2023) the bilevel-free set is extended to

$$S^+(1) = \left\{ (x, y) : -\frac{1}{3} \leq x \leq 2, \quad y \geq 1 \right\}.$$

With that, the computed cutting planes are equal for both methods from the literature, i.e.,  $-5x + 4y \leq 4$ . In Figure 3.3 we see that no cut is dominating the others w.r.t. the continuous relaxation of the shared constraint set. However, in this example our disjunctive cut is weaker in the sense that it does not separate the integer-feasible point  $(1, 3)$  in contrast to the cut computed by the other two methods.

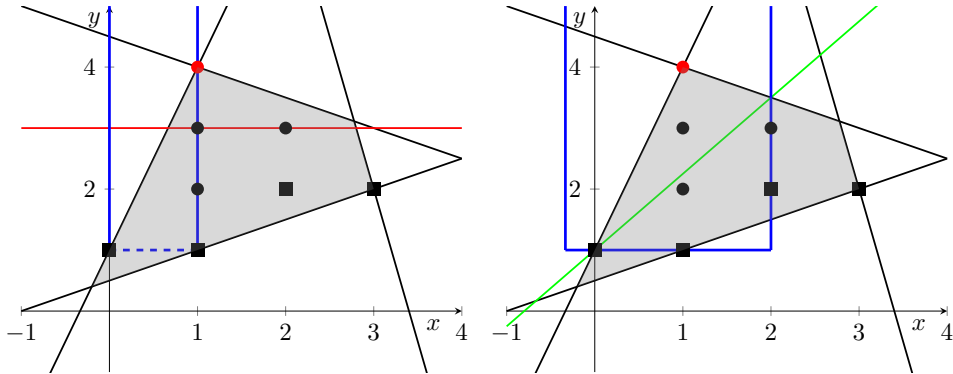


Figure 3.3: The shared constraint set of Problem (3.19) is depicted in gray. The point  $(1, 4)$  is the bilevel-infeasible solution to the HPR and the points  $(0, 1)$ ,  $(1, 1)$ ,  $(2, 2)$ , and  $(3, 2)$  are bilevel feasible. Our disjunctive cut (for  $\hat{y} = 1$ ; in red) is shown in the left figure. The intersection cut of Fischetti et al. (2018) and the disjunctive cut of Gaar et al. (2023) are computed using the extended bilevel-free set  $S^+(1)$ . Both cuts are the same as shown (in green) in the right figure.

Examples 3.5.1 and 3.5.2 show that our disjunctive cuts and the cuts derived from the methods presented in Fischetti et al. (2018) and Gaar et al. (2023) cannot be compared in general even if the same underlying bilevel-free set is used. Moreover, the cuts cannot be compared if one uses different bilevel-free sets for each method. This is because the choice of the bilevel-free set has a direct impact on the properties of the disjuncts and, hence, on the resulting cut.

### 3.5.2 Difficulties for the Mixed-Integer Setting

We now discuss difficulties arising when trying to generalize our method to the convex mixed-integer nonlinear bilevel setting. First, we focus on a general setting in which all variables may be fractional. Then, we consider a specific case where only the upper-level variables that do not appear in the lower level, i.e., the non-linking variables, are allowed to be fractional.

#### The General Mixed-Integer Setting

So far in this chapter, we have considered convex integer nonlinear bilevel problems as defined in (3.1). Currently, we cannot consider the general

mixed-integer variant of this problem, i.e., problems of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}, y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0, \quad x_i \in \mathbb{Z} \text{ for all } i \in \hat{I}_x, \\ & y \in \arg \min_{\bar{y} \in \mathbb{R}^{n_y}} \left\{ f(x, \bar{y}) : g(x, \bar{y}) \leq 0, \quad y_i \in \mathbb{Z} \text{ for all } i \in \hat{I}_y \right\}, \end{aligned} \quad (3.20)$$

where  $F, f : \mathbb{Q}^n \rightarrow \mathbb{Q}$ ,  $G : \mathbb{Q}^n \rightarrow \mathbb{Q}^m$ , as well as  $g : \mathbb{Q}^n \rightarrow \mathbb{Q}^l$  are continuous and jointly convex functions,  $\hat{I}_x \subset [n_x]$  and  $\hat{I}_y \subset [n_y]$ , and where Assumptions 2.1.1 and 2.1.2 hold. One difficulty when considering problems of the form (3.20) arises from the fact that integer no-good cuts cannot be derived in presence of fractional variables  $x$  or  $y$ ; see Section 2.6.4. If we are unable to derive a bilevel-free set as defined in Theorem 3.6 such that the bilevel-infeasible point that we want to separate lies inside the interior of this set, then we need to apply an integer no-good cut to at least one of the subproblems (3.11) or (3.12); see the discussion after Proposition 3.3.11. Example 3.5.3 illustrates this problem on an instance with fractional variables in the lower level. Note that one can easily construct a similar example in which only the upper level or both levels contain fractional variables.

**Example 3.5.3.** *Consider the problem*

$$\min_{x \in \{0,1\}, y} \quad -2x - y \quad \text{s.t.} \quad y \in \arg \min_{\bar{y} \in [0,0.5]} \{ \bar{y} : x + \bar{y} \geq 1 \}. \quad (3.21)$$

The bilevel-infeasible solution to the high-point relaxation is given by  $(\tilde{x}, \tilde{y}) = (1, 0.5)$ . For any  $\hat{y} \in \mathbb{Z}$ , a bilevel-free set is defined as

$$S(\hat{y}) := \{(x, y) : x + \hat{y} \geq 1, \quad y > \hat{y}\}.$$

To guarantee  $(1, 0.5) \in S(\hat{y})$ , we have to satisfy that  $1 + \hat{y} \geq 1$  and  $0.5 > \hat{y}$  holds, i.e.,  $\hat{y} = 0$  yields the only feasible bilevel-free set. However, the point  $(1, 0.5)$  lies on the boundary of  $S(0) := \{(x, y) : x \geq 1, y > 0\}$ . For iteration  $k$  in the cut generation, the resulting subproblems (3.11) and (3.12) read

$$\begin{aligned} \max_{x, y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & (x, y) \in \{(\bar{x}, \bar{y}) : \bar{x} \in \{0, 1\}, \bar{y} \in \mathbb{R} \cap [0, 0.5], \bar{x} + \bar{y} \geq 1\} \setminus \{(1, 0.5)\}, \\ & y \leq 0 \end{aligned}$$

and

$$\begin{aligned} \max_{x, y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & (x, y) \in \{(\bar{x}, \bar{y}) : \bar{x} \in \{0, 1\}, \bar{y} \in \mathbb{R} \cap [0, 0.5], \bar{x} + \bar{y} \geq 1\} \setminus \{(1, 0.5)\}, \\ & x \leq 1, \end{aligned}$$

which can be simplified to

$$\begin{aligned} \max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & x = 1, y = 0 \end{aligned}$$

and

$$\begin{aligned} \max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\ \text{s.t.} \quad & (x, y) \in \{(\bar{x}, \bar{y}) : \bar{x} \in \{0, 1\}, \bar{y} \in \mathbb{R} \cap [0, 0.5], \bar{x} + \bar{y} \geq 1\} \setminus \{(1, 0.5)\}. \end{aligned}$$

The latter problem has an open feasible set. Moreover, we cannot use an integer no-good cut to separate the point  $\{(1, 0.5)\}$  because it is not integer. Hence, we would not be able to solve Problem (3.21) using Algorithm 3.

As shown in Example 3.5.3, it is not possible for our method to tackle problems that allow fractional variables in general. This is because if we have a fractional bilevel-infeasible point that lies on the boundary of the bilevel-free set, we cannot use an integer no-good cut to separate it from the feasible regions of the subproblems (3.11) and (3.12). However, the bilevel-free set and, hence, also the disjuncts  $\mathcal{D}_i$  for  $i \in I_0$  are only constrained by the functions of the lower level, i.e., functions that depend on the upper-level linking variables and the lower-level variables. Therefore, we now consider a special case of Problem (3.20).

### Allowing Mixed-Integer Non-Linking Upper Level Variables

Consider Problem (3.20) with the specific condition that all linking variables and all lower-level variables are integer, i.e.,

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}, y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \leq 0, x_i \in \mathbb{Z} \text{ for all } i \in \hat{I}_x, \\ & y \in \arg \min_{\bar{y} \in \mathbb{Z}^{n_y}} \left\{ f(x_{\hat{I}_x}, \bar{y}) : g(x_{\hat{I}_x}, \bar{y}) \leq 0 \right\}. \end{aligned} \tag{3.22}$$

This is a rather strong assumption as none of the variables that appear in the lower-level functions are allowed to be fractional. It is possible to tackle problems of the form (3.22) using our approach but it would require to perform significant changes to our method that are out of scope of this thesis. These changes include adapting the Definition 3.3.2 of the disjunctive cuts, the scoop problem (3.16), (CGP), and the subproblems (3.11) and (3.12) to the mixed-integer setting. However, we briefly discuss the key ideas in the following.

Let  $(\tilde{x}, \tilde{y})$  be a bilevel-infeasible point that satisfies the integrality conditions of Problem (3.22). Then,  $(\tilde{x}_{\hat{I}_x}, \tilde{y})$  is integer and every point

in  $\mathcal{P} := \{(x, \tilde{y}) : x_{\hat{I}_x} = \tilde{x}_{\hat{I}_x}\}$  is bilevel infeasible. As mentioned in Section 3.5.2, there are cases in which we have to apply an integer no-good cut in the subproblems (3.11) and (3.12) to separate  $(\tilde{x}, \tilde{y})$  from the feasible regions, which is not possible if that point is not integer. However, if we consider problems of the form (3.22), we can use an integer no-good cut on  $(\tilde{x}_{\hat{I}_x}, \tilde{y})$  to separate every point in  $\mathcal{P}$  from the feasible regions of the subproblems (3.11) and (3.12). This allows every point in  $\mathcal{P}$  to be on the same side of the computed hyperplane as the point  $(\tilde{x}_{\hat{I}_x}, \tilde{y})$  and this can only be done because all those points are bilevel infeasible. Note that for the problem setting in (3.22), (CGP) becomes a semi-infinite problem over a nonconvex set, which adds a significant difficulty to our approach. One can still tackle this problem using tailored methods; see, e.g., Stein (2003) and the references therein. However, this is out of scope of this work. Moreover, to guarantee finite termination of the method like in Theorem 3.4.2, one needs to perform a refinement step as it is done in Fischetti et al. (2018); see also Section 3.4.3. For a point  $(\tilde{x}, \tilde{y})$ , the refinement step computes a bilevel-feasible point  $(\hat{x}, \hat{y})$  with  $\hat{x}_{\hat{I}_x} = \tilde{x}_{\hat{I}_x}$  that yields the best upper-level objective function value if any exists. With that, additionally to the disjunctive cut, one can add an integer no-good cut at  $\tilde{x}_{\hat{I}_x}$  to the node problem separating every point  $(x, y)$  with  $x_{\hat{I}_x} = \tilde{x}_{\hat{I}_x}$ . We will show in Chapter 4 of this thesis that the addition of such integer no-good cuts leads to finite termination of the branch-and-cut approach if there exist only finitely many vectors  $x_{\hat{I}_x}$  that lead to points that are feasible for the HPR. Note, that the latter holds for problems of the form (3.22) if Assumption 2.1.1 holds. In summary, it is possible to approach the slightly more general problem class introduced in (3.22) using the ideas presented in this work but that comes with further challenges, both theoretically and computationally.

## 3.6 Numerical Results

In this section, we present numerical results obtained by applying the proposed method to bilevel test instances from the literature. To the best of our knowledge, there is no publicly available solver that can handle general convex integer nonlinear bilevel problems for which the nonlinearities appear both in the objective function and in the constraints of the upper and the lower level. Consequently, to shed some light on the effectiveness of the developed disjunctive cuts, we compare the following three approaches:

**INGC:** a branch-and-cut algorithm in which only integer no-good cuts are used to cut off integer-feasible but bilevel-infeasible points;

**DC+FR:** a branch-and-cut algorithm based on the proposed disjunctive cuts and the separation procedure as described in Algorithm 3 using improving-solutions bilevel-free sets as discussed in Section 3.3.4;

**DC+Scoop:** a branch-and-cut algorithm based on disjunctive cuts and the separation procedure as described in Algorithm 3 using the improving-direction bilevel-free sets obtained by solving the scoop problem (3.16); see Section 3.3.4.

Note that the INGC method also solves problems of the form (3.1) or detects their infeasibility in a finite number of steps if Assumptions 2.1.1 and 2.1.2 hold. The proof is similar to the proof of Theorem 3.4.2 and based on the fact that we only have to separate a finite number of points. The code for each of the methods is publicly available at <https://github.com/Andreashorlaender/BnC-for-convex-integer-nonlinear-bilevel-problems>. In Section 3.6.1, we describe our computational setup and discuss some implementation details. Then, in Section 3.6.2, we describe the instance sets used for our experiments. Section 3.6.3 contains a discussion of the numerical results.

### 3.6.1 Hardware and Software Setup

We implemented Algorithm 3 in Python 3.10.16 using the branch-and-cut framework of CPLEX 22.1.1.0. The  $x$ -parameterized lower-level problem, the scoop problem (3.16), and the cut-generating problem (CGP) are also solved using CPLEX 22.1.1.0. However, the subproblems (3.11) and (3.12) are solved using Gurobi 12.0.0. The reason for the latter is that CPLEX does not support solving problems with nonlinear and nonconvex constraints. CPLEX would still work for problems for which the functions appearing in the lower-level problem (3.24) are linear in  $x$  and, hence, the resulting subproblems (3.11) and (3.12) are convex; see Lemma 3.3.8. However, since the implementation should also be general enough to be applied to other instances in the future as well, Gurobi is used for solving the decomposed cut-generating problems. Note that we cannot use Gurobi for the overall branch-and-cut framework because local cuts are currently not supported by Gurobi. Hence, we needed to use both solvers for our implementation. All cuts are implemented using the `add_local` function inside the `LazyConstraintCallback` environment of CPLEX. Moreover, we implemented the integer no-good cuts (INGCs) in the subproblems (3.11) and (3.12) with the help of a binary expansion of the  $x$  and  $y$  variables as discussed in Section 2.6.4.

We use depth-first search as our node selection strategy; see Section 2.5.3 for an overview. This is because we need to keep track on the precise definition of the feasible region  $\Omega_N$  in each node  $N$  of the branch-and-bound tree. This information is needed in Line 16 of Algorithm 3 for solving the subproblems (3.11) and (3.12). More precisely, we need to inherit disjunctive cuts (DCs) from the parent node. Both Python APIs of Gurobi and CPLEX do not provide the exact location of a node in the branch-and-bound tree, so that we have to track its location manually. Whenever we obtain an

integer-feasible but bilevel-infeasible node solution, we store the node by appending its variable bounds as well as the cutting plane derived for it to a list. With this at hand, we check if the variable bounds in the current node are tighter than the variable bounds of the last element in the list. If this is the case, we are in a successor node of the last node in the list and, hence, all local cuts stored in the list for the subproblems (3.11) and (3.12) are valid for the successor node as well. Otherwise, we remove the last node from the list and repeat the procedure. This procedure is repeated until we find an ancestor node in the list or until the list is empty, in which case we are in the root-node of the branch-and-bound tree. This strategy works for depth-first and breadth-first search but not for other, more sophisticated, node selection strategies. Our preliminary tests showed superior performance of depth-first search, which is why we focus on this setting. Moreover, we use the default branching strategy of CPLEX; see 2.5.4.

For problems of the form (3.1), the feasible region of every node in the branch-and-bound tree is convex but not necessarily a polyhedron. This also holds for our benchmark instances; see Section 3.6.2. Therefore, CPLEX uses a barrier method to solve the continuous relaxation at branching nodes. Hence, we may get a node solution  $(x^N, y^N)$  that is not an extreme point of the feasible region of that node although there exists an extreme point that yields the same objective function value as  $(x^N, y^N)$ . The latter holds because, w.l.o.g. we optimize a linear objective function over a convex set. In that case the computed hyperplane does not necessarily cut off the point  $(x^N, y^N)$ ; see Theorem 3.3.4. To detect this scenario, we compute  $\alpha^k x^N + \beta^k y^N - \tau$  every time we are in Line 24 of Algorithm 3. If this value is zero, we add an integer no-good cut rather than the computed valid inequality to separate the bilevel-infeasible node solution. However, this rare scenario occurs in only 8.8% of the “solvable” instances within our test set and in those specific instances it affects a median of 0.65% of the disjunctive cuts. Hence, the median over all solvable instance is zero for each instance set; see “INGCs” in Table 3.4.

All computations have been executed on the high performance cluster “Elwetritsch” at TU Kaiserslautern, which is part of the “Alliance of High Performance Computing Rheinland-Pfalz” (AHRP). We used a single Intel XEON SP 6126 core with 2.6 GHz and a maximum of 30 GB RAM. In our computational studies, we disabled heuristics and presolve to purely focus on the impact of the cuts. The integrality and feasibility tolerances in CPLEX and Gurobi are kept at their default values. The time limit is 2 hours.

### 3.6.2 Test Instances

We consider problems of the form

$$\begin{aligned} \min_{(x,y) \in \mathbb{Z}^n} \quad & c_x^\top x + c_y^\top y \\ \text{s.t.} \quad & Ax + By \leq a, \\ & y \in S(x), \end{aligned} \tag{3.23}$$

where  $S(x)$  is the set of optimal solutions to the  $x$ -parameterized lower-level problem

$$\min_{y \in \mathbb{Z}^{n_y}} \quad \frac{1}{2} y^\top Q y + d^\top y \tag{3.24a}$$

$$\text{s.t.} \quad (Cx + Dy)_1 + \frac{1}{2} y^\top P y \leq b_1, \tag{3.24b}$$

$$(Cx + Dy)_i \leq b_i, \quad i = 2, \dots, l, \tag{3.24c}$$

i.e., we have a quadratic lower-level objective as well as one quadratic lower-level constraint. Since the novelty of our approach is in handling nonlinearities in the lower-level problem, we keep the upper level linear. Note that one could also linearize the QP in the lower level using McCormick inequalities since all  $y$  variables are finitely bounded by Assumption 2.1.1. However, given that the purpose of the following numerical study is to shed some light on our method's capability to tackle nonlinearities in the lower-level problem, we keep the model as it is.

For our test set, we use a subset of the QBMKP instances used in Gaar et al. (2023). These are multidimensional knapsack problems derived from the SAC-94 library (Khuri et al. 1994) with 2 to 10 constraints and 10 to 105 items that have been translated into quadratic bilevel multiple knapsack problems (QBMKP); see Gaar et al. (2023) for further details. From those instances, we use the 100 binary and the 100 integer QBMKP instances with a single lower-level constraint. Furthermore, for each of those instances, we construct a new instance that contains the first  $\lfloor (m+l)/2 \rfloor$  constraints of the HPR in the upper level and we move the last  $\lceil (m+l)/2 \rceil$  constraints to the lower level. If an instance has only two constraints in total, we move both constraints in the lower level. We denote the latter instance set as QBMKP\_50/50. Since our method is capable of dealing with nonlinear but convex lower-level constraints, we additionally create instances with a convex-quadratic constraint contained in the lower level. Starting from a QBMKP instance, we convert the first linear constraint from the lower-level problem  $(Cx + Dy)_1 \leq b_1$  into a convex quadratic one  $(Cx + Dy)_1 + \frac{1}{2} y^\top P y \leq b_1$  as follows. A positive semi-definite matrix  $P = \tilde{P}^\top \tilde{P}$  is generated following the procedure described in Kleinert et al. (2021a) and Gaar et al. (2023). The entries of  $\tilde{P}$  are chosen uniformly at random from the interval  $[-\sqrt[4]{\sigma}, \sqrt[4]{\sigma}]$  with  $\sigma = \|b_1\|_\infty$ ; see the MATLAB

function of Kleinert and Schmidt (2021) which can be found at <https://github.com/m-schmidt-math-opt/qp-bilevel-matrix-generator>. The value of the right-hand side is set as  $b_1 \leftarrow b_1 + |b_1|$ .

For our experiments, we also use a subset of the MILP-MILP instances from Kleinert and Schmidt (2021). To fit in our setting, we apply integrality constraints on all  $x$  and  $y$  variables and randomly generate positive semi-definite matrices  $Q$  and  $P$  for the lower-level objective function (3.24a) and the lower-level constraint (3.24b) in the same way as we do it for the QBMKP instances. Note that we use  $\sigma = \|d\|_\infty$  to generate the entries of matrix  $Q$ . Again, we increase the right-hand side of the quadratic constraint by its absolute value. Since the vast majority of the modified instances from this collection can either not be solved within a time limit of 2 hours or are solved without using a single cut, we exclude all instance sets but the sets Denegre, XuWang, and XuLarge. Furthermore, we exclude 30 out of a total of 60 instances from the set XuLarge because the generation of the quadratic matrices exceeds the time limit of 2 hours. The unmodified versions of the instances are available in the BOBILib; see <https://bobilib.org> and Thürauf et al. (2024) for further details. The number of instances per instance set, as well as the sizes of the instances, are given in Table 3.1.

Table 3.1: Overview of the original test instances per instance class.

	Reference	Size	$n_x$		$n_y$		$m$		$l$	
			Min	Max	Min	Max	Min	Max	Min	Max
QBMKP	Gaar et al. (2023)	200	5	79	2	52	1	10	1	1
QBMKP_50/50	This work	200	5	79	2	52	1	10	2	5
Denegre	DeNegre and Ralphs (2009)	50	5	15	5	15	0	0	20	20
XuWang	Xu and Wang (2014)	100	10	460	10	460	4	184	4	184
XuLarge	Fischetti et al. (2017)	60	500	1000	500	1000	200	400	200	400

### 3.6.3 Discussion of the Results

Table 3.2: Overview of the modified test instances per subset that are presented in our numerical study.

	Solvable	Time limit	Trivial	$\Sigma$
QBMKP	62 (31.0%)	112 (56.0%)	26 (13.0%)	200
QBMKP_50/50	60 (30.0%)	113 (56.5%)	27 (13.5%)	200
Denegre	28 (56.0%)	16 (32.0%)	6 (12.0%)	50
XU	125 (96.2%)	0 (0.0%)	5 (3.8%)	130

In the following, we provide results of a computational comparison of the approaches INGC, DC+FR, and DC+Scoop. We divide the entire set of benchmark instances into four subsets; see Table 3.2. The subset XU

contains both the instances of `XuWang` and `XuLarge`. For the presentation of the numerical results, we exclude all instances that can be solved by all three methods in less than one second. We also exclude the instances that `DC+FR` and `DC+Scoop` solve without using a disjunctive cut and without pruning a node (as for those instances all methods are identical). We denote those instances as “trivial”. Conversely, we denote the remaining instances as “non-trivial”. Non-trivial instances are called “solvable” if at least one of the methods solves them within the time limit of 2 hours and they are labeled with “time limit” if none of the methods can solve them within the time limit. A classification of the instances per subset is given in Table 3.2.

The empirical cumulative distribution functions (ECDFs) w.r.t. idealized runtimes, node counts, and (relative) MIP gaps at termination of the three methods are given in Figures 3.4–3.7. They illustrate the proportion of instances within the test set that can be solved given a certain time or node count. To get the idealized runtime of the methods `DC+FR` and `DC+Scoop` for a given instance, we replace the runtime that it takes to solve all of the subproblems (3.11) and (3.12) successively (see Line 16 in Algorithm 3) by the maximum runtime that we need to solve one of the subproblems—hence mimicking as if we would solve all of the subproblems (3.11) and (3.12) in parallel; see Section 3.3.3.

The minimum, maximum, and median values for the idealized runtimes and for the node counts of the methods `INGC`, `DC+FR`, and `DC+Scoop` are given in Table 3.3 for each subset respectively. Similarly, Table 3.4 reports information w.r.t. the number of DCs, `INGCs`, and subproblems needed, respectively.

## QBMKP

As illustrated in Figure 3.4, which focuses on the `QBMKP` instances, both methods using disjunctive cuts outperform the standard approach `INGC` w.r.t. the running times and node counts. Moreover, `DC+FR` and `DC+Scoop` solve significantly more instances than `INGC`. For the unsolved instances of `QBMKP`, the MIP gaps are rather similar for the three approaches. It can be seen in Table 3.3 that the majority of instances in `QBMKP` is solved in the root-node using either `DC+FR` or `DC+Scoop`. Moreover, only very few disjunctive cuts are used in the median; see Table 3.4. Conversely, `INGC` uses a median of 59 023 branch-and-bound nodes and 29 378 integer no-good cuts to solve the instances. This shows that the disjunctive cuts are much stronger than the integer no-good cuts in practice. Hence, applying the former in the root-node of the branch-and-bound tree may already result in a solution to the problem, while for the latter further branching is required. Another explanation for the drastic difference in the node counts is that 29 out of the 174 non-trivial instances are proven to be infeasible by at least one of the methods. Here, `DC+FR` detects infeasibility in 26 of those instances already at the root node

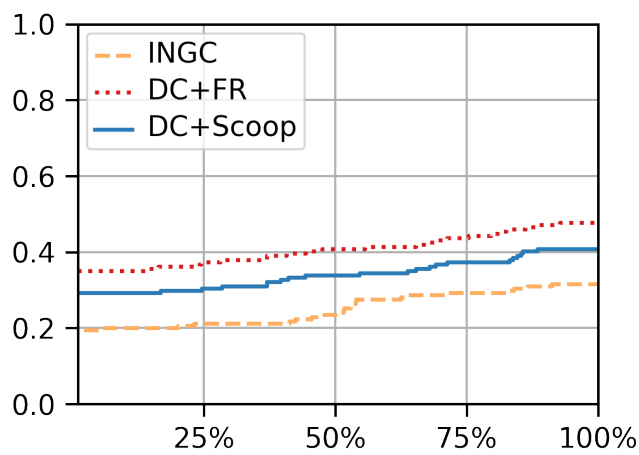
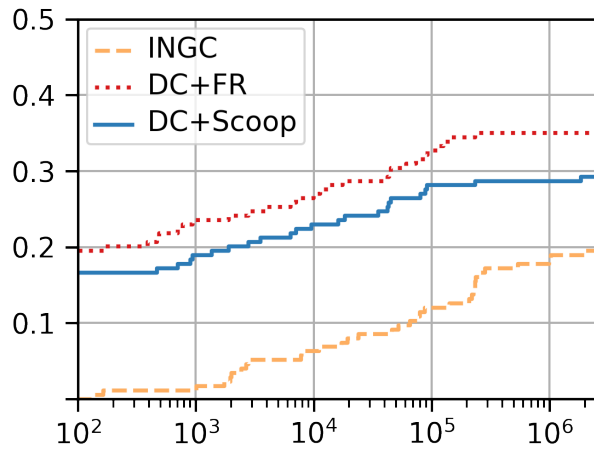
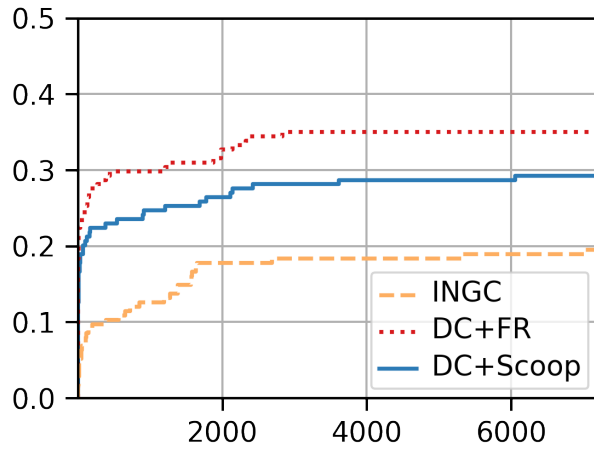


Figure 3.4: ECDFs for the “non-trivial” instances in the class QBMKP. Top: idealized runtimes. Middle: node counts. Bottom: MIP gaps.

(see Line 17 of Algorithm 3) and prunes it (see Line 18). This corresponds to roughly 43% of the non-trivial instances that DC+FR solves. Moreover, method DC+Scoop detects infeasibility of 20 instances in the root-node, which corresponds to 39% of the non-trivial instances that DC+Scoop solves. However, INGC is not capable of detecting bilevel infeasibility and, hence, removes (one by one) every point that is feasible for the high-point relaxation using an INGC. The latter approach almost always results in reaching the time limit of 2 hours.

We can see in Figure 3.4 and Table 3.3 that DC+FR performs better than DC+Scoop w.r.t. the runtimes and node counts. Moreover, the latter uses more cuts in the median. This implies that, for instances in QBMKP, using the follower’s response to derive a bilevel-free set leads to stronger disjunctive cuts compared to using the solution to the scoop problem (3.16). Moreover, DC+FR does not have the additional costs of solving Problem (3.16), rendering it the best of the three approaches to tackle the QBMKP instances.

Table 3.3: Detailed results per instance class for the solved instances that are “non-trivial”.

	Method	% solved	Idealized runtime in s			Nodes		
			Min	Max	Median	Min	Max	Median
QBMKP	INGC	19.5	0.3	7 027.3	291.6	131	1 984 795	59 023
	DC+FR	35.1	0.1	2 928.1	1.1	0	233 342	0
	DC+Scoop	29.3	0.1	6 257.8	6.6	0	1 838 541	0
QBMKP_50/50	INGC	27.2	0.2	4 355.0	214.8	8	839 823	3 041
	DC+FR	32.7	1.2	6 414.8	108.4	0	1 013 363	1 654
	DC+Scoop	31.8	2.2	6 854.3	67.7	0	891 682	1 016
Denegre	INGC	47.7	0.1	7 014.5	375.0	48	618 064	15 596
	DC+FR	54.5	2.5	6 461.7	200.9	31	174 597	1 965
	DC+Scoop	63.6	0.5	4 248.8	132.7	29	174 597	3 995
XU	INGC	100	0.1	5 668.7	223.3	24	39 423	3 367
	DC+FR	94.4	1.2	5 866.7	480.7	21	29 383	2 733
	DC+Scoop	97.6	1.0	6 969.2	486.8	21	42 300	2 825

Table 3.4: Total number of cuts and subproblems per instance class for the solved instances that are “non-trivial”.

	Method	DCs			INGCs			Subproblems (3.11) and (3.12)		
		Min	Max	Median	Min	Max	Median	Min	Max	Median
QBMKP	INGC	0	0	0	37	889875	29378	0	0	0
	DC+FR	0	301	1	0	5	0	2	131322	12
	DC+Scoop	0	3492	5	0	0	0	2	369348	202
QBMKP_50/50	INGC	0	0	0	2	159607	1206	0	0	0
	DC+FR	0	3837	57	0	26	0	72	256482	3462
	DC+Scoop	1	3554	27	0	2	0	117	152532	1518
Denegre	INGC	0	0	0	9	416396	12696	0	0	0
	DC+FR	3	11182	341	0	73	0	315	1167894	34125
	DC+Scoop	0	9132	155	0	59	0	126	1204392	37758
XU	INGC	0	0	0	1	11240	27	0	0	0
	DC+FR	0	1998	4	0	2	0	45	104040	3735
	DC+Scoop	0	234	3	0	0	0	45	95160	3675

### QBMKP\_50/50

Figure 3.5 reports the ECDFs for the QBMKP\_50/50 instances. We observe that the methods DC+FR and DC+Scoop perform better than INGC w.r.t. the runtimes and node counts; see Figure 3.5. However, the performance difference between the three methods for the instances in QBMKP\_50/50 is not as large as for the instances in QBMKP; compare Figure 3.4 and 3.5. Moreover, DC+FR and DC+Scoop perform equally well on the instances in QBMKP\_50/50. Table 3.4 shows that the median of the disjunctive cuts used in DC+Scoop is about half of the median for DC+FR, which indicates that the former method computes stronger cuts than the latter. However, this comes at the additional costs of solving the scoop problem (3.16). It is shown in Table 3.5 that solving the scoop problem indeed takes almost 30% of the overall median (idealized) runtime of the method DC+Scoop. This counterbalances the runtime that is saved by having less cuts overall, resulting in an equal performance for both methods.

We see in Table 3.3 that both methods DC+FR and DC+Scoop solve a similar percentage of instances compared to QBMKP while INGC solves significantly more. This may be because the lower level has more constraints, reducing the number of integer-feasible points that need to be separated to determine an optimal follower’s response for a given leader’s decision  $x$ . Therefore, we have an effect on the number of cuts needed, which benefits the integer no-good cuts more than the disjunctive cuts for the tested instances. It can be seen in Table 3.4 that the median of integer no-good cuts used in INGC for the instances in QBMKP\_50/50 is less than 5% of the value for the instances in QBMKP while solving more instances overall. Conversely, for the methods DC+FR and DC+Scoop we see in Table 3.4 that the medians of disjunctive cuts multiply compared to the medians for the set QBMKP. The same holds for the medians of the number of subproblems (3.11) and (3.12)

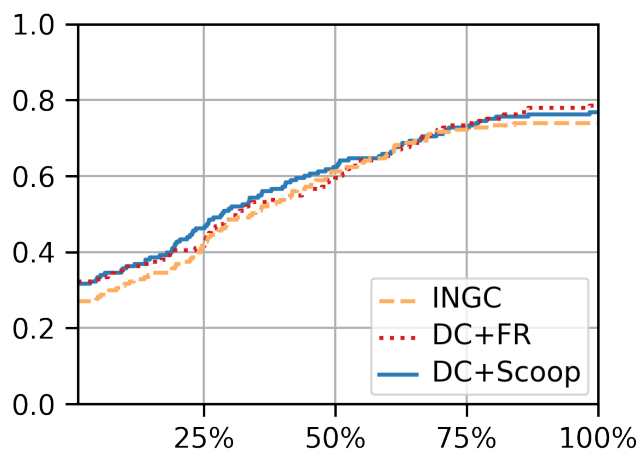
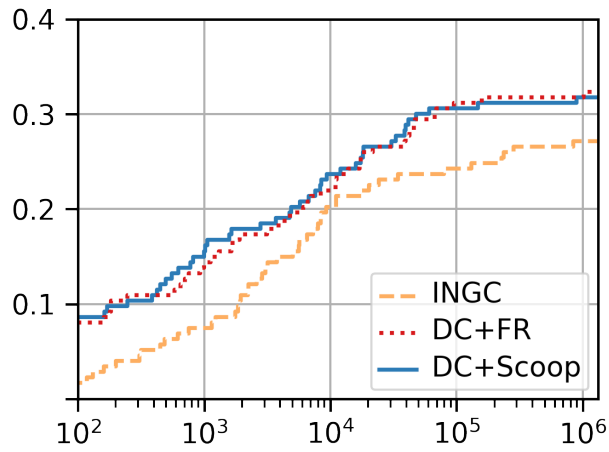
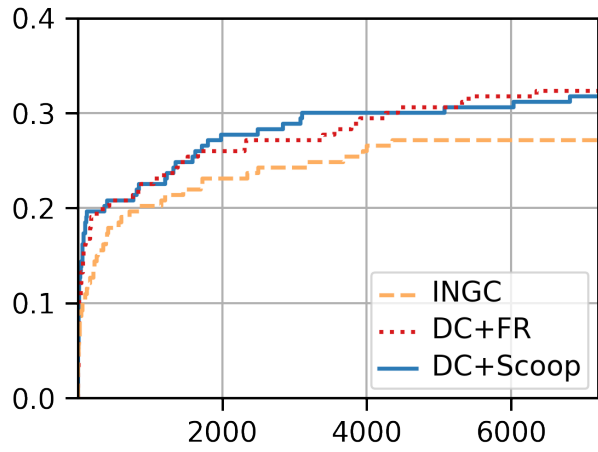


Figure 3.5: ECDFs for the “non-trivial” instances in the class QBMKP\_50/50. Top: idealized runtimes. Middle: node counts. Bottom: MIP gaps.

that are solved. One possible explanation for this is that 29 instances in QBMK are proven to be bilevel infeasible; see Section 3.6.3. For the majority of those instances, both methods DC+FR and DC+Scoop detect infeasibility in the root-node of the branch-and-bound tree using only very few cuts, i.e., very few subproblems are solved. These instances become bilevel feasible when we shift half of the constraints to the lower level, i.e., when they are in QBMKP\_50/50. Consequently, DC+FR and DC+Scoop both require a significantly higher number of cuts and subproblems to solve those problems in QBMKP\_50/50 compared to QBMK, which affects the respective medians.

For the instances in QBMK and QBMKP\_50/50 we conclude that having more constraints in the lower level is more advantageous for the INGC approach than for the methods using disjunctive cuts, i.e., DC+FR and DC+Scoop, because the number of required integer no-good cuts is drastically reduced. Nonetheless, both methods using disjunctive cuts perform better than INGC for both instance sets.

### Denegre and XU

Figure 3.6 illustrates the ECDFs for the instances in Denegre, where we see that DC+Scoop outperforms the other methods w.r.t. runtimes and node counts. As for the previous instance sets, INGC performs worst of all methods. The main difference to the results for QBMK and QBMKP\_50/50 is that DC+Scoop significantly performs better than DC+FR w.r.t. the runtimes and slightly better w.r.t. the node counts. We see in Table 3.4 that the median of disjunctive cuts used for DC+Scoop is less than half of the median for DC+FR. This is similar to the instances in QBMKP\_50/50. However, solving the scoop problem for the instances in Denegre is not as costly as for the other instance sets; see Table 3.5. Hence, the reduction of the number of disjunctive cuts by using improving direction bilevel-free sets outweighs the additional costs that it takes to solve Problem (3.16).

For the instances in XU, see Figure 3.7, INGC slightly outperforms both other methods w.r.t. the running times. However, the node counts for all three methods are almost equal and the methods DC+Scoop and DC+FR both need less cuts to solve the problems than INGC; see Table 3.4. This is because the instances in XU are less challenging compared to the instances in all other subsets. Indeed, all instances of the subset XU could be solved within the time limit (see Table 3.2) and each method solves over 94% of the non-trivial instances; see Table 3.3. As the instances pose little computational challenge, we omit the plot for the MIP gaps in Figure 3.7, as it would offer little additional insight.

The advantage of integer no-good cuts is their fast separation that cannot be outperformed by any other type of cutting planes, but their drawback is that they only cut off a single point, i.e., they are very shallow and, hence, typically many of them are required to prove optimality. This is the

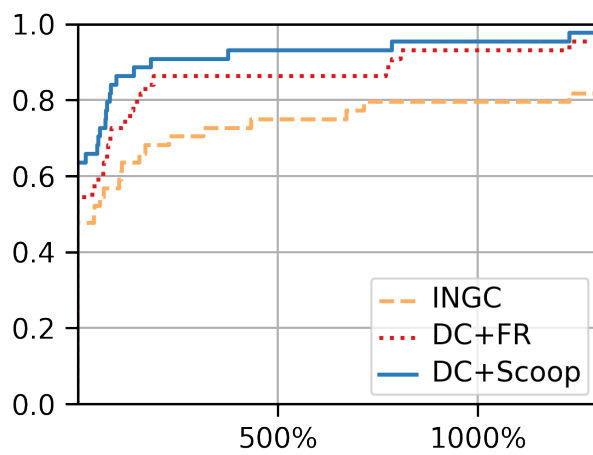
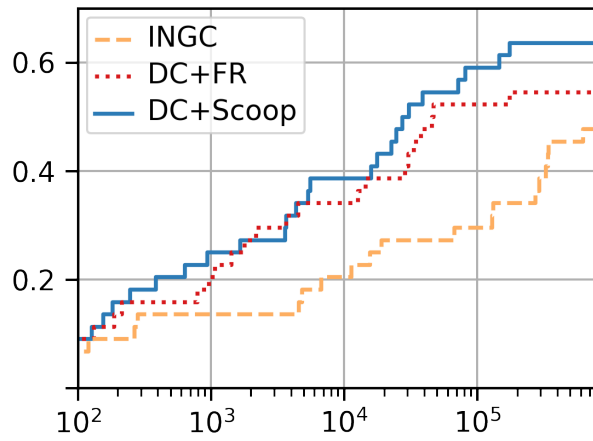
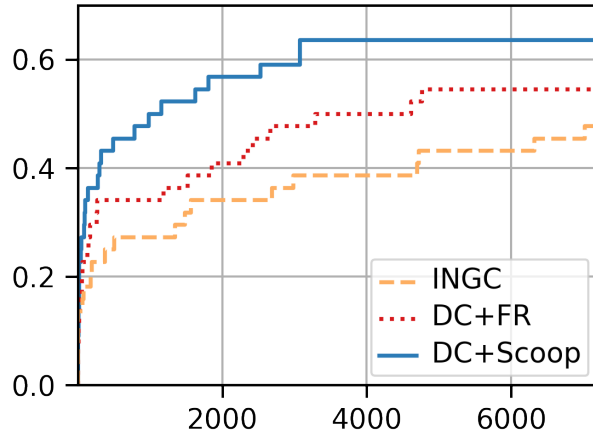


Figure 3.6: ECDFs for the “non-trivial” instances in the class Denegre. Top: idealized runtimes. Middle: node counts. Bottom: MIP gaps.

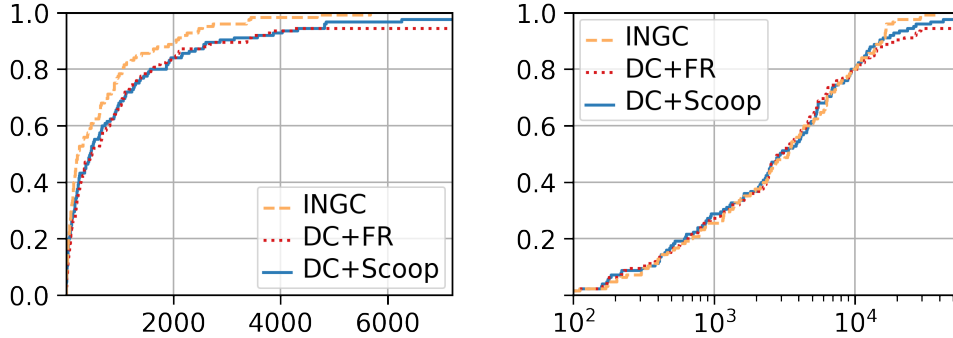


Figure 3.7: ECDFs for the “non-trivial” instances in the class XU. Left: idealized runtimes. Right: node counts.

reason why DC+FR and DC+Scoop outperform INGC in the previous instance sets. Regarding the disjunctive cuts, there is a trade-off between the high computational effort of the separation procedure, and the depth of the cuts, which does not pay off for instances like the ones in XU, where only very few INGCs are sufficient to prove optimality. Indeed, the median of the INGCs used by INGC is 27, see Table 3.4, which is by far the lowest value across all subsets. Similarly, it only takes four disjunctive cuts for DC+FR and three for DC+Scoop in the median to solve the instances in XU; see Table 3.4. However, the median of the running times of DC+Scoop and DC+FR is more than twice as large as the median of INGC. Additionally, these running times are much larger compared to the other median values for other subsets; see Table 3.3. This is due to the size of the instances in XU, which are much larger compared to the instances of the other subsets; see Table 3.1. Hence, although XU instances are easy to solve, they still need some time to be solved because of their size. This has an even larger effect on the methods DC+FR and DC+Scoop, because not only the  $x$ -parameterized lower-level problem (3.1c), but also (CGP), the subproblems (3.11) and (3.12), and, in the latter approach, the scoop problem (3.16), are affected by the size of the input instance. Moreover, due to the size of the instances in XU, the majority of disjunctive cuts are added at depths below level 30 in the branch-and-bound tree; see Table 3.7 for the statistics of DC+Scoop. This is in contrast to every other instance set and showcases that the difficulty of the instances in XU mainly comes from the size of the instances and the resulting depth of the branch-and-bound trees.

### Further Statistics for Algorithm 3

In this section, we focus on statistics for the individual steps in Algorithm 3. Table 3.5 shows the median values of the percentages of the major operations

in Algorithm 3 for the respective instance classes. Note that we consider idealized runtimes, i.e., we replace the runtime that it takes to solve all of the subproblems (3.11) and (3.12) successively by the maximum runtime that we need to solve one of the subproblems. We see that solving the scoop problems requires around 30% of the overall runtimes for the sets QBMKP, QBMKP\_50/50, and XU. Therefore, DC+Scoop could not outperform DC+FR in those instances although the former method needs less cuts in most cases. For the instances in Denegre, solving the scoop problems takes a much smaller fraction of the total runtime. Moreover, it greatly reduces the costs of solving the subproblems (3.11) and (3.12) leading to a better performance of DC+Scoop compared to DC+FR.

Table 3.5 indicates that for DC+Scoop, solving the subproblems (3.11) and (3.12) is not the most expensive operation in general. This may be surprising as we have to solve a series of INLPs. Since we consider idealized runtimes, the number of subproblems that are solved in each iteration of the adversarial approach has much less impact on this statistic than the overall number of iterations. We see in Table 3.6 that we generally only need very few iterations to compute a disjunctive cut. This holds for both methods DC+FR and DC+Scoop for all of the tested instances sets. This could be a reason why the runtimes to solve the subproblems (3.11) and (3.12) do not take a larger proportion of the total idealized runtimes.

We see in Table 3.6 that the percentage of nodes that are pruned in Line 18 of Algorithm 3 is almost the same for both methods DC+FR and DC+Scoop in all instance sets. However, there is a difference for the median value of the instance set Denegre that aligns with the overall better performance of DC+Scoop compared to DC+FR.

Table 3.7 shows the distribution of all disjunctive cuts across all instances in the respective instance set, categorized by the depth at which they are invoked in the branch-and-bound tree. It can be seen that for the instances sets QBMKP, QBMKP\_50/50, and Denegre the majority of disjunctive cuts are added early in the branch-and-bound tree, i.e., up to depth 20. Moreover, over 95% are invoked up to depth 40. This is in contrast to the instances in XU where about half of the cuts are added at depth 31 or later. One can conclude that the instances in XU need larger branch-and-bound trees to be solved compared to instances in the other sets. This is mainly due to the size of the instances, i.e., the number of variables. Nonetheless, the processing of every node, i.e., the application of Algorithm 3 has the highest percentage value of the total idealized runtime in the median across all instance sets; see Table 3.5. This is because the size of the instances affects the runtimes of all of the auxiliary problems that are solved in Algorithm 3.

Table 3.5: Median percentages of the idealized runtimes used to handle the subproblems that are used in Algorithm 3. For the subproblems (3.11) and (3.12), the idealized runtimes are used as discussed in Section 3.6.3.

		Median percentages of the idealized runtime spent for the				
		Lower level	Scoop problem	(CGP)	Subproblems (3.11) and (3.12)	Entire node processing
QBMKP	DC+FR	8.95 %	0.00 %	13.31 %	58.20 %	82.05 %
	DC+Scoop	4.88 %	30.88 %	8.40 %	23.16 %	86.18 %
QBMKP_50/50	DC+FR	39.02 %	0.00 %	11.84 %	35.26 %	89.28 %
	DC+Scoop	33.72 %	29.77 %	10.17 %	6.94 %	90.21 %
Denegre	DC+FR	9.93 %	0.00 %	7.81 %	68.08 %	88.16 %
	DC+Scoop	21.83 %	11.78 %	10.90 %	17.13 %	80.07 %
XU	DC+FR	53.90 %	0.00 %	2.78 %	31.87 %	90.89 %
	DC+Scoop	45.57 %	29.34 %	2.31 %	10.91 %	93.06 %

Table 3.6: Median values for performance indicators of Algorithm 3.

		Percentage of nodes pruned	Median of the average number of Problems (3.11) and (3.12) per cut	Median of the average number of iterations to compute one cut
QBMKP	DC+FR	27.90 %	2.0	1.1
	DC+Scoop	26.92 %	2.3	1.4
QBMKP_50/50	DC+FR	14.63 %	9.4	2.7
	DC+Scoop	13.85 %	8.6	2.2
Denegre	DC+FR	24.25 %	43.1	2.1
	DC+Scoop	29.40 %	27.9	1.3
XU	DC+FR	0.67 %	156.1	1.2
	DC+Scoop	0.73 %	139.2	1.1

Table 3.7: Distribution of disjunctive cuts across different depths of the branch-and-bound tree for method DC+Scoop. The table reports the total number of cuts applied at different depth ranges, summed over all instances that method DC+Scoop solves.

		Depth of the branch-and-bound tree						$\Sigma$
		0–10	11–20	21–30	31–40	41–50	> 50	
QBMKP	Number of cuts	409	4063	1038	1754	132	0	7396
	In percent	5.53 %	54.94 %	14.03 %	23.72 %	1.78 %	0.00 %	
QBMKP_50/50	Number of cuts	798	5332	2138	1076	331	1	9682
	In percent	8.24 %	55.07 %	22.08 %	11.11 %	3.42 %	0.01 %	
Denegre	Number of cuts	533	13879	12456	442	0	0	27310
	In percent	1.95 %	50.82 %	45.61 %	1.62 %	0.00 %	0.00 %	
XU	Number of cuts	205	274	209	144	146	416	1394
	In percent	14.71 %	19.66 %	14.99 %	10.33 %	10.47 %	29.84 %	

### Comparison to the Method of Gaar et al. (2023)

We also compare Algorithm 3 with the method presented in Gaar et al. (2023) that is designed for integer bilevel problems with second-order cone constraints in the upper level and a convex-quadratic objective function in the lower level. Note that the problem setting considered in this work is more general since we allow for nonlinear constraints in the lower level and since we do not restrict ourselves to second-order cone constraints in the upper level. Furthermore, we allow for an arbitrary jointly convex lower-level objective function; see Problem (3.1).

Hence, for the comparison, we use the same QBMKP instances as used in Gaar et al. (2023), i.e., 300 instances in total. Moreover, we exclude all instances that both methods can solve in less than one second and label them with “trivial” as in the previous tests; see Section 3.6.2. The remaining instances are labeled as “non-trivial”. We compare both methods DC+FR and DC+Scoop with the parameterization BC-Best (Gaar et al. 2023) as this configuration performs best on their respective numerical studies.

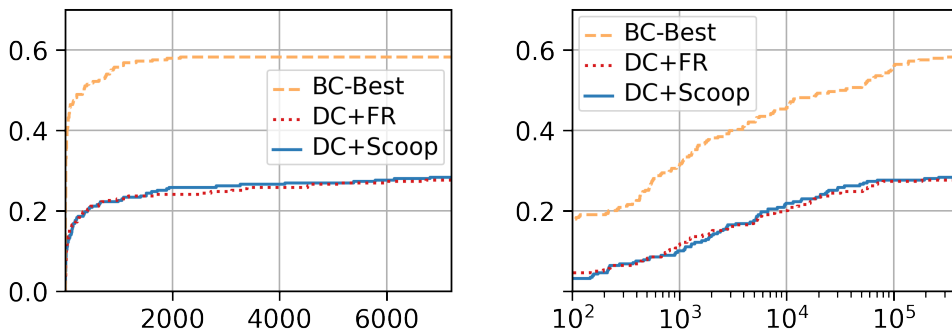


Figure 3.8: ECDFs for the QBMKP instances from Gaar et al. (2023) Left: runtimes for BC-Best and idealized runtimes for DC+FR and DC+Scoop. Right: node counts.

It can be seen in Figure 3.8 that the configuration BC-Best of the method proposed in Gaar et al. (2023) clearly outperforms our approach on the QBMKP instances both w.r.t. the running times and node counts. This can be expected because our cut generating procedure as described in Sections 3.3.2–3.3.3 is much more expensive than the approach proposed in Gaar et al. (2023). While we use an adversarial approach to generate a disjunctive cut, which involves solving a series of integer and possibly nonconvex subproblems, Gaar et al. (2023) solve a single second-order cone problem; see (CG-SOCP) in the respective paper. This showcases the additional effort that it takes to be able to tackle more general problems of the form (3.1) compared to the problem class discussed in Gaar et al. (2023). Note that we do not compare

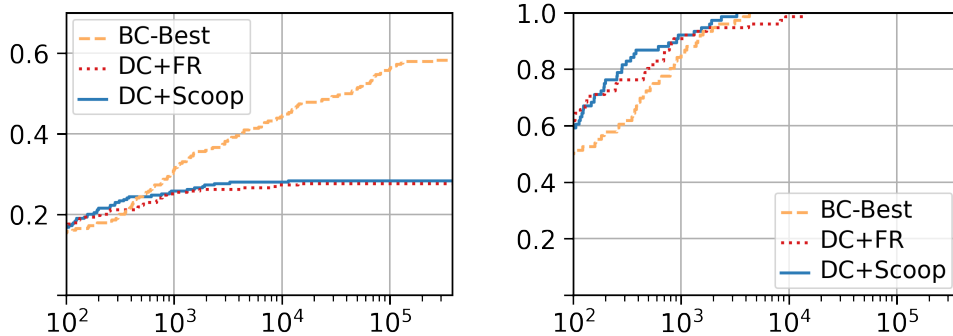


Figure 3.9: ECDFs for the number of disjunctive cuts used to solve the QBMKP instances from Gaar et al. (2023). Left: non-trivial instances. Right: non-trivial instances that both methods can solve.

the MIP gaps in Figure 3.9. This is because every instance that BC-Best could not solve hits the memory limit. Hence, we cannot make a fair comparison for the gaps.

We can see in Figure 3.9 that we solve instances using notably less cuts than BC-Best indicating that DC+FR and DC+Scoop provide stronger disjunctive cuts than BC-Best. If we only consider the non-trivial instances that all methods solve within the time limit of two hours, DC+Scoop performs slightly better than DC+FR and BC-Best w.r.t. the number of disjunctive cuts used; see Figure 3.9. A possible explanation is that our disjunctive cuts can cut into the disjuncts  $\mathcal{D}_i$  while the disjunctive cuts computed by BC-Best cannot; see Example 3.5.1. Therefore, we can get stronger cuts. However, compared to BC-Best, the computational effort that it takes to generate our cuts does not compensate for their (potential) strength. For the instances that both methods can solve, DC+Scoop still performs worse regarding the running times and node counts. Furthermore, BC-Best solves some of the tested instances using only a few disjunctive cuts while DC+Scoop does not solve them at all. Hence, DC+Scoop is not suitable for solving problems that are less general than (3.1) because their structure can be exploited better by a problem-tailored method leading to a more time-efficient cut generation.

### Further Algorithmic Techniques

We now discuss the numerical test for the ideas presented in Section 3.4.3. Therefore, we use DC+Scoop as our benchmark. We compare it with DC+Scoop+Ref and DC+Scoop+SNP, which additionally use the refinement procedure and the sibling node pruning, respectively. Moreover, we test DC+Scoop using the 1-norm in the objective function of the scoop problem (3.16) as discussed in Remark 3.3.12. The latter is denoted with

DC+Scoop\_1-Norm.

We also test different ways to handle the subproblems (3.11) and (3.12) that are discussed in Section 3.4.3. Therefore, we test an early termination of the solution process of the subproblems and we test a strategy where we do not solve all subproblems in every iteration of the cut generation; see Section 3.4.3. Both techniques have little to no effect on the running times and node counts for DC+Scoop. Hence, we do not discuss them in the following.

We do not test the initialization of  $\mathcal{Z}^0$  with bilevel-feasible points as discussed in Section 3.4.3 for two reasons. First, this technique cannot straightforwardly be implemented because we require an exact description of the set  $\Omega_N$  for every node  $N$ , which comes with additional effort; see Section 3.6.1. Second, we have seen in Table 3.6 that if we initialize  $\mathcal{Z}^0 = \emptyset$ , we generally only need very few iterations to compute a disjunctive cut. Hence, it cannot be expected that initializing  $\mathcal{Z}^0$  with bilevel-feasible points leads to a significant increase of the performance of our method.

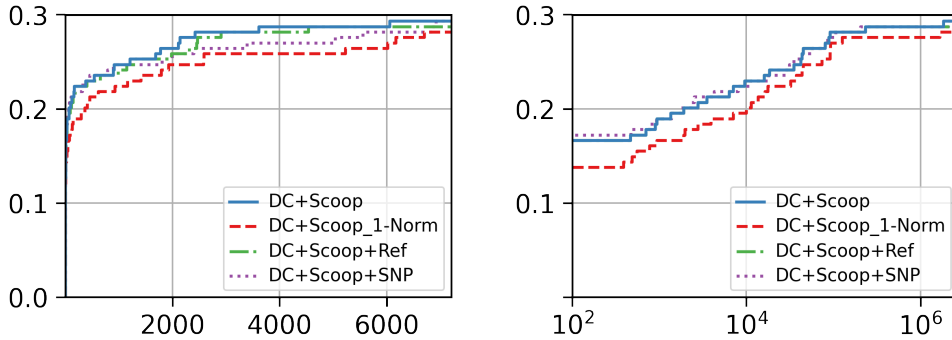


Figure 3.10: ECDFs for the non-trivial instances in QKP using further algorithmic techniques. Left: idealized runtimes. Right: node counts.

We can see in Figures 3.10–3.13 that none of the tested techniques of Section 3.4.3 leads to an improvement of our method DC+Scoop. Using the refinement procedure is only valuable for us if the lower level has multiple globally optimal solutions for different  $x$ -parameterizations. For our test instances, however, this scenario rarely occurs. The performance of DC+Scoop on the instances in QBMKP and QBMKP\_50/50 w.r.t. runtimes and node counts is unaffected by the use of the refinement procedure; see Figures 3.10 and 3.11. For the instances in Denegre and XU, we see a small and a significant deterioration of the running times, respectively, if we add the refinement step; see Figures 3.12 and 3.13. For the latter, the performance decrease is due to the size of the instances that affects the size of the restricted HPR that is solved in the refinement step.

It can be seen in Figures 3.10–3.13 that the sibling node pruning does

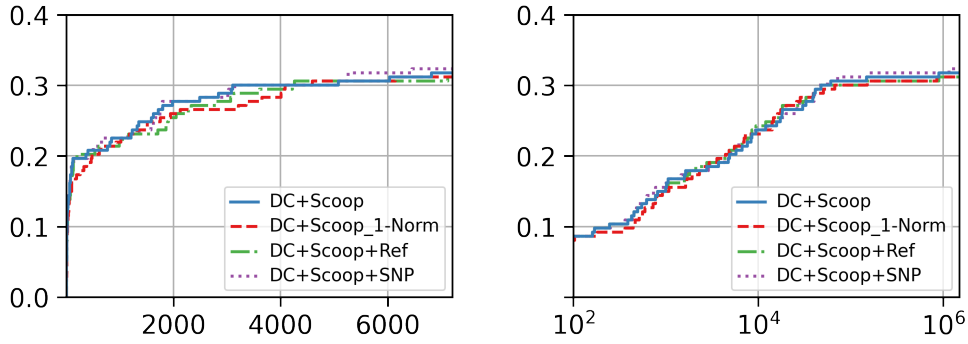


Figure 3.11: ECDFs for the non-trivial instances in QKP\_50/50 using further algorithmic techniques. Left: idealized runtimes. Right: node counts.

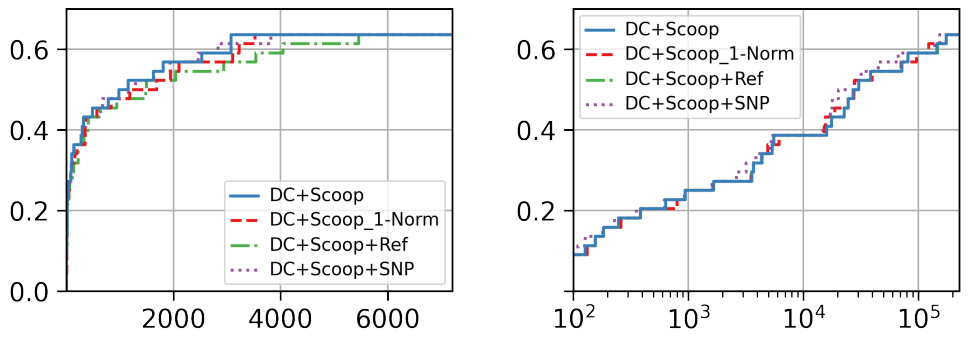


Figure 3.12: ECDFs for the non-trivial instances in Denegre using further algorithmic techniques. Left: idealized runtimes. Right: node counts.

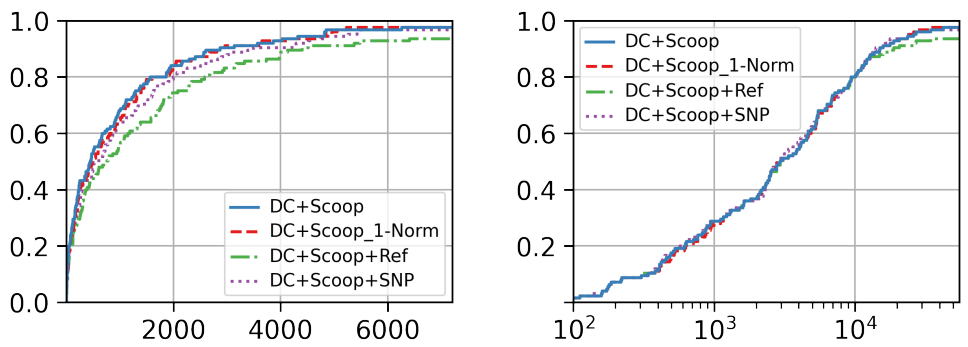


Figure 3.13: ECDFs for the non-trivial instances in XU using further algorithmic techniques. Left: idealized runtimes. Right: node counts.

not lead to a notable decrease of the number of branch-and-bound nodes in general. Hence, the runtimes of **DC+Scoop** increase for the vast majority of instances if we add this technique. Note that this increase is rather small, even for the instances in **XU**. For the latter, this is because less than one percent of the nodes gets pruned in the median; see Table 3.6. Therefore, we hardly ever use the sibling node pruning when we solve those instances; see Section 3.4.3.

Finally, Figures 3.12 and 3.13 show that using the 1-norm in the objective function of the scoop problem (3.16) for the method **DC+Scoop** has little to no impact on the overall performance of the method for the instances in **Denegre** and **XU**. However, we see a performance degradation for the instances in **QBMKP** and **QBMKP\_50/50**; see Figures 3.10 and 3.11. For the instances in **QBMKP**, both the runtimes and the node counts of **DC+Scoop\_1-Norm** are significantly higher compared to **DC+Scoop**. A possible explanation for this is that the instances in **QBMKP** only have a single lower-level constraint; see Section 3.6.2. Therefore, the scoop problem (3.16) contains just two slack variables, i.e.,  $s_0$  and  $s_1$ . Maximizing the 1-norm, i.e., the sum of the two variables, often results in one variable taking a large value while the other stays near zero, as this typically yields a higher objective function value compared to maximizing a lower bound  $t$ , as done in **DC+Scoop**. Consequently, bilevel-free sets that are computed in **DC+Scoop\_1-Norm** contain node solutions close to their boundary, which potentially leads to the generation of weak cuts. In contrast, instances with more lower-level constraints introduce more slack variables  $s_i$  in the scoop problem (3.16), resulting in the computation of bilevel-free sets that are similar for the two approaches **DC+Scoop** and **DC+Scoop\_1-Norm**.

## Summary

Our empirical study shows that the method using disjunctive cuts outperforms the branch-and-cut algorithm that uses only INGCs. This is true for both variants **DC+FR** and **DC+Scoop** in terms of node counts, running times, and the number of cuts used. However, there are instances for which the standard INGC-based approach performs better than our approach, e.g., the instances in **XU**. These instances have in common that they can be solved using only few integer no-good cuts and, hence, they are too easy to justify the high computational effort required for separating disjunctive cuts. For the instances that are more challenging to solve, the number of integer no-good cuts outnumbers the number of disjunctive cuts, clearly rendering our approach the method of choice. Finally, the addition of the algorithmic techniques discussed in Section 3.4.3 does not improve our method.

Method **DC+FR** outperforms **DC+Scoop** for the modified **QBMKP** instances. The opposite holds for the instances in **Denegre**. Moreover, both methods perform equally well for the sets **QBMKP\_50/50** and **XU**. Hence,

none of the two approaches has a clear advantage over the other. Both methods DC+FR and DC+Scoop are outperformed by the method proposed in Gaar et al. (2023) on the QBMKP instances that are used in the same paper. This was to be expected, since the latter is the state-of-the-art solver tailored for the tested problem class while our method is designed to solve more general problems of the form (3.1). However, for some instances, we are able to create deeper cuts compared to BC-Best but that does not compensate for the higher computational effort needed to compute the cuts.

In our numerical experiments, we focus on convex integer nonlinear bilevel problems where the only nonlinearities appear in the lower-level objective function and in a single lower-level constraint; see Section 3.6.2. After this modification, only 38.4% of the non-trivial instances in QBMKP, QBMKP\_50/50, and Denegre are solvable within the time limit; see Table 3.2. It can be expected that this proportion would decrease further if additional nonlinearities were introduced into the problem formulation. This is particularly true when considering a nonconvex lower-level objective function  $f$ ; see Remark 3.4.3.

## Chapter 4

# Mixed-Integer Bilevel Optimization with Nonconvex Quadratic Lower-Level Problems: Complexity and a Solution Method

This chapter contains the theoretical and numerical results from the Article [AH2]. It is structured as follows. Section 4.1 positions the content of this chapter within the overall framework of the thesis. In Section 4.2, we state the problem under consideration for which we prove  $\Sigma_2^P$ -hardness in Section 4.3. Our solution approach is discussed in detail in Section 4.4, and we present the numerical results in Section 4.5.

### 4.1 Motivation

In the previous chapter, we demonstrated that convex integer nonlinear bilevel problems can be solved using tailored disjunctive cuts within a branch-and-cut framework. This method, however, critically depends on the convexity of the node problems, which is why we focused on a setting in which all functions  $F$ ,  $G$ ,  $f$ , and  $g$  are jointly convex. We have seen in Section 3.5.2 that a generalization of the proposed method to solve mixed-integer bilevel problems can only be done for specific problem settings and requires additional theoretical and computational effort.

As a next step, we shift our focus from convex to nonconvex lower-level problems. Hence, in this chapter, we study mixed-integer bilevel problems

with integer linking variables and a purely continuous but nonconvex lower-level problem. In particular, we consider the case of a quadratic lower-level problem in which the lower level has a polyhedral feasible set but a nonconvex quadratic objective function. We study the computational complexity of the considered problem class and design an algorithm to solve these problems. Their nonconvexity is the main reason for the hardness of the problem, and we thus focus on this aspect both in terms of computational complexity as well as with the design of our solution method.

This chapter offers three main contributions:

- (i) We prove that the problems under consideration are  $\Sigma_2^P$ -hard in general, by a restriction to a special variant of the bilevel knapsack problem studied by Caprara et al. (2014); see Theorem 4.3.7.
- (ii) We derive an iterative procedure that successively tightens lower and upper bounds. The lower-bounding step consists in a KKT-based single-level relaxation of the problem, which is only a relaxation and not a reformulation since the KKT conditions are only necessary and not sufficient in general. Finiteness and correctness of the method is further ensured by using integer no-good cuts and optionally, a simple optimality cut; see Theorem 4.4.8.
- (iii) We present a numerical study showcasing the applicability of our approach; see Section 4.5. To the best of our knowledge, this is the first numerical study for this problem class using an extensive test set that also includes larger-sized instances—compared to other instances used in the literature that often have not more than 5 to 10 variables.

Very closely related to our approach is the method developed by Mitsos et al. (2008), where a purely continuous bilevel problem is considered which may contain nonconvex nonlinearities. The approach has been generalized later to the mixed-integer setting by Mitsos (2010). The key idea presented by Mitsos et al. (2008) is a lower-bounding procedure using an iterative tightening of a relaxation of an optimal-value function constraint, which leads to a finitely terminating algorithm when included in a Blankenship–Falk like algorithm; see Blankenship and Falk (1976). This approach can be extended by the KKT conditions of the lower level (which are only necessary and not sufficient in the nonconvex setting), which is also what we do in this chapter. However, their strategy of guaranteeing finite termination is completely different to ours. Moreover, our approach does not need comparably strong assumptions as Mitsos et al. (2008) do; see Assumption 3 in the cited paper, which excludes instances that have  $x$ -dependent equality constraints in the lower level; see Page 477 in Mitsos et al. (2008) for a brief discussion of their assumptions. However, there is a later follow-up paper by Mitsos and some other co-authors (Djelassi et al. 2019) in which they present an extension of

the former methods with which it is then also possible to tackle  $x$ -dependent lower-level equality constraints. Still, the size of the instances considered in the numerical results is as small as reported above. Let us mention that Mitsos and Barton (2006) showed that one cannot work with convex relaxations of the lower-level problem, such as the  $\alpha$ BB underestimator introduced in Androulakis et al. (1995).

## 4.2 Problem Statement

We study mixed-integer quadratic bilevel problems of the form

$$\begin{aligned}
\min_{x,y} \quad & F(x,y) = \frac{1}{2}x^\top H_x x + c_x^\top x + \frac{1}{2}y^\top H_y y + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \\
& x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \quad \text{for all } i \in I \subseteq [n_x], \\
& x_i \in \mathbb{R} \quad \text{for all } i \in [n_x] \setminus I, \\
& y \in \arg \min_{\bar{y}} \left\{ f(\bar{y}) = \frac{1}{2}\bar{y}^\top Q \bar{y} + d^\top \bar{y} : Cx_I + D\bar{y} \geq b, \bar{y} \in \mathbb{R}^{n_y} \right\},
\end{aligned} \tag{4.1}$$

where  $H_x \in \mathbb{Q}^{n_x \times n_x}$  as well as  $H_y \in \mathbb{Q}^{n_y \times n_y}$  are symmetric and positive semidefinite matrices and  $Q \in \mathbb{Q}^{n_y \times n_y}$  is symmetric but possibly indefinite. Furthermore, we have vectors  $c_x \in \mathbb{Q}^{n_x}$ ,  $c_y, d \in \mathbb{Q}^{n_y}$ , matrices  $A \in \mathbb{Q}^{m \times n_x}$ ,  $B \in \mathbb{Q}^{m \times n_y}$ ,  $C \in \mathbb{Q}^{l \times |I|}$ ,  $D \in \mathbb{Q}^{l \times n_y}$ , finite bounds  $x_i^- \leq x_i^+$  for the integer variables  $x_i$ ,  $i \in I$ , as well as right-hand side vectors  $a \in \mathbb{Q}^m$  and  $b \in \mathbb{Q}^l$ . The variables  $x$  contain the integer ( $x_I$ ) and continuous ( $x_{[n_x] \setminus I}$ ) upper-level variables and  $y$  denotes the (continuous) lower-level variables. In this setup, the upper-level problem is a convex mixed-integer quadratic problem (MIQP) and for fixed integer linking variables  $x_I$ , the lower level is a possibly nonconvex quadratic problem (QP). This class of problems covers applications in which the lower-level problem is concave, e.g., due to economies of scale. Moreover, to give another example, it also allows to address clique-related problems in the lower-level problem via the Motzkin–Straus approach (Motzkin and Straus 1965) and its regularizations (Bomze 1997; Hungerford and Rinaldi 2019).

In what follows, we make Assumption 2.1.1, i.e., we assume boundedness of the HPR. Note that Assumption 2.1.3, i.e., boundedness of the linking variables, is satisfied by the problem setting in (4.1). If some integer upper-level variables in Problem (4.1) do not appear in the lower-level problem, the corresponding columns in the matrix  $C$  would be set to zero. As discussed in Section 2.1, the HPR of Problem (4.1) is given by

$$\min\{F(x,y) : (x,y) \in \tilde{\Omega}\},$$

with

$$\tilde{\Omega} := \{(x, y) : Ax + By \geq a, Cx_I + Dy \geq b, x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \text{ for all } i \in I\}.$$

### 4.3 $\Sigma_2^P$ -Hardness

In this section, we show that the bilevel problem (4.1) is  $\Sigma_2^P$ -hard; see Section 2.2 for preliminaries on the complexity of bilevel problems. For some of the proofs in this section, we will make use of the following lemma.

**Lemma 4.3.1.** *Let  $\alpha \in [0, 1]$  and  $y \in [0, \frac{1}{2}]$ . Then, it holds*

$$\alpha y - \alpha^2 y^2 \leq y - y^2.$$

*Proof.* It holds

$$\begin{aligned} y \leq \frac{1}{2} \leq \frac{1}{1+\alpha} &\implies y(1-\alpha^2) \leq 1-\alpha \\ \implies \alpha - \alpha^2 y \leq 1-y &\implies \alpha y - \alpha^2 y^2 \leq y - y^2. \quad \square \end{aligned}$$

In what follows, we consider the problem

$$\begin{aligned} \min_{x \in \mathbb{Z}, y} & -c_x x - c_y^\top y \\ \text{s.t.} & x^- \leq x \leq x^+, \\ & y \in \arg \min_{\bar{y}} \left\{ -d^\top \bar{y} + M \left( \sum_{i=1}^{n_y} \bar{y}_i - \bar{y}_i^2 \right) : -d^\top \bar{y} \geq -x, \bar{y} \in [0, 1]^{n_y} \right\}, \end{aligned} \quad (4.2)$$

where  $0 \neq d \in \mathbb{Z}_+^{n_y}$ ,  $c_y \in \mathbb{Z}_+^{n_y}$ ,  $c_x, x^-, x^+ \in \mathbb{Z}_+$ , and  $M > 0$ . Problem (4.2) is a special case of Problem (4.1). The quadratic term  $M(\sum_{i=1}^{n_y} \bar{y}_i - \bar{y}_i^2)$  in the lower-level objective function can be restated as  $M(\bar{y}^\top Q \bar{y} + e^\top \bar{y})$ , where  $e$  is the vector of all ones and  $Q = \text{diag}(-e)$ . It is strictly concave, hence nonconvex, and corresponds to a penalty term that is positive if  $\bar{y}$  is fractional. Obviously, Problem (4.2) is equivalent to

$$\begin{aligned} \max_{x \in \mathbb{Z}, y} & c_x x + c_y^\top y \\ \text{s.t.} & x^- \leq x \leq x^+, \\ & y \in \arg \max_{\bar{y}} \left\{ d^\top \bar{y} - M \left( \sum_{i=1}^{n_y} \bar{y}_i - \bar{y}_i^2 \right) : d^\top \bar{y} \leq x, \bar{y} \in [0, 1]^{n_y} \right\}, \end{aligned} \quad (4.3)$$

with the same range specification for the parameters  $d, c_x, c_y, x^\pm, M$ . We will show that Problem (4.3) covers a special variant of a bilevel knapsack problem for which  $\Sigma_2^P$ -hardness is shown in the literature. From this,  $\Sigma_2^P$ -hardness of the problem class considered in this work follows immediately.

First we show that for every  $M > 0$ , a globally optimal solution  $\hat{y}$  to the  $x$ -parameterized lower level of Problem (4.3) has at most one fractional component.

**Lemma 4.3.2.** *Let  $x$  be feasible for the upper level of Problem (4.3) and let  $\hat{y}$  be a globally optimal solution to the  $x$ -parameterized lower level. Assume that there exists an index  $j$  such that  $\hat{y}_j \in (0, 1)$ . Then, it holds  $\hat{y}_i \in \{0, 1\}$  for all  $i \neq j$ .*

*Proof.* Assume that there exists an index  $k \neq j$  such that  $\hat{y}_k \in (0, 1)$ . It holds

$$\hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2 > 0$$

and, since  $\hat{y}$  is globally optimal for the lower level,  $d_j > 0$  and  $d_k > 0$  holds. We define

$$\begin{aligned} \zeta &:= \min \left\{ \hat{y}_j + \frac{d_k}{d_j} \hat{y}_k, 1 \right\}, & \xi &:= \max \left\{ 0, \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j - \frac{d_j}{d_k} \right\}, \\ \rho &:= \min \left\{ \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j, 1 \right\}, & \theta &:= \max \left\{ 0, \hat{y}_j + \frac{d_k}{d_j} \hat{y}_k - \frac{d_k}{d_j} \right\}. \end{aligned}$$

Note that  $\zeta$  and  $\xi$  cannot both be fractional simultaneously. Also the simultaneous case

$$\hat{y}_j + \frac{d_k}{d_j} \hat{y}_k > 1 \quad \text{and} \quad \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j < \frac{d_j}{d_k},$$

which would imply  $(\zeta, \xi) = (1, 0)$ , is impossible. The same holds for  $\theta$  and  $\rho$ . Therefore, it holds

$$d_j \zeta + d_k \xi = d_j \theta + d_k \rho = d_j \hat{y}_j + d_k \hat{y}_k,$$

i.e., we can replace  $\hat{y}_j$  and  $\hat{y}_k$  either with  $\zeta$  and  $\xi$  or with  $\theta$  and  $\rho$  and obtain the same values in the linear part of the lower-level objective function value. We now show that either

$$\zeta - \zeta^2 + \xi - \xi^2 < \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2$$

or

$$\theta - \theta^2 + \rho - \rho^2 < \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2$$

holds. This means that we can always choose either  $\zeta$  and  $\xi$  or  $\theta$  and  $\rho$  so that the penalization of the fractionality in the lower-level objective function gets smaller than the one for  $\hat{y}_j$  and  $\hat{y}_k$ . To this end, we distinguish the four following cases.

**Case 1:**  $\hat{y}_j \geq \frac{1}{2}$  and  $\hat{y}_k \leq \frac{1}{2}$ : It holds  $\frac{1}{2} \leq \hat{y}_j < \zeta$  and  $\xi < \hat{y}_k \leq \frac{1}{2}$  and, hence,

$$\zeta - \zeta^2 + \xi - \xi^2 < \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2.$$

**Case 2:**  $\hat{y}_j \leq \frac{1}{2}$  and  $\hat{y}_k \geq \frac{1}{2}$ : We have  $\theta < \hat{y}_j \leq \frac{1}{2}$  and  $\frac{1}{2} \leq \hat{y}_k < \rho$  and, hence,

$$\theta - \theta^2 + \rho - \rho^2 < \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2.$$

**Case 3:**  $\hat{y}_j \leq \frac{1}{2}$  and  $\hat{y}_k \leq \frac{1}{2}$ : In case  $d_k \leq d_j$ , it holds  $\zeta = \hat{y}_j + \frac{d_k}{d_j}\hat{y}_k$ ,  $\xi = 0$ , and, hence,

$$\begin{aligned}
\zeta - \zeta^2 + \xi - \xi^2 &= \zeta - \zeta^2 \\
&= \hat{y}_j + \frac{d_k}{d_j}\hat{y}_k - \left(\hat{y}_j + \frac{d_k}{d_j}\hat{y}_k\right)^2 \\
&= \hat{y}_j - \hat{y}_j^2 + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k^2}{d_j^2}\hat{y}_k^2 - 2\hat{y}_j \frac{d_k}{d_j}\hat{y}_k \\
&= \hat{y}_j - \hat{y}_j^2 + (1 - 2\hat{y}_j)\frac{d_k}{d_j}\hat{y}_k - \frac{d_k^2}{d_j^2}\hat{y}_k^2 \\
&< \hat{y}_j - \hat{y}_j^2 + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k^2}{d_j^2}\hat{y}_k^2 \\
&\leq \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2
\end{aligned}$$

due to Lemma 4.3.1 with  $\alpha = d_k/d_j$  and  $y = \hat{y}_k$ . In the opposite case  $d_k > d_j$ , we have  $\rho = \hat{y}_k + \frac{d_j}{d_k}\hat{y}_j$ ,  $\theta = 0$ , and, hence,

$$\begin{aligned}
\rho - \rho^2 + \theta - \theta^2 &= \rho - \rho^2 \\
&= \hat{y}_k + \frac{d_j}{d_k}\hat{y}_j - \left(\hat{y}_k + \frac{d_j}{d_k}\hat{y}_j\right)^2 \\
&= \hat{y}_k - \hat{y}_k^2 + \frac{d_j}{d_k}\hat{y}_j - \frac{d_j^2}{d_k^2}\hat{y}_j^2 - 2\hat{y}_k \frac{d_j}{d_k}\hat{y}_j \\
&= \hat{y}_k - \hat{y}_k^2 + (1 - 2\hat{y}_k)\frac{d_j}{d_k}\hat{y}_j - \frac{d_j^2}{d_k^2}\hat{y}_j^2 \\
&< \hat{y}_k - \hat{y}_k^2 + \frac{d_j}{d_k}\hat{y}_j - \frac{d_j^2}{d_k^2}\hat{y}_j^2 \\
&\leq \hat{y}_k - \hat{y}_k^2 + \hat{y}_j - \hat{y}_j^2,
\end{aligned}$$

also due to Lemma 4.3.1 with  $\alpha = d_j/d_k$  and  $y = \hat{y}_j$ .

**Case 4:**  $\hat{y}_j \geq \frac{1}{2}$  and  $\hat{y}_k \geq \frac{1}{2}$ : In case  $d_k \leq d_j$ , it holds  $\rho = 1$ ,  $\theta = \hat{y}_j + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k}{d_j}$ , leading to

$$\begin{aligned}
\rho - \rho^2 + \theta - \theta^2 &= \theta - \theta^2 \\
&= \hat{y}_j + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k}{d_j} - \left(\hat{y}_j + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k}{d_j}\right)^2 \\
&= \hat{y}_j - \hat{y}_j^2 + \frac{d_k}{d_j}\hat{y}_k - \frac{d_k}{d_j} - \frac{d_k^2}{d_j^2}\hat{y}_k^2 - \frac{d_k^2}{d_j^2} - 2\hat{y}_j \frac{d_k}{d_j}\hat{y}_k + 2\frac{d_k}{d_j}\hat{y}_j + 2\frac{d_k^2}{d_j^2}\hat{y}_k \\
&= \hat{y}_j - \hat{y}_j^2 + (1 - 2\hat{y}_j)\frac{d_k}{d_j}\hat{y}_k - \left(\frac{d_k}{d_j} - 2\frac{d_k}{d_j}\hat{y}_j\right) + \frac{d_k^2}{d_j^2}(2\hat{y}_k - 1 - \hat{y}_k^2)
\end{aligned}$$

$$\begin{aligned}
&= \hat{y}_j - \hat{y}_j^2 + (2\hat{y}_j - 1) \frac{d_k}{d_j} (1 - \hat{y}_k) - \frac{d_k^2}{d_j^2} (1 - \hat{y}_k)^2 \\
&< \hat{y}_j - \hat{y}_j^2 + \frac{d_k}{d_j} (1 - \hat{y}_k) - \frac{d_k^2}{d_j^2} (1 - \hat{y}_k)^2 \\
&\leq \hat{y}_j - \hat{y}_j^2 + (1 - \hat{y}_k) - (1 - \hat{y}_k)^2 \\
&= \hat{y}_j - \hat{y}_j^2 + \hat{y}_k - \hat{y}_k^2,
\end{aligned}$$

because of  $1 - \hat{y}_j \leq 1/2$  and Lemma 4.3.1 with  $\alpha = d_k/d_j$  and  $y = 1 - \hat{y}_k$ . In the opposite case  $d_k > d_j$ , we have  $\zeta = 1$ ,  $\xi = \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j - \frac{d_j}{d_k}$ , which implies

$$\begin{aligned}
&\zeta - \zeta^2 + \xi - \xi^2 \\
&= \xi - \xi^2 \\
&= \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j - \frac{d_j}{d_k} - \left( \hat{y}_k + \frac{d_j}{d_k} \hat{y}_j - \frac{d_j}{d_k} \right)^2 \\
&= \hat{y}_k - \hat{y}_k^2 + \frac{d_j}{d_k} \hat{y}_j - \frac{d_j}{d_k} - \frac{d_j^2}{d_k^2} \hat{y}_j^2 - \frac{d_j^2}{d_k^2} - 2\hat{y}_k \frac{d_j}{d_k} \hat{y}_j + 2\frac{d_j}{d_k} \hat{y}_k + 2\frac{d_j^2}{d_k^2} \hat{y}_j \\
&= \hat{y}_k - \hat{y}_k^2 + (1 - 2\hat{y}_k) \frac{d_j}{d_k} \hat{y}_j - \left( \frac{d_j}{d_k} - 2\frac{d_j}{d_k} \hat{y}_k \right) + \frac{d_j^2}{d_k^2} (2\hat{y}_j - 1 - \hat{y}_j^2) \\
&= \hat{y}_k - \hat{y}_k^2 + (2\hat{y}_k - 1) \frac{d_j}{d_k} (1 - \hat{y}_j) - \frac{d_j^2}{d_k^2} (1 - \hat{y}_j)^2 \\
&< \hat{y}_k - \hat{y}_k^2 + \frac{d_j}{d_k} (1 - \hat{y}_j) - \frac{d_j^2}{d_k^2} (1 - \hat{y}_j)^2 \\
&\leq \hat{y}_k - \hat{y}_k^2 + (1 - \hat{y}_j) - (1 - \hat{y}_j)^2 \\
&= \hat{y}_k - \hat{y}_k^2 + \hat{y}_j - \hat{y}_j^2,
\end{aligned}$$

because of  $1 - \hat{y}_j \leq 1/2$  and Lemma 4.3.1 with  $\alpha = d_j/d_k$  and  $y = 1 - \hat{y}_j$ .

Thus, we can replace  $\hat{y}_j$  and  $\hat{y}_k$  either with  $\zeta$  and  $\xi$  or with  $\theta$  and  $\rho$  to obtain a better lower-level objective function value. This holds for all  $M > 0$ . Therefore,  $\hat{y}$  is not optimal for the lower level, which is a contradiction to our assumption. This proves that  $\hat{y}_j$  and  $\hat{y}_k$  cannot be both fractional at the same time.  $\square$

In the special case of  $\|d\|_\infty = 1$ , we show that if  $\hat{y}$  is a globally optimal solution to the  $x$ -parameterized lower level, then there exists no  $j$  with  $\hat{y}_j \in (0, 1)$ .

**Lemma 4.3.3.** *Assume that  $\|d\|_\infty = 1$ . Let  $x$  be feasible for the upper level of (4.3). Then, every globally optimal solution to the  $x$ -parameterized lower level is binary.*

*Proof.* If  $\|d\|_\infty = 1$ , then  $d_i \in \{0, 1\}$  for  $i \in [n_y]$ . Let  $J := \{i \in [n_y] : d_i = 1\}$ . If  $|J| \leq x$ , then  $d^\top y \leq x$  for all  $y \in [0, 1]^{n_y}$ . Consequently, every point  $\hat{y}$

with  $\hat{y}_i = 1$  for  $i \in J_x$  and  $\hat{y}_i \in \{0, 1\}$  for  $i \notin J_x$  is lower-level optimal. Now assume that at least  $x$ -many entries of  $d$  are one, i.e.,  $|J| \geq x$ . Let  $J_x$  be any subset of  $J$  with  $|J_x| = x$ . Set  $\hat{y}_i = 1$  for  $i \in J_x$  and  $\hat{y}_i = 0$  for  $i \notin J_x$ . Then, the lower-level objective function value is given by

$$d^\top \hat{y} - M \left( \sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2 \right) = x.$$

Hence,  $\hat{y}$  is globally optimal for the lower level. Furthermore, every point  $\bar{y}$  with  $d^\top \bar{y} = x$  and  $\bar{y}_j \in (0, 1)$  for one  $j$  yields a lower-level objective function value of

$$d^\top \bar{y} - M \left( \sum_{i=1}^{n_y} \bar{y}_i - \bar{y}_i^2 \right) < x,$$

i.e., it is not globally optimal.  $\square$

For  $\|d\|_\infty \geq 2$  we use Lemma 4.3.2 to show that a fractional component of the globally optimal solution to the  $x$ -parameterized lower level of Problem (4.3) cannot be arbitrarily close to zero or one.

**Lemma 4.3.4.** *Assume that  $\|d\|_\infty \geq 2$ . Let  $x$  be feasible for the upper level of (4.3) and let  $\hat{y}$  be a globally optimal solution to the  $x$ -parameterized lower level. Let  $j$  be an index such that  $\hat{y}_j \in (0, 1)$ . Then, it holds*

$$d^\top \hat{y} = x$$

and

$$\frac{1}{\|d\|_\infty} \leq \hat{y}_j \leq 1 - \frac{1}{\|d\|_\infty}.$$

*Proof.* For the first part assume that  $d^\top \hat{y} < x$ . If  $\hat{y}$  is globally optimal for the lower level, we know that  $d_j > 0$  because if  $d_j = 0$ , then  $\hat{y}_j \in \{0, 1\}$  would lead to a better lower-level objective function value. Furthermore, we have

$$\sum_{i=1, i \neq j}^{n_y} d_i \hat{y}_i \leq d^\top \hat{y} - M(\hat{y}_j - \hat{y}_j^2),$$

i.e., the lower-level objective function value is at least as good as in the case in which we set  $\hat{y}_j$  to zero. It follows

$$d_j \hat{y}_j \geq M(\hat{y}_j - \hat{y}_j^2) \quad \text{and} \quad d_j \geq M(1 - \hat{y}_j). \quad (4.4)$$

Since  $d^\top \hat{y} < x$  and  $\hat{y}_j \in (0, 1)$ , we can find an  $\varepsilon > 0$  such that  $\hat{y}_j + \varepsilon < 1$  and

$$d^\top \hat{y} < d^\top \hat{y} + d_j \varepsilon \leq x.$$

With (4.4) we get that

$$\begin{aligned}
& M(\hat{y}_j + \varepsilon) > 0 \\
\iff & M(1 - \hat{y}_j) - M(1 - 2\hat{y}_j - \varepsilon) > 0 \\
\implies & d_j - M(1 - 2\hat{y}_j - \varepsilon) > 0 \\
\iff & d_j\varepsilon - M(\varepsilon - 2\varepsilon\hat{y}_j - \varepsilon^2) > 0 \\
\iff & d_j(\hat{y}_j + \varepsilon) - M(\hat{y}_j + \varepsilon - (\hat{y}_j + \varepsilon)^2) > d_j\hat{y}_j - M(\hat{y}_j - \hat{y}_j^2).
\end{aligned}$$

From the last inequality it follows that, if we replace  $\hat{y}_j$  with  $\hat{y}_j + \varepsilon$ , we get a better lower-level objective function value, i.e.,  $\hat{y}$  is not globally optimal. This proves  $d^\top \hat{y} = x$ .

For the second part assume that there exists an index  $j$  such that  $\hat{y}_j \in (0, 1)$  and  $\hat{y}_j < 1/\|d\|_\infty$ . It holds  $d_j > 0$  because if  $d_j = 0$ , then  $\hat{y}_j \in \{0, 1\}$  would lead to a better lower-level objective function value. With this we have

$$0 < d_j\hat{y}_j < \frac{d_j}{\|d\|_\infty} \leq 1.$$

As shown before, it holds  $d^\top \hat{y} = x$ , leading to

$$x - 1 < \sum_{i=1, i \neq j}^{n_y} d_i \hat{y}_i < x.$$

From  $x \in \mathbb{Z}$  it follows that  $\sum_{i=1, i \neq j}^{n_y} d_i \hat{y}_i \notin \mathbb{Z}$  and, hence, there exists at least one index  $k \neq j$  with  $\hat{y}_k \in (0, 1)$  and  $d_k > 0$ . With Lemma 4.3.2 we get that  $\hat{y}$  cannot be optimal, which is a contradiction to our assumption. Thus, it has to hold

$$\frac{1}{\|d\|_\infty} \leq \hat{y}_j.$$

Assume now that there exists an index  $j$  such that  $\hat{y}_j \in (0, 1)$  and  $\hat{y}_j > 1 - 1/\|d\|_\infty$ . Again, we have  $d_j > 0$  because if  $d_j = 0$ , then  $\hat{y}_j \in \{0, 1\}$  would lead to a better lower-level objective function value. Furthermore, it holds

$$d_j - 1 \leq d_j \left(1 - \frac{1}{\|d\|_\infty}\right) < d_j\hat{y}_j < d_j.$$

We showed that  $d^\top \hat{y} = x$ . Consequently, we have

$$x - d_j < \sum_{i=1, i \neq j}^{n_y} d_i \hat{y}_i < x - d_j + 1,$$

i.e.,  $\sum_{i=1, i \neq j}^{n_y} d_i \hat{y}_i \notin \mathbb{Z}$ . Again, there exists at least one index  $k \neq j$  with  $\hat{y}_k \in (0, 1)$  and  $d_k > 0$ . Using Lemma 4.3.2, we see that  $\hat{y}$  cannot be optimal and, thus,

$$\hat{y}_j \leq 1 - \frac{1}{\|d\|_\infty}$$

holds. □

With Lemma 4.3.4 we immediately get the following result.

**Lemma 4.3.5.** *Assume that  $\|d\|_\infty \geq 2$ . Let  $(\hat{x}, \hat{y})$  be bilevel feasible for Problem (4.3) with  $\hat{y}_j \in (0, 1)$  for one index  $j$ . Then, it holds*

$$\sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2 \geq \frac{1}{\|d\|_\infty} - \frac{1}{\|d\|_\infty^2}.$$

*Proof.* From Lemma 4.3.2 we know that  $\hat{y}_j$  is the only fractional component in  $\hat{y}$ . Assume that  $\hat{y}_j \leq 1/2$ . Then, with Lemma 4.3.4 we have

$$\frac{1}{\|d\|_\infty} - \frac{1}{\|d\|_\infty^2} \leq \hat{y}_j - \hat{y}_j^2 = \sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2.$$

Now assume that  $\hat{y}_j \geq 1/2$ . Then, it holds

$$\frac{1}{\|d\|_\infty} - \frac{1}{\|d\|_\infty^2} = 1 - \frac{1}{\|d\|_\infty} - \left(1 - \frac{1}{\|d\|_\infty}\right)^2 \leq \hat{y}_j - \hat{y}_j^2 = \sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2. \quad \square$$

Lemma 4.3.5 states a lower bound for the penalization in the lower-level objective function of a bilevel-feasible point that is not integer. Now, we revisit the bilevel problem

$$\begin{aligned} \max_{x \in \mathbb{Z}, y} \quad & c_x x + c_y^\top y \\ \text{s.t.} \quad & x^- \leq x \leq x^+, \\ & y \in \arg \max_{\bar{y}} \left\{ d^\top \bar{y} : d^\top \bar{y} \leq x, \bar{y} \in \{0, 1\}^{n_y} \right\}, \end{aligned} \quad (4.5)$$

where  $0 \neq d \in \mathbb{Z}_+^{n_y}$ ,  $c_y \in \mathbb{Z}_+^{n_y}$ , and  $c_x, x^-, x^+ \in \mathbb{Z}_+$ ; see Problem (2.9) in Section 2.2. The problem has been introduced in Dempe and Richter (2000) and its  $\Sigma_2^P$ -hardness has been shown in Section 3.1 in Caprara et al. (2014), where it is called ‘‘DeRi’’.

We show the equivalence of Problem (4.3) and Problem (4.5) for  $M$  being sufficiently large but still of polynomial size (in the size of the input data of the problem).

**Theorem 4.3.6.** *Let  $\mathcal{B}_1$  be the set of bilevel-feasible points of Problem (4.3) and let  $\mathcal{B}_2$  be the set of bilevel-feasible points of Problem (4.5). If  $\|d\|_\infty = 1$ , then for any  $M > 0$  it holds  $\mathcal{B}_1 = \mathcal{B}_2$ . Furthermore, if  $\|d\|_\infty \geq 2$ , then, for*

$$M > \frac{n_y \|d\|_\infty^3}{\|d\|_\infty - 1},$$

*it holds  $\mathcal{B}_1 = \mathcal{B}_2$ .*

*Proof.* For  $\|d\|_\infty = 1$ , Lemma 4.3.3 ensures  $\mathcal{B}_1 = \mathcal{B}_2$ .

Now assume that  $\|d\|_\infty \geq 2$ . First, we show that  $\mathcal{B}_1 \subseteq \mathcal{B}_2$ . Let  $(\hat{x}, \hat{y})$  be a bilevel-feasible point for Problem (4.3). Assume that there exists an index  $j$  such that  $\hat{y}_j \in (0, 1)$ . From Lemma 4.3.2 we know that  $\hat{y}_i \in \{0, 1\}$  for all  $i \neq j$  and with Lemma 4.3.5, we get

$$\sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2 \geq \frac{1}{\|d\|_\infty} - \frac{1}{\|d\|_\infty^2}.$$

For  $M > n_y \|d\|_\infty^3 / (\|d\|_\infty - 1)$ , we have

$$\begin{aligned} d^\top \hat{y} - M \left( \sum_{i=1}^{n_y} \hat{y}_i - \hat{y}_i^2 \right) &\leq d^\top \hat{y} - M \left( \frac{1}{\|d\|_\infty} - \frac{1}{\|d\|_\infty^2} \right) \\ &< d^\top \hat{y} - n_y \|d\|_\infty \\ &\leq 0 = \min \left\{ d^\top y : y \in [0, 1]^{n_y} \right\}. \end{aligned} \tag{4.6}$$

Any  $\tilde{y} \in \{0, 1\}^{n_y}$  satisfies  $M \left( \sum_{i=1}^{n_y} \tilde{y}_i - \tilde{y}_i^2 \right) = 0$  and, hence, leads to a better lower-level objective function value than  $\hat{y}$ . Consequently,  $\hat{y}$  is not bilevel feasible. With that, every point being bilevel feasible for Problem (4.3) maximizes  $d^\top y$  subject to  $d^\top y \leq x$  and  $y \in \{0, 1\}^{n_y}$ , i.e., it is bilevel feasible for Problem (4.5).

Now we show that  $\mathcal{B}_2 \subseteq \mathcal{B}_1$ . Let  $(\hat{x}, \hat{y})$  be bilevel feasible for Problem (4.5). With (4.6) we get that  $\hat{y}$  has to be optimal for the  $\hat{x}$ -parameterized lower level of Problem (4.3). Thus,  $(\hat{x}, \hat{y})$  is bilevel feasible for Problem (4.3).  $\square$

To sum up, we have shown that Problem (4.3) is equivalent to Problem (4.2), which is a special case of Problem (4.1). Furthermore, there exists a constant  $M > 0$  of polynomial size (in the input data of the problem) so that Problem (4.3) is equivalent to Problem (4.5); see Theorem 4.3.6. Hence, we have the following result.

**Theorem 4.3.7.** *Problem (4.1) is  $\Sigma_2^P$ -hard.*

## 4.4 Solution Approach

We tackle Problem (4.1) by using an iterative method, which improves the lower and upper bound for the optimal objective function value until global optimality is certified. In Section 4.4.1 and 4.4.2 we describe how we get lower and upper bounds before we then state the algorithm in Section 4.4.3 and prove its correctness.

#### 4.4.1 Computing Lower Bounds

To compute lower bounds for Problem (4.1), we replace the lower level with its KKT conditions. This yields the optimization problem

$$\begin{aligned}
\min_{x,y,\lambda} \quad & F(x,y) = \frac{1}{2}x^\top H_x x + c_x^\top x + \frac{1}{2}y^\top H_y y + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \quad Cx_I + Dy \geq b, \\
& x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \quad \text{for all } i \in I, \\
& Qy + d - D^\top \lambda = 0, \quad \lambda \geq 0, \\
& \lambda^\top (Cx_I + Dy - b) = 0.
\end{aligned} \tag{4.7}$$

For Problem (4.7), the following property holds.

**Proposition 4.4.1.** *Problem (4.7) is a relaxation of problem (4.1) in the following sense: Let  $(x^*, y^*)$  be a globally optimal solution to the bilevel problem (4.1) and let  $(\hat{x}, \hat{y}, \hat{\lambda})$  be a globally optimal solution to the KKT relaxation (4.7). Then, it holds  $F(\hat{x}, \hat{y}) \leq F(x^*, y^*)$ . Furthermore, if  $(\hat{x}, \hat{y})$  is bilevel feasible, it holds  $F(\hat{x}, \hat{y}) = F(x^*, y^*)$ , i.e.,  $(\hat{x}, \hat{y})$  is bilevel optimal.*

*Proof.* We define the set of KKT points of the  $x$ -parameterized lower level as follows:

$$\begin{aligned}
\text{KKT}(x) := \{ & y \in \mathbb{R}^{n_y} : Dy \geq b - Cx_I, \\
& \exists \lambda \in \mathbb{R}_+^m : Qy + d - D^\top \lambda = 0, \lambda^\top (Cx_I + Dy - b) = 0 \}.
\end{aligned}$$

Since all constraints of the lower-level problem are linear, the KKT theorem leads to  $S(x) \subseteq \text{KKT}(x)$  for all  $x$ . The remainder is evident.  $\square$

Proposition 4.4.1 states that we obtain a lower bound for the optimal objective value of the bilevel problem (4.1) by solving the KKT relaxation (4.7), which is an MIQP with complementarity constraints. We handle the latter KKT complementarity conditions via SOS1-type constraints; see, Section 2.5.

In our iterative method, we improve this lower bound by successively reducing the feasible region of Problem (4.7). We do this by adding cutting planes of the form  $h(x, y) \geq 0$ ; see Section 2.6.1 for a detailed discussion on cutting planes. Here,  $h$  is a function that satisfies  $h(x, y) < 0$  for all  $(x, y) \in V'$  and  $h(x, y) \geq 0$  for all  $(x, y) \notin V'$ , where  $V'$  is a certain subset of  $\tilde{\Omega}$ . In our application,  $h$  is linear, i.e., the incorporation of these cuts can easily be embedded into our general framework by suitable extension of  $A$  and  $b$ , and no constraint qualifications are needed. The set  $V'$  will, among others, contain points that we already visited in previous iterations. Note that a solution to Problem (4.7) that additionally contains cutting planes still provides a lower bound as long as the cuts do not cut off all bilevel-optimal points. This follows directly from Proposition 4.4.1 and is formalized in the following result.

**Proposition 4.4.2.** *Let  $k$  be a positive integer. For  $i = 1, \dots, k$ , let  $V_i \subseteq \tilde{\Omega}$  and let  $h_i$  be a function that separates  $V_i$  and  $P \setminus V_i$ , i.e.,*

$$h_i(x, y) < 0 \text{ for all } (x, y) \in V_i \quad \text{and} \quad h_i(x, y) \geq 0 \text{ for all } (x, y) \in P \setminus V_i.$$

*Consider Problem (4.7) with the additional restrictions  $h_i(x, y) \geq 0$  for  $i = 1, \dots, k$ , and let  $(\hat{x}, \hat{y}, \hat{\lambda})$  be a globally optimal solution to that problem. If there exists a point  $(x^*, y^*) \notin V := \bigcup_{i=1}^k V_i$  that is globally optimal for the bilevel problem (4.1), then it holds*

$$F(\hat{x}, \hat{y}) \leq F(x^*, y^*).$$

From Proposition 4.4.1 we know that if a globally optimal solution to the KKT relaxation (4.7) is bilevel feasible, it is also globally optimal for the bilevel problem (4.1). On the other hand, if it is not bilevel feasible, we remove it from the feasible region of Problem (4.7) by using a cutting plane. If we re-solve the problem, we get a lower bound that is not smaller than the lower bound obtained by the previous solution. Hence, in an iterative method that adds cutting planes to Problem (4.7) in each iteration, we successively improve the lower bound. The derivation of the cutting planes is discussed in Section 4.4.3.

#### 4.4.2 Computing Upper Bounds

Every bilevel-feasible point provides an upper bound (and every optimal solution to Problem (4.7), which is bilevel feasible, is also bilevel optimal by Proposition 4.4.1). However, even if a globally optimal solution  $(\hat{x}, \hat{y}, \hat{\lambda})$  to the KKT relaxation (4.7) is bilevel infeasible, we still use it to obtain an upper bound for the optimal objective function value of Problem (4.1). To do so, we first solve the  $\hat{x}$ -parameterized lower level (a nonconvex QP) to obtain

$$\Phi(\hat{x}) := \min_y \left\{ \frac{1}{2} y^\top Q y + d^\top y : C \hat{x}_I + D y \geq b, y \in \mathbb{R}^{n_y} \right\}.$$

Afterward, we apply a refinement procedure as described in Section 3.4.3; see also Fischetti et al. (2018). Hence, we solve the restricted high-point relaxation, a QP with an additional nonconvex quadratic constraint (QCQP)

$$\begin{aligned} \min_{x, y} \quad & F(x, y) \\ \text{s.t.} \quad & (x, y) \in P, \\ & x_I = \hat{x}_I, \\ & f(y) \leq \Phi(\hat{x}); \end{aligned} \tag{4.8}$$

see Section 2.6.3. If the refinement problem (4.8) is feasible, we obtain a bilevel-feasible point, and, hence, an upper bound. Otherwise, every point  $(x, y)$  with  $x_I = \hat{x}_I$  is bilevel infeasible.

### 4.4.3 The Algorithm

The proposed method successively generates upper bounds for the optimal objective function value of Problem (4.1) by computing bilevel-feasible points. Furthermore, it iteratively improves the lower bound. For the former, in each iteration, we compute a bilevel-feasible point as explained in Section 4.4.2 if possible and for the latter, we shrink the feasible region of Problem (4.7) by adding cutting planes as explained in Section 4.4.1. We use the relative optimality gap

$$\gamma := \frac{|u - \ell|}{10^{-10} + |u|} \quad (4.9)$$

as a stopping criterion, where  $\ell$  is the best known lower bound and  $u$  is the incumbent; see Section 2.4 for further details. As cutting planes we use integer no-good cuts on the linking variables  $x_I$ .

**Remark 4.4.3.** *We have discussed in Section 2.6.4 and demonstrated in Section 3.6 that integer no-good cuts are algorithmically inefficient. However, we cannot derive other cutting planes such as intersection or disjunctive cuts as done in Fischetti et al. (2018), Gaar et al. (2023), or in Chapter 3 of this thesis to add them to Problem (4.7). This is because the lower level of (4.1) is continuous, i.e., a disjunctive or intersection cut can be arbitrarily weak w.r.t. the directions of the  $y$  variables. Moreover, it remains unclear how to compute such a cut; see Example 3.5.3. In addition to that, the existence of a disjunctive or intersection cut cannot be guaranteed since Problem (4.7) is nonconvex; see the proof of Theorem 3.3.4. This is why we recall to integer no-good cuts which we only apply on the (integer) linking variables  $x_I$ . To do so, we first represent the variables  $x_i$  for  $i \in I$  with binary vectors  $v^i$  (see Section 2.6.4), i.e.,*

$$x_i = -2^{s_i} v_{s_i}^i + \sum_{k=0}^{s_i-1} 2^k v_k^i,$$

where

$$s_i := \lceil \log_2 (\max \{|x_i^-|, |x_i^+|\}) + 1 \rceil.$$

Now, let  $(\hat{x}, \hat{y}, \hat{\lambda})$  be a point that is feasible for the KKT relaxation (4.7), and let  $\{\hat{v}^i : i \in I\}$  be the binary vectors representing  $\hat{x}_I$ . After we perform a refinement step as explained in Section 4.4.2, we use the inequality

$$\sum_{i \in I} \left( \sum_{k \in S_i: \hat{v}_k^i = 0} v_k^i + \sum_{k \in S_i: \hat{v}_k^i = 1} (1 - v_k^i) \right) \geq 1 \quad (4.10)$$

with  $S_i := \{0, \dots, s_i\}$  to cut off every point  $(x, y)$  with  $x_I = \hat{x}_I$ .

Note that the integer no-good cut in (4.10) may cut off multiple points. This can be done because after performing the refinement step, none of the points  $(x, y)$  with  $x_I = \hat{x}_I$  can improve the upper bound  $u$  anymore. In addition to integer no-good cuts, we add an optimality cut of the form  $F(x, y) \leq u$  to Problem (4.7). Adding both types of inequalities to the KKT relaxation leads to the problem

$$\begin{aligned}
\min_{x, y, \lambda} \quad & F(x, y) = \frac{1}{2}x^\top H_x x + c_x^\top x + \frac{1}{2}y^\top H_y y + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \quad Cx_I + Dy \geq b, \\
& x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \text{ for all } i \in I, \\
& Qy + d - D^\top \lambda = 0, \quad \lambda \geq 0, \\
& \lambda^\top (Cx_I + Dy - b) = 0, \\
& \text{INGC (4.10) on } \tilde{x}_I \text{ for all } \tilde{x}_I \in \tilde{X}, \\
& F(x, y) \leq u.
\end{aligned} \tag{4.11}$$

The set  $\tilde{X}$  contains the values of the linking variables of points that are feasible for Problem (4.7) and gets updated throughout the procedure. The optimality cut in the last constraint is optional; see Section 4.5.3 for a thorough discussion. Note that Problem (4.11) is, in general, not a relaxation of the bilevel problem (4.1) anymore, because the optimality- and integer no-good cuts may separate bilevel-feasible points. However, we never cut off points that are bilevel feasible and that can improve the incumbent in the respective iteration. Hence, we get the following corollary.

**Corollary 4.4.4.** *Let  $(x^*, y^*)$  with  $x_I^* \notin \tilde{X}$  be a globally optimal solution to the bilevel problem (4.1) and let  $(\hat{x}, \hat{y}, \hat{\lambda})$  be a globally optimal solution to Problem (4.11). Then, it holds  $F(\hat{x}, \hat{y}) \leq F(x^*, y^*)$ .*

Corollary 4.4.4 is a special case of Proposition 4.4.2, where each function  $h_i(x, y)$  represents one integer no-good cut and  $V = \{(x, y) : x_I \in \tilde{X}\}$ . Note that the optimality cut does not separate bilevel-optimal points. Hence, its presence in Problem (4.11) does not interfere with the statement in Corollary 4.4.4. Based on Problem (4.11) we propose the procedure in Algorithm 4 to solve the bilevel problem (4.1).

In Line 2 of Algorithm 4 we solve Problem (4.11). Note that this problem is the KKT relaxation (4.7) for  $k = 0$ . By Assumption 2.1.1, we know that if it is feasible, we get a solution  $(x^k, y^k, \lambda^k)$  in Line 4. However, for some iteration  $k$ , Problem (4.11) with  $\tilde{X} := \tilde{X}_k$  might be infeasible. This can happen when  $\tilde{X}_k$  contains every bilevel-optimal point. In that case,  $u$  equals the optimal objective function value of Problem (4.1), see Line 13, and the optimality cut prevents  $F(x^k, y^k)$  from exceeding  $u$ . Moreover, the last bilevel-feasible point that lead to an update of  $u$  in a previous iteration is

---

**Algorithm 4:** Iterative method to solve the bilevel problem (4.1).

---

**Input:** An instance of Problem (4.1),  $\ell = -\infty$ ,  $u = \infty$ ,  $\gamma = \infty$ ,  $k = 0$ , and  $\tilde{X}_0 = \emptyset$ .

**Output:** A globally optimal solution  $(x^*, y^*)$  to the bilevel problem (4.1) or a correct indication of infeasibility.

```

1 if  $\gamma > 0$  then
2   Solve Problem (4.11) with  $\tilde{X} = \tilde{X}_k$  to global optimality.
3   if Problem (4.11) is feasible then
4     Let  $(x^k, y^k, \lambda^k)$  be the solution to Problem (4.11). Set
        $\ell \leftarrow F(x^k, y^k)$ .
5     Solve the  $x^k$ -parameterized lower-level problem to compute an
       optimal follower's response  $\bar{y}^k$  as well as  $\Phi(x^k)$ .
6     if  $(x^k, y^k)$  is bilevel-feasible, i.e.,  $f(y^k) = f(\bar{y}^k)$  then
7       | Set  $(x^*, y^*) \leftarrow (x^k, y^k)$  and go to Line 25.
8     else
9       | Solve the refinement problem (4.8).
10      | if the refinement problem (4.8) is feasible then
11        | Denote the obtained bilevel-feasible point  $(\bar{x}, \bar{y})$ . It
          | holds  $\bar{x}_I = x_I^k$ .
12        | if  $F(\bar{x}, \bar{y}) < u$  then
13          | | Set  $u \leftarrow F(\bar{x}, \bar{y})$  and  $(x^*, y^*) \leftarrow (\bar{x}, \bar{y})$ .
14        | if  $u < \infty$  then
15          | | Set  $\gamma \leftarrow |u - \ell| / (10^{-10} + |u|)$ .
16        | Set  $\tilde{X}_{k+1} \leftarrow \tilde{X}_k \cup \{x_I^k\}$  and  $k \leftarrow k + 1$ . Go to Line 2.
17      | end
18    else if  $u < \infty$  then
19      | Go to Line 25.
20    end
21  else
22    | return "The bilevel problem (4.1) is infeasible."
23  end
24 else
25 | return A globally optimal solution  $(x^*, y^*)$  to Problem (4.1).
26 end

```

---

bilevel optimal. Hence, the algorithm proceeds in Line 18 and verifies that  $u$  is finite. It then terminates in Line 25 with a globally optimal solution to Problem (4.1).

Otherwise, i.e., if Problem (4.11) is feasible, the value  $F(x^k, y^k)$  is always a lower bound on the optimal upper-level objective function value; see Corollary 4.4.4. Hence, we update the lower bound  $\ell$  in Line 4. Then,

we solve the  $x^k$ -parameterized lower-level problem in Line 5 to obtain an optimal response  $\bar{y}^k$  for the given  $x^k$ . It is possible that  $(x^k, y^k)$  is already bilevel feasible, i.e.,  $f(y^k) = f(\bar{y}^k)$ . Then, the point is also bilevel optimal, see Proposition 4.4.5 below, and the algorithm terminates. The computed point  $(x^k, \bar{y}^k)$  is bilevel feasible if it satisfies the upper-level constraints. Moreover, the  $x^k$ -parameterized lower-level could have multiple globally optimal solutions that lead to different values in the upper-level objective. To address both subjects, we perform a refinement step in Line 9 of Algorithm 4; see Section 4.4.2 and Fischetti et al. (2018) for further details. Therefore, we solve the restricted high-point relaxation (4.8). If it is feasible, then we obtain a bilevel-feasible point in Line 11 and we improve the incumbent  $u$  if possible; see Line 13. Note that this also updates the optimality cut given in Problem (4.11). Otherwise, every pair  $(x, y)$  with  $x_I = x_I^k$  and  $y$  being an optimal solution to the  $x_I^k$ -parameterized lower level violates the upper-level constraints. Hence, there exists no bilevel-feasible point  $(\bar{x}, \bar{y})$  with  $\bar{x}_I = x_I^k$ . In both cases, we checked every point  $(x, y)$  with  $x_I = x_I^k$  for bilevel feasibility. Hence, we use an integer no-good cut as described in (4.10) to remove all points  $(x, y)$  with  $x_I = x_I^k$  from our search space. We do this by setting  $\tilde{X}_{k+1}$  to  $\tilde{X}_k \cup \{x_I^k\}$  in Line 16 of Algorithm 4.

The algorithm terminates if the optimality gap  $\gamma$  as defined in (4.9) is zero at the start of any iteration; see Line 1. In that case, the last point  $(x^*, y^*)$  that improved our incumbent  $u$  is bilevel optimal; see Line 25. Once the method updates the upper bound in Line 13,  $\gamma$  gets updated in every iteration; see Line 15. The algorithm also terminates if for a solution  $(x^k, y^k, \lambda^k)$  obtained in Line 4, the point  $(x^k, y^k)$  is bilevel feasible; see Lines 6–7 and Proposition 4.4.5 below. In both cases, the algorithm terminates with a globally optimal solution to the bilevel problem (4.1). However, the algorithm also detects infeasibility of the bilevel problem. If Problem (4.11) with  $\tilde{X} := \tilde{X}_k$  is infeasible for some iteration  $k$ , then the algorithm checks in Line 18 if  $u < \infty$ , i.e., if a bilevel-feasible point exists. If this is the case, then  $(x^*, y^*)$ , i.e., the last point that improved the incumbent, is a globally optimal solution to the bilevel problem (4.1). Otherwise, the bilevel problem (4.1) is infeasible; see Line 22. Note that the check  $\gamma > 0$  in the very first line can also be replaced by  $\gamma \geq \varepsilon$  in order to be able to terminate earlier with  $\varepsilon$ -optimal points.

First, we prove that the termination criterion in Lines 6–7 is correct.

**Proposition 4.4.5.** *If for any iteration  $k \geq 0$  the solution  $(x^k, y^k)$  to Problem (4.11) with  $\tilde{X} := \tilde{X}_k$  is bilevel feasible, then it is bilevel optimal.*

*Proof.* Let  $(x^*, y^*)$  be a bilevel-optimal solution. If  $x_I^* \in \tilde{X}_k$  for iteration  $k$ , i.e., we already identified the point  $(x^*, y^*)$  as bilevel feasible in a previous iteration  $k' < k$ , then we have  $u = F(x^*, y^*)$ . This is because  $F(x^*, y^*)$  yields the best possible upper bound. Due to the optimality cut  $F(x, y) \leq u$ , in iteration  $k$  of Algorithm 4, we exclude every non-optimal bilevel-feasible

point in Problem (4.11) and, hence,  $(x^k, y^k)$  has to be bilevel optimal as well. On the other hand, if  $x_I^* \notin \tilde{X}_k$ , then it holds  $F(x^k, y^k) \leq F(x^*, y^*)$  due to Corollary 4.4.4 and  $F(x^*, y^*) \leq F(x^k, y^k)$  because  $(x^k, y^k)$  is bilevel feasible. Hence, it is bilevel optimal.  $\square$

Note, that Proposition 4.4.5 does not hold true in general if we omit the optimality cut in Problem (4.11), which we illustrate using the following example.

**Example 4.4.6.** Consider the problem

$$\begin{aligned} \min_{x \in \{0,1\}, y} \quad & x + y^2 \\ \text{s.t.} \quad & y \in \arg \min \{ -y^2 : y \geq -x, -2y \geq 3x - 2 \}; \end{aligned} \quad (4.12)$$

see Figure 4.1 for an illustration. For  $x = 0$ , the follower can choose  $y \in [0, 1]$  and for  $x = 1$  it is  $y \in [-1, -0.5]$ . The KKT conditions of the lower level are given by

$$\begin{aligned} -2y - \lambda_1 + 2\lambda_2 &= 0, \\ \lambda_1, \lambda_2 &\geq 0, \\ y + x &\geq 0, \quad -2y - 3x + 2 \geq 0, \\ \lambda_1(y + x) &= 0, \quad \lambda_2(-2y - 3x + 2) = 0, \end{aligned}$$

where  $\lambda_1$  and  $\lambda_2$  are the respective Lagrange multipliers. For  $x = 0$ , the KKT points are given by  $(y, \lambda_1, \lambda_2) \in \{(0, 0, 0), (1, 0, 1)\}$ . For  $x = 1$ , there is one KKT point given by  $(y, \lambda_1, \lambda_2) = (-1, 2, 0)$ . Therefore, the KKT relaxation (4.7) has three feasible points out of which only  $(1, 0, 1)$  and  $(-1, 2, 0)$  correspond to bilevel-feasible points. The first one,  $(1, 0, 1)$  for  $x = 0$  is the uniquely determined bilevel-optimal point. When we solve the KKT relaxation (4.7) in Line 2 in iteration  $k = 0$  of Algorithm 4, we get the solution  $(x^0, y^0) = (0, 0)$ , which is bilevel infeasible. Then, we apply the refinement step in Line 9 and compute the bilevel-feasible point  $(\bar{x}, \bar{y}) = (0, 1)$  that is also globally optimal for the bilevel-problem. We update the incumbent to  $u = 1$  and set  $(x^*, y^*) = (0, 1)$ , see Line 13. Then, we add the no-good cut  $x \geq 1$  (note, that  $x$  is already binary) in Line 16. Now, if we consider Problem (4.11) in iteration  $k = 1$  with  $\tilde{X} = \{1\}$  and with the optimality cut  $F(x, y) \leq 1$ , the problem is infeasible. Hence, we terminate with the bilevel-optimal solution  $(x^*, y^*) = (0, 1)$ ; see Lines 18 and 25. However, if we omit the optimality cut in Problem (4.11), the point  $(x, y) = (1, -1)$  remains feasible in iteration  $k = 1$  and this will then be its solution. The point is also bilevel feasible, which means the algorithm would wrongly terminate with the non-optimal solution  $(x^*, y^*) = (1, -1)$ ; see Lines 6–7 and 25.

Since we use an integer no-good cut in every iteration of Algorithm 4, it is easy to see that the feasible region of problem (4.11) is reduced.

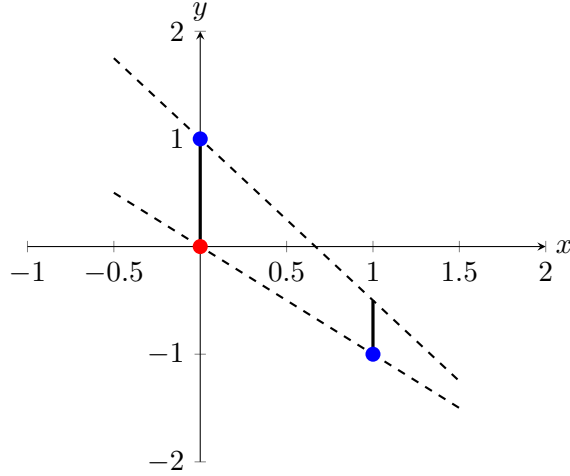


Figure 4.1: The feasible region of Problem (4.12). The point  $(0,0)$  is the bilevel-infeasible solution of the KKT relaxation. The points  $(0,1)$  and  $(1,-1)$  are bilevel feasible.

**Corollary 4.4.7.** *Let  $\mathcal{F}^k$  denote the feasible region of problem (4.11) in iteration  $k$ . Then, it holds  $\mathcal{F}^{k+1} \subsetneq \mathcal{F}^k$ .*

From this, it is obvious that the lower bound  $\ell$  is non-decreasing. With this at hand, we prove the correctness of the method.

**Theorem 4.4.8.** *Algorithm 4 terminates after finitely many steps either with a globally optimal solution of the bilevel problem (4.1) or with a correct indication of infeasibility.*

*Proof.* All upper-level variables that appear in the lower level, i.e.,  $x_i$  with  $i \in I$  are bounded integers; see Assumption 2.1.3. Hence, the set

$$T := \{x_I : x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \text{ for all } i \in I\}$$

is finite.

Let us first consider the case in which the bilevel problem (4.1) is infeasible. Then, the refinement problem that we solve in Line 9 is also infeasible. Hence, we never update the incumbent  $u$  since we never reach Line 13. However, we still apply an integer no-good cut that removes all points  $(x, y)$  with  $x_I = x_I^k$  in every iteration. After at most  $|T| < \infty$  many iterations, Problem (4.11) with  $\tilde{X} := \tilde{X}_{|T|}$  becomes infeasible and the algorithm terminates in Line 22 with a correct indication of infeasibility.

Now assume that the bilevel problem is feasible. If a solution  $(x^k, y^k)$  to Problem (4.11) with  $\tilde{X} := \tilde{X}_k$  is not bilevel feasible, we apply the refinement procedure in Line 9. Among all points  $(x, y) \in \tilde{\Omega}$  with  $x_I = x_I^k$  we compute a bilevel-feasible point that minimizes the upper-level objective function

value, if any exists. If the computed point improves the incumbent, we store it and update  $u$  in Line 13. Otherwise, there exists no point  $(x, y) \in \tilde{\Omega}$  with  $x_I = x_I^k$  that is bilevel feasible and improves the incumbent. In any case, we remove the points  $(x, y) \in \tilde{\Omega}$  with  $x_I = x_I^k$  by using an integer no-good cut on the linking variables of  $x^k$  in Line 16. Therefore, we cannot exclude bilevel-feasible points that would further improve our incumbent. After  $k^* \leq |T| < \infty$  many iterations, Problem (4.11) with  $\tilde{X} := \tilde{X}_{k^*}$  becomes either infeasible or has a solution that is bilevel feasible. In the first case, we know that we observed every point  $(x, y) \in \tilde{\Omega}$  because infeasibility can only result from the integer no-good cuts. Since the bilevel problem is feasible, we already obtained a bilevel-optimal point in a previous iteration and it is given by  $(x^*, y^*)$ . In the latter case, the algorithm terminates correctly with  $(x^{k^*}, y^{k^*})$  as a bilevel-optimal solution as shown in Proposition 4.4.5.  $\square$

## 4.5 Numerical Results

In this section we present the numerical results obtained with our method, applied to bilevel test instances from the literature. To the best of our knowledge, there is no publicly available solver that is tailored for mixed-integer quadratic bilevel problems with a nonconvex QP in the lower level. Note that there are methods that can solve the more general case of mixed-integer nonconvex bilevel problems; see, e.g., Mitsos (2010). However, their computational study is limited to only a few and small instances, and we are not aware of any publicly available implementation that we could use for a comparison. Since the high-point relaxation, i.e., Problem (4.11) without the KKT conditions of the lower-level problem, is the most classic relaxation in bilevel optimization (Kleinert et al. 2021b), we take this alternative relaxation as the baseline for our numerical study. Therefore, we compare the following two approaches:

**KKT:** Algorithm 4 with Problem (4.11) including the KKT conditions of the lower-level problem;

**HPR:** Algorithm 4 with Problem (4.11) not including the KKT conditions of the lower-level problem.

The implementation of both methods is publicly available at <https://github.com/AndreasHorlaender/Solver-for-MIQP-QP-Bilevel-Problems>. In Section 4.5.1, we briefly describe our computational setup. Then, in Section 4.5.2, we describe the instance sets used for our experiments. Finally, Section 4.5.3 contains the discussion of the numerical results.

### 4.5.1 Hardware and Software Setup

We implemented Algorithm 4 in Python 3.9.7 and we used Gurobi 11.0.2 to solve all occurring single-level problems. In our computational study we

used the default settings of **Gurobi**. All computations have been executed on the high performance cluster “Elwetritsch” at the TU Kaiserslautern, which is part of the “Alliance of High Performance Computing Rheinland-Pfalz” (AHRP). We used a single Intel XEON SP 6126 core with 2.6 GHz and a maximum of 30 GB RAM. Our time limit was 2 hours.

## 4.5.2 Test Instances

For our numerical tests we consider problems of the form

$$\min_{x \in \mathbb{Z}^{n_x}, y} c_x^\top x + c_y^\top y \quad \text{s.t.} \quad Ax + By \geq a, \quad y \in S(x),$$

where  $S(x)$  is the set of optimal solutions of the  $x$ -parameterized lower-level problem

$$\min_{y \in \mathbb{R}^{n_y}} \frac{1}{2} y^\top Q y + d^\top y \quad \text{s.t.} \quad Cx + Dy \geq b.$$

As for the numerical studies in Chapter 3, we use MILP-MILP problem instances from the **BOBILib**. Other instance libraries such as **BASBLib** (Paulavicius and Adjiman 2017) or **BOLIB** (Zhou et al. 2018) are not suitable for our purposes since they only include purely continuous bilevel instances. An overview of the used instance sets used in our numerical study is given in Table 4.1. We modify the instances by adding a nonconvex quadratic term to the lower-level objective function. Therefore, the symmetric and indefinite matrix  $Q$  is generated using the **MATLAB** function `sprandsym`. It takes the input  $(n_y, \text{density}, \text{eigenvalues})$ , where the density is computed according to the **MATLAB** function of Kleinert and Schmidt (2021), and the eigenvalues are randomly chosen integers in  $[-1000, 1000]$ . Since the entries of the computed matrix are fractional, we round them to obtain integer data.

For our numerical tests, we exclude the instance sets **Clique**, **Inter-Clique**, **Generalized**, **KP**, **Inter-Fire**, and **OR**, since almost none of those instances (extended with indefinite lower-level objective matrices) can be solved within the time limit of 2 hours. Hence, our test set consists of instances of the classes **QBMKP**, **Denegre**, **XuWang**, **XuLarge**, **Inter-Assig**, and **Inter-KP**; see Thürauf et al. (2024) for the details. As in Section 3.6.3, we exclude those instances which both methods can solve within 1 second. They are labeled “trivial” in Table 4.1 and we denote the remaining instances as “non-trivial”. Instances that neither **KKT** nor **HPR** can solve within the time limit of 2 hours are labeled with “time limit”. We also did this for instances, for which the square matrix  $Q$  cannot be computed within the time limit due to their size. In the latter case, we remove the instances from our test set. This affects 30 out of 31 instances in **XuLarge** that are labeled with time limit; see Table 4.1. All non-trivial instances for which either approach terminates within the time limit are labeled “solvable”. This includes the instances for which infeasibility is detected within the time limit.

Table 4.1: Overview of the test instances per instance class. The first group is the subset of instance test sets that we use in our numerical study.

Instance class	Total	Solvable (in %)	Time limit (in %)	Trivial (in %)
Denegre	50	34 (68.00 %)	2 (4.00 %)	14 (28.00 %)
Inter-Assig	24	24 (100.00 %)	0 (0.00 %)	0 (0.00 %)
Inter-KP	99	54 (54.55 %)	45 (45.45 %)	0 (0.00 %)
QBMKP	200	112 (56.00 %)	78 (39.00 %)	10 (5.00 %)
XuLarge	60	29 (48.33 %)	31 (51.67 %)	0 (0.00 %)
XuWang	100	93 (93.00 %)	0 (0.00 %)	7 (7.00 %)
Clique	60	6 (10.00 %)	54 (90.00 %)	0 (0.00 %)
Generalized	90	1 (1.11 %)	89 (98.89 %)	0 (0.00 %)
Inter-Clique	80	1 (1.25 %)	79 (98.75 %)	0 (0.00 %)
Inter-Fire	72	0 (0.00 %)	72 (100.00 %)	0 (0.00 %)
KP	450	3 (0.67 %)	447 (99.33 %)	0 (0.00 %)
OR	810	6 (0.74 %)	801 (98.89 %)	3 (0.37 %)
$\Sigma$	2095	363 (17.33 %)	1698 (81.05 %)	34 (1.62 %)

### 4.5.3 Discussion of the Results

In Algorithm 4, we solve Problem (4.11) in every iteration. We compare this approach to the case in which we omit the KKT conditions of the lower level in Problem (4.11) and use the high-point relaxation as a basis instead, i.e., we solve the problem

$$\begin{aligned}
\min_{x,y} \quad & F(x,y) = \frac{1}{2}x^\top H_x x + c_x^\top x + \frac{1}{2}y^\top H_y y + c_y^\top y \\
\text{s.t.} \quad & Ax + By \geq a, \quad Cx_I + Dy \geq b, \\
& x_i \in \mathbb{Z} \cap [x_i^-, x_i^+] \quad \text{for all } i \in I, \\
& \text{INGC (4.10) on } \tilde{x}_I \quad \text{for all } \tilde{x}_I \in \tilde{X}, \\
& F(x,y) \leq u.
\end{aligned} \tag{4.13}$$

In the former approach (called KKT in what follows), Problem (4.11) is a nonconvex mixed-integer problem and its feasible region only consists of points  $(x, y, \lambda)$  such that  $(y, \lambda)$  is a KKT point of the  $x$ -parameterized lower level. In the latter approach (denoted by HPR in the following), Problem (4.13) is a mixed-integer convex-quadratic problem with a feasible region that may contain points which do not lead to KKT points of the  $x$ -parameterized lower level.

The empirical cumulative distribution functions (ECDFs) w.r.t. runtimes, node counts, number of INGCs, and (relative) gaps at termination of KKT and HPR are given in Figure 4.2 and 4.3. In addition to the runtime results for the methods KKT and HPR, Figure 4.2 includes an ECDF illustrating

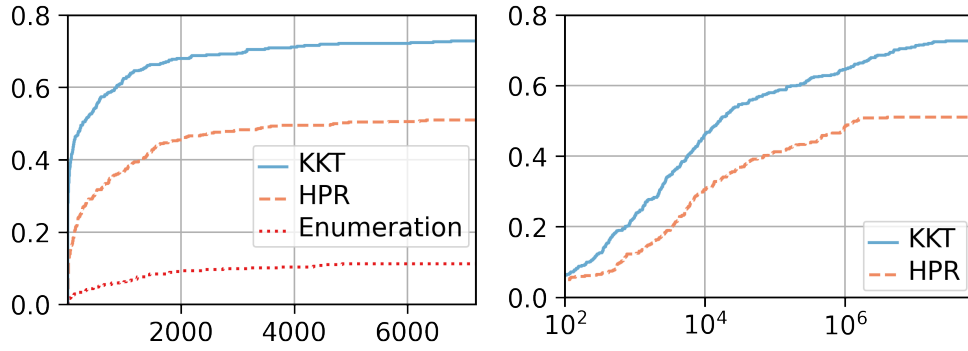


Figure 4.2: ECDFs for the non-trivial instances in our test set. Left: runtimes. Right: node counts.

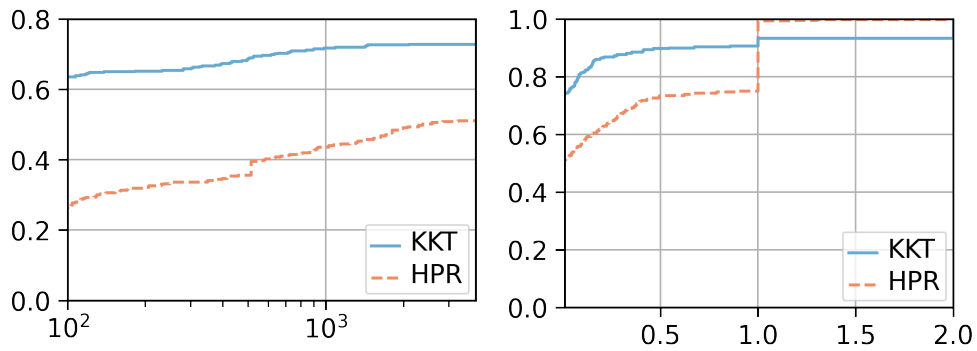


Figure 4.3: ECDFs for the non-trivial instances in our test set. Left: number of INGCs. Right: gaps.

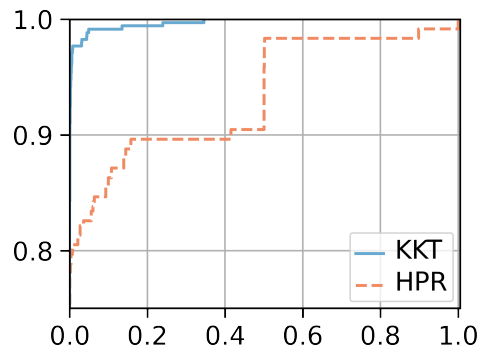


Figure 4.4: ECDF of  $(\#INGC)2^{n_b}$  for the non-trivial instances in our test set solved by the respective methods KKT and HPR.

the runtimes of a full-enumeration approach. In this approach, for every possible combination of variables  $x_I$ , we solve the  $x_I$ -parameterized lower-level problem and apply a refinement step afterward, see Section 4.4.2. Note that  $x_i$  is integer and bounded by  $x_i^- \leq x_i \leq x_i^+$  for  $i \in I$ , i.e., the number of possible combinations is finite. A bilevel-optimal solution is then chosen among the computed bilevel-feasible points that yield the best upper-level objective function value.

As illustrated in Figure 4.2, KKT clearly outperforms HPR w.r.t. node counts (see Figure 4.2) and the number of integer no-good cuts (see Figure 4.3). This can be expected since the feasible region of Problem (4.11) is, in general, much smaller than the one of Problem (4.13) used in HPR. However, the method KKT also outperforms method HPR w.r.t. the runtimes. Therefore, one can conclude that the benefit of having less iterations overall compensates for the issue that Problem (4.11) is harder to solve than Problem (4.13) due to the present nonconvexities.

In most of the instances that both methods cannot solve, the gaps are similarly small; see Figure 4.3 (right). In around 10 % of the instances, KKT does neither find an upper bound nor a lower bound, hence, no gap could be computed. However, for the same instances HPR finds both an upper and a lower bound. This is because in those instances, Problem (4.7) is already too hard to be solved in the time limit of 2 hours. In around 7 % of the instances, the method HPR finds a lower but no upper bound. In additional 17 % of instances, the method finds a lower bound of zero and a finite upper bound. In both cases, i.e., in approximately 24 % of the instances, we get a gap of 1. This is also reflected by the “jump” in Figure 4.3 (right).

We showcase the efficiency of the methods KKT and HPR compared to a full enumeration method in Figure 4.4. On the  $x$ -axis, we have  $(\#INGC)/2^{n_b}$ , where  $\#INGC$  is the number of integer no-good cuts used by either KKT or HPR and  $n_b$  is the number of binary variables needed to represent all integer linking variables of the instance. Hence, a full enumeration requires  $2^{n_b}$  many iterations. The plot shows that for roughly 90 % of all instances solved by the HPR approach, it requires less than 20 % of the iterations that are required by a full enumeration. Finally, for the KKT approach, the effect is even more pronounced. For roughly 99 % of all instances solved by the KKT approach, it requires less than 5 % of the iterations that are required by a full enumeration. This shows that the novel approach presented in this work clearly outperforms a full enumeration.

## Optimality Cuts

The optimality cut prevents a solution  $(x, y, \lambda)$  to Problem (4.11) to have a worse upper-level objective function value than the current incumbent  $u$ . However, the bilevel-feasible point computed in Line 11 of Algorithm 4 may violate this constraint. Hence, to test the efficiency of the optimality cut, we

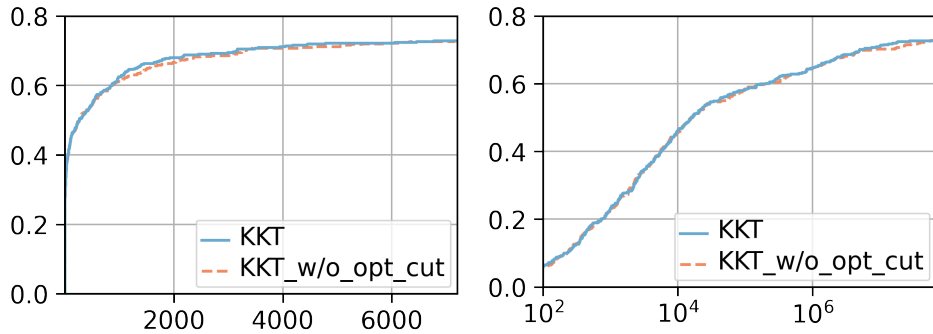


Figure 4.5: ECDFs for the non-trivial instances in our test set. Left: runtimes. Right: node counts.

tested the KKT approach without using it in Problem (4.11). As mentioned in Example 4.4.6, one then has to be careful w.r.t. Lines 6 and 7 of Algorithm 4. To avoid incorrect termination of the algorithm in some special cases, we have to replace Line 7 with

---

```

if  $\ell \leq u$  then
  | Set  $(x^*, y^*) \leftarrow (x^k, y^k)$ .
  Go to Line 25.

```

---

As shown in Figure 4.5, the optimality cut does not lead to a significant improvement w.r.t. runtimes and node counts. This is because the optimality cut only separates points that lead to a large upper-level objective function value, i.e., its absence does not change the overall solution to Problem (4.11), if any exists. However, we decided to keep it, since it also has no negative effect on the overall performance of the algorithm and, for some instances, it may lead to small improvements w.r.t. the runtimes.

## Summary

The numerical study in this section shows that method KKT outperforms HPR in terms of runtimes, node counts, and the number of integer no-good cuts required to solve the instances in our test set. For the instances solved, both KKT and HPR explore only a small fraction of the overall search space, indicating that Algorithm 4 performs significantly better than exhaustive enumeration. Lastly, we observe that the inclusion of the optional optimality cut does not yield a noticeable improvement in the performance of the proposed method.

## Chapter 5

# P-Splits for Bilevel Optimization

This chapter is organized as follows. In Section 5.1, we position the content of this chapter within the overall framework of this dissertation. Section 5.2 introduces the bilevel problem under consideration, along with its KKT reformulation, which we exploit in subsequent sections. In Section 5.3, we discuss the big- $M$  and convex-hull reformulations of the KKT problem. Then, in Section 5.4, we explore the family of so-called  $P$ -split formulations for the KKT complementarity conditions and use them to reformulate the KKT-based problem. Therefore, we apply the ideas of Kronqvist et al. (2022) to our specific setup. Finally, we test the applicability of the derived reformulations in a preliminary numerical study in Section 5.5.

### 5.1 Motivation

In Chapter 4, we proposed a lower and upper bounding scheme to solve MIQP-QP bilevel problems with nonconvex lower levels. This iterative method starts with solving the KKT relaxation of the original problem, in which the arising complementarity conditions are handled using SOS1 formulations. As discussed in Section 2.3, the SOS1 approach and big- $M$  formulations are state-of-the-art technique in bilevel optimization for addressing KKT complementarity conditions of the lower level; see, e.g., Kleinert and Schmidt (2023). Both methods yield a single-level problem that results in an MIQP, which is then solved via branch-and-bound. However, it is well known that the root-node relaxation of this MIQP can be quite weak, leading to weak initial lower bounds on the optimal objective function value of the bilevel problem.

In this chapter, we explore alternative reformulations of the KKT complementarity conditions that may produce tighter continuous relaxations than those derived from SOS1 or big- $M$  formulations. However, this is at

the cost of introducing additional variables and constraints. To this end, we consider bilevel problems in which the lower level is a convex-quadratic problem, a setting that has been addressed in several works; see, e.g., Dempe et al. (2015) or Kleinert et al. (2021a). In this case, the KKT conditions of the lower-level problem are both necessary and sufficient. We represent the nonconvex complementarity conditions as disjunctions of linear expressions and reformulate them using the convex hull of the respective disjunctions.

Additionally, we investigate the family of so-called  $P$ -split formulations. This approach is introduced by Kronqvist et al. (2022) to address general mixed-integer (nonlinear) problems with disjunctive constraints. The computational results in Kronqvist et al. (2022) indicate that  $P$ -split formulations of disjunctions can outperform traditional big- $M$  and convex hull approaches in terms of computation time and number of explored branch-and-bound nodes.

Hence, we apply the  $P$ -split approach to the disjunctions that we obtain from the complementarity conditions in the KKT reformulation. We evaluate and compare the computational performance of the classic big- $M$  formulation, the convex-hull reformulation, and the  $P$ -split formulation. Therefore, we consider both the strength of the root-node relaxation and the overall solver performance in terms of computation times and branch-and-bound node counts.

## 5.2 Problem Statement

We consider the optimistic bilevel optimization problem

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}, y} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \geq 0, \\ & x \geq 0, \\ & y \in S(x), \end{aligned} \tag{5.1}$$

where  $S(x)$  is the solution set of the convex-quadratic  $x$ -parameterized lower-level problem

$$\begin{aligned} \min_{y \in \mathbb{R}^{n_y}} \quad & \frac{1}{2} y^\top Q y + d^\top y \\ \text{s.t.} \quad & Cx + Dy = b, \\ & y \geq 0. \end{aligned} \tag{5.2}$$

We have  $n := n_x + n_y$ ,  $F : \mathbb{Q}^n \rightarrow \mathbb{Q}$ ,  $G : \mathbb{Q}^n \rightarrow \mathbb{Q}^m$ , as well as  $d \in \mathbb{Q}^{n_y}$ ,  $C \in \mathbb{Q}^{l \times n_x}$ ,  $D \in \mathbb{Q}^{l \times n_y}$ , and  $b \in \mathbb{Q}^l$ . Moreover,  $Q \in \mathbb{Q}^{n_y \times n_y}$  is a symmetric and positive semidefinite matrix. The high-point relaxation of Problem (5.1)

is given by

$$\begin{aligned} \min_{x,y} \quad & F(x, y) \\ \text{s.t.} \quad & (x, y) \in \Omega \end{aligned}$$

with  $\Omega = \tilde{\Omega} := \{(x, y) \in \mathbb{R}_{\geq 0}^n : G(x, y) \geq 0, Cx + Dy = b\}$ ; see Section 2.1. We again make Assumptions 2.1.1 and 2.1.2, i.e., we assume boundedness of the HPR and the  $x$ -parameterized lower level for all  $x \in \tilde{\Omega}_x$ . Since all constraints of the lower-level problem are linear, the Abadie constraint qualification holds and the KKT conditions are both necessary and sufficient for the lower-level problem; see Section 2.3. These conditions are given by

$$\begin{aligned} Qy + d - D^\top \lambda - \mu &= 0, \\ Cx + Dy - b &= 0, \\ y, \mu &\geq 0, \\ \mu^\top y &= 0. \end{aligned}$$

Here,  $\lambda \in \mathbb{R}^l$  are the free Lagrange multipliers of the equality constraints  $Cx + Dy = b$  and  $\mu \geq 0$  are the Lagrange multipliers for the constraints  $y \geq 0$ . Written in complementarity form, we get

$$\begin{aligned} 0 \leq y \perp Qy + d - D^\top \lambda &\geq 0, \\ Cx + Dy - b &= 0. \end{aligned}$$

The first part can be seen as a disjunction, so that we obtain

$$\forall i \in [n_y] : \begin{bmatrix} Q_{i \cdot} y + d_i = (D^\top \lambda)_i \\ y_i \geq 0 \end{bmatrix} \vee \begin{bmatrix} y_i = 0 \\ Q_{i \cdot} y + d_i \geq (D^\top \lambda)_i \end{bmatrix}, \quad (5.3)$$

along with the lower-level's equality constraint  $Cx + Dy - b = 0$ . Here,  $Q_{i \cdot}$  denotes the  $i$ th row of the matrix  $Q$ . Hence, the bilevel problem is equivalent to the KKT reformulation

$$\begin{aligned} \min_{x,y,\lambda} \quad & F(x, y) \\ \text{s.t.} \quad & G(x, y) \geq 0, \quad x \geq 0, \\ & Cx + Dy = b, \\ & \forall i \in [n_y] : \begin{bmatrix} Q_{i \cdot} y + d_i = (D^\top \lambda)_i \\ y_i \geq 0 \end{bmatrix} \vee \begin{bmatrix} y_i = 0 \\ Q_{i \cdot} y + d_i \geq (D^\top \lambda)_i \end{bmatrix}. \end{aligned} \quad (5.4)$$

In what follows, we rely on the existence of lower and upper bounds for the dual variables  $\lambda$ . Therefore, we make the following assumption.

**Assumption 5.2.1.** *The dual variables  $\lambda$  in the KKT reformulation (5.4) can be bounded by bilevel-correct values  $\underline{\lambda}_i \leq \lambda_i \leq \bar{\lambda}_i$  for  $i \in [n_y]$ .*

The notion “bilevel correct” states that adding the bounds  $\underline{\lambda}$  and  $\bar{\lambda}$  in the KKT reformulation (5.4) preserves all bilevel-optimal solutions. For LP-LP bilevel problems, Buchheim (2023) showed that such bilevel-correct values exist for the KKT reformulation. However, the resulting bounds can be very weak. While they are computable in polynomial time, finding the tightest possible bounds is NP-hard; see Kleinert et al. (2020). It is an open question whether the results in Buchheim (2023) and Kleinert et al. (2020) can be generalized to our more general setting.

Using Assumptions 5.2.1 and 2.1.1, we introduce bounds  $\underline{\lambda}, \bar{\lambda} \in \mathbb{R}^l$  on the variables  $\lambda$  and upper bounds  $\bar{y} \in \mathbb{R}^{n_y}$  on the variables  $y$ , respectively. With that, we rewrite the KKT reformulation as

$$\min_{x,y,\lambda} F(x,y) \tag{5.5a}$$

$$\text{s.t. } G(x,y) \geq 0, \quad x \geq 0, \tag{5.5b}$$

$$Cx + Dy = b, \tag{5.5c}$$

$$(y, \lambda) \in \mathcal{D}_{i,1} \cup \mathcal{D}_{i,2}, \quad i \in [n_y], \tag{5.5d}$$

with

$$\mathcal{D}_{i,1} := \left\{ y \in [0, \bar{y}], \lambda \in [\underline{\lambda}, \bar{\lambda}] : Q_i \cdot y - \left( D^\top \lambda \right)_i = -d_i, \quad y_i \geq 0 \right\},$$

$$\mathcal{D}_{i,2} := \left\{ y \in [0, \bar{y}], \lambda \in [\underline{\lambda}, \bar{\lambda}] : Q_i \cdot y - \left( D^\top \lambda \right)_i \geq -d_i, \quad y_i = 0 \right\}.$$

The sets  $\mathcal{D}_{i,1}$  and  $\mathcal{D}_{i,2}$  model the feasible region of the two disjuncts in the  $i$ th disjunction in (5.3) along with the introduced bound constraints. Note that if the KKT reformulation (5.5) is feasible, then for every  $i \in [n_y]$ , the sets  $\mathcal{D}_{i,1}$  and  $\mathcal{D}_{i,2}$  cannot be empty simultaneously. Moreover, we assume that we can replace the  $i$ th disjunction in (5.5d) with one of the disjuncts if the other disjunct yields an empty set. Consequently, we make the following assumption.

**Assumption 5.2.2.** *The sets  $\mathcal{D}_{i,1}$  and  $\mathcal{D}_{i,2}$  are not empty for every  $i \in [n_y]$ .*

### 5.3 The Big- $M$ and Convex-Hull Reformulations

The classic way to tackle the disjunctions in (5.5d) is to reformulate them using additional binary variables  $z \in \{0, 1\}^{n_y}$  and big- $M$  constraints; see

Section 2.3. The resulting problem reads

$$\begin{aligned}
& \min_{x,y,\lambda,z} F(x,y) \\
& \text{s.t.} \quad G(x,y) \geq 0, \quad x \geq 0, \\
& \quad \quad \quad Cx + Dy = b, \\
& \quad \quad \quad Qy + d - D^\top \lambda \geq 0, \quad y \geq 0, \\
& \quad \quad \quad Q_i y + d_i - \left(D^\top \lambda\right)_i \leq M_i(1 - z_i), \quad i \in [n_y], \\
& \quad \quad \quad y_i \leq \bar{y}_i z_i, \quad i \in [n_y], \\
& \quad \quad \quad z \in \{0, 1\}^{n_y},
\end{aligned} \tag{5.6}$$

where  $M_i$  is a sufficiently large constant for each  $i \in [n_y]$ . It is known that the continuous relaxation of this big- $M$  reformulation of the disjunctions is not tight, i.e., it is not equivalent to the convex hull of the disjunction; see, e.g., Hooker (2009) or Kronqvist et al. (2022).

We denote the continuous relaxation of the KKT reformulation (5.6), i.e.,  $z \in [0, 1]^{n_y}$ , as the *big- $M$  relaxation* of Problem (5.5). If we solve the big- $M$  relaxation, we get a lower bound on the optimal objective function value of Problem (5.5). Another way to get a relaxation of Problem (5.5) is to use the following well-known theorem by Balas.

**Theorem 5.3.1** (see Theorem 2.1 in Balas (2018)). *Let  $\hat{\mathcal{D}} = \bigcup_{h \in H} \hat{\mathcal{D}}_h$  with sets  $\hat{\mathcal{D}}_h = \{\sigma \in \mathbb{R}^{n_\sigma} : A^h \sigma \geq a_0^h, \sigma \geq 0\}$ , a matrix  $A^h \in \mathbb{R}^{m_h \times n_\sigma}$ , a vector  $a_0^h \in \mathbb{R}^{m_h}$ , and a finite index set  $H$  be given. Define*

$$H^* = \{h \in H : \hat{\mathcal{D}}_h \neq \emptyset\} \quad \text{and} \quad \vartheta := \left( \sigma, \left(\zeta^h\right)_{h \in H^*}, \left(\zeta_0^h\right)_{h \in H^*} \right) \in \mathbb{R}^w,$$

with  $w = n_\sigma(1 + 2H^*)$  and vectors  $\zeta^h \in \mathbb{R}^{n_\sigma}$  and  $\zeta_0^h \in \mathbb{R}^{n_\sigma}$  for  $h \in H^*$ . Moreover, define

$$\mathfrak{B} := \left\{ \vartheta \in \mathbb{R}^w \left| \begin{array}{l} \sigma = \sum_{h \in H^*} \zeta^h, \\ A^h \zeta^h - a_0^h \zeta_0^h \geq 0, \quad h \in H^*, \\ \left(\zeta^h, \zeta_0^h\right) \geq 0, \quad h \in H^*, \\ \sum_{h \in H^*} \zeta_0^h = 1, \quad h \in H^* \end{array} \right. \right\}.$$

If  $\hat{\mathcal{D}} \neq \emptyset$ , then the closure of the convex hull of  $\hat{\mathcal{D}}$  is given by

$$\text{cl}\left(\text{conv}\left(\hat{\mathcal{D}}\right)\right) = \left\{ \sigma \in \mathbb{R}^{n_\sigma} : \exists \zeta^h, \zeta_0^h \in \mathbb{R}^{n_\sigma}, h \in H^* : \vartheta \in \mathfrak{B} \right\}.$$

We use Theorem 5.3.1 to get a description of the convex hull of each disjunction in (5.5d) for all  $i \in [n_y]$ . By Assumption 5.2.2, we have two non-empty disjuncts  $\mathcal{D}_{i,1}$  and  $\mathcal{D}_{i,2}$  for the  $i$ th disjunction in (5.5d), i.e.,  $H = H^*$ . Hence, we introduce two vectors  $v^{i,1}, v^{i,2}$  to represent the lower-level variables  $y$  and two vectors  $\gamma^{i,1}, \gamma^{i,2}$  to represent the dual variables  $\lambda$ , respectively. Furthermore, we introduce the vectors  $z$  and  $\tilde{z}$ . With Theorem 5.3.1, the convex hull of  $\mathcal{D}_{i,1} \cup \mathcal{D}_{i,2}$  for some  $i \in [n_y]$  is given by all points  $(y, \lambda) \in \mathbb{R}^{n_y+l}$  that satisfy the conditions

$$\begin{aligned}
y &= v^{i,1} + v^{i,2}, \quad \lambda = \gamma^{i,1} + \gamma^{i,2}, \\
Q_i v^{i,1} - \left( D^\top \gamma^{i,1} \right)_i &= -d_i z_i, \\
Q_i v^{i,2} - \left( D^\top \gamma^{i,2} \right)_i &\geq -d_i \tilde{z}_i, \quad v_i^{i,2} = 0, \\
0 \leq v_j^{i,1} &\leq \bar{y}_j z_i, \quad j \in [n_y], \\
0 \leq v_j^{i,2} &\leq \bar{y}_j (1 - z_i), \quad j \in [n_y], \\
\lambda_i z_i &\leq \gamma_j^{i,1} \leq \bar{\lambda}_i z_i, \quad j \in [n_y], \\
\lambda_i (1 - z_i) &\leq \gamma_j^{i,2} \leq \bar{\lambda}_i (1 - z_i), \quad j \in [n_y], \\
z_i + \tilde{z}_i &= 1, \quad z_i, \tilde{z}_i \geq 0.
\end{aligned} \tag{5.7}$$

We simplify (5.7) by setting  $\tilde{z}_i = 1 - z_i$  for all  $i \in [n_y]$ . Then, we replace the disjunctions in (5.5d) by the conditions in (5.7) for every  $i \in [n_y]$  and apply binary constraints to the variables  $z$ . The resulting problem reads

$$\min_{x, y, \lambda, \pi^{\text{ch}}} F(x, y) \tag{5.8a}$$

$$\text{s.t. } G(x, y) \geq 0, \quad x \geq 0, \tag{5.8b}$$

$$Cx + Dy = b, \quad y \geq 0, \tag{5.8c}$$

$$y = v^{i,1} + v^{i,2}, \quad \lambda = \gamma^{i,1} + \gamma^{i,2}, \quad i \in [n_y], \tag{5.8d}$$

$$0 \leq v_j^{i,1} \leq \bar{y}_j z_i, \quad i \in [n_y], \quad j \in [n_y], \tag{5.8e}$$

$$0 \leq v_j^{i,2} \leq \bar{y}_j (1 - z_i), \quad i \in [n_y], \quad j \in [n_y], \tag{5.8f}$$

$$v_i^{i,2} = 0, \quad i \in [n_y], \tag{5.8g}$$

$$\lambda_i z_i \leq \gamma_j^{i,1} \leq \bar{\lambda}_i z_i, \quad i \in [n_y], \quad j \in [n_y], \tag{5.8h}$$

$$\lambda_i (1 - z_i) \leq \gamma_j^{i,2} \leq \bar{\lambda}_i (1 - z_i), \quad i \in [n_y], \quad j \in [n_y], \tag{5.8i}$$

$$Q_i v^{i,1} - \left( D^\top \gamma^{i,1} \right)_i = -d_i z_i, \quad i \in [n_y], \tag{5.8j}$$

$$Q_i v^{i,2} - \left( D^\top \gamma^{i,2} \right)_i \geq -d_i (1 - z_i), \quad i \in [n_y], \tag{5.8k}$$

$$z_i \in \{0, 1\}, \quad i \in [n_y], \tag{5.8l}$$

with  $\pi^{\text{ch}} = ((v^{i,1}, v^{i,2}, \gamma^{i,1}, \gamma^{i,2})_{i \in [n_y]}, z)$ . We call Problem (5.8) the *convex-hull reformulation* of Problem (5.5) and refer to its continuous relaxation,

i.e.,  $z_i \in [0, 1]$  for  $i \in [n_y]$ , as *convex-hull relaxation*.

If the variable  $z_i = 0$  for some  $i \in [n_y]$ , then the vectors  $v^{i,1}$  and  $\gamma^{i,1}$  are set to zero; see (5.8e) and (5.8h). Consequently, Condition (5.8j) is trivially satisfied and Constraints (5.8d), (5.8f), (5.8g), (5.8i), and (5.8k) model the feasible region of  $\mathcal{D}_{i,2}$ . Conversely, if  $z_i = 1$  for some  $i \in [n_y]$ , then the vectors  $v^{i,2}$  and  $\gamma^{i,2}$  are set to zero due to Constraints (5.8f) and (5.8i). Furthermore, Constraint (5.8k) is trivially satisfied and Constraints (5.8d), (5.8e), (5.8h), and (5.8j) model the feasible region of  $\mathcal{D}_{i,1}$ . Therefore, Problem (5.8) is a reformulation of Problem (5.5).

Note that the existence of bounds  $\lambda$ ,  $\bar{\lambda}$ , and  $\bar{y}$  plays an important role in Problem (5.8) as without them, we can only obtain a relaxation of Problem (5.5) rather than a reformulation.

## 5.4 The $P$ -Split Reformulation

Another idea to further reformulate the KKT reformulation (5.5) consists of using the  $P$ -split formulations introduced by Kronqvist et al. (2022). They represent a hierarchy of formulations for a set of disjunctions. Their lowest level is equivalent to the big- $M$  reformulation and their highest level is equivalent to the disjunction's convex hull.

### 5.4.1 General Idea in Kronqvist et al. (2022)

We now briefly sketch the idea of the  $P$ -split techniques for general disjunctions. For a more detailed discussion and examples on this topic we refer to Kronqvist et al. (2022). In their work, the authors consider disjunctions of the form

$$\bigvee_{h \in H} \left[ \begin{array}{l} g_h(\sigma) \leq b_h \\ \sigma \in \Sigma \end{array} \right] \quad (5.9)$$

where  $\Sigma \subset \mathbb{R}^{n_\sigma}$  is a convex and compact set and  $H$  is a finite index set. Additionally, the functions  $g_h : \mathbb{R}^{n_\sigma} \rightarrow \mathbb{R}$  are bounded over  $\Sigma$  and additively separable, i.e.,  $g_h(\sigma) = \sum_{i=1}^{n_\sigma} r_{i,h}(\sigma_i)$  where  $r_{i,h} : \mathbb{R} \rightarrow \mathbb{R}$  are convex functions. Note that the latter property also always holds in our setting. The variables  $\sigma$  are partitioned into  $P$  sets with corresponding index sets  $I_1, \dots, I_P$ , such that all variables are included in exactly one set. Next, auxiliary vectors  $\alpha^h \in \mathbb{R}^P$  for  $h \in H$  are introduced to rewrite (5.9) as

$$\bigvee_{h \in H} \left[ \begin{array}{l} \sum_{k=1}^P \alpha_k^h \leq b_h, \\ \sum_{i \in I_1} r_{i,h}(\sigma_i) \leq \alpha_1^h \\ \vdots \\ \sum_{i \in I_P} r_{i,h}(\sigma_i) \leq \alpha_P^h, \\ \sigma \in \Sigma, \\ \alpha^h \in \mathbb{R}^P, h \in H \end{array} \right]. \quad (5.10)$$

The constraints  $\sum_{i \in I_k} r_{i,h}(\sigma_i) \leq \alpha_k^h$  for  $k \in [P]$  and  $h \in H$  can be strengthened to equality constraints if the left-hand side is affine for  $k \in [P]$ . Due to the boundedness of the disjuncts, the variables  $\alpha_k^h$  can be bounded by

$$\underline{\alpha}_k^h = \min_{\sigma \in \Sigma} \sum_{i \in I_k} h_{i,k}(\sigma_i), \quad \bar{\alpha}_k^h = \max_{\sigma \in \Sigma} \sum_{i \in I_k} h_{i,k}(\sigma_i).$$

The bound constraints are added to (5.10) and the disjunction is relaxed by moving all constraints containing  $\sigma$  out of it. Based on this, a so-called *P-split representation* of Disjunction (5.9) is given by

$$\begin{aligned} \bigvee_{h \in H} \left[ \begin{array}{l} \sum_{k=1}^P \alpha_k^h \leq b_h, \\ \underline{\alpha}_k^h \leq \alpha_k^h \leq \bar{\alpha}_k^h, \quad k \in [P] \end{array} \right], \\ \sum_{i \in I_k} r_{i,h}(\sigma_i) \leq \alpha_k^h, \quad k \in [P], \quad h \in H \\ \sigma \in \Sigma, \quad \alpha^h \in \mathbb{R}^P, \quad h \in H; \end{aligned} \quad (5.11)$$

see Definition 2 in Kronqvist et al. (2022). Finally, a *P-split reformulation* of Disjunction (5.9) is obtained by applying a convex-hull reformulation on (5.11) using Theorem 5.3.1.

#### 5.4.2 Application to KKT Complementarity Conditions

The KKT complementarity conditions in Problem (5.5) are restated as two-term disjunctions ( $H = \{1, 2\}$ ) where the disjuncts depend on  $y$  and  $\lambda$ . To apply the *P-split* approach discussed in Section 5.4.1 on each disjunction, we do the following. For the  $i$ th disjunction with  $i \in [n_y]$ , the variables  $y$  and  $\lambda$  are split into  $\hat{P}_i \in [n_y]$  and  $\tilde{P}_i \in [l]$  many subsets, respectively. To do so, we consider the index sets  $I_1^i, \dots, I_{\hat{P}_i}^i$  and  $J_1^i, \dots, J_{\tilde{P}_i}^i$  such that

$$I_k^i \cap I_{k'}^i = \emptyset \quad \text{for all } k, k' \in [\hat{P}_i] := \{1, \dots, \hat{P}_i\} \text{ with } k \neq k', \quad (5.12)$$

$$J_k^i \cap J_{k'}^i = \emptyset \quad \text{for all } k, k' \in [\tilde{P}_i] := \{1, \dots, \tilde{P}_i\} \text{ with } k \neq k', \quad (5.13)$$

and

$$\bigcup_{k \in [\hat{P}_i]} I_k^i = [n_y] \setminus \{i\}, \quad \bigcup_{k \in [\tilde{P}_i]} J_k^i = [l] \quad (5.14)$$

holds. For the sake of simplicity, we make the following assumption.

**Assumption 5.4.1.** *Let  $P \in \{1, \dots, \kappa\}$  for  $\kappa = \min\{n_y, l\}$ . For every index  $i \in [n_y]$ , we use the same number of partitions for the  $y$  and  $\lambda$  variables, i.e.,  $\hat{P}_i := P$  and  $\tilde{P}_i := P$ .*

**Example 5.4.2.** *Let  $n_y = 9$ ,  $l = 8$ , and  $P = 3$ . Then, an exemplary partition of  $[n_y]$  and  $[l]$  for  $i = 5$  that satisfies (5.12)–(5.14) is given by*

$$I^5 = \{\{1, 2\}, \{3, 4, 6, 7, 8\}, \{9\}\}, \quad J^5 = \{\{1\}, \{2\}, \{3, 4, 5, 6, 7, 8\}\}.$$

As in Section 5.4.1, we introduce auxiliary vectors  $\alpha^i \in \mathbb{R}^P$  and  $\beta^i \in \mathbb{R}^P$  to re-write the  $i$ th disjunction in (5.3) as

$$\left[ \begin{array}{l} Q_{ii}y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) = -d_i, \\ y_i \geq 0, \\ \alpha_1^i = \sum_{j \in I_1^i} Q_{ij}y_j, \\ \vdots \\ \alpha_P^i = \sum_{j \in I_P^i} Q_{ij}y_j, \\ \beta_1^i = \sum_{j \in J_1^i} D_{ji}\lambda_j, \\ \vdots \\ \beta_P^i = \sum_{j \in J_P^i} D_{ji}\lambda_j \end{array} \right] \vee \left[ \begin{array}{l} y_i = 0, \\ Q_{ii}y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) \geq -d_i, \\ \alpha_1^i = \sum_{j \in I_1^i} Q_{ij}y_j, \\ \vdots \\ \alpha_P^i = \sum_{j \in I_P^i} Q_{ij}y_j, \\ \beta_1^i = \sum_{j \in J_1^i} D_{ji}\lambda_j, \\ \vdots \\ \beta_P^i = \sum_{j \in J_P^i} D_{ji}\lambda_j \end{array} \right].$$

Note that the terms in both disjuncts in (5.3) for a fixed  $i \in [n_y]$  are the same. Hence, the same variables  $\alpha_k^i$  and  $\beta_k^i$  can be used to represent them; see Kronqvist et al. (2022). The superscript index  $i$  is therefore used to refer to the corresponding disjunction rather than to distinguish between the disjuncts as it is done with index  $h$  in Section 5.4.1.

Using Assumptions 2.1.2 and 5.2.1, we define finite bounds on the auxiliary variables for  $k \in [P]$  via

$$\alpha_k^i := \min_{y \in [0, \bar{y}]} \left\{ \sum_{j \in I_k^i} Q_{ij}y_j \right\}, \quad \bar{\alpha}_k^i := \max_{y \in [0, \bar{y}]} \left\{ \sum_{j \in I_k^i} Q_{ij}y_j \right\},$$

$$\underline{\beta}_k^i := \min_{\lambda \in [\underline{\lambda}, \bar{\lambda}]} \left\{ \sum_{j \in J_k^i} D_{ji}\lambda_j \right\}, \quad \bar{\beta}_k^i := \max_{\lambda \in [\underline{\lambda}, \bar{\lambda}]} \left\{ \sum_{j \in J_k^i} D_{ji}\lambda_j \right\}.$$

By adding the bound constraints and moving the constraints depending on  $y$  and  $\lambda$  out of the disjunction, we get

$$\left[ \begin{array}{l} Q_{ii}y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) = -d_i, \\ y_i \geq 0, \\ \alpha_k^i \leq \alpha_k^i \leq \bar{\alpha}_k^i, \quad k \in [P], \\ \underline{\beta}_k^i \leq \beta_k^i \leq \bar{\beta}_k^i, \quad k \in [P] \end{array} \right] \vee \left[ \begin{array}{l} y_i = 0, \\ Q_{ii}y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) \geq -d_i, \\ \alpha_k^i \leq \alpha_k^i \leq \bar{\alpha}_k^i, \quad k \in [P], \\ \underline{\beta}_k^i \leq \beta_k^i \leq \bar{\beta}_k^i, \quad k \in [P] \end{array} \right],$$

$$\alpha_k^i = \sum_{j \in I_k^i} Q_{ij}y_j, \quad \beta_k^i = \sum_{j \in J_k^i} D_{ji}\lambda_j, \quad k \in [P].$$

We call (5.15) the  $i$ - $P$ -split representation of the  $i$ th disjunction in (5.5d). Note that one could equivalently state Problem (5.15) by defining the bound constraints globally. However, we keep them in each disjunct to reformulate

Problem (5.5) as

$$\begin{aligned}
& \min_{x,y,\lambda,\alpha,\beta} && F(x,y) \\
& \text{s.t.} && G(x,y) \geq 0, \quad x \geq 0, \\
& && Cx + Dy = b, \\
& && \alpha_k^i = \sum_{j \in I_k^i} Q_{ij} y_j, \quad \beta_k^i = \sum_{j \in J_k^i} D_{ji} \lambda_j, \quad k \in [P], \\
& && (y, \lambda) \in \tilde{\mathcal{D}}_{i,1} \cup \tilde{\mathcal{D}}_{i,2}, \quad i \in [n_y],
\end{aligned} \tag{5.15}$$

with  $\alpha = (\alpha^i)_{i \in [n_y]}$ ,  $\beta = (\beta^i)_{i \in [n_y]}$ , and

$$\begin{aligned}
\tilde{\mathcal{D}}_{i,1} &:= \left\{ (y_i, \alpha^i, \beta^i) \in \mathcal{B}_i : Q_{ii} y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) = -d_i, y_i \geq 0 \right\}, \\
\tilde{\mathcal{D}}_{i,2} &:= \left\{ (y_i, \alpha^i, \beta^i) \in \mathcal{B}_i : Q_{ii} y_i + \sum_{k=1}^P (\alpha_k^i - \beta_k^i) \geq -d_i, y_i = 0 \right\}, \\
\mathcal{B}_i &:= [0, \bar{y}_i] \times [\underline{\alpha}^i, \bar{\alpha}^i] \times [\underline{\beta}^i, \bar{\beta}^i] \subset \mathbb{R}^{1+2P}.
\end{aligned}$$

We assume that the sets  $\tilde{\mathcal{D}}_{i,1}$  and  $\tilde{\mathcal{D}}_{i,2}$  are nonempty; see Assumption 5.2.2. For  $i \in [n_y]$ , we apply the extended convex hull formulation by Balas (1998) to the disjunctions  $\tilde{\mathcal{D}}_{i,1} \cup \tilde{\mathcal{D}}_{i,2}$ ; see Section 5.3. We introduce two auxiliary variables  $v_i^1, v_i^2 \in \mathbb{R}$  to represent  $y_i$ , two auxiliary vectors  $w^{i,1}, w^{i,2} \in \mathbb{R}^P$  for each  $\alpha^i$ , and two auxiliary vectors  $\tau^{i,1}, \tau^{i,2} \in \mathbb{R}^P$  for each  $\beta^i$ . As stated Theorem 5.3.1, the convex hull of  $\tilde{\mathcal{D}}_{i,1} \cup \tilde{\mathcal{D}}_{i,2}$  is given by the set of points  $(y_i, \alpha^i, \beta^i) \in \mathbb{R}^{1+2P}$  that satisfy the conditions

$$y_i = v_i^1 + v_i^2, \tag{5.16a}$$

$$\alpha_k^i = w_k^{i,1} + w_k^{i,2}, \quad k \in [P], \tag{5.16b}$$

$$\beta_k^i = \tau_k^{i,1} + \tau_k^{i,2}, \quad k \in [P], \tag{5.16c}$$

$$Q_{ii} v_i^1 + \sum_{k=1}^P (w_k^{i,1} - \tau_k^{i,1}) = -d_i z_i, \tag{5.16d}$$

$$Q_{ii} v_i^2 + \sum_{k=1}^P (w_k^{i,2} - \tau_k^{i,2}) \geq -d_i (1 - z_i), \tag{5.16e}$$

$$v_i^1 \geq 0, \quad v_i^2 = 0, \tag{5.16f}$$

$$\underline{\alpha}_k^i z_i \leq w_k^{i,1} \leq \bar{\alpha}_k^i z_i, \quad k \in [P], \tag{5.16g}$$

$$\underline{\beta}_k^i z_i \leq \tau_k^{i,1} \leq \bar{\beta}_k^i z_i, \quad k \in [P], \tag{5.16h}$$

$$\underline{\alpha}_k^i (1 - z_i) \leq w_k^{i,2} \leq \bar{\alpha}_k^i (1 - z_i), \quad k \in [P], \tag{5.16i}$$

$$\underline{\beta}_k^i (1 - z_i) \leq \tau_k^{i,2} \leq \bar{\beta}_k^i (1 - z_i), \quad k \in [P], \tag{5.16j}$$

$$z_i \in [0, 1]. \tag{5.16k}$$

Note that we can trivially eliminate the variables  $v_i^1$  and  $v_i^2$  for  $i \in [n_y]$ . We use (5.15) to restate (5.16b) and (5.16c) as

$$\sum_{j \in I_k^i} Q_{ij} y_j = w_k^{i,1} + w_k^{i,2}, \quad \sum_{j \in J_k^i} D_{ji} \lambda_j = \tau_k^{i,1} + \tau_k^{i,2}, \quad k \in [P].$$

With that, the conditions in (5.16) model the convex hull of the feasible region of the  $i$ - $P$ -split representation (5.15). Now, we add binary constraints on the  $z$  variables to obtain the convex-hull reformulation of Problem (5.15), i.e.,

$$\min_{x,y,\lambda,\pi^P} F(x, y) \tag{5.17a}$$

$$\text{s.t. } G(x, y) \geq 0, \quad x \geq 0, \tag{5.17b}$$

$$Cx + Dy = b, \quad y \geq 0, \tag{5.17c}$$

$$y_i \leq \bar{y}_i z_i, \quad i \in [n_y], \tag{5.17d}$$

$$\sum_{j \in I_k^i} Q_{ij} y_j = w_k^{i,1} + w_k^{i,2}, \quad k \in [P], \quad i \in [n_y], \tag{5.17e}$$

$$\sum_{j \in J_k^i} D_{ji} \lambda_j = \tau_k^{i,1} + \tau_k^{i,2}, \quad k \in [P], \quad i \in [n_y], \tag{5.17f}$$

$$\underline{\alpha}_k^i z_i \leq w_k^{i,1} \leq \bar{\alpha}_k^i z_i, \quad k \in [P], \quad i \in [n_y], \tag{5.17g}$$

$$\underline{\beta}_k^i z_i \leq \tau_k^{i,1} \leq \bar{\beta}_k^i z_i, \quad k \in [P], \quad i \in [n_y], \tag{5.17h}$$

$$\underline{\alpha}_k^i (1 - z_i) \leq w_k^{i,2} \leq \bar{\alpha}_k^i (1 - z_i), \quad k \in [P], \quad i \in [n_y], \tag{5.17i}$$

$$\underline{\beta}_k^i (1 - z_i) \leq \tau_k^{i,2} \leq \bar{\beta}_k^i (1 - z_i), \quad k \in [P], \quad i \in [n_y], \tag{5.17j}$$

$$Q_{ii} y_i + \sum_{k=1}^P (w_k^{i,1} - \tau_k^{i,1}) = -d_i z_i, \quad i \in [n_y], \tag{5.17k}$$

$$\sum_{k=1}^P (w_k^{i,2} - \tau_k^{i,2}) \geq -d_i (1 - z_i), \quad i \in [n_y], \tag{5.17l}$$

$$z_i \in \{0, 1\}, \quad i \in [n_y], \tag{5.17m}$$

with  $\pi^P = ((w^{i,1}, w^{i,2}, \tau^{i,1}, \tau^{i,2})_{i \in [n_y]}, z)$ . The model in (5.17) is equivalent to Problem (5.15) for the same reason that the convex-hull reformulation (5.8) is equivalent to the KKT reformulation (5.5); see Section 5.3. Since (5.15) itself reformulates (5.5), it follows that (5.17) is also a reformulation of (5.5). We refer to (5.17) as the  $P$ -split reformulation of Problem (5.5) and we denote its continuous relaxation, i.e.,  $z_i \in [0, 1]$  for  $i \in [n_y]$ , with  $P$ -split relaxation. From Kronqvist et al. (2022), we obtain the following results that demonstrate the relationship between the  $P$ -split reformulations and the big- $M$  and convex-hull reformulations w.r.t. their continuous relaxations.

**Theorem 5.4.3** (see Kronqvist et al. (2022)). Let  $\Upsilon^M$  and  $\Upsilon^{ch}$  denote the projections of the feasible regions of the big- $M$  and convex-hull relaxations onto the  $(y, \lambda)$ -space, respectively. Moreover, let  $\Upsilon^P$  denote the projections of the feasible regions of the  $P$ -split relaxations onto the  $(y, \lambda)$ -space for  $P \in \{1, \dots, \kappa\}$  with  $\kappa = \min\{n_y, l\}$ . Then, it holds

$$\Upsilon^{ch} \subseteq \Upsilon^\kappa \subseteq \Upsilon^{\kappa-1} \subseteq \dots \subseteq \Upsilon^2 \subseteq \Upsilon^1 = \Upsilon^M.$$

Moreover, if  $n_y = l$ , then  $\Upsilon^\kappa = \Upsilon^{ch}$ .

We see that the  $P$ -split reformulations (5.17) lie between the big- $M$  and the convex-hull reformulations in terms of relaxation strength. The number of partitions, i.e., the value of  $P$ , can be used to balance the trade-off between relaxation strength and size of the resulting reformulation; see Kronqvist et al. (2022). Note that it is not guaranteed that all of the inclusions in Theorem 5.4.3 are strict. However, if  $\Upsilon^{ch} \subset \Upsilon^M$ , then at least one inclusion  $\Upsilon^P \subset \Upsilon^{P-1}$  is strict for some  $P \in \{2, \dots, \kappa\}$ . In the following section, we test the applicability of  $P$ -split reformulations for different values of  $P$ .

## 5.5 Numerical Results

In this section, we provide a preliminary numerical study on the performance of the SOS1, big- $M$ , convex hull, and  $P$ -split reformulations of the KKT reformulation (5.5) in terms of computation times and node counts. Therefore, we compare the following approaches:

**SOS1:** We solve the KKT reformulation (5.5) for which the complementarity conditions are rephrased as (see Section 2.3)

$$e_j = Qy + d - D^\top \lambda, \quad \{e_j, \lambda_j\} \text{ is SOS1}, \quad j \in [n_y];$$

**Big- $M$ :** We solve the big- $M$  reformulation (5.6);

**C-Hull:** We solve the convex-hull reformulation (5.8);

**$P$ -Split:** We solve the  $P$ -split reformulation (5.17) for  $P \in \{2, 3, 4, 5\}$ .

Note that SOS1 is the only exact approach while the correctness of the other methods depends on the validity of the big- $M$  values used. However, in the numerical tests we observed that all of the methods returned the same solution values on the considered instances. The implementation of all models is publicly available at <https://github.com/AndreasHorlaender/P-split-s-for-QP-Lower-Levels>.

### 5.5.1 Hardware and Software Setup

We implemented and solved all models using Python 3.9.7 and Gurobi 11.0.2. In our computational study we set the feasibility tolerance for the constraints to  $10^{-6}$  and the integer feasibility tolerance to  $10^{-8}$ . Moreover, we set the parameter `NumericFocus` in Gurobi to the maximum value of three to avoid numerical issues. We chose the big- $M$  constant to be  $10^5$  and applied the bounds  $\lambda \in [-10^5, 10^5]^{n_y}$  to compute  $\underline{\beta}$  and  $\bar{\beta}$ . The upper bounds on the  $y$  variables are obtained from the instances. We computed the index sets  $I^i$  and  $J^i$  in the following way. For every  $i \in [n_y]$ , we kept the order of the indices in the index sets  $[n_y] \setminus \{i\}$  and  $[l]$ . Then, we partitioned the sets  $[n_y] \setminus \{i\}$  in  $P$  many subsets, where the first  $(n_y - 1 \bmod P)$ -many subsets contain  $(\lfloor n_y/P \rfloor + 1)$ -many indices and the others contain  $\lfloor n_y/P \rfloor$ -many indices. The partitioning for the set  $[l]$  is the same for every  $i \in [n_y]$ . The first  $(l \bmod P)$ -many subsets contain  $(\lfloor l/P \rfloor + 1)$ -many indices and the others contain  $\lfloor l/P \rfloor$ -many indices. To give an example, for  $[n_y] = \{1, 2, 3, 4\}$  and  $P = 2$ , we get

$$\begin{aligned} I^1 &= \{\{2, 3\}, \{4\}\}, & I^2 &= \{\{1, 3\}, \{4\}\}, & I^3 &= \{\{1, 2\}, \{4\}\}, \\ I^4 &= \{\{1, 2\}, \{3\}\}, & J^i &= \{\{1, 2\}, \{3, 4\}\}, & i &\in \{1, 2, 3, 4\}. \end{aligned}$$

All computations have been executed on the high performance cluster “Elwetritsch” at the TU Kaiserslautern, which is part of the “Alliance of High Performance Computing Rheinland-Pfalz” (AHRP). We used a single Intel XEON SP 6126 core with 2.6 GHz and a maximum of 30 GB RAM. Our time limit was 2 hours.

### 5.5.2 Test Instances

As for the previous chapters, we use MILP-MILP problem instances from the BOBILib and add quadratic matrices  $Q$  for the lower-level objective function using the MATLAB function of Kleinert and Schmidt (2021); see Chapter 3. The instance sets are listed in Table 5.1. Furthermore, we relax all integrality constraints which leads to instances of the form

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}, \hat{y}} \quad & c_x^\top x + \hat{c}_y^\top \hat{y} \\ \text{s.t.} \quad & Ax + \hat{B}\hat{y} - a \geq 0, \\ & x \geq 0, \\ & \hat{y} \in S(x), \end{aligned} \tag{5.18}$$

where  $S(x)$  is the solution set of the  $x$ -parameterized lower-level problem

$$\min_{\hat{y} \in \mathbb{R}^{\hat{n}_y}} \frac{1}{2} \hat{y}^\top \hat{Q} \hat{y} + \hat{d}^\top \hat{y} \quad (5.19a)$$

$$\text{s.t. } Cx + \hat{D}\hat{y} - b \geq 0, \quad (5.19b)$$

$$\hat{y} \geq 0. \quad (5.19c)$$

To fit into the framework of (5.2), we introduce slack variables  $s \in \mathbb{R}^l$  for the lower-level constraints (5.19b) to restate the above problem as

$$\begin{aligned} \min_{x \in \mathbb{R}^{n_x}, y} \quad & c_x^\top x + c_y^\top y \\ \text{s.t.} \quad & Ax + By - a \geq 0, \\ & x \geq 0, \\ & y \in \arg \min_{\bar{y} \in \mathbb{R}^{n_y}} \left\{ \frac{1}{2} \bar{y}^\top Q \bar{y} + d^\top \bar{y} : Cx + D\bar{y} - b = 0, \bar{y} \geq 0 \right\}, \end{aligned}$$

with  $n_y := \hat{n}_y + l$  and

$$\begin{aligned} y &:= \begin{pmatrix} \hat{y} \\ s \end{pmatrix} \in \mathbb{R}^{n_y}, \quad c_y := \begin{pmatrix} \hat{c}_y \\ 0 \end{pmatrix} \in \mathbb{R}^{n_y}, \quad B := [\hat{B} \quad 0] \in \mathbb{R}^{m \times n_y}, \\ d &:= \begin{pmatrix} \hat{d} \\ 0 \end{pmatrix} \in \mathbb{R}^{n_y}, \quad Q := \begin{bmatrix} \hat{Q} & 0 \\ 0 & 0 \end{bmatrix} \in \mathbb{R}^{n_y \times n_y}, \quad \text{and } D := [\hat{D} \quad -I] \in \mathbb{R}^{l \times n_y}. \end{aligned}$$

Here,  $I \in \mathbb{R}^{l \times l}$  is the identity matrix. The upper bounds for the slack variables  $s_i$  for  $i \in [l]$  that are embedded in  $y$  are computed as the maximal possible slack of the  $i$ th lower-level constraint in (5.19b).

As in the numerical studies presented in previous chapters, we categorize instances as “trivial,” “solvable,” or “time limit” and discuss the results for the “solvable” instances, i.e., they form our test set. An overview of the classification of the instances per instance set is given in Table 5.1. We use SOS1 as our benchmark. We label an instance as “trivial”, if SOS1 solves it within 1 second, and as “time limit”, if SOS1 fails to solve it within the 2 hour time limit. This differs from earlier chapters, where an instance was labeled “trivial” or “time limit” only if all compared methods solved it within 1 second or failed to solve it within 2 hours, respectively. We do this because the majority of “trivial” instances in Table 5.1 cannot be solved by the other methods within 1 second. Since this affects hundreds of instances, including them as “solvable” would considerably obfuscate the numerical results. Furthermore, based on the outcomes of the “solvable” instances, we do not expect that many of the “time limit” cases would be successfully solved by the alternative methods.

We label 30 instances in XuLarge as “time limit” for which the square matrix  $Q$  cannot be computed within the time limit due to their size. In addition, we exclude 14 infeasible instances in QBMKP and 72 infeasible instances in Generalized as “trivial” in Table 5.1.

Table 5.1: Overview of the test instances per instance class. The first group is the subset of instance test sets that we use in our numerical study.

Instance class	Total	Solvable (in %)	Time limit (in %)	Trivial (in %)
Inter-Assig	24	24 (100.00 %)	0 (0.00 %)	0 (0.00 %)
Inter-Clique	80	38 (47.50 %)	30 (37.50 %)	12 (15.00 %)
Inter-KP	99	40 (40.40 %)	0 (0.00 %)	59 (59.60 %)
KP	450	27 (6.00 %)	351 (78.00 %)	72 (16.00 %)
QBMKP	200	9 (4.50 %)	0 (0.00 %)	191 (95.50 %)
OR	810	107 (13.21 %)	626 (77.28 %)	77 (9.51 %)
XuLarge	60	30 (50.00 %)	30 (50.00 %)	0 (0.00 %)
XuWang	100	50 (50.00 %)	0 (0.00 %)	50 (50.00 %)
Clique	60	0 (0.00 %)	54 (90.00 %)	6 (10.00 %)
Denegre	50	0 (0.00 %)	0 (0.00 %)	50 (100.00 %)
Generalized	90	0 (0.00 %)	18 (20.00 %)	72 (80.00 %)
Inter-Fire	72	0 (0.00 %)	72 (100.00 %)	0 (0.00 %)
$\Sigma$	2095	325 (15.51 %)	1181 (56.37 %)	589 (28.11 %)

### 5.5.3 Discussion of the Results

The empirical cumulative distribution functions (ECDFs) w.r.t. computation times and node counts for different  $P$ -split reformulations (5.17) are given in Figure 5.1. For the computation times, we exclude the time to compute the bounds  $\bar{y}$ ,  $\underline{\alpha}^i$ ,  $\bar{\alpha}^i$ ,  $\underline{\beta}^i$ , and  $\bar{\beta}^i$  for  $i \in [n_y]$  to focus purely on the solution time of the respective model. Figure 5.1 shows that **2-Split** dominates  $P$ -**Split** for  $P \in \{3, 4, 5\}$  both in terms of computation times and node counts. Moreover, **3-Split** slightly outperforms **5-Split**. This is likely due to the additional variables and constraints that are added to the  $P$ -split reformulation (5.17) for increasing value of  $P$ . Furthermore, it can be seen that **3-Split** performs slightly better than **4-Split** on the majority of instances, but this trend reverses for the most challenging ones. One of the possible explanations for this behavior could be that **Gurobi** finds better ways to exploit the structure of few large-sized instances for the 4-split reformulation than for the other tested  $P$ -split reformulations. This leads to a small decrease in the number of nodes that the solver requires to solve the problems. Consequently, we see an effect in the computation times for the large instances, since each node needs a considerable amount of time to be solved.

Since **2-Split** performs best, we compare it against **SOS1**, **Big- $M$** , and **C-Hull**. The corresponding ECDFs w.r.t. computation times and node counts are given in Figure 5.2. It can be seen that **SOS1** outperforms the other approaches for the computation times. This outcome is somewhat unexpected, as in a recent study of Kleinert and Schmidt (2023), the presented numerical results indicated an advantage of the **big- $M$**  reformulation over the **SOS1**

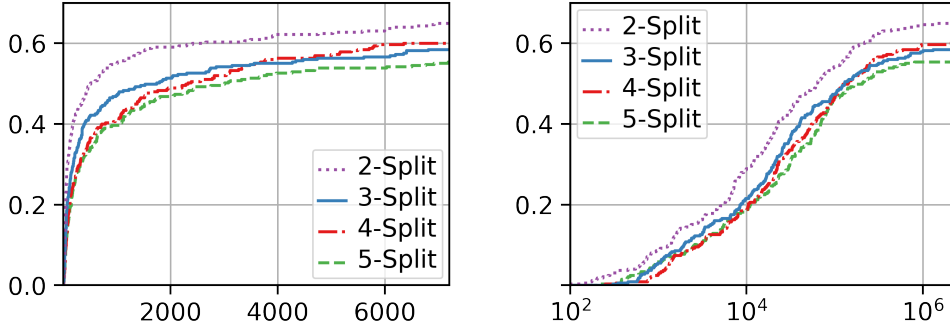


Figure 5.1: ECDFs of the instances in our test set. Left: computation time for  $P$ -Split for  $P \in \{2, 3, 4, 5\}$ . Right: node counts.

approach. This contrast can be due to differences in the test sets, the problem formulations, the choices for the tolerances in combination with the big- $M$  value, or the use of different solvers.

For the  $P$ -split reformulations, we see in Figure 5.2 that 2-Split performs worse w.r.t. computation times and node counts compared to SOS1 or Big- $M$  but outperforms C-Hull. A possible explanation for the comparably bad performance of the  $P$ -split reformulations lies in the size of the considered bounds for  $y$  and  $\lambda$ ; see Section 5.5.1. Large values for  $\lambda$ ,  $\bar{\lambda}$ , and  $\bar{y}$  lead to weak bounds for the auxiliary variables  $\alpha$  and  $\beta$  and thus to weak  $P$ -split relaxations; see Kronqvist et al. (2022). Moreover, both the  $P$ -split and the convex-hull reformulations require the introduction of a considerable number of auxiliary variables and constraints, making them computationally more expensive to solve than SOS1 or big- $M$  formulations. Note that this is especially true, if the initial instance is of large size.

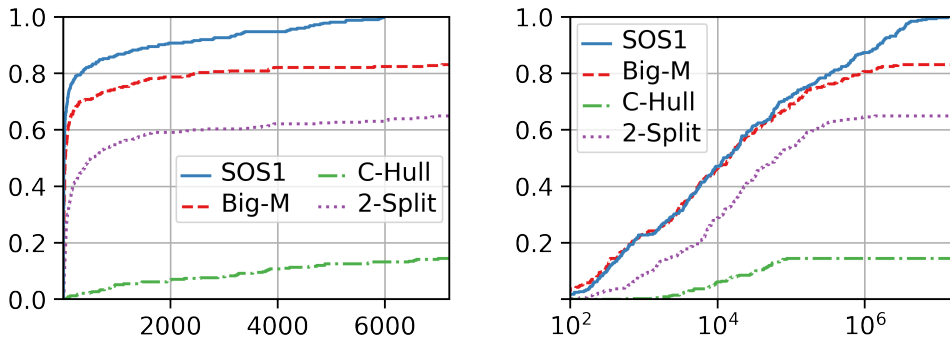


Figure 5.2: ECDFs of the instances in our test set. Left: computation time for SOS1, Big- $M$ , C-Hull, and 2-Split. Right: node counts.

We additionally tested whether the solutions to the continuous relaxations of the convex-hull reformulations (5.8) or the  $P$ -split reformulation (5.17) for  $P \in \{2, 3, 4, 5\}$  can provide stronger lower bounds than those obtained from the HPR. To this end, we solved the respective reformulations with a node limit of zero. However, we almost never observed an improvement in the bound, and in the few instances where a stronger bound was obtained, we cannot rule out the possibility that it was due to numerical inaccuracies. Again, a potential reason for this is that the values used to bound  $y$  and  $\lambda$  are not tight and, hence, the feasible region of the  $P$ -split relaxations remains large for different values of  $P$ . However, choosing smaller big- $M$  values, e.g., for the bounds of the variables  $\lambda$  leads to wrong solutions for some instances.

#### 5.5.4 Summary

In this first preliminary numerical study we have seen that the  $P$ -split reformulations get outperformed by the SOS1 or big- $M$  reformulation on the instances of the considered test set. We have also seen that  $P$ -split reformulations perform best if the value of  $P$  is small. However, there are many aspects that are not yet covered. For example, one could compare the discussed reformulations using instances-tailored tight bounds for the variables  $y$  and  $\lambda$ . Moreover, one could consider different values of  $\hat{P}_i$  and  $\tilde{P}_i$  for different  $i \in [n_y]$ , i.e., omit Assumption 5.4.1. Finally, one could test different strategies to determine the sets  $I_k^i$  for  $i \in [n_y]$ ,  $k \in \{1, \dots, P\}$ , and a given value of  $P$ .

## Chapter 6

# Conclusion and Outlook

In this dissertation, we studied different classes of bilevel problems with nonlinear lower levels and developed tailored solution methods that we analyzed theoretically and tested in extensive numerical studies.

**Disjunctive Cuts for Convex Integer Bilevel Problems** In Chapter 3, we proposed a branch-and-cut algorithm based on novel disjunctive cuts that is able to solve convex integer nonlinear bilevel problems. We proved the correctness of the resulting branch-and-cut method and compared the new cuts with those from the literature, both in theory and in practice. The results obtained in the numerical study demonstrate that the advantages of using disjunctive cuts outweigh their higher computational costs, highlighting the applicability of the approach.

We demonstrated that the presented disjunctive cuts cannot be used to handle mixed-integer variables in the upper- or lower-level problem in general; see Section 3.5.2. Despite this, we discuss a specific mixed-integer problem setting, to which the method could be adapted and outline the challenges involved. Moreover, the presented method is capable of handling nonconvex lower-level objective functions. This property may be exploited to tackle nonconvex lower-level constraints using penalty methods; see Remark 3.4.3. However, it is certainly possible that the computation of disjunctive cuts in this setting is too expensive using our approach. Further research regarding the generalization of the method is valuable. However, bilevel problems containing nonconvexities in the upper level remain out of scope for our approach as convexity in the node problems is crucial; see the proof of Theorem 3.3.4. Still, it may be possible to develop convexification techniques that allow our method to be applied such settings.

The method proposed in Chapter 3 relies on the use of bilevel-free sets; see Definition 3.6 and Section 3.3.4. We have seen in the numerical experiments that the choice of the bilevel-free sets used in the presented method is important for improving the performance of the approach. A well-chosen

bilevel-free set can significantly impact the effectiveness of the generated cuts and, consequently, the overall solution process. However, the derivation of such a set may be very costly. Hence, one promising direction to improve the proposed method is the development of novel bilevel-free sets or new techniques that improve existing ones.

**Complexity and a Solution Method for Nonconvex MIQP-QP Bilevel Problems** In Chapter 4, we studied MIQP-QP bilevel problems with integer linking variables and a nonconvex and continuous QP in the lower level for which we proved  $\Sigma_2^P$ -hardness. It remains open whether this problem is  $\Sigma_2^P$ -complete. One can try to address this question using the results in Henke et al. (2025a) and Henke et al. (2025b). It may be possible to show that there is no difference in the complexity-theoretical hardness of the considered MIQP-QP bilevel problem with and without coupling constraints. Then, the remainder follows from the definition of  $\Sigma_2^P$  in Woeginger (2021); see also Definition 2.2.1 and Theorem 4.3.7.

We additionally presented an iterative method to solve the problems under consideration and provided an extensive numerical study to showcase its applicability. The method is based on a single-level relaxation of the bilevel problem using the lower level’s KKT conditions. Note that our approach can also handle problems with more general upper levels, as long as a suitable solver for tackling the resulting single-level relaxation is at hand. Even nonlinear lower-level constraints can be considered if a reasonable constraint qualification can be guaranteed. The proposed method uses integer no-good cuts to separate points which do not improve an upper bound for the optimal upper-level objective function value. It would be valuable to have different types of cutting planes, tailored to separate bilevel-infeasible points, since this could significantly improve the performance of the presented method. To this end, we would need to solve the following separation problem:

- Input:** An MIQP-QP bilevel problem with integer linking variables and a nonconvex and continuous lower-level QP as well as a point  $(\tilde{x}, \tilde{y}, \tilde{\lambda}) \in P \cap (\text{KKT}(\tilde{x}) \setminus S(\tilde{x}))$ .
- Output:** A function  $h(x, y)$  such that  $h(\tilde{x}, y) \geq 0$  for all  $y \in S(\tilde{x})$  and  $h(\tilde{x}, y) < 0$  for all  $y \in \text{KKT}(\tilde{x}) \setminus S(\tilde{x})$ .

Here,  $S(\tilde{x})$  is the rational reaction set of the follower given  $\tilde{x}$  and  $\text{KKT}(\tilde{x})$  is the set of KKT points of the  $\tilde{x}$ -parameterized lower level. To the best of our knowledge, there are no approaches in the literature to solve this separation problem. Hence, any advancement in this respect would lead both to a better understanding on how to improve existing methods and to new approaches tailored for this class of bilevel problems.

Let us also mention that another important question for future research is to study the convergence of methods such as the one presented in Chapter 4 if the lower-level problem is only solved approximately. However, this might

be a challenging task; see, e.g., Beck et al. (2023b).

***P*-Splits for Continuous Lower-Level QPs** Finally, in Chapter 5, we studied bilevel problems with continuous upper- and lower-level variables as well as convex-quadratic lower-level QPs. We considered the KKT reformulation of the given bilevel problem and expressed the arising nonconvex complementarity constraints as a set of disjunctions. Then, we applied the ideas of Kronqvist et al. (2022) for general disjunctive constraints to the complementarity conditions. Therefore, we explored a family of *P*-split reformulations whose continuous relaxations are located between the big-*M* and convex hull relaxation of the disjunctions. Note that we can straightforwardly apply this strategy to a generalized problem setting containing mixed-integer upper-level variables. Finally, we tested the applicability of the *P*-split formulations in a preliminary numerical study.

Although the *P*-split approach could not beat state-of-the-art methods such as SOS1 or big-*M* reformulations in our numerical tests, we discussed several ideas that have not yet been tested. This includes different partitioning strategies and the number of subsets per disjunction, i.e., the value of  $P_i$  for the *i*th disjunction. Moreover, we discussed that tightness of the bounds for the lower-level primal and dual variables plays a crucial role for the relaxation strength of the *P*-split reformulations. Hence, an extensive numerical study addressing different parameterizations for the partitioning using instance-tailored tight bounds would be valuable to examine the efficiency of the *P*-split approach.

**General Outlook** We made all of our code publicly available to ensure reproducibility of the presented methods and to motivate further research in this directions. In all of our numerical experiments we used MILP-MILP bilevel instances from the literature. Due to the lack of nonlinear test instances, we generated quadratic matrices for modeling nonlinear terms in the lower level. Having benchmark sets that contain mixed-integer nonlinear bilevel instances would not only greatly benefit numerical studies for future methods, they could also be used to improve the quality of experiments for existing approaches. Fortunately, there has been recent progress in collecting bilevel instances on a large scale; see <https://bobilib.org> and Thürauf et al. (2024).

While the methods presented in this thesis can address various classes of bilevel problems with nonlinear lower levels, none are capable of handling general mixed-integer, nonlinear, and nonconvex bilevel problems without strong additional assumptions. Achieving this remains one of the main goals in the field of bilevel optimization, and this work represents an initial step toward that direction.

# Bibliography

- Achterberg, T. (2009). “SCIP: solving constraint integer programs.” In: *Mathematical Programming Computation* 1, pp. 1–41. DOI: [10.1007/s12532-008-0001-1](https://doi.org/10.1007/s12532-008-0001-1).
- Achterberg, T., T. Koch, and A. Martin (2005). “Branching rules revisited.” In: *Operations Research Letters* 33.1, pp. 42–54. DOI: [10.1016/j.orl.2004.04.002](https://doi.org/10.1016/j.orl.2004.04.002).
- Androulakis, I. P., C. D. Maranas, and C. A. Floudas (1995). “ $\alpha$ BB: A global optimization method for general constrained nonconvex problems.” In: *Journal of Global Optimization* 7.4, pp. 337–363. DOI: [10.1007/BF01099647](https://doi.org/10.1007/BF01099647).
- Balas, E. (1998). “Disjunctive programming: Properties of the convex hull of feasible points.” In: *Discrete Applied Mathematics* 89.1, pp. 3–44. DOI: [10.1016/S0166-218X\(98\)00136-X](https://doi.org/10.1016/S0166-218X(98)00136-X).
- (2018). *Disjunctive Programming*. Springer. DOI: [10.1007/978-3-030-00148-3](https://doi.org/10.1007/978-3-030-00148-3).
- Bazaraa, M. S., H. D. Sherali, and C. M. Shetty (2006). *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons. DOI: [10.1002/0471787779](https://doi.org/10.1002/0471787779).
- Beale, E. M. L. and J. A. Tomlin (1970). “Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables.” In: *OR* 69.447-454, p. 99. URL: [https://www.researchgate.net/publication/313166553\\_Special\\_facilities\\_in\\_a\\_general\\_mathematical\\_programming\\_system\\_for\\_nonconvex\\_problems\\_using\\_ordered\\_sets\\_of\\_variables](https://www.researchgate.net/publication/313166553_Special_facilities_in_a_general_mathematical_programming_system_for_nonconvex_problems_using_ordered_sets_of_variables).
- Beck, Y., I. Ljubić, and M. Schmidt (2023a). “A survey on bilevel optimization under uncertainty.” In: *European Journal of Operational Research* 311.2, pp. 401–426. DOI: [10.1016/j.ejor.2023.01.008](https://doi.org/10.1016/j.ejor.2023.01.008).
- Beck, Y. and M. Schmidt (2021). *A Gentle and Incomplete Introduction to Bilevel Optimization*. Lecture notes. URL: <https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/392>.
- Beck, Y., M. Schmidt, J. Thürauf, and D. Bienstock (2023b). “On a Computationally Ill-Behaved Bilevel Problem with a Continuous and Nonconvex Lower Level.” In: *Journal of Optimization Theory and Applications*. DOI: [10.1007/s10957-023-02238-9](https://doi.org/10.1007/s10957-023-02238-9).

- Ben-Tal, A., A. Nemirovski, and L. El Ghaoui (2009). *Robust Optimization*. Princeton university press. DOI: [10.1515/9781400831050](https://doi.org/10.1515/9781400831050).
- Bertsimas, D., D. B. Brown, and C. Caramanis (2011). “Theory and Applications of Robust Optimization.” In: *SIAM review* 53.3, pp. 464–501. DOI: [10.1137/080734510](https://doi.org/10.1137/080734510).
- Blankenship, J. W. and J. E. Falk (1976). “Infinitely constrained optimization problems.” In: *Journal of Optimization Theory and Applications* 19.2, pp. 261–281. DOI: [10.1007/BF00934096](https://doi.org/10.1007/BF00934096).
- Bomze, I. M. (1997). “Evolution towards the Maximum Clique.” In: *Journal of Global Optimization* 10.2, pp. 143–164. DOI: [10.1023/A:1008230200610](https://doi.org/10.1023/A:1008230200610).
- Bracken, J. and J. T. McGill (1973). “Mathematical programs with optimization problems in the constraints.” In: *Operations Research* 21.1, pp. 37–44. DOI: [10.1287/opre.21.1.37](https://doi.org/10.1287/opre.21.1.37).
- Buchheim, C. (2023). “Bilevel linear optimization belongs to NP and admits polynomial-size KKT-based reformulations.” In: *Operations Research Letters* 51.6, pp. 618–622. DOI: [10.1016/j.orl.2023.10.006](https://doi.org/10.1016/j.orl.2023.10.006).
- Caprara, A., M. Carvalho, A. Lodi, and G. J. Woeginger (2014). “A study on the computational complexity of the bilevel knapsack problem.” In: *SIAM Journal on Optimization* 24.2, pp. 823–838. DOI: [10.1137/130906593](https://doi.org/10.1137/130906593).
- (2016). “Bilevel knapsack with interdiction constraints.” In: *INFORMS Journal on Computing* 28.2, pp. 319–333. DOI: [10.1287/ijoc.2015.0676](https://doi.org/10.1287/ijoc.2015.0676).
- Cerulli, M. (2021). “Bilevel optimization and applications.” PhD thesis. Institut Polytechnique de Paris. URL: <https://theses.hal.science/tel-03587548/>.
- Conforti, M., G. Cornuéjols, and G. Zambelli (2014). *Integer Programming*. Springer. DOI: [10.1007/978-3-319-11008-0](https://doi.org/10.1007/978-3-319-11008-0).
- Dempe, S. (2002). *Foundations of Bilevel Programming*. Springer. DOI: [10.1007/b101970](https://doi.org/10.1007/b101970).
- Dempe, S. and J. Dutta (2012). “Is bilevel programming a special case of a mathematical program with complementarity constraints?” In: *Mathematical Programming* 131.1. <https://doi.org/10.1007/s10107-010-0342-1>, pp. 37–48.
- Dempe, S., J. Dutta, and B. Mordukhovich (2007). “New necessary optimality conditions in optimistic bilevel programming.” In: *Optimization* 56.5-6, pp. 577–604. DOI: [10.1080/02331930701617551](https://doi.org/10.1080/02331930701617551).
- Dempe, S., V. Kalashnikov, G. A. Pérez-Valdés, and N. Kalashnykova (2015). “Bilevel programming problems.” In: *Energy Systems. Springer, Berlin* 10.978-3, pp. 53–56. DOI: [10.1007/978-3-662-45827-3](https://doi.org/10.1007/978-3-662-45827-3).
- Dempe, S. and P. Mehlitz (2025). “Duality-based single-level reformulations of bilevel optimization problems.” In: *Journal of Optimization Theory and Applications* 205.2, p. 26. DOI: [10.1007/s10957-025-02627-2](https://doi.org/10.1007/s10957-025-02627-2).
- Dempe, S. and K. Richter (2000). “Bilevel Programming with Knapsack Constraints.” In: *Central European Journal of Operations Research* 8.2,

- pp. 93–107. URL: [https://www.researchgate.net/publication/2376759\\_Bilevel\\_Programming\\_With\\_Knapsack\\_Constraints](https://www.researchgate.net/publication/2376759_Bilevel_Programming_With_Knapsack_Constraints).
- Dempe, S. and A. Zemkoho (2020). *Bilevel Optimization*. Vol. 161. Springer. DOI: [10.1007/978-3-030-52119-6](https://doi.org/10.1007/978-3-030-52119-6).
- DeNegre, S. and T. K. Ralphs (2009). “A branch-and-cut algorithm for integer bilevel linear programs.” In: *Operations research and cyber-infrastructure*. Springer, pp. 65–78. DOI: [10.1007/978-0-387-88843-9\\_4](https://doi.org/10.1007/978-0-387-88843-9_4).
- DeNegre, S. T. (2011). *Interdiction and discrete bilevel linear programming*. PhD thesis. Lehigh University. URL: <http://coral.ie.lehigh.edu/~ted/files/papers/ScottDeNegreDissertation11.pdf>.
- Djelassi, H., M. Glass, and A. Mitsos (2019). “Discretization-based algorithms for generalized semi-infinite and bilevel programs with coupling equality constraints.” In: *Journal of Global Optimization* 75.2, pp. 341–392. DOI: [10.1007/s10898-019-00764-3](https://doi.org/10.1007/s10898-019-00764-3).
- Eggermont, C. E. and G. J. Woeginger (2013). “Motion planning with pulley, rope, and baskets.” In: *Theory of Computing Systems* 53.4, pp. 569–582. DOI: [10.1007/s00224-013-9445-4](https://doi.org/10.1007/s00224-013-9445-4).
- Fischetti, M., F. Glover, and A. Lodi (2005). “The feasibility pump.” In: *Mathematical Programming* 104, pp. 91–104. DOI: [10.1007/s10107-004-0570-3](https://doi.org/10.1007/s10107-004-0570-3).
- Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017). “A new general-purpose algorithm for mixed-integer bilevel linear programs.” In: *Operations Research* 65.6, pp. 1615–1637. DOI: [10.1287/opre.2017.1650](https://doi.org/10.1287/opre.2017.1650).
- (2018). “On the use of intersection cuts for bilevel optimization.” In: *Mathematical Programming* 172.1, pp. 77–103. DOI: [10.1007/s10107-017-1189-5](https://doi.org/10.1007/s10107-017-1189-5).
- Fischetti, M., A. Lodi, and A. Tramontani (2011). “On the separation of disjunctive cuts.” In: *Mathematical Programming* 128.1-2, pp. 205–230. DOI: [10.1007/s10107-009-0300-y](https://doi.org/10.1007/s10107-009-0300-y).
- Fortuny-Amat, J. and B. McCarl (1981). “A representation and economic interpretation of a two-level programming problem.” In: *Journal of the Operational Research Society* 32.9, pp. 783–792. DOI: [10.1057/jors.1981.156](https://doi.org/10.1057/jors.1981.156).
- Gaar, E., J. Lee, I. Ljubić, M. Sinnl, and K. Tanınmış (2023). “On SOCP-based disjunctive cuts for solving a class of integer bilevel nonlinear programs.” In: *Mathematical Programming*, pp. 1–34. DOI: [10.1007/s10107-023-01965-1](https://doi.org/10.1007/s10107-023-01965-1).
- Gabriel, S. A., A. J. Conejo, J. D. Fuller, B. F. Hobbs, and C. Ruiz (2012). *Complementarity Modeling in Energy Markets*. Vol. 180. Springer Science & Business Media. DOI: [10.1007/978-1-4419-6123-5](https://doi.org/10.1007/978-1-4419-6123-5).
- Garey, M. R. and D. S. Johnson (2002). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Vol. 29. W.H. Freeman New York.

- Gorissen, B. L., I. Yanıkođlu, and D. den Hertog (2015). “A practical guide to robust optimization.” In: *Omega* 53, pp. 124–137. DOI: [10.1016/j.omega.2014.12.006](https://doi.org/10.1016/j.omega.2014.12.006).
- Grimm, V., T. Kleinert, F. Liers, M. Schmidt, and G. Zöttl (2019). “Optimal price zones of electricity markets: a mixed-integer multilevel model and global solution approaches.” In: *Optimization Methods and Software* 34.2, pp. 406–436. DOI: [10.1080/10556788.2017.1401069](https://doi.org/10.1080/10556788.2017.1401069).
- Gurobi Optimization, LLC (2025). *Gurobi Optimizer Reference Manual*. URL: <https://www.gurobi.com>.
- Hansen, P., B. Jaumard, and G. Savard (1992). “New branch-and-bound rules for linear bilevel programming.” In: *SIAM Journal on Scientific and Statistical Computing* 13.5, pp. 1194–1217. DOI: [10.1137/0913069](https://doi.org/10.1137/0913069).
- Henke, D., H. Lefebvre, M. Schmidt, and J. Thürauf (2025a). “On coupling constraints in linear bilevel optimization.” In: *Optimization Letters* 19.3, pp. 689–697. DOI: [10.1007/s11590-024-02156-3](https://doi.org/10.1007/s11590-024-02156-3).
- (2025b). *On Coupling Constraints in Pessimistic Linear Bilevel Optimization*. URL: <https://arxiv.org/abs/2503.01563>.
- Hooker, J. N. (2009). “A principled approach to mixed integer/linear problem formulation.” In: *Operations research and cyber-infrastructure*. Springer, pp. 79–100. DOI: [10.1007/978-0-387-88843-9\\_5](https://doi.org/10.1007/978-0-387-88843-9_5).
- Huang, L., X. Chen, W. Huo, J. Wang, F. Zhang, B. Bai, and L. Shi (2021). *Branch and bound in mixed integer linear programming problems: A survey of techniques and trends*. URL: <https://arxiv.org/abs/2111.06257>.
- Hungerford, J. T. and F. Rinaldi (2019). “A general regularized continuous formulation for the maximum clique problem.” In: *Mathematics of Operations Research* 44.4, pp. 1161–1173. DOI: [10.1287/moor.2018.0954](https://doi.org/10.1287/moor.2018.0954).
- IBM Corporation (2023). *IBM ILOG CPLEX Optimization Studio Documentation*. URL: <https://www.ibm.com/docs/en/icos>.
- Jeroslow, R. G. (1985). “The polynomial hierarchy and a simple model for competitive analysis.” In: *Mathematical Programming* 32.2, pp. 146–164. DOI: [10.1007/BF01586088](https://doi.org/10.1007/BF01586088).
- Khuri, S., T. Baeck, and J. Heitkoetter (1994). *SAC94 suite: Collection of multiple knapsack problems*. URL: <http://www.cs.cmu.edu/Groups/AI/areas/genetic/ga/test/sac/0.html>.
- Kleinert, T., V. Grimm, and M. Schmidt (2021a). “Outer approximation for global optimization of mixed-integer quadratic bilevel problems.” In: *Mathematical Programming* 188.2, pp. 461–521. DOI: [10.1007/s10107-020-01601-2](https://doi.org/10.1007/s10107-020-01601-2).
- Kleinert, T., M. Labbé, I. Ljubić, and M. Schmidt (2021b). “A survey on mixed-integer programming techniques in bilevel optimization.” In: *EURO Journal on Computational Optimization* 9, p. 100007. DOI: [10.1016/j.ejco.2021.100007](https://doi.org/10.1016/j.ejco.2021.100007).
- Kleinert, T., M. Labbé, F. Plein, and M. Schmidt (2020). “There’s no free lunch: on the hardness of choosing a correct big-M in bilevel optimization.”

- In: *Operations Research* 68.6, pp. 1716–1721. DOI: [10.1287/opre.2019.1944](https://doi.org/10.1287/opre.2019.1944).
- Kleinert, T. and M. Schmidt (2021). “Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method.” In: *INFORMS Journal on Computing* 33.1, pp. 198–215. DOI: [10.1287/ijoc.2019.0945](https://doi.org/10.1287/ijoc.2019.0945).
- (2023). “Why there is no need to use a big- $M$  in linear bilevel optimization: A computational study of two ready-to-use approaches.” In: *Computational Management Science*. DOI: [10.1007/s10287-023-00435-5](https://doi.org/10.1007/s10287-023-00435-5).
- Kleniati, P.-M. and C. S. Adjiman (2015). “A generalization of the Branch-and-Sandwich algorithm: From continuous to mixed-integer nonlinear bilevel problems.” In: *Computers & Chemical Engineering* 72, pp. 373–386. DOI: [10.1016/j.compchemeng.2014.06.004](https://doi.org/10.1016/j.compchemeng.2014.06.004).
- Kleniati, P.-M. and C. S. Adjiman (2014a). “Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part I: Theoretical development.” In: *Journal of Global Optimization* 60.3, pp. 425–458. DOI: [10.1007/s10898-013-0121-7](https://doi.org/10.1007/s10898-013-0121-7).
- (2014b). “Branch-and-Sandwich: a deterministic global optimization algorithm for optimistic bilevel programming problems. Part II: Convergence analysis and numerical results.” In: *Journal of Global Optimization* 60.3, pp. 459–481. DOI: [10.1007/s10898-013-0120-8](https://doi.org/10.1007/s10898-013-0120-8).
- Kronqvist, J., R. Misener, and C. Tsay (2022). *P-split formulations: A class of intermediate formulations between big- $M$  and convex hull for disjunctive constraints*. DOI: [10.48550/ARXIV.2202.05198](https://doi.org/10.48550/ARXIV.2202.05198).
- Labbé, M. and A. Violin (2013). “Bilevel programming and price setting problems.” In: *4OR* 11, pp. 1–30. DOI: [10.1007/s10288-012-0213-0](https://doi.org/10.1007/s10288-012-0213-0).
- Land, A. and A. Doig (1960). “An Automatic Method of Solving Discrete Programming Problems.” In: *Econometrica* 28.3, pp. 497–520. DOI: [10.1007/978-3-540-68279-0\\_5](https://doi.org/10.1007/978-3-540-68279-0_5).
- Lee, J. and S. Leyffer (2011). *Mixed Integer Nonlinear Programming*. Vol. 154. Springer Science & Business Media. DOI: [10.1007/978-1-4614-1927-3](https://doi.org/10.1007/978-1-4614-1927-3).
- Locatelli, M. and F. Schoen (2013). *Global Optimization: Theory, Algorithms, and Applications*. SIAM. DOI: [10.1137/1.9781611972672](https://doi.org/10.1137/1.9781611972672).
- Lodi, A., M. Tanneau, and J.-P. Vielma (2023). “Disjunctive cuts in mixed-integer conic optimization.” In: *Mathematical Programming* 199.1-2, 671–719. DOI: [10.1007/s10107-022-01844-1](https://doi.org/10.1007/s10107-022-01844-1).
- Lozano, L. and J. C. Smith (2017). “A value-function-based exact approach for the bilevel mixed-integer programming problem.” In: *Operations Research* 65.3, pp. 768–786. DOI: [10.1287/opre.2017.1589](https://doi.org/10.1287/opre.2017.1589).
- Mansi, R., C. Alves, J. Valério de Carvalho, and S. Hanafi (2012). “An Exact Algorithm for Bilevel 0-1 Knapsack Problems.” In: *Mathematical Problems in Engineering* 2012.1, p. 504713. DOI: [10.1155/2012/504713](https://doi.org/10.1155/2012/504713).
- Marcotte, P. (1986). “Network design problem with congestion effects: A case of bilevel programming.” In: *Mathematical Programming* 34.2, pp. 142–162. DOI: [10.1007/BF01580580](https://doi.org/10.1007/BF01580580).

- Meyer, A. R. and L. J. Stockmeyer (1972). “The equivalence problem for regular expressions with squaring requires exponential space.” In: *SWAT*. Vol. 72, pp. 125–129. DOI: [10.1109/SWAT.1972.29](https://doi.org/10.1109/SWAT.1972.29).
- Mitsos, A. (2010). “Global solution of nonlinear mixed-integer bilevel programs.” In: *Journal of Global Optimization* 47.4, pp. 557–582. DOI: [10.1007/s10898-009-9479-y](https://doi.org/10.1007/s10898-009-9479-y).
- Mitsos, A. and P. I. Barton (2006). *Issues in the development of global optimization algorithms for bilevel programs with a nonconvex inner program*. Tech. rep. URL: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.4195&rep=rep1&type=pdf>.
- Mitsos, A., P. Lemonidis, and P. I. Barton (2008). “Global solution of bilevel programs with a nonconvex inner program.” In: *Journal of Global Optimization* 42.4, pp. 475–513. DOI: [10.1007/s10898-007-9260-z](https://doi.org/10.1007/s10898-007-9260-z).
- Moore, J. T. and J. F. Bard (1990). “The mixed integer linear bilevel programming problem.” In: *Operations Research* 38.5, pp. 911–921. DOI: [10.1287/opre.38.5.911](https://doi.org/10.1287/opre.38.5.911).
- Motzkin, T. S. and E. G. Straus (1965). “Maxima for Graphs and a New Proof of a Theorem of Turán.” In: *Canadian Journal of Mathematics* 17, pp. 533–540. DOI: [10.4153/CJM-1965-053-6](https://doi.org/10.4153/CJM-1965-053-6).
- Pastor, R. and A. Corominas (2000). “Strategies of node selection in search procedures for solving combinatorial optimization problems: a survey and a general formalization.” In: *Top* 8, pp. 111–134. DOI: [10.1007/BF02564831](https://doi.org/10.1007/BF02564831).
- Paulavicius, R. and C. S. Adjiman (2017). *BASBLib - a library of bilevel test problems, v2.2 [Data set]*. DOI: [10.5281/zenodo.897966](https://doi.org/10.5281/zenodo.897966).
- Pineda, S. and J. M. Morales (2019). “Solving linear bilevel problems using big-Ms: Not all that glitters is gold.” In: *IEEE Transactions on Power Systems* 34.3, pp. 2469–2471. DOI: [10.1109/TPWRS.2019.2892607](https://doi.org/10.1109/TPWRS.2019.2892607).
- Rodrigues, B., M. Carvalho, M. F. Anjos, and N. Sugishita (2025). *Unboundedness in bilevel optimization*. Tech. rep. G-2025-22. URL: <https://www.gerad.ca/fr/papers/G-2025-22.pdf>.
- Stein, O. (2003). *Bi-level Strategies in Semi-infinite Programming*. Vol. 71. Springer Science & Business Media. DOI: [10.1007/978-1-4419-9164-5](https://doi.org/10.1007/978-1-4419-9164-5).
- Stockmeyer, L. J. (1976). “The polynomial-time hierarchy.” In: *Theoretical Computer Science* 3.1, pp. 1–22. DOI: [10.1016/0304-3975\(76\)90061-X](https://doi.org/10.1016/0304-3975(76)90061-X).
- Tahernejad, S. and T. K. Ralphs (2020). *Valid Inequalities for Mixed Integer Bilevel Linear Optimization Problems*. Tech. rep. URL: <https://optimization-online.org/wp-content/uploads/2020/11/8096.pdf>.
- Tambe, M. (2011). *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press. DOI: [10.5555/2124410](https://doi.org/10.5555/2124410).
- Thürauf, J., T. Kleinert, I. Ljubić, T. Ralphs, and M. Schmidt (2024). *BO-BILib: Bilevel Optimization (Benchmark) Instance Library*. Tech. rep. URL: <https://optimization-online.org/?p=27063>.

- Vicente, L., G. Savard, and J. Júdice (1994). “Descent approaches for quadratic bilevel programming.” In: *Journal of Optimization Theory and Applications* 81.2, pp. 379–399. DOI: [10.1007/BF02191670](https://doi.org/10.1007/BF02191670).
- von Stackelberg, H. (1934). *Marktform und Gleichgewicht*.
- Wallace, C. (2010). “ZI round, a MIP rounding heuristic.” In: *Journal of Heuristics* 16, pp. 715–722. DOI: [10.1007/s10732-009-9114-6](https://doi.org/10.1007/s10732-009-9114-6).
- Wang, L. and P. Xu (2017). “The watermelon algorithm for the bilevel integer linear programming problem.” In: *SIAM Journal on Optimization* 27.3, pp. 1403–1430. DOI: [10.1137/15M1051592](https://doi.org/10.1137/15M1051592).
- Wang, Z., Y. Yin, and B. An (2016). “Computing Optimal Monitoring Strategy for Detecting Terrorist Plots.” In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*. Ed. by D. Schuurmans and M. P. Wellman. AAAI Press, pp. 637–643. DOI: [10.1609/aaai.v30i1.10028](https://doi.org/10.1609/aaai.v30i1.10028).
- Woeginger, G. J. (2021). “The trouble with the second quantifier.” In: *4OR* 19.2, pp. 157–181. DOI: [10.1007/s10288-021-00477-y](https://doi.org/10.1007/s10288-021-00477-y).
- Xu, P. and L. Wang (2014). “An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions.” In: *Computers & Operations Research* 41, pp. 309–318. DOI: [10.1016/j.cor.2013.07.016](https://doi.org/10.1016/j.cor.2013.07.016).
- Ye, J. J. and D. Zhu (1995). “Optimality conditions for bilevel programming problems.” In: *Optimization* 33.1, pp. 9–27. DOI: [10.1080/02331939508844060](https://doi.org/10.1080/02331939508844060).
- Zare, M. H., J. S. Borrero, B. Zeng, and O. A. Prokopyev (2019a). “A note on linearized reformulations for a class of bilevel linear integer problems.” In: *Annals of Operations Research* 272, pp. 99–117. DOI: [10.1007/s10479-017-2694-x](https://doi.org/10.1007/s10479-017-2694-x).
- (2019b). “A note on linearized reformulations for a class of bilevel linear integer problems.” In: *Annals of Operations Research* 272, pp. 99–117. DOI: [10.1007/s10479-017-2694-x](https://doi.org/10.1007/s10479-017-2694-x).
- Zhou, S., A. B. Zemkoho, and A. Tin (2018). *BOLIB: Bilevel Optimization LIBrary of test problems*. URL: <https://arxiv.org/abs/1812.00230>.