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

Andreas Horländer, Ivana Ljubić, and Martin Schmidt

Abstract. We present a branch-and-cut method for solving convex integer nonlinear bilevel problems, i.e., bilevel models with nonlinear but convex objective functions and constraints in both the upper and the lower level. To this end, we generalize the idea of using disjunctive cuts to cut off integer-feasible but bilevel-infeasible points. These cuts can be obtained by solving a cut-generating problem, which itself is a single-level but nonconvex integer nonlinear problem. We show that this cut-generating problem can be decomposed into a series of smaller subproblems. These can all be solved in parallel and we state sufficient conditions to ensure that they are convex integer nonlinear problems. Moreover, we develop additional algorithmic techniques such as tailored pruning rules to further speed up our method. We finally prove the correctness of the method and test it in a numerical study that shows the applicability of the method.

1. Introduction

Bilevel optimization is an important area of mathematical optimization that has gained increasing attention over the last years and decades. Bilevel problems model hierarchical situations in which two agents with potentially conflicting interests make decisions that influence each other and, therefore, cannot be considered independently; see Dempe and Zemkoho (2020) or Bard (2013) for more detailed introductions. Hence, bilevel optimization is a powerful tool to study 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), or pricing (Labbé and Violin 2013). However, bilevel problems are very hard to solve both in theory and practice. Even in the easiest case in which both levels are linear programs, the problem is known to be strongly NP-hard (Hansen et al. 1992) and even checking local optimality for a given point is NP-hard; see Vicente et al. (1994).

For bilevel problems with 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) for further information. Both mathematically and algorithmically more challenging classes of bilevel problems are those with nonconvexities in the lower-level problem. In such cases, there are no compact necessary and sufficient optimality conditions in general. Therefore, approaches such as the KKT reformulation or reformulations based on strong-duality theorems (Zare et al. 2019) are not applicable. An important example of such a problem class are bilevel problems for which both levels are mixed-integer linear problems. Here, nonconvexities are due to integrality constraints. In Jeroslow (1985) and Lodi et al. (2014) it is shown that bilevel problems with mixed-integer linear models on both levels are Σ_2^P-hard in general.

Date: March 21, 2024.

2020 *Mathematics Subject Classification.* 90C10, 90C11, 90C26.

Key words and phrases. Bilevel optimization, Branch-and-cut, Mixed-integer nonlinear optimization, Disjunctive cuts.

Hence, most solution methods use some kind of branch-and-cut framework, where branching is performed on integrality constraints and cutting planes are used to cut off integer-feasible but bilevel-infeasible points. The cut generation usually needs to solve an NP-hard problem. 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); see Kleinert et al. (2021b) for a recent overview. An overview of suitable cutting planes in bilevel optimization is given in Tahernejad and Ralphs (2020).

Even more challenging than MILP-MILP bilevel problems are problems with integrality conditions as well as nonlinearities in both levels. In the recent work by Gaar et al. (2023), the authors consider convex integer nonlinear bilevel problems with second-order cone constraints in the upper level and a convex quadratic objective function and linear constraints in the lower level. The authors developed problem-specific 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 programs 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 we allow for nonlinearities in the lower-level constraints and since we do not restrict ourselves to second-order cone constraints.

Let us also mention methods that consider even more general classes of bilevel problems. For instance, in Mitsos (2010), general bilevel MINLPs are considered. The approach is an extension of the method in Mitsos et al. (2008) for purely continuous bilevel problems. Furthermore, there is a series of papers on the so-called branch-and-sandwich approach for bilevel MINLPs; see, e.g., Kleniati and Adjiman (2014b) and Kleniati and Adjiman (2014a) on purely continuous bilevel problems and the extension to the mixed-integer case in Kleniati and Adjiman (2015). Due to the overall hardness of the problems under consideration and the very general assumptions made, the computational study in Kleniati and Adjiman (2015) deals with rather small problems with up to 12 variables and 7 constraints. A different setting is considered in Lozano and Smith (2017). Here, all objective and constraint functions need to be continuous but can be nonconvex. However, the authors use special separability properties and assume that some functions only take integer values.

In the view of these references and to the best of our knowledge, we are the first ones presenting a method that is tailored for solving general convex integer nonlinear bilevel problems. By exploiting the problem's special structure, we are able to tackle much larger instances having a few hundred variables and constraints. Hence, and to sum up, our contribution is the development of disjunctive cuts for our specific setup. Furthermore, we develop a novel branch-and-cut framework and also present additional algorithmic techniques to enhance our method. Finally, we present extensive numerical results that show the applicability of our method.

The remainder of the paper is organized as follows. In Section 2, we describe our problem setting and introduce those standard notions of bilevel optimization that will be used afterward. Then, in Section 3, we discuss disjunctive cuts in the context of our setup and give an in-depth discussion on how to compute them. Next, we describe the overall branch-and-cut framework including the use of disjunctive cuts in Section 4 and we prove the correctness of the method. In Section 5, we discuss algorithmic techniques that we develop to enhance the proposed algorithm.

Section 6 contains the numerical results. We compare our 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. Finally, we summarize in Section 7 and discuss some topics for future research.

2. Problem Statement and Important Relaxations

We consider the problem

$$\min_{x \in \mathbb{Z}^{n_x}, y} \quad F(x, y) \tag{1a}$$

$$\text{s.t.} \quad G(x, y) \leq 0, \tag{1b}$$

$$y \in \operatorname*{arg\,min}_{y' \in \mathbb{Z}^{n_y}} \{f(x, y') \colon g(x, y') \leq 0\}, \tag{1c}$$

where $n := n_x + n_y$ and $F, f : \mathbb{R}^n \to \mathbb{R}$, $G : \mathbb{R}^n \to \mathbb{R}^m$, as well as $g : \mathbb{R}^n \to \mathbb{R}^l$ are continuous and jointly convex functions. Note that the constraints in (1b) and (1c) may also contain bounds on the x- and y-variables. Without loss of generality, we can assume that F is linear, because we can solve the epigraph reformulation of Problem (1) otherwise. We refer to (1a) and (1b) as the leader's optimization or upper-level problem and to (1c) as the follower's optimization or lower-level problem. In the case that the lower level has multiple optimal solutions, we choose the one leading to minimum costs $F(x, y)$, i.e., we consider the optimistic version of the given bilevel optimization problem.

The value-function reformulation of Problem (1) reads[1]

$$\min_{x \in \mathbb{Z}^{n_x}, y \in \mathbb{Z}^{n_y}} \quad F(x, y) \tag{2a}$$

$$\text{s.t.} \quad G(x, y) \leq 0, \tag{2b}$$

$$g(x, y) \leq 0, \tag{2c}$$

$$(x, y) \in \mathbb{Z}^n, \tag{2d}$$

$$f(x, y) \leq \Phi(x) \tag{2e}$$

with

$$\Phi(x) := \min_{y \in \mathbb{Z}^{n_y}} \{f(x, y) \colon g(x, y) \leq 0\}. \tag{3}$$

Dropping Condition (2e) leads to the high-point relaxation (HPR), i.e., the convex integer nonlinear problem (INLP)

$$\min_{(x,y) \in \tilde{\Omega}} F(x, y) \tag{4}$$

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

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

Further removing the integrality constraints on the variables x and y leads to the continuous high-point relaxation (C-HPR), which is the convex optimization problem

$$\min_{(x,y) \in \Omega} F(x, y) \tag{5}$$

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

Similar to Gaar et al. (2023), we make the following assumptions.

Assumption 2.1. *The HPR's feasible set is bounded and the HPR has a solution.*

[1]Throughout this paper, we abbreviate $(x^\top, y^\top)^\top$ by (x, y) for the ease of better reading.

Assumption 2.2. *For all $x \in \tilde{\Omega}_x$, the set $\tilde{\Omega}_l(x) := \{y \in \mathbb{Z}^{n_y} : g(x,y) \leq 0\}$ is non-empty and bounded.*

Note that if Assumption 2.1 holds, the C-HPR (5) cannot be unbounded due to the convexity of the problem. Hence, this implies the boundedness of the set Ω. Assumption 2.2 ensures that for all $x \in \tilde{\Omega}_x$, the x-parameterized lower-level problem (1c) has a solution.

3. Disjunctive Cuts

We tackle Problem (1) by 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 Algorithm 2 in Fischetti et al. (2018) for a generic branch-and-bound scheme for mixed-integer linear bilevel problems (MILP-MILP). 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 3.1. *For any $\hat{y} \in \mathbb{Z}^{n_y}$, the set*

$$S(\hat{y}) := \{(x,y) \in \mathbb{R}^n \colon g(x,\hat{y}) \leq 0,\ f(x,y) > f(x,\hat{y})\} \tag{6}$$

does not contain any bilevel-feasible point.

The set $S(\hat{y})$ in (6) is commonly denoted as the bilevel-free set; see Fischetti et al. (2018) or Gaar et al. (2023). We will later use the interior of $S(\hat{y})$ and its complement to derive valid inequalities to cut off bilevel-infeasible points. To this end, we make the following assumption.

Assumption 3.2. *Let $(\tilde{x},\tilde{y}) \in \Omega$ be a bilevel-infeasible point. We assume that there exists a point $\hat{y} \in \mathbb{Z}^{n_y}$ such that $(\tilde{x},\tilde{y}) \in int(S(\hat{y}))$, where for an arbitrary set A we denote with $int(A)$ the interior of A.*

Note that the point $(\tilde{x},\tilde{y})$ in the last assumption 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. We assume that the bilevel-infeasible point $(\tilde{x},\tilde{y})$ is in the interior of a bilevel-free set $S(\hat{y})$ to guarantee that the inequality we generate actually cuts off the point. In Section 3.4, we discuss how to compute a suitable bilevel-free set (if it exists at all). Furthermore, in Section 4.2, we discuss the case in which the bilevel-infeasible point $(\tilde{x},\tilde{y})$ is on the boundary of the bilevel-free set $S(\hat{y})$.

Given $\hat{y} \in \mathbb{Z}^{n_y}$ such that Assumption 3.2 is satisfied, we now define

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

for $i \in I := \{1,\dots,l\}$ 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 them has an empty intersection with the interior of the bilevel-free set $S(\hat{y})$. Moreover, we have

$$\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) = \operatorname{int}(S(\hat{y}))^{\mathrm{c}}, \tag{7}$$

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

3.1. **Computing Disjunctive Cuts.**

We now give a formal definition of disjunctive cuts in our context.

Definition 3.3. *Let $\Theta \subseteq \Omega$ be non-empty and convex and let $(\tilde{x}, \tilde{y}) \in \Theta$ be a bilevel-infeasible point. Moreover, let*

$$\alpha^\top x + \beta^\top y - \tau = 0 \tag{8}$$

be a hyperplane that is parameterized by $\alpha \in \mathbb{R}^{n_x}$, $\beta \in \mathbb{R}^{n_y}$, and $\tau \in \mathbb{R}$. Furthermore, 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

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

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

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 Θ.

A disjunctive cut as stated in Definition 3.3 separates a bilevel-infeasible point $(\tilde{x}, \tilde{y})$ from all integer-feasible points that are inside Θ but outside of $\operatorname{int}(S(\hat{y}))$ for a given $\hat{y}$. To compute a disjunctive cut, we solve the cut-generating problem

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

The norm constraint ensures that the (CGP) is bounded. We use the ℓ_1-norm as it is commonly done in the literature; see, e.g., Fischetti et al. (2011) and Lodi et al. (2023). Note that the (CGP) cannot be infeasible because $(\alpha, \beta, \tau) = (0, 0, 0) \in \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}$ is feasible for (CGP). If a solution (α, β, τ) to the (CGP) yields a strictly positive objective function value, then the resulting hyperplane separates the point $(\tilde{x}, \tilde{y})$ from the set $\Theta \cap \mathbb{Z}^n \cap \left(\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right)$. The following theorem makes a statement about the solvability of (CGP).

Theorem 3.4. *Suppose that there is a bilevel-infeasible point $(\tilde{x}, \tilde{y})$ that is an extreme point of Θ and that belongs to the interior of the bilevel-free set $S(\hat{y})$ for some $\hat{y} \in \mathbb{Z}^{n_y}$. Then, there exists a disjunctive cut that separates $(\tilde{x}, \tilde{y})$ from $\Theta \cap \mathbb{Z}^n \cap \left(\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right)$ and it can be obtained by solving the (CGP).*

Proof. The proof is similar to the proof of Theorem 1 in Gaar et al. (2023). If the point $(\tilde{x}, \tilde{y})$ lies in the interior of the bilevel-free set $S(\hat{y})$, then it is not in $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ due to (7). Moreover, since $(\tilde{x}, \tilde{y})$ is an extreme point of Θ, it cannot be represented as a proper convex combination of two different points inside $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$. Therefore, the point is not in the convex hull of $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$. Hence, we can separate $(\tilde{x}, \tilde{y})$ from $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ via a linear inequality. Such an inequality is a disjunctive cut according to Definition 3.3 and is obtained when solving the (CGP). □

In the context of a branch-and-bound algorithm, the set Θ may correspond to a subset of the search space determined by the branching decisions together with additional constraints generated along the branching process.

Remark 3.5. *In the standard theory of disjunctive cuts one typically separates points from the convex hull of a disjunction; see, e.g., Balas (2018). In our setup, we want to separate $(\tilde{x}, \tilde{y})$ from a discrete set. Therefore, we do not need to work with the convex hull of $\mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ because we can obtain an implicit representation of the discrete set as shown in the next section.*

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

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

where $\mathcal{Z}^k \subseteq \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ is a discrete and finite set. Note that (RCGP) can be reformulated as a linear optimization problem in (α, β, τ). Hence, it can be solved in polynomial time. The set $\mathcal{Z}^0$ can be initialized with the empty set but in Section 5.2 we will also give a condition under which we can initialize $\mathcal{Z}^0$ with bilevel-feasible points. After solving the (RCGP), we obtain an optimal solution $(\alpha^k, \beta^k, \tau^k)$ and check if this solution satisfies the constraint

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

i.e., we check if

$$\begin{aligned} & \Psi(\alpha^k, \beta^k, \tau^k) \\ & := \max_{x,y} \left\{ (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k : (x, y) \in \Theta \cap \mathbb{Z}^n \cap \left(\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}) \right) \right\} \leq 0 \end{aligned} \tag{9}$$

holds. Problem (9) is bounded due to the boundedness of $\Theta \subseteq \Omega$. If Problem (9) is infeasible, the set Θ does not contain bilevel-feasible points as stated in the following lemma.

Lemma 3.6. *Let $S(\hat{y})$ be any bilevel-free set. If Problem (9) is infeasible, then the set $\Theta \subseteq \Omega$ is bilevel-free.*

Proof. If Problem (9) is infeasible, we have

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

see (7) for the first equality. It follows that

$$\Theta \cap \mathbb{Z}^n \subseteq \operatorname{int}(S(\hat{y}))$$

holds, i.e., every integer-feasible point inside Θ is inside a bilevel-free set. Therefore, Θ does not contain any bilevel-feasible point. □

Note that the feasibility of Problem (9) does not depend on the iteration k. To obtain $\Psi(\alpha^k, \beta^k, \tau^k)$ we can solve the equivalent problem

$$\begin{aligned}
\max_{x,y,b} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k && \text{(10a)}\\
\text{s.t.} \quad & (x,y) \in \Theta \cap \mathbb{Z}^n, && \text{(10b)}\\
& f(x,\hat{y}) - f(x,y) \geq -M_0(1-b_0), && \text{(10c)}\\
& g_i(x,\hat{y}) \geq -M_i(1-b_i), \quad i \in I, && \text{(10d)}\\
& b_i \in \{0,1\}, \quad i \in I_0, && \text{(10e)}\\
& \sum_{i \in I_0} b_i \geq 1, && \text{(10f)}
\end{aligned}$$

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 (10c) and (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 (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 (10), we optimize over the set $\Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$.

3.3. **Decomposition of (CGP).** To solve Problem (10), we decompose it into the $l+1$ subproblems

$$\begin{aligned}
\max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\
\text{s.t.} \quad & (x,y) \in \Theta \cap \mathbb{Z}^n, \\
& f(x,\hat{y}) - f(x,y) \geq 0
\end{aligned} \tag{11}$$

and

$$\begin{aligned}
\max_{x,y} \quad & (\alpha^k)^\top x + (\beta^k)^\top y - \tau^k \\
\text{s.t.} \quad & (x,y) \in \Theta \cap \mathbb{Z}^n, \\
& g_i(x,\hat{y}) \geq 0
\end{aligned} \tag{12}$$

for $i \in I$. In each of the subproblems (11) and (12) one aims to find a point $(x,y) \in \Theta \cap \mathbb{Z}^n \cap \mathcal{D}_i(\hat{y})$, which yields the largest objective value for the current hyperplane that is parameterized by α^k, β^k, and τ^k.

Note that the subproblems (11) and (12) are generally nonconvex and, thus, hard to solve. They are bounded due to the boundedness of Θ. It is possible that some or even all of the subproblems (11) and (12) are infeasible, which depends on $\hat{y}$ and Θ. 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 (9) is infeasible as well and, hence, Θ is bilevel-free; see Lemma 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

$$(\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0. \tag{13}$$

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 (x^j, y^j). Then, we set

$$\mathcal{Z}^{k+1} \leftarrow \mathcal{Z}^k \cup \left\{(x^i, y^i)^k : (\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0,\, i \in I_0^{\text{feas}}\right\} \tag{14}$$

and repeat the procedure by solving the (RCGP) with the updated set $\mathcal{Z}^{k+1}$. If, on the other hand, there is no $(x^i, y^i)^k$ satisfying (13) for some $i \in I_0^{\text{feas}}$, then $(\alpha^k, \beta^k, \tau^k)$ strictly separates (x^*, y^*) from all points $(x,y) \in \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ and we found an appropriate cut.

One advantage of considering the subproblems (11) and (12) instead of Problem (10) is that we neither need big-M constants nor the respective binary variables. Moreover, we can solve all subproblems in parallel. We now show that computing a disjunctive cut with the described adversarial approach only takes a finite amount of iterations.

Lemma 3.7. *Suppose that there is a bilevel-infeasible point $(\tilde{x}, \tilde{y})$ that is an extreme point of Θ and that belongs to the interior of the bilevel-free set $S(\hat{y})$ for some $\hat{y} \in \mathbb{Z}^{n_y}$. Then, computing the disjunctive cut from Theorem 3.4 can be done in finitely many steps.*

Proof. Solving the (RCGP) can be done in polynomial time as it is a linear problem. Since the subproblems (11) and (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 Ω, see Assumption 2.1, which implies the boundedness of Θ. Furthermore, in every iteration k, we enlarge the set $\mathcal{Z}^k$ by at least one point if we have not found a cutting hyperplane yet; see (14). Hence, we only need finitely many updates until $\mathcal{Z}^{k'} = \Theta \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ holds for some $k' < \infty$. □

In general, the subproblems (11) and (12) are nonconvex INLPs but the following proposition gives a condition under which the problems are convex INLPs.

Proposition 3.8. *Let the functions f and g_i be jointly convex and linear in x for all $i \in I$. Then, the problems (11) and (12) are convex INLPs for all $i \in I$.*

Proof. If the functions f and g_i, $i \in I$, are jointly convex and linear in x, then the condition

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

in Problem (11) is convex in x and y. Furthermore, the condition

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

in Problem (12) is also convex in x for all $i \in I$. Note that $\hat{y}$ is a constant in this context. □

3.4. **Computing Bilevel-Free Sets.**

So far we assumed that we are given a bilevel-free set such that the bilevel-infeasible point $(\tilde{x}, \tilde{y})$ is in the interior of this set; see Assumption 3.2. In this section, we show how do derive a suitable bilevel-free set for a given integer-feasible point $(\tilde{x}, \tilde{y})$.

The idea of Fischetti et al. (2018) for mixed-integer bilevel linear programs (MIBLPs) consists of using the optimal follower's response for a given $\tilde{x}$ as $\hat{y}$ to create a bilevel-free set. Therefore, in Fischetti et al. (2018), $S(\hat{y})$ is 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). However, applying the same techniques is not possible in our nonlinear setting.

Instead, we derive the bilevel-free set by solving an auxiliary problem. Therefore, we generalize the type-1 scoop problem of the watermelon approach presented by Wang and Xu (2017), where bilevel integer linear optimization 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 Δy such that the bilevel-free set $S(\tilde{y} + \Delta y)$ contains the bilevel-infeasible point $(\tilde{x}, \tilde{y})$ in its interior, i.e., $(\tilde{x}, \tilde{y}) \in \operatorname{int}(S(\tilde{y} + \Delta y))$. Note that $\tilde{y} + \Delta y$ takes the role of $\hat{y}$ of the previous notation. The bilevel-free set of the point $\tilde{y} + \Delta y$ is given by

$$S(\tilde{y} + \Delta y) := \{(x, y) \in \mathbb{R}^n : g(x, \tilde{y} + \Delta y) \leq 0,\ f(x, y) > f(x, \tilde{y} + \Delta y)\} \tag{15}$$

and the point $(\tilde{x}, \tilde{y})$ belongs to $\operatorname{int}(S(\tilde{y} + \Delta y))$ 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 Δy, we solve the convex MINLP

$$\begin{aligned} \max_{\Delta y \in \mathbb{R}^{n_y}, s \in \mathbb{R}^{l+1}, t \in \mathbb{R}} \quad & t && \text{(16a)}\\ \text{s.t.} \quad & t \le s_i, \quad i \in I_0, && \text{(16b)}\\ & g_i(\tilde{x}, \tilde{y} + \Delta y) + s_i \le 0, \quad i \in I, && \text{(16c)}\\ & f(\tilde{x}, \tilde{y} + \Delta y) - f(\tilde{x}, \tilde{y}) + s_0 \le 0, && \text{(16d)}\\ & \Delta y \in \mathbb{Z}^{n_y}, && \text{(16e)}\\ & s_i \ge 0, \quad i \in I_0, && \text{(16f)} \end{aligned}$$

which is an extension of the type-1 scoop problem in Wang and Xu (2017) to the nonlinear case. Problem (16) is convex in our setting because we assume that f and g_i are jointly convex. Hence, the left-hand sides of Constraints (16c) and (16d) are convex in Δy. 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 (16) such that (16c) and (16d) are strictly satisfied, i.e., all slack variables s_i for $i \in I_0$ are strictly positive, then $(\tilde{x}, \tilde{y}) \in \operatorname{int}(S(\tilde{y} + \Delta y))$ holds. To achieve this, 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 Δy variables because $\tilde{y} + \Delta y$ has to be integer for being able to apply Theorem 3.1. We now derive some properties of Problem (16).

Proposition 3.9. *Let $(\tilde{x}, \tilde{y}) \in \Theta \subseteq \Omega$ be given. Then, Problem (16) is always solvable and the solution $(\Delta y^*, s^*, t^*)$ of Problem (16) yields a non-negative optimal objective value t^*.*

Proof. Due to Assumption 2.2, the constraints in (16c) bound the Δy-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 (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 (16) and yields a non-negative objective value, i.e., the feasible region of (16) is non-empty. With the theorem of Weierstraß we thus get the existence of an optimal solution for the scoop problem. □

Proposition 3.10. *Let $(\tilde{x}, \tilde{y}) \in \Theta \subseteq \Omega$ be an integer-feasible point. Furthermore, let $(\Delta y^*, s^*, t^*)$ be an optimal solution of Problem (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 6. □

Proposition 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 of Problem (16) that is parameterized by $(\tilde{x}, \tilde{y})$ and suppose $t^* = 0$ holds. Then, there exists no $\Delta y \in \mathbb{Z}^{n_y}$ for which 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 there is at least one slack variable s_i^*, $i \in I_0$, with $s_i^* = 0$. Therefore, at least one inequality constraint of $S(\tilde{y} + \Delta y^*)$ is satisfied with equality at the point $(\tilde{x}, \tilde{y})$ and hence this point is on the boundary of the bilevel-free set $S(\tilde{y} + \Delta y^*)$. Since t is maximized, there is no other bilevel-free set containing $(\tilde{x}, \tilde{y})$ in its interior. □

Proposition 3.10 requires that the solution $\Delta\tilde{y}$ of Problem (16) has a strictly positive objective value. However, it is not guaranteed that such a solution exists. In the case $t^* = 0$, we cannot satisfy the assumptions of Theorem 3.4, and, hence, it may be impossible to compute a disjunctive cut as defined in Definition 3.3. Therefore, in this scenario, we use an integer no-good cut to separate the bilevel-infeasible point; see Section 4.2.

4. A Branch-and-Cut Method

In this section, we describe how to use the results of Section 3 in a branch-and-cut procedure to solve convex integer nonlinear bilevel problems. A general branch-and-bound framework for mixed-integer linear bilevel problems can be found in Fischetti et al. (2018). A formal listing of our method is given in Algorithm 1.

Algorithm 1: Processing Node j

```
Input: A node problem of the form ($N_j$) and an incumbent value $u$.
1  Solve node problem ($N_j$).
2  if Problem ($N_j$) is infeasible then
3      fathom the current node.
4  Let $(x^j, y^j)$ denote the solution of Problem ($N_j$).
5  if $F(x^j, y^j) \geq u$ then
6      fathom the current node.
7  if $(x^j, y^j) \notin \mathbb{Z}^n$ then
8      apply integrality branching.
9  Determine $\bar{y} \in \arg\min_{y \in \mathbb{Z}^{n_y}} \{f(x^j, y) \colon g(x^j, y) \leq 0\}$ and $\Phi(x^j)$.
10 if $G(x^j, \bar{y}) \leq 0$ then
11     The point $(x^j, \bar{y})$ is bilevel-feasible. Set $u \leftarrow \min\{u, F(x^j, \bar{y})\}$.
12 if $f(x^j, y^j) > \Phi(x^j)$ then
13     solve Problem (16) to obtain an optimal solution $(\Delta y^j, s^j, t^j)$.
14     if $t^j > 0$ then
15         set $\Theta := \Omega_j$ and $\hat{y} := y^j + \Delta y^j$.
16         for $k = 0, 1, 2, \dots$ do
17             solve the (RCGP) and obtain the solution $(\alpha^k, \beta^k, \tau^k)$.
18             Given $(\alpha^k, \beta^k, \tau^k)$, solve the subproblems (11) and (12) for all
               $i \in I_0$. Let $I_{0,j}^{\text{feas}}$ denote the index set of the feasible subproblems.
19             if $I_{0,j}^{\text{feas}} = \emptyset$ then
20                 fathom the current node.
21             else
22                 compute solutions $(x^i, y^i)^k$ with $i \in I_{0,j}^{\text{feas}}$. Set
                   $\mathcal{M} := \{(x^i, y^i)^k : (\alpha^k)^\top (x^i)^k + (\beta^k)^\top (y^i)^k - \tau^k > 0,\ i \in I_{0,j}^{\text{feas}}\}$.
23                 if $\mathcal{M} \neq \emptyset$ then
24                     Set $\mathcal{Z}^{k+1} \leftarrow \mathcal{Z}^k \cup \mathcal{M}$.
25                 else
26                     add the cut $\alpha^k x + \beta^k y - \tau \leq 0$ to Problem ($N_j$) and go to
                       Line 1.
27     else
28         derive an integer no-good cut and add it to Problem ($N_j$); see
           Section 4.2. Go to Line 1.
29 else
30     The node solution $(x^j, y^j)$ is bilevel-feasible. Set $u \leftarrow \min\{u, F(x^j, y^j)\}$.
       Fathom the current node.
```

Starting with the continuous HPR (5) as the root-node relaxation, we solve in every node j of the branch-and-bound tree problems of the form

$$\min_{(x,y)\in\Omega_j} F(x,y), \tag{N_j}$$

where $\Omega_j := \Omega \cap \{(x,y) \in \mathbb{R}^n : A^j x + B^j y \leq a^j\}$; see Line 1. The linear constraints $A^j x + B^j y \leq a^j$ contain cuts that we already added, branching decisions, and potential other cuts of the underlying MINLP solver. Note that due to Assumption 2.1, each node problem is bounded. As stated in Section 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 (1) otherwise.

As usual in branch-and-bound, we fathom a node j if it is infeasible; see Lines 2 and 3. Otherwise, we denote a solution of Problem (N_j) with (x^j, y^j); see Line 4. Note that in our setting we can always obtain a node solution (x^j, y^j), which is an extreme point of Ω_j. This is because we minimize a linear function over a continuous, convex, and bounded set. If the objective value of the node solution (x^j, y^j) exceeds an upper bound u for the objective value of the bilevel-optimal solution, we prune the node j; see Lines 5 and 6. Otherwise, we check if the solution is integer and if it is fractional, we perform the branching step in Line 8 of Algorithm 1.

On the other hand, if it is integer-feasible, we compute an optimal follower's response $\bar{y}$ and the corresponding objective value $\Phi(x^j)$ for the leader's decision x^j; see Line 9. If the point $(x^j, \bar{y})$ satisfies every constraint of the upper level, it is bilevel-feasible and, hence, we update the incumbent u with the minimum of u and $F(x^j, \bar{y})$; see Lines 10 and 11. In Line 12 we check if the integer-feasible node solution (x^j, y^j) is bilevel-feasible by comparing its lower-level objective function value with the optimal objective function value of the x^j-parameterized lower level. If the point (x^j, y^j) is bilevel-feasible, we update the incumbent u and prune the current node; see Line 30. Note that at this point we need to update the incumbent to hedge against the case that (x^j, y^j) is bilevel-feasible but we obtain a $\bar{y} \neq y^j$ such that $(x^j, \bar{y})$ violates the upper-level constraints. Then 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 $x \in \tilde{\Omega}_x$.

If the node solution (x^j, y^j) is integer-feasible but bilevel-infeasible, we compute a disjunctive cut separating the node solution (x^j, y^j) from all bilevel-feasible points inside Ω_j. Therefore, we first solve the scoop problem (16) with $(\tilde{x}, \tilde{y}) := (x^j, y^j)$ and obtain an optimal solution $(\Delta y^j, s^j, t^j)$; see Line 13. If the optimal objective value t^j is strictly greater than zero, we have found a bilevel-free set $S(y^j + \Delta y^j)$ such that the node solution (x^j, y^j) is in its interior. Hence, the assumptions in Theorem 3.4 are satisfied and we compute a disjunctive cut to separate the bilevel-infeasible node solution by using the adversarial approach as discussed in Section 3.2.

Iteratively, we first solve the (RCGP) to obtain a hyperplane separating the node solution (x^j, y^j) from every point in $\mathcal{Z}^k$ for the current iteration k; see Lines 16 and 17. Afterward, we check if the computed hyperplane separates (x^j, y^j) from $\Omega_j \cap \mathbb{Z}^n \cap (\bigcup_{i\in I_0} \mathcal{D}_i(y^j + \Delta y^j))$. Therefore, in Line 18 of Algorithm 1, we determine for every set $\mathcal{D}_i(y^j + \Delta y^j)$, $i \in I_0$, a point $(x^i, y^i)^k \in \Omega_j \cap \mathbb{Z}^n \cap \mathcal{D}_i(y^j + \Delta y^j)$, which maximizes the value of the left hand-side of the given hyperplane. We do this by solving the subproblems (11) and (12) with $\Theta := \Omega_j$ and $\hat{y} := y^j + \Delta y^j$. Since the feasibility of these subproblems depends on Ω_j and $y^j + \Delta y^j$, we denote the index set of feasible subproblems with $I_{0,j}^{\text{feas}}$.

If each of the subproblems is infeasible, i.e., $I_{0,j}^{\text{feas}} = \emptyset$, we prune the current node due to Lemma 3.6; see Line 20. This is because Ω_j does not contain any bilevel-feasible point. The correctness of this pruning step is shown in Lemma 4.1, below. Otherwise, we consider the solutions $(x^i, y^i)^k$ with $i \in I_0^{\text{feas}}$ of 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^j, y^j); see Line 23. If this is the case, we extend $\mathcal{Z}^k$ by those points; see Line 24. Otherwise, we add the hyperplane as a local constraint to the node problem (N_j), i.e., it only affects the nodes of the subtree rooted in node j. 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.

If, on the other hand, $t^j = 0$ holds, there is no bilevel-free set containing the node solution (x^j, y^j) in its interior; see Proposition 3.11. Hence, we derive an integer no-good cut and add it as a local constraint to the node problem (N_j); see Line 28. A derivation of such an integer no-good cut is given in Section 4.2.

4.1. Correctness of the Method.

The following lemma makes a statement about the correctness of the pruning step in Line 20.

Lemma 4.1. *Let Ω_j be the feasible region of node j and let $S(\hat{y})$ be any bilevel-free set. If the subproblems (11) and (12) with $\Theta := \Omega_j$ are infeasible for all $i \in I$, then we can prune node j.*

Proof. Let $\Theta := \Omega_j$. If the subproblems (11) and (12) are infeasible for all $i \in I$ and any $\hat{y} \in \mathbb{Z}^n$, Problem (9) is infeasible as well. Due to Lemma 3.6 we have $\Omega_j \cap \mathbb{Z}^n \subseteq \operatorname{int}(S(\hat{y}))$. Therefore, Ω_j is bilevel-free and, hence, we can prune node j. □

Note that in Line 20 of Algorithm 1, we check the feasibility of the subproblems (11) and (12), which are parameterized by $\hat{y} := y^j + \Delta y^j$; see Line 15. As stated in Lemma 4.1, it is also possible to use any other parameterization $\hat{y} \in \mathbb{Z}^n$. We discuss this briefly in Section 5.1. With the above lemma we finally show the correctness of Algorithm 1.

Theorem 4.2. *Suppose that the Assumptions 2.1 and 2.2 hold. Then, the branch-and-cut procedure based on Algorithm 1 terminates after a finite number of steps with a globally optimal solution of the bilevel problem (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 (1c) for every parameterization $x \in \tilde{\Omega}_x$, and, therefore, also to the bilevel problem (1). Let j be an arbitrary node in the branch-and-bound tree. The set $\Omega_j \cap \mathbb{Z}^n$ is finite due to Assumption 2.1. Since every node problem (N_j) is convex, the node solution (x^j, y^j) we obtain in Line 4 of Algorithm 1 is globally optimal for Problem (N_j). Let $\mathcal{B}^j = \{(x, y) \in \mathbb{Z}^n \colon f(x, y) \leq \Phi(x)\} \cap \Omega_j$ be the set of bilevel-feasible points in the feasible region of node j. If we get a node solution $(x^j, y^j) \in \mathbb{Z}^n$ that is bilevel-infeasible, i.e., $(x^j, y^j) \notin \mathcal{B}^j$, Algorithm 1 either adds a constraint to Ω_j, which excludes at least this point from the finite set $\Omega_j \cap \mathbb{Z}^n$, see Line 26, or prunes the node j; see Line 20. Note that the pruning step is correct due to Lemma 4.1.

From Lemma 3.7, we know that computing a disjunctive cut only takes finitely many steps. The same holds true for computing an integer-no-good cut, which can easily be seen in Section 4.2. Furthermore, it only takes finitely many cuts to separate every integer-feasible but bilevel-infeasible point in Ω_j. A complete evaluation of the branch-and-bound tree also takes finitely many steps. Therefore, if $\bigcup_j \mathcal{B}^j$ is not empty, the algorithm terminates after a finite amount of steps with a bilevel-optimal solution

$$(x^*, y^*) \in \operatorname*{arg\,min}_{x,y} \left\{ F(x,y) \colon (x,y) \in \bigcup_j \mathcal{B}^j \right\}.$$

Otherwise, if $\bigcup_j \mathcal{B}^j$ is empty, we need only finitely many steps to exclude every integer-feasible point in Ω, which results in a correct indication of infeasibility. □

Note that the correctness of Algorithm1 remains true even if we only use integer no-good cuts. This is due to finiteness of $\Omega \cap \mathbb{Z}^n$; see Assumption 2.1.

4.2. **Integer No-Good Cuts.** We now briefly discuss how to compute an integer no-good cut in the case that the optimal objective function value t^j for a node j is zero, see Lines 14 and 28 of Algorithm 1, and we cannot compute a disjunctive cut; see Proposition 3.11 and Theorem 3.4. For binary variables x and y we use the inequality

$$\sum_{i \in N_x : x_i^j = 0} x_i + \sum_{i \in N_x : x_i^j = 1} (1 - x_i) + \sum_{i \in N_y : y_i^j = 0} y_i + \sum_{i \in N_y : y_i^j = 1} (1 - y_i) \geq 1$$

with $N_x := \{1, \dots, n_x\}$ and $N_y := \{1, \dots, n_y\}$ to cut off the bilevel-infeasible node solution (x^j, y^j). In the case of integer variables x and y with bounds $l_x \leq x \leq u_x$ and $l_y \leq y \leq u_y$, we represent them with binary variables v and w first, i.e.,

$$x_i = -2^{m_x} v_{i,m_x} + \sum_{k=0}^{m_x - 1} 2^k v_{i,k}, \quad y_i = -2^{m_y} w_{i,m_y} + \sum_{k=0}^{m_y - 1} 2^k w_{i,k}$$

with

$$m_x := \lceil \log_2 (\max\{|l_x|, |u_x|\}) + 1 \rceil, \; m_y := \lceil \log_2 (\max\{|l_y|, |u_y|\}) + 1 \rceil.$$

5. Further Algorithmic Techniques

Now we discuss further algorithmic techniques, which we implemented in Algorithm 1 to enhance our method.

5.1. **Sibling Node Pruning.** In Line 20 of Algorithm 1, we implemented a pruning technique based on Lemma 3.6. Therefore, we check the feasibility of the subproblems (11) and (12) that are parameterized by $y^j + \Delta y^j$. This is equivalent to check if $\Omega_k \cap \mathbb{Z}^n \subseteq \operatorname{int}(S(y^j + \Delta y^j))$ holds. As stated in Lemma 4.1, one can use any other bilevel-free set $S(\hat{y})$ different from $S(y^j + \Delta y^j)$ as well. Based on this, if a node j gets pruned due to Lemma 3.6 in Line 20 of Algorithm 1, we can check if $\Omega_k \cap \mathbb{Z}^n \subseteq \operatorname{int}(S(y^j + \Delta y^j))$ holds. Here, Ω_k is the feasible region of the sibling node k of node j and Δy^j is a solution to the scoop problem (16) parameterized by a node solution (x^j, y^j) of node j. We do this by checking the feasibility of the subproblems (11) and (12) with $\Theta := \Omega_k$ and $\hat{y} := y^j + \Delta y^j$. If we verify that the sibling node k of j is also contained in $S(y^j + \Delta y^j)$, we prune node k. 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 j. That is because the sets Ω_k and Ω_j only differ in the bounds of one variable, i.e., their difference is as small as possible. This approach is useful in scenarios in which the sibling node k 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 1.

5.2. **Initializing $\mathcal{Z}^0$.** In Line 17, we solve the relaxed cut-generating problem (RCGP). In the first iteration $k = 0$, one can initialize the set $\mathcal{Z}^0$ with the empty set. The following lemma gives a condition under which one can initialize $\mathcal{Z}^0$ with a bilevel-feasible point.

Lemma 5.1. *Let $(\hat{x}, \hat{y})$ be a bilevel-feasible point, which is feasible for node j, i.e., $(\hat{x}, \hat{y}) \in \Omega_j$. Then, $(\hat{x}, \hat{y}) \in \Omega_j \cap \mathbb{Z}^n \cap (\bigcup_{i \in I_0} \mathcal{D}_i(\hat{y}))$ holds.*

Proof. If the point $(\hat{x}, \hat{y}) \in \Omega_j$ is bilevel-feasible, it is also integer-feasible. Furthermore, it is not contained in $S(y^j + \Delta y^j)$ and with (7) we have $(\hat{x}, \hat{y}) \in \bigcup_{i \in I_0} \mathcal{D}_i(\hat{y})$. □

One can make use of this lemma as follows. When a bilevel-feasible point $(x^j, \bar{y})$ in Line 9 of Algorithm 1 is computed, one can check if $(x^j, \bar{y}) \in \Omega_j$ holds. If this is the case, one can initialize $\mathcal{Z}^0$ with that point. However, in our setup it is too costly to be implemented, so this remains a theoretical result.

5.3. **Refinement Procedure.** As in Fischetti et al. (2018), we can apply a refinement procedure to our algorithm to obtain better upper bounds for the optimal objective function value. If the lower-level objective function is not strictly convex, we may not have uniqueness of the solution of the x^j-parameterized lower-level problem. In this case, after computing $\Phi(x^j) \in \mathbb{R}$ in Line 9 of Algorithm 1, we can solve a restricted HPR in which we temporarily fix all upper-level variables $x = x^j$ and add the constraint $f(x^j, y) \leq \Phi(x^j)$. Solving this restricted HPR leads to the x^j-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. We will, however, later not test this idea numerically since almost all of our tested instances do not have multiple lower-level solutions.

5.4. **Recycling Bilevel-Free Sets.** In Algorithm 1, before solving the x^j-parameterized lower level for an integer-feasible node solution (x^j, y^j) in Line 9, one could check if $(x^j, y^j) \in S(\hat{y})$ holds for any bilevel-free set $S(\hat{y})$ observed so far. If this is the case, the node solution (x^j, y^j) is bilevel-infeasible. Hence, we do not need to solve the lower-level problem. Instead, we then check if (x^j, y^j) is inside the interior of that bilevel-free set and, if it is, we also do not need to solve the scoop problem because we can simply use the set $S(\hat{y})$ to derive the disjunctive cut. This may not lead to the optimal bilevel-free set in terms of maximizing the distance of (x^j, y^j) to its boundary as it is done in the scoop problem (16). But in return, we may save computational effort; see Table 8 in Appendix B.

5.5. **Handling the Subproblems (11) and (12).** The feasibility of the subproblems (11) and (12) does not depend on the parameters α^k, β^k, and τ^k. Hence, if one of those problems is infeasible for $k = 0$, we do not solve them again for any iteration $k \geq 1$.

If we obtain a non-positive upper bound for the objective value of one of the subproblems (11) and (12), we terminate the solution process for this specific subproblem. This is because there is no point that is feasible for the subproblem and that lies on the wrong side of the hyperplane.

Furthermore, we can optionally decide if we solve the subproblems (11) and (12) to global optimality in every iteration k or if we terminate the solution process of a problem as soon as we find a strictly positive lower bound of the objective function value. In the latter case we still obtain a point, that lies on the wrong side of the hyperplane but this point may not have the largest distance to it. 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 in each iteration gets smaller.

We can also optionally decide if we solve every feasible subproblem in each iteration k or if we only consider those subproblems in iteration $k+1$ that led to a point with positive objective function value in iteration k. The latter case may prevent solving subproblems multiple times that are not relevant for computing the hyperplane. In iteration k', if there is no subproblem left, we perform a correction step in which we solve every feasible subproblem again for the 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.

6. 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 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 two approaches:

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

B+DC: our branch-and-cut algorithm based on disjunctive cuts and the separation procedure as described in Algorithm 1.

We also test the additional algorithmic techniques as discussed in Section 5. In Section 6.1, we describe our computational setup and discuss some implementation details. Then, in Section 6.2, we describe the instance sets used for our experiments. Finally, Section 6.3 contains a discussion of the numerical results.

6.1. **Hardware and Software Setup.**

We implemented Algorithm 1 in Python 3.9.7 and we use the branch-and-cut (BnC) framework of CPLEX 22.1.1.0. The x-parameterized lower-level problem, the scoop problem (16), and the cut-generating problem (CGP) are also solved using CPLEX 22.1.1.0. The subproblems (11) and (12) are solved using Gurobi 9.5.1. The reason for the latter is that CPLEX is not capable of solving problems having nonlinear and nonconvex constraints. CPLEX would still work for problems for which the functions appearing in the lower-level problem (18) are linear in x and, hence, the resulting subproblems (11) and (12) are convex; see Lemma 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. All cuts are implemented using the add_local function inside the LazyConstraintCallback environment of CPLEX.

We use depth-first search as our node selection strategy. This is because we need to keep track on the precise definition of the feasible region Ω_j in each node j of the BnC tree; this information is needed in Line 18 of Algorithm 1 for solving the subproblems (11) and (12). More precisely, we need to inherit DCs and INGCs from the parent node. Both Gurobi and CPLEX do not provide the exact location of a node in the BnC tree, and 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 (11) and (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 BnC 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 in the remainder of this paper.

We implemented the INGCs with the help of a binary expansion of the x- and y-variables as discussed in Section 4.2. It is worth mentioning that for the problem type we consider in our numerical study, i.e., convex-quadratic integer bilevel problems (see below for the details), one can also use the approach given in Tahernejad and Ralphs (2020), which results in convex-quadratic INGCs. This approach would use less variables but is ruled out since it is not possible to add nonlinear cuts via the add_local function inside the LazyConstraintCallback environment of CPLEX.

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. Moreover, we prioritize branching on x- and y-variables compared to the auxiliary binary variables introduced for the binary representation of these variables.

All computations are 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 use a single Intel XEON SP 6126 core with 2.6 GHz and a maximum of 32 GB RAM.

6.2. **Test Instances.**

We consider problems of the form

$$\begin{aligned} \min_{(x,y)\in\mathbb{Z}^n} \quad & c_x^\top x + c_y^\top y \\ \text{s.t.} \quad & Ax + By \le a, \\ & y \in S(x), \end{aligned} \tag{17}$$

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

$$\min_{y\in\mathbb{Z}^{n_y}} \quad \frac{1}{2}y^\top Q y + d^\top y \tag{18a}$$

$$\text{s.t.} \quad (Cx + Dy)_i \le b_i, \quad i = 1, \dots, l-1, \tag{18b}$$

$$(Cx + Dy)_l + \frac{1}{2}y^\top P y \le b_l, \tag{18c}$$

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. 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 got translated into quadratic bilevel multiple knapsack problems (QBMKP); see Gaar et al. (2023) for further details. From those instances we used the 100 binary and the 100 integer QBMKP instances with a single lower-level constraint. Furthermore, for each of those instances, we constructed a new instance that contains the first $\lfloor (m+l)/2 \rfloor$ constraints of the HPR in the upper level and we moved the last $\lceil (m+l)/2 \rceil$ constraints to the lower level. If an instance only has two constraints in total, we moved 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

TABLE 1. Overview of the original test instances per instance class

			n_x		n_y		m		l	
	Reference	Size	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 paper	200	5	79	2	52	1	10	2	5
DENEGRE	DeNegre (2011)	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)	30	500	700	500	700	200	280	200	280

TABLE 2. Overview of the modified test instances per subset

	without quadratic constraint				with quadratic constraint			
	solvable	time limit	trivial	$\sum$	solvable	time limit	trivial	$\sum$
QBMKP_sim	45	144	11	200	54	121	25	200
QBMKP_opp	52	147	1	200	105	94	1	200
QBMKP_50/50_sim	54	135	11	200	52	63	85	200
QBMKP_50/50_opp	52	147	1	200	33	106	61	200
DENEGRE	0	0	0	0	29	15	6	50
XU	0	0	0	0	123	0	7	130

additionally created the instances with a convex-quadratic constraint contained in the lower level. Starting from a QBMKP instance, we converted the last linear constraint from the lower-level problem $(Cx + Dy)_l \leq b_l$ into a convex quadratic one $(Cx + Dy)_l + \frac{1}{2}y^\top Py \leq b_l$ 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 $\sigma = ||b_l||_\infty$, see the MATLAB function of Kleinert and Schmidt (2021).[2] The value of the right-hand side is set as $b_l \leftarrow b_l + |b_l|$.

Moreover, we tested all of the instances w.r.t. minimization and maximization of the upper-level objective function. The upper-level objective function coefficients are always positive since they represent the value gained for packing the knapsack. The lower-level objective function of the QBMKP instances only consist of a quadratic term, i.e., $d = 0$. Therefore, minimizing the upper-level objective function leads to a case where both players objective functions point towards a similar direction w.r.t. the y-variables while maximizing the upper-level objective function leads to the case that the two objectives point in opposite directions. Hence, we can investigate both scenarios, either when both players optimize in similar or in opposite directions. These instance sets are denoted by "sim" and "opp", respectively.

For our experiments we also used a subset of the MILP-MILP instances from Kleinert and Schmidt (2021). To fit in our setting, we applied integrality constraints on all x- and y-variables and randomly generated positive semi-definite matrices Q and P for the lower-level objective function (18a) and the lower-level constraint (18c) in the same way as we did it with the QBMKP instances. Note that we used $\sigma = ||d||_\infty$ to generate the entries of matrix Q. Again, we increased the right-hand side of the quadratic constraint by its absolute value. Since almost all of the modified instances from this collection could either not be solved within a time limit of 2 hours or were solved without using a single cut, we excluded all instance sets but the sets DENEGRE, XUWANG, and XULARGE.

The sizes of the instance sets used can be found in Table 1.

[2]The MATLAB function can be found under https://github.com/m-schmidt-math-opt/qp-bilevel-matrix-generator.

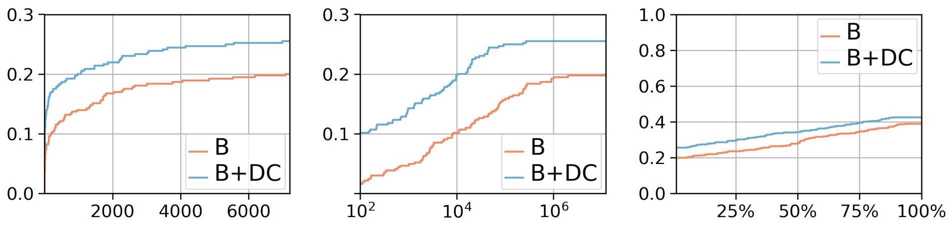

FIGURE 1. ECDFs of the instances in the class QBMKP_sim. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

6.3. **Discussion of the Results.** In the following we provide results of a computational comparison of our B+DC approach with the B approach. We divided the entire set of benchmark instances into six subsets; see Table 2. The first two subsets contain each instance of QBMKP once with and once without having a quadratic constraint in the lower level as described in Section 6.2. In the first subset, both the upper and the lower level objective function are minimized, i.e., both players optimize in a similar direction w.r.t. y because all coefficients in the objective functions are positive. Therefore, we denote this set as QBMKP_sim. In the second set, the upper level objective function is maximized, i.e., both players optimize in opposite directions w.r.t. y. We denote this second set as QBMKP_opp. The next two sets are constructed the same way but with instances of QBMKP_50/50, i.e., they are denoted with QBMKP_50/50_sim and QBMKP_50/50_opp. We consider the instance set DENEGRE as our fifth subset and the last subset contains the instances of XUWANG and XULARGE and is denoted as XU.

For the presentation of the numerical results, we exclude all instances that could be solved by both methods B and B+DC in less than one second. We also exclude the instances that B+DC solves without using a disjunctive cut and without pruning a node (as for those instances both methods are identical). We denote those instances as trivial and the other ones as non-trivial. A classification of the instances per subset is given in Table 2. Here, instances are called "solvable" if they are non-trivial and if at least one of the methods solves them within the time limit of 2 hours. They are labeled with "time limit" when none of the methods could solve them within the time limit and "trivial" if they are trivial.

The empirical cumulative distribution functions (ECDFs) w.r.t. idealized runtimes, node counts, and (relative) MIP gaps at termination of B+DC and B are given in the Figures 1–6. To get the idealized runtime of method B+DC for a given instance, we replace the runtime that it takes to solve all of the subproblems (11) and (12) successively (see Line 18 in Algorithm 1) by the maximum runtime that we need to solve one of the subproblems—hence mimicking as if we would solve all of the subproblems (11) and (12) in parallel; see Section 3.3.

The minimum, maximum, and median values for the idealized runtimes and for the node counts of the methods B and B+DC are given in Table 3 for each subset respectively. Similarly, Table 4 reports that information w.r.t. the number of DCs, INGCs, and subproblems needed, respectively.

6.3.1. *QBMKP.* As illustrated in Figure 1, which focuses on the instances in QBMKP_sim, our approach B+DC clearly outperforms the standard approach B w.r.t. the runtimes and the number of nodes. In addition, our method separates a median of 8 DCs and no INGCs while the method B has a median of 11 724 INGCs. Moreover, B+DC solves 25.5 % of the non-trivial instances while the standard approach only solves 20.1 %; see Table 3. For the unsolved instances of QBMKP_sim, the MIP

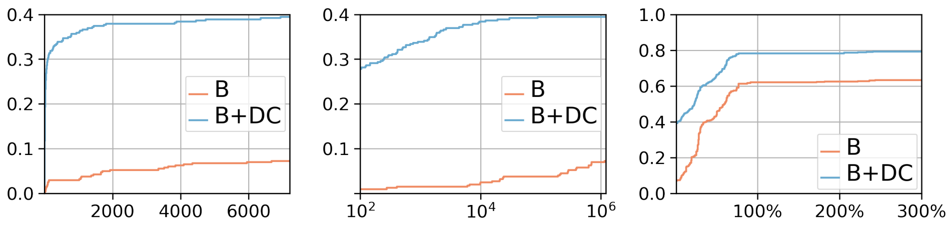

FIGURE 2. ECDFs of the instances in the class QBMKP_opp. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

TABLE 3. Detailed results for the "solvable" instances per instance class

			idealized runtime in s			nodes		
	method	% solved	min	max	median	min	max	median
QBMKP_sim	B	20.1	0.1	7 082.1	185.8	0	11 738 740	8 173
	B+DC	25.5	0.1	7 070.9	30.6	0	267 923	701
QBMKP_opp	B	7.3	2.2	6 663.5	1 083.7	0	1 149 025	23 833
	B+DC	39.4	0.1	6 941.6	14.4	0	88 297	0
QBMKP_50/50_sim	B	33.9	0.1	6 835.0	32.7	4	7 724 085	3 032
	B+DC	32.9	0.6	7 068.6	55.2	0	107 491	1 997
QBMKP_50/50_opp	B	12.4	0.1	6 353.9	138.8	0	991 462	16 333
	B+DC	24.9	1.5	5 073.2	94.6	0	359 734	554
DENEGRE	B	45.5	0.1	6 124.0	293.3	5	340 344	13 453
	B+DC	65.9	1.4	5 670.4	156.8	0	174 201	4 186
XU	B	100	1.0	5 723.6	258.7	45	39 423	3 381
	B+DC	95.9	1.7	6 800.1	556.0	37	31 920	2 859

gaps are rather similar for the two approaches, with B+DC having a slight advantage over B.

Figure 2 illustrates the ECDFs of the non-trivial instances in QBMKP_opp. Again, B+DC outperforms approach B, but the difference in performance is much more pronounced than for the QBMKP_sim instances. When both the upper- and lower-level objective function point in a similar direction, it is likely that a solution to a node problem in the BnC tree is "close" to a bilevel-feasible point and, hence, only a few INGCs are required. Therefore, also method B performs well in those cases. Conversely, when both objective functions point in opposite directions, a BnC algorithm using only INGCs will likely need many more cuts to separate bilevel-infeasible points in each node. This observation is confirmed in Table 4, where the standard approach B needs a lot more INGCs to prove optimality for the instances in QBMKP_opp, compared to the instances in QBMKP_sim. While the performance of a BnC method using only INGCs seems to highly depend on the correlation of the two objective functions, method B+DC using DCs performs well independent of that.

It can be seen in Table 3 that the standard approach B only solves 7.3 % while the method B+DC solves 39.4 % of the non-trivial QBMKP_opp instances. Moreover, the median number of BnC nodes for B+DC is zero, i.e., most of the instances are solved in the root node, whereas median for B is 23 833 nodes. One possible reason for this drastic difference lies in the fact that around 11 % of the non-trivial instances of this subset are bilevel-infeasible but their high-point relaxations are feasible. Hence, our B+DC method detects the bilevel-infeasibility already at the

TABLE 4. Total number of cuts and subproblems for the "solvable" instances per instance class

		DCs			INGCs			Subproblems (11) and (12)		
	Method	Min	Max	Median	Min	Max	Median	Min	Max	Median
QBMKP_sim	B	0	0	0	1	889 875	11 724	0	0	0
	B+DC	0	11 586	8	0	163	0	2	258 236	572
QBMKP_opp	B	0	0	0	328	1 107 808	169 175	0	0	0
	B+DC	0	11 082	10	0	16	0	2	252 200	382
QBMKP_50/50_sim	B	0	0	0	1	415 973	495	0	0	0
	B+DC	0	6 684	7	0	155	0	3	228 072	1 045
QBMKP_50/50_opp	B	0	0	0	3	967 094	13 986	0	0	0
	B+DC	2	5 623	66	0	2 411	0	20	230 313	4 096
DENEGRE	B	0	0	0	2	209 584	10 047	0	0	0
	B+DC	1	6 876	114	0	46 745	20	147	1 154 937	32 823
XU	B	0	0	0	1	11 240	27	0	0	0
	B+DC	0	193	3	0	72	1	45	69 030	3 145

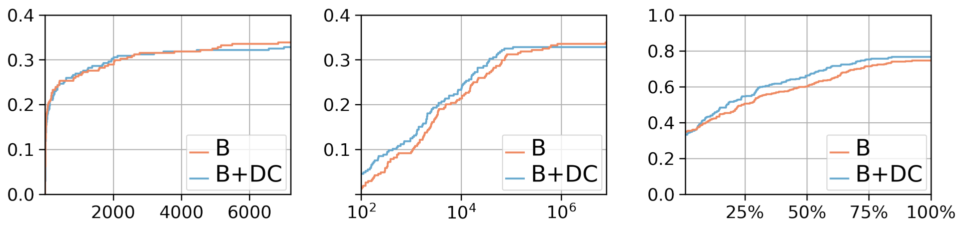

FIGURE 3. ECDFs of the instances in the class QBMKP_50/50_sim. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

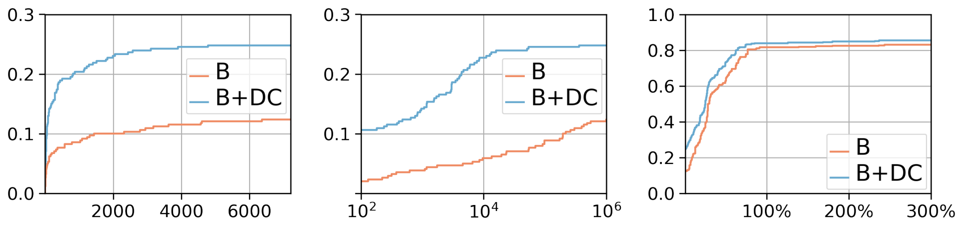

FIGURE 4. ECDFs of the instances in the class QBMKP_50/50_opp. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

root node (see Line 19 of Algorithm 1) and prunes it (see Line 20). However, the approach B is not capable of detecting bilevel-infeasibility and, hence, removes 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.

6.3.2. *QBMKP_50/50.* Figure 3 reports the ECDF for QBMKP_50/50_sim instances. We observe that both methods B and B+DC perform equally well for this subset. However, if we consider the QBMKP_50/50_opp instances, then our approach again clearly outperforms the standard approach; see Figure 4. As discussed above, a lot more INGCs are needed when both objective functions point towards an opposite direction, as opposed to the problems where both objective functions point towards a similar direction. This can also be verified in Table 4.

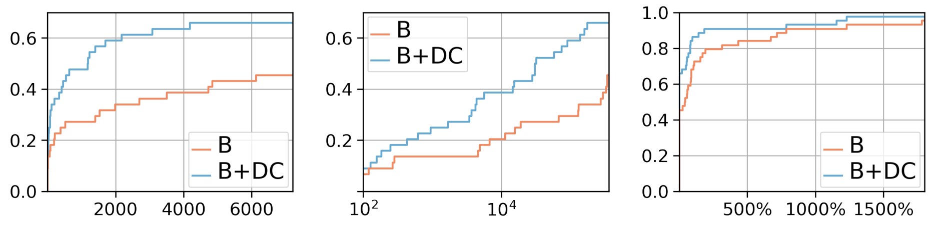

FIGURE 5. ECDFs of the instances DENEGRE. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

However, the performance difference between the two methods for the instances in QBMKP_50/50 is not as large as for the instances in QBMKP; compare Figure 1 vs. 3 and Figure 2 vs. 4. This may be because shifting more upper-level constraints to the lower level shrinks the feasible region of the x-parameterized lower-level problems. Hence, finding the optimal follower's response to a given leaders decision x takes less INGCs the more constraints are in the lower level. This could explain why the method B performs better for the instances in QBMKP_50/50 than for the instances in QBMKP.

Additionally, having more constraints in the lower level leads to more subproblems in (12) that have to be solved in Line 18 of Algorithm 1 in each iteration k. If one compares the median values of the total number of subproblems (11) and (12) needed for the method B+DC of the instances in QBMK with those of the instances in QBMK_50/50, one can see that they are significantly higher for the latter instances; see Table 4. The same holds for the medians of the running times for the method B+DC. Therefore, B+DC may perform better on instances with less constraints in the lower level. However, the total number of subproblems (11) and (12) does not only depend on the number of lower-level constraints but also on the number of disjunctive cuts needed. Hence, an instance with many lower-level constraints that only needs a few disjunctive cuts to be solved may need less subproblems to be solved than a problem that has only a few lower-level constraints but that needs a lot of disjunctive cuts to be solved. In Table 4 one can see that the median of the number of disjunctive cuts needed for the instances in the subset QBMKP_50/50_opp (66) is over six times as large as for the instances in the subset QBMKP_opp (10) while the median of the number of subproblems increased more than tenfold from 382 to 4096. On the other hand, the median of the number of disjunctive cuts needed for the instances in the subset QBMKP_50/50_sim (7) is almost the same as for the instances in the subset QBMKP_sim (8) while the median of the number of subproblems almost doubles from 572 to only 1045. Therefore, it can be seen that the total number of subproblems (11) and (12) needed depends even more on the number of disjunctive cuts needed than on the number of lower-level constraints.

6.3.3. *DENEGRE and XU.* Figure 5 illustrates the ECDFs for the instances in DENEGRE, where we see that B+DC outperforms B w.r.t. runtimes and node counts. The same holds for the number of cuts used; see Table 4. However, for the instances in XU (see Figure 6), B+DC is slightly outperformed by B w.r.t. the running times and also slightly outperformed w.r.t. the node counts, but it still needs less cuts to solve the problems; see Table 4. This is because the instances in XU are rather easy compared to the instances in all other subsets. Indeed, all instances of the subset XU could be solved within the time limit (see Table 2) and both methods B and B+DC solve over 95 % of the non-trivial instances; see Table 3. The advantage

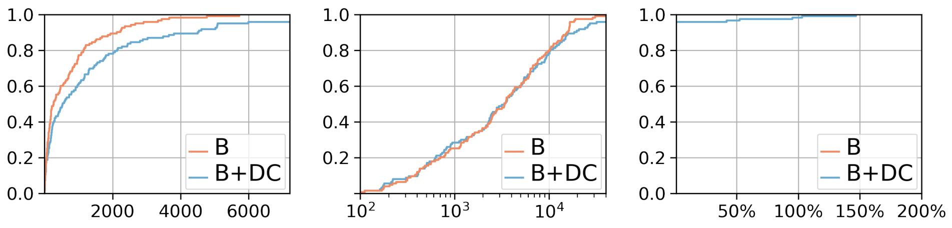

FIGURE 6. ECDFs of the instances XU. Left: idealized runtimes. Middle: node counts. Right: MIP gaps.

of INGCs 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. Regarding the DCs, 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 B is 27, see Table 4, which is by far the lowest value across all subsets. Similarly, it only takes three DCs and one INGC in the median to solve these instances with B+DC; see Table 4. However, the median of the running times of B+DC is more than twice as large as the median of B. Additionally, these running times are much larger compared to the other median values for other subsets; see Table 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 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 method B+DC, because not only the x-parameterized lower-level (1c), used in both B and B+DC, but also the scoop problem (16), the (CGP), and the subproblems (11) and (12) are affected by the size of the input instance.

6.3.4. *Summary.* Our empirical study shows that B+DC outperforms the BnC algorithm that uses only INGCs in most of the tested instances. This is true in terms of node counts, running times, and the number of cuts used. Furthermore, there are instances where both methods are equally good, e.g., the instances in QBMKP_50/50_sim, and also instances where the standard approach performs better than B+DC, e.g., the instances in XU. However, those 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 that comes with separating disjunctive cuts. For the instances that are harder to solve, the number of integer no-good cuts outnumbers the number of disjunctive cuts, making B+DC a clearly more efficient method than the branch-and-cut method using only integer no-good cuts.

For the QBMKP instances, the benefit of using B+DC over B is largest when the two players optimize in opposite directions w.r.t. y. This is also the more realistic and the more interesting case in bilevel optimization compared to the setting in which both players optimize in similar directions. B+DC will likely perform better for instances that have fewer lower-level constraints than for instances with many lower-level constraints due to the time it needs to solve the subproblems (11) and (12) in each iteration k in Line 18. However, the effect of the number of lower-level constraints on the performance of the method does not seem to be relevant for instances of small to medium size.

We also compared the computational performance of instances with and without an additional quadratic constraint in the lower level, see Appendix A. This analysis shows that there is no significant trend depending on having a quadratic constraint benefits our method compared to method B or not. This is because modifying a constraint as described in Section 6.2 does not a priori decrease the feasible region of the follower and, hence, we do not have the same effect as when we shift upper-level constraints to the lower level as it is discussed in Section 6.3.2.

Finally, we tested the further algorithmic techniques discussed in Section 5 for the subsets DENEGRE, XU, and on the instances in QBMK_opp and QBMK_50/50_opp that have a quadratic constraint. None of the described enhancements lead to a significant improvement of our method for the instances tested in this paper. There may still be other applications for which they could make a difference. The numerical results for these tests are given in the Appendix B.

7. Conclusion

We have presented a branch-and-cut algorithm based on disjunctive cuts that is able to solve convex integer nonlinear bilevel problems. These problems contain nonlinearities both in the upper- and lower-level objective function as well as in the upper- and lower-level constraints. We derive tailored disjunctive cuts and show how to separate them. Moreover, we prove the correctness of the resulting branch-and-cut method. Furthermore, we discussed additional techniques to enhance our method that may be useful in certain scenarios; see Section 5. We also compared our approach to a branch-and-cut method that only uses integer no-good cuts and show that the benefit of using disjunctive cuts outweighs the higher computational effort that it takes to generate them, which shows the applicability of our novel approach.

Acknowledgements

The first author thanks the DFG for their support within RTG 2126 “Algorithmic Optimization”. The computations were 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 kindly acknowledge the support of RHRK.

References

Balas, E. (2018). *Disjunctive programming.* Springer. DOI: 10.1007/978-3-030-00148-3.

Bard, J. F. (2013). *Practical bilevel optimization: algorithms and applications.* Vol. 30. Springer Science & Business Media. DOI: 10.1007/978-1-4757-2836-1.

Dempe, S. and A. Zemkoho (2020). “Bilevel optimization.” In: *Springer optimization and its applications.* Vol. 161. Springer. DOI: 10.1007/978-3-030-52119-6.

DeNegre, S. (2011). “Interdiction and discrete bilevel linear programming.” PhD thesis.

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.

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.

– (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.

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.

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.

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.

Gorissen, B. L., I. Yanıkoğlu, 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.

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.

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.

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.

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.

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.

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.

Kleniati, P.-M. and C. S. Adjiman (2014a). “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.

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.

Kleniati, P.-M. and C. S. Adjiman (2014b). “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.

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.

Lodi, A., T. K. Ralphs, and G. J. Woeginger (2014). “Bilevel programming and the separation problem.” In: *Mathematical Programming* 146.1-2, pp. 437–458. DOI: 10.1007/s10107-013-0700-x.

Lodi, A., M. Tanneau, and J.-P. Vielma (2023). “Disjunctive cuts in mixed-integer conic optimization.” In: *Mathematical Programming* 199.1-2, pp. 671–719. DOI: 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.

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.

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.

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.

Tahernejad, S. and T. K. Ralphs (2020). *Valid Inequalities for Mixed Integer Bilevel Linear Optimization Problems.* Tech. rep.

Tambe, M. (2011). *Security and game theory: algorithms, deployed systems, lessons learned.* Cambridge university press. DOI: 10.5555/2124410.

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.

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.

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.

Zare, M. H., J. S. Borrero, B. Zeng, and O. A. Prokopyev (2019). “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.

Appendix A. ECDFs for the QBMKP Instances With and Without a Quadratic Constraint in the Lower Level

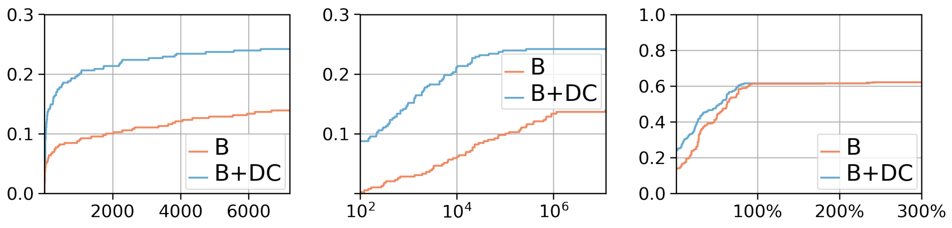

Figure 7. ECDFs of the instances in QBMKP without a quadratic lower-level constraint

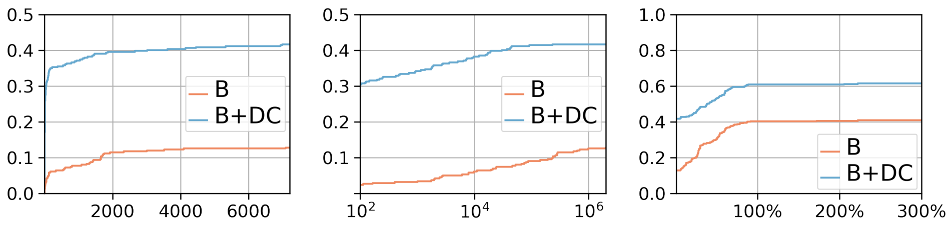

Figure 8. ECDFs of the instances in QBMKP with a quadratic lower-level constraint

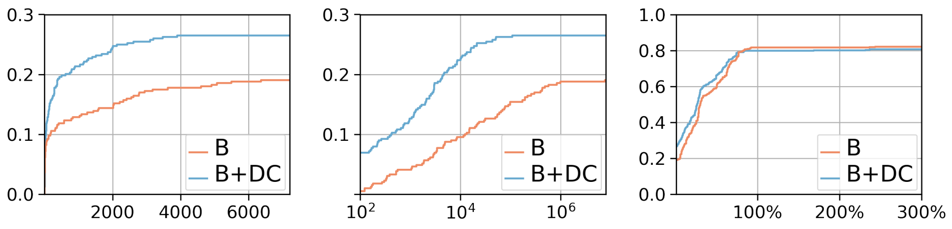

Figure 9. ECDFs of the instances in QBMKP_50/50 without a quadratic lower-level constraint

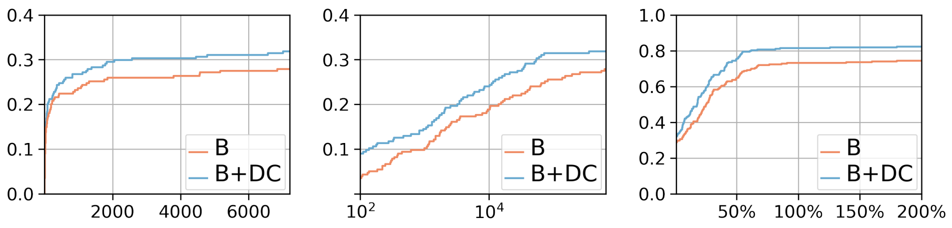

Figure 10. ECDFs of the instances in QBMKP_50/50 with a quadratic lower-level constraint

Appendix B. Numerical Results for the Techniques of Section 5

Table 5. Detailed results for Algorithm 1 with and without sibling node pruning (SNP). Here, the instances per instance class are considered that are quadratic and "solvable" with at least one node getting pruned due to Line 20 of Algorithm 1.

			idealized runtime in s			nodes		
	method	% solved	min	max	median	min	max	median
QBMKP_sim	B+DC	26.2	0.1	7 070.9	10.3	0	233 342	30
	B+DC+SNP	26.2	0.1	6 937.5	9.1	0	204 054	27
QBMKP_opp	B+DC	29.9	1.6	6 941.6	55.3	0	29 837	112
	B+DC+SNP	30.6	0.1	4 573.7	47.5	0	20 297	48
QBMKP_50/50_sim	B+DC	41.6	1.0	7 068.6	69.1	59	72 205	2 510
	B+DC+SNP	41.6	1.0	6 045.0	65.1	59	466 220	3 229
QBMKP_50/50_opp	B+DC	14.4	3.3	5 073.2	99.9	10	359 734	1 681
	B+DC+SNP	14.4	3.6	3 566.1	122.7	10	48 823	1 688
DENEGRE	B+DC	64.3	2.2	5 670.4	287.5	67	174 201	4 330
	B+DC+SNP	64.3	2.0	7 110.0	211.2	67	152 658	3 960
XU	B+DC	95.9	1.7	6 800.1	567.6	37	31 920	2 938
	B+DC+SNP	92.6	2.4	6 962.7	599.3	37	31 798	2 777

Table 6. Detailed results for Algorithm 1 with and without early termination (E) of the subproblems (11) and (12) as discussed in Section 5.5. Here, the instances per instance class are considered that are quadratic and "solvable".

			idealized runtime in s			nodes		
	method	% solved	min	max	median	min	max	median
QBMKP_sim	B+DC	29.1	0.1	7 070.9	9.3	0	233 342	0
	B+DC+E	29.7	0.1	6 899.2	11.4	0	233 342	0
QBMKP_opp	B+DC	52.8	0.1	6 941.6	5.5	0	29 837	0
	B+DC+E	54.8	0.2	4 893.9	5.1	0	28 892	0
QBMKP_50/50_sim	B+DC	42.6	0.6	7 068.6	66.3	0	72 205	2 103
	B+DC+E	41.7	0.6	6 009.5	63.3	0	72 205	2 077
QBMKP_50/50_opp	B+DC	23.0	1.5	5 073.2	28.5	0	359 734	67
	B+DC+E	22.3	2.0	2 016.0	23.9	0	53 156	64
DENEGRE	B+DC	65.9	1.4	5 670.4	156.8	0	174 201	4 186
	B+DC+E	65.9	1.5	5 320.0	143.4	0	174 201	3 911
XU	B+DC	95.9	1.7	6 800.1	556.0	37	31 920	2 859
	B+DC+E	96.7	2.1	6 757.7	523.3	37	31 798	2 938

(A. Horländer, M. Schmidt) Trier University, Department of Mathematics, Universitätsring 15, 54296 Trier, Germany
Email address: `horlaender@uni-trier.de`
Email address: `martin.schmidt@uni-trier.de`

(I. Ljubić) ESSEC Business School of Paris, Cergy-Pontoise, France
Email address: `ljubic@essec.edu`

Table 7. Detailed results for Algorithm 1 with and without strategic solving (S) of the subproblems (11) and (12) as described in Section 5.5. Here, the instances per instance class are considered that are quadratic and “solvable”.

			idealized runtime in s			nodes		
	method	% solved	min	max	median	min	max	median
QBMKP_sim	B+DC	29.1	0.1	7 070.9	9.3	0	233 342	0
	B+DC+S	29.1	0.1	6 516.5	8.6	0	233 342	0
QBMKP_opp	B+DC	52.8	0.1	6 941.6	5.5	0	29 837	0
	B+DC+S	52.8	0.1	4 156.3	4.7	0	29 873	0
QBMKP_50/50_sim	B+DC	42.6	0.6	7 068.6	66.3	0	72 205	2 103
	B+DC+S	41.7	0.8	5 995.7	59.2	0	72 205	2 077
QBMKP_50/50_opp	B+DC	23.0	1.5	5 073.2	28.5	0	359 734	67
	B+DC+S	23.0	1.5	1 860.5	30.9	0	53 229	86
DENEGRE	B+DC	65.9	1.4	5 670.4	156.8	0	174 201	4 186
	B+DC+S	65.9	1.3	3 343.4	144.0	0	174 201	4 186
XU	B+DC	95.9	1.7	6 800.1	556.0	37	31 920	2 859
	B+DC+S	96.7	1.8	6 058.6	525.6	37	31 635	2 938

Table 8. Detailed results for Algorithm 1 with and without recycling of bilevel-free sets (R) as described in Section 5.4. Here, the instances per instance class are considered that are quadratic and “solvable”.

			idealized runtime in s			nodes		
	method	% solved	min	max	median	min	max	median
QBMKP_sim	B+DC	29.1	0.1	7 070.9	9.3	0	233 342	0
	B+DC+R	26.3	0.1	6 310.7	4.5	0	236 113	0
QBMKP_opp	B+DC	52.8	0.1	6 941.6	5.5	0	29 837	0
	B+DC+R	55.3	0.1	6 056.8	3.8	0	21 188	0
QBMKP_50/50_sim	B+DC	42.6	0.6	7 068.6	66.3	0	72 205	2 103
	B+DC+R	49.6	0.6	7 030.3	67.6	0	728 602	12 607
QBMKP_50/50_opp	B+DC	23.0	1.5	5 073.2	28.5	0	359 734	67
	B+DC+R	20.9	1.1	6 195.9	14.1	0	53 115	42
DENEGRE	B+DC	65.9	1.4	5 670.4	156.8	0	174 201	4 186
	B+DC+R	68.2	1.6	5 521.8	230.1	0	191 509	5 861
XU	B+DC	95.9	1.7	6 800.1	556.0	37	31 920	2 859
	B+DC+R	96.7	1.2	5 290.3	353.3	37	41 104	3 783