A Penalty Branch-and-Bound Method for Mixed Binary Linear Complementarity Problems

A Penalty Branch-and-Bound Method for Mixed-Binary Linear Complementarity Problems

Marianna De Santis, Sven de Vries, Martin Schmidt, Lukas Winkel

Abstract. Linear complementarity problems (LCPs) are an important modeling tool for many practically relevant situations but also have many important applications in mathematics itself. Although the continuous version of the problem is extremely well studied, much less is known about mixed-integer LCPs (MILCPs) in which some variables have to be integer-valued in a solution. In particular, almost no tailored algorithms are known besides reformulations of the problem that allow to apply general-purpose mixed-integer linear programming solvers. In this paper, we present, theoretically analyze, enhance, and test a novel branch-and-bound method for MILCPs. The main property of this method is that we do not "branch" on constraints as usual but by adding suitably chosen penalty terms to the objective function. By doing so, we can either provably compute an MILCP solution if one exists or compute an approximate solution that minimizes an infeasibility measure combining integrality and complementarity conditions. We enhance the method by MILCP-tailored valid inequalities, node selection strategies, branching rules, and warmstarting techniques. The resulting algorithm is shown to clearly outperform two benchmark approaches from the literature.

1. Introduction

The linear complementarity problem (LCP) is the task to find a vector $z \in \mathbb{R}^n$ that satisfies

$$z \geq 0, \tag{1a}$$

$$q + Mz \geq 0, \tag{1b}$$

$$z^\top (q + Mz) = 0 \tag{1c}$$

or to show that no such vector exists. The problem is denoted by $\mathrm{LCP}(q, M)$ for a given real matrix $M \in \mathbb{R}^{n \times n}$ and a real vector $q \in \mathbb{R}^n$. LCPs are an important tool for the modeling and analysis of equilibrium problems in economics, mechanics, and other applied fields, but also have many important applications in mathematics itself. We refer to the seminal textbook by Cottle et al. (2009) for a general overview and to the book by Gabriel, Conejo, Fuller, et al. (2012) for a large collection of applications in energy markets. As defined above, the classic LCP is stated in terms of a continuous variable vector z. In practice, however, one also faces situations in which a subset of variables is restricted to take integer values, i.e., $z_i \in \mathbb{Z}$ for a given index set $I \subseteq \{1, \ldots, n\}$. This is relevant in situations in which an LCP solution needs to satisfy additional integrality constraints. A particular example for such a situation are market equilibrium problems (that can be modeled as LCPs) for which certain equilibrium quantities need to be integer. Other situations include those in which further combinatorial constraints are imposed on the LCP's solution such as equity-enforcement conditions for network flows or integer production levels.

Date: May 30, 2022.

2020 *Mathematics Subject Classification.* 90-08, 90Bxx, 90C11, 90C33.

Key words and phrases. Mixed-Integer Programming, Linear Complementarity Problems, Mixed-Integer Linear Complementarity Problems, Branch-and-Bound, Penalty Methods.

For applications related to mixed-integer LCPs (MILCPs) we refer to, e.g., Gabriel (2017), Gabriel, Conejo, Ruiz, et al. (2013), Gabriel, Siddiqui, et al. (2013), and Weinhold and Gabriel (2020).

1.1. **Literature Review.** Although integer solutions of LCPs have been mentioned first in 1940 by Val (1940), the study of the connection between LCPs and integer programming (IP) started in the late 1980s, when it was shown that LCPs and IP are equivalent (Pardalos 1988; Pardalos and Rosen 1988); see also Pardalos (1994) or Pardalos (1996) for survey articles that also discuss the relation of LCPs and IP problems. Some applications of LCPs with mixed-integer variables are also discussed by Pardalos and Nagurney (1990), where the authors mention, e.g., polymatrix games in pure strategies, general economic equilibria with integer activity levels, or spatial price equilibrium problems with discrete commodities. To the best of our knowledge, Pardalos and Nagurney (1990) also present the first algorithm to solve integer LCPs. However, the idea of this algorithm is simple: enumerate all continuous LCP solutions and check if any of them are purely integer-valued.

For continuous LCPs two of the classic questions are those of existence and uniqueness of solutions, which are closely related to the study of matrix classes; see, e.g., Cottle et al. (2009) for an overview. For purely integer-valued LCP solutions, the respective matrix class I has been introduced by Chandrasekaran et al. (1998) and Cunningham and Geelen (1998) and necessary as well as sufficient conditions are derived for that a given matrix is in I. The class I is particularly important for mixed-integer LCPs since it contains those integer-valued matrices for which there is an integral solution to the $\mathrm{LCP}(q, M)$, with q integral as well, whenever there is a continuous solution at all. Chandrasekaran et al. (1998) also discuss the "peeling algorithm", which is an iterative method to find integer solutions of LCPs and show that it is correct when applied to specific matrix classes. Further and more recent studies of existence of integer solutions are given by Dubey and Neogy (2018) and Sumita et al. (2018).

Compared to the larger number of theoretical papers mentioned so far, the literature on algorithms for mixed-integer LCPs is rather sparse. In the context of energy applications, Gabriel, Conejo, Ruiz, et al. (2013) reformulate the mixed-integer LCP as a mixed-integer linear problem (MILP) using disjunctive constraints and big-M constants. Unfortunately, this cannot be seen as a general-purpose method since such big-Ms are not always available. Gabriel (2017) exploits the relation of complementarity problems to the median function (Gabriel 1998; Gabriel and Moré 1997), which again is used to state a proper MILP to solve the mixed-integer LCP. This approach, however, again relies on the choice of big-M constants. Furthermore, Fomeni et al. (2019a,b) exploit reformulation-linearization techniques (RLT) to solve the mixed-integer LCP. Finally, Gabriel, Leal, et al. (2021) use purely continuous reformulations of MILCPs to solve these problems to local optimality by exploiting general-purpose nonlinear programming solvers. To sum up, almost all known solution approaches are based on a reformulation of the mixed-integer LCP so that state-of-the-art MILP solvers can be used.

1.2. **Contribution.** We present a novel branch-and-bound method that explicitly exploits the structure of mixed-integer LCPs and that has the following main properties. First, in classic branch-and-bound methods, relaxations are solved and the feasible set is tightened again by branching on, e.g., integer variables or constraints. Consequently, the objective function stays the same in the entire branch-and-bound tree but the constraint set is extended as one goes down the tree starting from the root node. Our branch-and-bound approach for mixed-integer LCPs follows a different rationale. Here, the constraint set does not change

over the entire tree and "branching" is realized by adding suitably chosen penalty terms to the objective, i.e., the number of terms in our objective function grows if we go down the tree starting from the root node. In order to obtain efficiently solvable problems in the nodes of our branch-and-bound tree, we restrict ourselves here to positive semi-definite LCP matrices M. Second, for practically relevant instances, the existence of mixed-integer feasible solutions of LCPs is often hard to achieve. Thus, it is crucial for practice to algorithmically deal with the case of non-existence of solutions as well. Since non-existence is often the case, one is also interested in approximate feasible solutions, i.e., points that minimize a certain infeasibility measure that combines both the violation of integrality conditions as well as of complementarity constraints; see, e.g., Gabriel, Conejo, Ruiz, et al. (2013), where similar relaxations are considered. Our penalty-based branch-and-bound approach is explicitly tailored to deal with the case of non-existence: if a mixed-integer feasible LCP solution exists, our algorithm provably finds one—otherwise it computes an approximate feasible solution that minimizes a certain infeasibility measure. Third, our branch-and-bound framework allows to incorporate further algorithmic enhancements. Here, we present mixed-integer LCP specific valid inequalities, discuss tailored node selection strategies, branching rules, and warmstarting techniques. These all are general-purpose enhancements that can be applied to any given MILCP. However, it is a dedicated feature of our approach that it can be extended by problem-specific techniques (such as problem-specific cutting planes or branching rules) if they are at hand for a concrete application problem. Fourth and finally, we test the penalty branch-and-bound method numerically and compare it with two benchmarks: an MILP reformulation of the mixed-integer LCP using big-Ms as well as a straightforward MIQP reformulation—both solved with the state-of-the-art mixed-integer programming solver Gurobi. It turns out that our method (extended by the above mentioned enhancements) clearly outperforms both benchmark approaches.

1.3. **Organization of the Paper.** The remainder of the paper is organized as follows. In Section 2, we present the problem statement and discuss a reformulation of the mixed-integer LCP that is later needed for the derivation of the penalty branch-and-bound method. The method is described in Section 3, where we also prove the correctness of the method by proving the correctness of the branching as well as of the bounding step. Afterward, in Section 4, we discuss further enhancements of the algorithm to improve its performance, which we evaluate in a numerical study in Section 5. Finally, we conclude the paper in Section 6.

2. Problem Statement and Reformulations

One important tool in the analysis and the resolution of the LCP (1) is the following reformulation as a quadratic problem (QP):

$$\min_{z\in\mathbb{R}^n} \quad z^\top(q+Mz) \tag{2a}$$

$$\text{s.t.} \quad z \geq 0, \quad q+Mz \geq 0. \tag{2b}$$

It is easy to see that the LCP (1) has a solution if and only if the QP (2) has an optimal solution with objective function value of zero. The mixed-integer linear complementarity problem (MILCP) that we consider in this paper is the task to

find a vector $z \in \mathbb{R}^n$ that satisfies

$$z \geq 0, \tag{3a}$$
$$q + Mz \geq 0, \tag{3b}$$
$$z^\top (q + Mz) = 0, \tag{3c}$$
$$z_i \in \{0, 1\} \quad \text{for } i \in I, \tag{3d}$$

or to show that no such vector exists. Here, $I \subseteq \{1, \ldots, n\}$ is the set of indices for which we require the variable to be binary. We denote this problem by $\mathrm{LCP}(q, M, I)$. For the ease of presentation, we restrict ourselves to mixed-binary LCPs, which is, of course, equivalent to considering mixed-integer LCPs for bounded integers.

Besides these solution approaches from the literature that are all based on reformulating the problem as an MILP, another straightforward possibility to compute a solution is by solving the reformulation as the mixed-integer quadratic problem (MIQP)

$$\min_{z \in \mathbb{R}^n} \quad z^\top (q + Mz) \tag{4a}$$
$$\text{s.t.} \quad z \geq 0, \; q + Mz \geq 0, \tag{4b}$$
$$z_i \in \{0, 1\} \text{ for } i \in I. \tag{4c}$$

Note that this MIQP is bounded from below by 0. Hence, the MILCP (3) has a solution if and only if there is an optimal solution of the MIQP (4) with objective function value of 0. In practice, however, different mathematical challenges arise. The combination of integrality and complementarity conditions may make it unlikely that there exists a solution of the MILCP. Since the existence of solutions cannot be expected in general, approximate solutions are of interest in many cases. Here, approximate solutions are points that violate the "challenging conditions", i.e., the integrality conditions as well as the complementarity constraints, as little as possible. This means, we call points (ρ, σ)-approximate solutions if they are feasible for the relaxed version

$$z \geq 0, \tag{5a}$$
$$q + Mz \geq 0, \tag{5b}$$
$$z^\top (q + Mz) \leq \rho, \tag{5c}$$
$$z_i \in [0, \sigma_i] \cup [1 - \sigma_i, 1] \quad \text{for} \quad i \in I, \tag{5d}$$

of (3). Here, $\rho \geq 0$ measures the violation of the complementarity constraints, while $\sigma_i \geq 0, i \in I$, measure the violation of integrality. Of course, one wants to choose these parameters as small as possible. We recover the original MILCP (3) for $\rho = 0$ and $\sigma_i = 0$, $i \in I$. For what follows, we do not consider a relaxation of the linear constraints (3a) and (3b) as they are usually less restrictive compared to the combination of integrality and complementarity conditions of the problem.

In the next section, we propose a nonconvex reformulation of Problem (3) that includes a relaxation of the integrality and complementarity conditions as shown in (5). We use this reformulation to derive an algorithm to solve this reformulation to global optimality. This especially means that if the MILCP is solvable, we compute a solution for the MILCP in finite time. On the other hand, if the MILCP is not solvable, we compute a solution that minimizes a certain measure of violation of the integrality and complementarity conditions. We call the proposed algorithm a "penalty branch-and-bound method" since we do branch without introducing further inequality constraints to the problem in the parent node but via introducing suitably chosen penalty terms. Thus, while branching, our feasible set stays the same and the objective function is extended by adding the penalty terms, which is

exactly the other way around compared to classic branch-and-bound in mixed-integer optimization. All details are presented in the next section. To obtain tractable problems to be solved at every node of the branch-and-bound tree, we restrict ourselves to matrices M that are positive semi-definite.

3. A Penalty Branch-and-Bound Method

We look for approximate solutions of (3), i.e., feasible solutions of (5) for which ρ and σ are minimized. To this end, we consider an optimization problem in which a combination of the violation of the complementarity conditions (3c) and the violation of the binary conditions (3d) is minimized over the feasible set

$$Z := \{z \in \mathbb{R}^n : z \geq 0,\ q + Mz \geq 0,\ z_i \leq 1 \text{ for } i \in I\}.$$

Note that Z is defined by the linear constraints (3a) and (3b) together with the continuous relaxation of the binary conditions. A suitable measure for the violation of the complementarity conditions (3c) is simply given by $z^\top(q + Mz)$, which is non-negative on Z. On the other hand, the violation of the binary conditions (3d) can be measured in different ways. Several penalty functions have been proposed in the literature; see, e.g., De Santis et al. (2013), Giannessi and Tardella (1998), Lucidi and Rinaldi (2010), Rinaldi (2009), and Zhu (2003). For our purposes, a concave and piecewise linear penalty function is preferred, as it permits to consider only linear objective terms along the nodes of our branch-and-bound tree. Thus, we choose the classic penalty function

$$\sum_{i \in I} \min\{z_i, 1 - z_i\}$$

and obtain the following nonlinear, nonconvex, and nonsmooth reformulation

$$\min_{z \in Z} \quad f(z) := \alpha z^\top(q + Mz) + (1 - \alpha) \sum_{i \in I} \min\{z_i, 1 - z_i\}. \tag{6}$$

Here, $\alpha \in (0, 1)$ is a parameter controlling the emphasis that is put on each of the two penalty terms. Note that if (3) is solvable, Problem (6) is an equivalent reformulation, i.e., its global solutions are solutions of (3)—and vice versa. Otherwise, if (3) is not solvable, the global solutions of problem (6) are approximate solutions of (3) with the smallest violation of the complementarity and the integrality conditions. Here, the violation of both the complementarity as well as the integrality conditions are measured in the ℓ_1 norm. There is no clear relationship between the parameter α used in (6) and the parameters ρ and σ used in (5). However, we can expect that choosing α close to 1 would lead to (ρ,σ)-approximate solutions with small ρ, or, in other words, (ρ,σ)-approximate solutions closer to satisfy the complementarity conditions. On the other hand, choosing α close to zero, would put more emphasis to the penalization of the integrality constraints and then should lead to smaller $\sigma_i, i \in I$.

It is our aim now to devise a penalty branch-and-bound method to solve Problem (6) that exploits the specific structure of the objective function f in (6).

In classic branch-and-bound approaches for mixed-integer problems, branching is done by starting with the continuous relaxation and by creating different subproblems in which variables, which are fractional in the relaxation's solution, are fixed to certain values or the feasible set is divided into disjoint sets using inequalities. Global upper bounds are derived by feasible points and local lower bounds are obtained from solving the optimization problems in the nodes of the branch-and-bound tree. In our method, however, we start with an α-scaled objective function of the continuous relaxation (2) and create subproblems by adding different penalty terms (for fractional variables) to the objective function. Thus, in contrast to classic

branch-and-bound methods, the feasible set remains the same over the entire tree. The bounding is done in analogy to a classic branch-and-bound method as the objective values of the node problems yield local lower bounds and any feasible solution yields a global upper bound.

3.1. **Branching.** At the root node of the branch-and-bound tree, we solve the convex relaxation

$$\min_{z \in Z} \quad \alpha z^\top (q + Mz)$$

of Problem (6) that is obtained by neglecting the second term in the objective function. Afterward, two child nodes are created as follows. Each node corresponds to a new convex QP, in which we add either $(1-\alpha)z_j$ or $(1-\alpha)(1-z_j)$ to the objective function. To this end, we choose an index $j \in I$ that satisfies $\min\{z_j^*, 1 - z_j^*\} > 0$ in the solution z^* of the root node relaxation. We will discuss more sophisticated branching rules in Section 4.

In particular, the problem of the first child node reads

$$\min_{z \in Z} \quad f_1(z) := \alpha z^\top (q + Mz) + (1 - \alpha) z_j,$$

which aims to drive z_j to 0 in the respective subtree, while the problem of the second child node is given by

$$\min_{z \in Z} \quad f_2(z) := \alpha z^\top (q + Mz) + (1 - \alpha)(1 - z_j),$$

which aims to drive z_j to 1 in the respective subtree. The idea is to split the term $\min\{z_j, 1 - z_j\}$ occurring in (6) into two new problems and taking the minimum of both these problems, i.e.,

$$\min_{z \in Z} \alpha z^\top (q + Mz) + (1 - \alpha) \min\{z_j, 1 - z_j\} = \min \left\{ \min_{z \in Z} f_1(z), \min_{z \in Z} f_2(z) \right\}.$$

Note that both z_j and $1 - z_j$ are non-negative on the feasible set Z. At an arbitrary node of the branch-and-bound tree, we thus have a convex-quadratic problem in which non-negative terms $(1 - \alpha)z_j$ and $(1 - \alpha)(1 - z_k)$, $j, k \in I$, have been added to the convex-quadratic function $\alpha z^\top (q + Mz)$. We define I_0 to be the set of indices $j \in I$ for which $(1 - \alpha)z_j$ has been added and I_1 to be the set of indices $j \in I$ for which $(1 - \alpha)(1 - z_j)$ has been added. This definition is in close analogy to the sets of fixed variables in a classic branch-and-bound method for mixed-binary problems. Consequently, every node N is uniquely determined by $N = (I_0, I_1)$. In the following, the objective function at a node $N = (I_0, I_1)$ is denoted by

$$f_N(z) = \alpha z^\top (q + Mz) + (1 - \alpha) \left(\sum_{j \in I_0} z_j + \sum_{j \in I_1} (1 - z_j) \right)$$

and the optimal solution of that node is denoted by z_N^*. Thus, the problem that has to be addressed at a node $N = (I_0, I_1)$ is of the form

$$\min_{z \in Z} \quad f_N(z). \tag{7}$$

Without loss of generality, we assume that the problems in the first child nodes are always obtained adding the terms $(1 - \alpha)z_j$, $j \in I_0$, which we call "downwards branching". The problems in the second child nodes are obtained adding the terms $(1 - \alpha)(1 - z_j)$, $j \in I_1$, which we call "upwards branching".

As a first result, we show that enumerating all possible partitions (I_0, I_1) of I, i.e., $I = I_0 \cup I_1$ with $I_0 \cap I_1 = \emptyset$, yield the optimal value of Problem (6). In other words, we show that the minimum among the optimal solutions of the problems of

all leaf nodes of the fully enumerated branch-and-bound tree is the optimal solution of Problem (6).

Lemma 1. *Let z^* be an optimal solution of Problem (6). Then, it holds*

$$f(z^*) = \min\left\{f_N(z^*_N)\colon N = (I_0, I_1) \text{ with } I_0 \cup I_1 = I \text{ and } I_0 \cap I_1 = \emptyset\right\}.$$

Proof. Note that the feasible set does not change from one N to another. Hence, all optimal points are feasible for all nodes. Let $N^* = (I_0^*, I_1^*)$ be the leaf with $I_0^* := \{i \in I\colon z_i^* \leq 1 - z_i^*\}$ and $I_1^* := \{i \in I\colon z_i^* > 1 - z_i^*\}$. We then have

$$\begin{aligned} f(z^*) &= \alpha(z^*)^\top(q + Mz^*) + (1-\alpha)\sum_{j\in I}\min\left\{z_j^*, 1 - z_j^*\right\} \\ &= \alpha(z^*)^\top(q + Mz^*) + (1-\alpha)\sum_{j\in I_0^*} z_j^* + (1-\alpha)\sum_{j\in I_1^*}(1 - z_j^*) \\ &= f_{N^*}(z^*) \geq f_{N^*}(z^*_{N^*}). \end{aligned}$$

Hence,

$$f(z^*) \geq \min\left\{f_N(z^*_N)\colon N = (I_0, I_1) \text{ with } I_0 \cup I_1 = I \text{ and } I_0 \cap I_1 = \emptyset\right\}$$

holds. To show the other inequality, we assume that there exists a node $N' = (I_0', I_1')$ with $I_0' \cup I_1' = I$ and $I_0' \cap I_1' = \emptyset$ such that

$$f_{N'}(z^*_{N'}) < f(z^*)$$

holds. We thus obtain $f_{N'}(z^*_{N'}) < f(z^*_{N'})$ or, equivalently,

$$\begin{aligned} &\alpha(z^*_{N'})^\top(q + Mz^*_{N'}) + (1-\alpha)\sum_{j\in I_0'} z^*_{N',j} + (1-\alpha)\sum_{j\in I_1'}(1 - z^*_{N',j}) \\ &< \alpha(z^*_{N'})^\top(q + Mz^*_{N'}) + (1-\alpha)\sum_{j\in I}\min\left\{z^*_{N',j}, 1 - z^*_{N',j}\right\}. \end{aligned}$$

This implies

$$\sum_{j\in I_0'}\left(z^*_{N',j} - \min\left\{z^*_{N',j}, 1 - z^*_{N',j}\right\}\right) + \sum_{j\in I_1'}\left(1 - z^*_{N',j} - \min\left\{z^*_{N',j}, 1 - z^*_{N',j}\right\}\right) < 0,$$

which is impossible as

$$z^*_{N',j} \geq \min\left\{z^*_{N',j}, 1 - z^*_{N',j}\right\}$$

and

$$1 - z^*_{N',j} \geq \min\left\{z^*_{N',j}, 1 - z^*_{N',j}\right\}.$$

Hence,

$$f(z^*) \leq \min\left\{f_N(z^*_N)\colon N = (I_0, I_1) \text{ with } I_0 \cup I_1 = I \text{ and } I_0 \cap I_1 = \emptyset\right\}$$

holds and the claim follows. □

As a consequence of this lemma, we know that by iterating over all possible partitions of I, we get an optimal solution of Problem (6), which is key to prove the correctness of the overall penalty branch-and-bound method.

3.2. **Bounding.** Similar to classic branch-and-bound methods, we can establish local lower bounds on the optimal solution for the different nodes and global upper bounds for the optimal solution of Problem (6). Obviously, the value of f at any feasible point is a global upper bound. Hence, $f(z^*) \leq f(z^*_N)$ holds with N being an arbitrary node of the branch-and-bound tree. We denote by z^*_{inc} the incumbent, i.e., the point so that $f(z^*_{\text{inc}})$ constitutes the best known upper bound for Problem (6) found so far.

Next we prove that the optimal value of the problem defined at a certain node is a lower bound for the optimal value of the problem defined at any of its successor

nodes. While this is rather trivial for classic branch-and-bound methods, this result is harder to establish for the penalty branch-and-bound method considered here.

Lemma 2. *Let $N' = (I'_0, I'_1)$ be a successor of some node $N = (I_0, I_1)$ in the branch-and-bound tree, i.e., $I_0 \subseteq I'_0$ and $I_1 \subseteq I'_1$ holds. Then,*

$$f_N(z^*_N) \leq f_{N'}(z^*_{N'})$$

holds.

Proof. Since the feasible set does not change during the branching process all feasible points remain feasible for all nodes. Thus,

$$\begin{aligned}
f_{N'}(z^*_{N'}) &= \alpha\ (z^*_{N'})^\top (q + M z^*_{N'}) + (1-\alpha) \sum_{j \in I'_0} z^*_{N',j} + (1-\alpha) \sum_{j \in I'_1} (1 - z^*_{N',j}) \\
&= \alpha\ (z^*_{N'})^\top (q + M z^*_{N'}) + (1-\alpha) \sum_{j \in I_0} z^*_{N',j} + (1-\alpha) \sum_{j \in I_1} (1 - z^*_{N',j}) \\
&\quad + (1-\alpha) \sum_{j \in I'_0 \setminus I_0} z^*_{N',j} + (1-\alpha) \sum_{j \in I'_1 \setminus I_1} (1 - z^*_{N',j}) \\
&\geq \alpha\ (z^*_{N'})^\top (q + M z^*_{N'}) + (1-\alpha) \sum_{j \in I_0} z^*_{N',j} + (1-\alpha) \sum_{j \in I_1} (1 - z^*_{N',j}) \\
&= f_N(z^*_{N'}) \geq f_N(z^*_N).
\end{aligned}$$

Note that the first inequality is due to the fact that $z^*_{N',j} \geq 0$ and $(1 - z^*_{N',j}) \geq 0$ for $j \in I$ on the feasible set. The second inequality follows from optimality. □

Lemma 2 implies that in the case that Problem (7) at node N leads to a solution z^*_N such that

$$f_N(z^*_N) \geq f(z^*_{\text{inc}})$$

holds, we have that every leaf of the subtree rooted in N cannot yield a better solution than the best known solution z^*_{inc}. Hence, we can prune the subtree rooted in N. Note that in our branch-and-bound method, there is no direct analogy to pruning due to feasibility as done in classic branch-and-bound methods: As soon as we find a feasible solution for Problem (3) we stop the algorithm. Pruning because of infeasibility is also not possible here, since the feasible set does not change throughout the process, except for the cuts introduced later.

3.3. **The Algorithm.** We are now ready to formally state the basic scheme of our penalty branch-and-bound method in Algorithm 1.

Theorem 1. *Algorithm 1 terminates after finitely many steps with a global optimal solution of Problem (6).*

Proof. The algorithm terminates after finitely many steps since the set I is finite. Thus, at some point, $I = I_0 \cup I_1$ holds and we can no longer find a branching variable in the node and no child node can be generated. Assume now that $f_N(z^*_N) < f(z^*_{\text{inc}})$ always holds in the second if-clause. Then the correctness of the algorithm follows from Lemma 1, as we iterate through the complete branch-and-bound tree. Finally, in the cases, in which $f_N(z^*_N) \geq f(z^*_{\text{inc}})$ holds, the nodes that are not added can be excluded due to Lemma 2. □

4. Further Algorithmic Enhancements

Similar to classic branch-and-bound methods, there are different possibilities to improve the performance of the overall algorithm. Both the choice of the next node to be solved and the choice of the next index to branch on are not yet specified in Algorithm 1—although it is known that these aspects have a significant impact

Algorithm 1 A Penalty Branch-and-Bound Algorithm for MILCPs

Input: $q \in \mathbb{R}^n$, $M \in \mathbb{R}^{n\times n}$, $\emptyset \subseteq I \subseteq [n]$, $\alpha \in (0,1)$
Output: A global optimum z^* of Problem (6).
Set $\mathcal{N} \leftarrow \{(\emptyset, \emptyset)\}$.
Set $f_{\text{inc}} \leftarrow \infty$.
Set $z^*_{\text{inc}} \leftarrow$ none.
while $\mathcal{N} \neq \emptyset$ **do**
  Choose $N = (I_0, I_1) \in \mathcal{N}$.
  Set $\mathcal{N} \leftarrow \mathcal{N} \setminus \{N\}$.
  Compute $z^*_N \in \arg\min\{f_N(z) : z \in Z\}$.
  if $f(z^*_N) < f_{\text{inc}}$ **then**
    Set $z^*_{\text{inc}} \leftarrow z^*_N$.
    Set $f_{\text{inc}} \leftarrow f(z^*_N)$.
  if $f_N(z^*_N) < f_{\text{inc}}$ **and** $I \setminus (I_0 \cup I_1) \neq \emptyset$ **then**
    Choose $j \in I \setminus (I_0 \cup I_1)$.
    Set $\mathcal{N} \leftarrow \mathcal{N} \cup \{(I_0 \cup \{j\}, I_1), (I_0, I_1 \cup \{j\})\}$.
return z^*_{inc}

on the performance of the algorithm. We address these choices in Sections 4.1 and 4.2. In Section 4.3, we also discuss the possibility of warmstarting the problem of each node with the optimal basis of the parent node. One of the most important ingredients of classic branch-and-bound algorithms are additional valid inequalities that can be incorporated in the problems of the nodes to improve the dual bound. We investigate two different types of valid inequalities for MILCPs in Section 4.4.

4.1. **Node Selection.** In our implementation of the penalty branch-and-bound algorithm, we consider three different node selection strategies. The first two are depth- and breadth-first search. The third strategy is based on Lemma 2 and will be referred to as the "lower-bound-push strategy".

From Lemma 2, we know that the optimal value $f_N(z^*_N)$ of the problem defined at a node N is a local lower bound for the subtree rooted in N. Hence, the global lower bound is the smallest value among the lower bounds obtained from nodes that have unsolved children. As the node to be solved next, we thus select a child of the node N that has the lowest objective value $f_N(z^*_N)$. When both children of N are not yet solved, we take the left child if $z^*_{N,j} \leq 0.5$ with $j \in I$ being the index that has been branched on last and the right child otherwise. Then, we choose the child node with the smaller value as we would expect this to result in a smaller lower bound. This lower bound may then be improved in the new node.

In our numerical experiments, we consider depth- and breadth-first search strategies as a benchmark for the lower-bound-push strategy; see Section 5.

4.2. **Branching Rules.** It is known that the performance of branch-and-bound methods for mixed-integer problems strongly depends on the branching strategy (Achterberg et al. 2005), i.e., on how to select the next variable to branch on.

In every node of our penalty branch-and-bound method, we need to choose an index $j \in I$ to define the objective functions of the problems in the child nodes. Clearly, we do not branch on a variable z_j, $j \in I$, if this variable is integer-valued in this node.

In the following, we propose two different branching strategies: "pseudocost branching", which is well-known from mixed-integer programming and "MIQP-based branching". In our numerical experiments, we compare these two strategies with

random branching (i.e., the naive approach of choosing the index $j \in I$ at random) and most-violated branching (i.e., choosing the index of the variable closest to 1/2).

4.2.1. *Pseudocost Branching.* As a first advanced approach, we consider a variant of pseudocost branching—a technique commonly used in branch-and-bound algorithms for mixed-integer programs that goes back to Benichou et al. (1971). The idea of pseudocost branching is to measure the expected objective gain when branching on a specific variable index. The strategy is to keep track of the change in the objective function when an index $j \in I$ has been chosen to be branched on. The rule then chooses the index that is predicted to have the largest impact on the objective function based on these past changes.

We transfer this idea to our context in the following. Let $\varphi^1_{N,j}$ be the objective gain per unit change when branching upwards on variable $j \in I$ at node N:

$$\varphi^1_{N,j} := \frac{f(z^*_{N_1}) - f(z^*_N)}{\lceil z^*_{N_1,j} \rceil - z^*_{N_1,j}}.$$

Here, N_1 is the child of N created by upwards branching. We denote by ψ^1_j the expected objective gain per unit change when branching upwards on variable j. To this end, let N^j be the set of nodes where $j \in I$ is chosen as the variable to branch on. Then, we define ψ^1_j as

$$\psi^1_j := \frac{1}{|N^j|} \sum_{N \in N^j} \varphi^1_{N,j}.$$

Analogously, we define $\varphi^0_{N,j}$ and ψ^0_j for branching downwards on variable $j \in I$. The average gain is then calculated as

$$\begin{aligned} s_j := \mu \min \left\{ \psi^0_j \cdot (z^*_{N_0,j} - \lfloor z^*_{N_0,j} \rfloor),\, \psi^1_j \cdot (\lceil z^*_{N_1,j} \rceil - z^*_{N_1,j}) \right\} \\ + (1-\mu) \max \left\{ \psi^0_j \cdot (z^*_{N_0,j} - \lfloor z^*_{N_0,j} \rfloor),\, \psi^1_j \cdot (\lceil z^*_{N_1,j} \rceil - z^*_{N_1,j}) \right\} \end{aligned}$$

with $\mu \in (0,1)$. The pseudocost-based branching candidate then is the index $j \in I$ with the largest score s_j. At the beginning of our branch-and-bound, we initialize the average $\psi^{0,1}_j$ with 1. If at a certain node N, we have not yet branched on a candidate $j \in I$, namely $N^j = \emptyset$, we initialize that $\psi^{0,1}_j$ with the average of all other $\psi^{0,1}_i$ for $i \in I$ with $i \neq j$.

4.2.2. *MIQP-Based Branching.* As a further branching rule that we use in the penalty branch-and-bound method, we propose a strategy based on solving a single-binary-variable MIQP for each integer variable in the presolve phase of the algorithm. Again, we aim at sorting the indices $j \in I$ so that we branch on those indices first that are expected to give good lower bounds on the optimal solution. For every index $j \in I$, we solve the following MIQP with a single integer variable:

$$\min_{z \in \mathbb{R}^n} \quad z^\top (q + Mz) \tag{8a}$$

$$\text{s.t.} \quad q + Mz \geq 0,\ z \geq 0, \tag{8b}$$

$$z_j \in \{0, 1\}. \tag{8c}$$

As discussed in the introduction, we know that it is likely that the overall MILCP has no solution and that this is due to the combination of complementarity as well as integrality conditions. By solving all $|I|$ many MIQPs (8) we measure the impact of the ith binary variable on the infeasibility of the problem (if it is infeasible at all). The indices $j \in I$ are then sorted with decreasing optimal objective function values of Problem (8). Moreover, infeasible problems are formally assigned the objective function value ∞. The resulting branching strategy then chooses the branching candidate on top of the list while skipping all integer-feasible indices as

well as all indices that have been branched on already. Additionally, we can use the optimal solutions of each of these MIQPs to constitute a first upper bound on our branch-and-bound process, as each of the points is also feasible for our method.

4.3. **Warmstarting.** Recall that all nodes of the search tree share the same feasible set and that the objective functions change only slightly from a parent node to its child nodes. This allows for warmstarting the QP solver for solving the child nodes. To this end, we take the optimal primal basis of the parent node as the starting basis for the child nodes.

4.4. **Valid Inequalities.** In this section, we propose two classes of inequalities that are not valid for the overall problem (6) in the classic sense but that are valid locally.

4.4.1. *Simple Cuts.* The first class of inequalities are called simple cuts. Assume that we just solved node N and that we decide to branch on the variable z_j, $j \in I$. Then, in the nodes corresponding to the downwards branching subtree, we add the bound constraint $z_j \leq 0.5$, while in the nodes belonging to the upwards branching subtree, we add the bound constraint $z_j \geq 0.5$. Although this sounds rather simple, the effect of including these cuts is significant (see Section 5) and proving the correctness of these inequalities is also not as easy as stating them.

We first show that the optimal solution of Problem (6) is not cut off when introducing the simple cuts. To this end, we prove that the minimum among the optimal solutions of the leaf problems that we obtain when including the simple cuts still is the optimal solution of Problem (6).

Lemma 3. *Let*

$$z_N^* \in \arg\min \{f_N(z) \colon z \in Z,\, z_{I_0} \leq 0.5,\, z_{I_1} \geq 0.5\}$$

be an optimal solution at node N when simple cuts are included. Then,

$$f(z^*) = \min \{f_N(z_N^*) \colon N = (I_0, I_1) \text{ with } I_0 \cup I_1 = I\}$$

holds.

Proof. Let $N^* = (I_0^*, I_1^*)$ be the leaf defined by $I_0^* := \{j \in I \colon z_j^* \leq 1 - z_j^*\}$ and $I_1^* := \{j \in I \colon z_j^* > 1 - z_j^*\}$. We then have

$$\begin{aligned} f(z^*) &= \alpha(z^*)^\top (q + Mz^*) + (1-\alpha) \sum_{i \in \mathcal{I}} \min\{z_i^*, 1 - z_i^*\} \\ &= \alpha(z^*)^\top (q + Mz^*) + (1-\alpha) \sum_{j \in I_0^*} z_j^* + (1-\alpha) \sum_{j \in I_1^*} (1 - z_j^*) \\ &= f_{N^*}(z^*) \geq f_{N^*}(z_{N^*}^*). \end{aligned}$$

The last inequality holds because, by definition, we have $z_j^* \leq 0.5$ for all $j \in I_0^*$ and $z_j^* \geq 0.5$ for all $j \in I_1^*$. Thus, z^* is feasible for $N = (I_0^*, I_1^*)$, which is a leaf by definition. Hence,

$$f(z^*) \geq \min \{f_N(z_N^*) \colon N = (I_0, I_1) \text{ with } I_0 \cup I_1 = I\}$$

holds. To show the other inequality, we assume that there exists a node $N' = (I_0', I_1')$ with $I_0' \cup I_1' = I$ such that

$$f_{N'}(z_{N'}^*) < f(z^*)$$

holds. With $f(z^*) \leq f(z_{N'}^*)$, we obtain

$$f_{N'}(z_{N'}^*) < f(z_{N'}^*)$$

or, equivalently,

$$\begin{aligned}&\alpha(z^*_{N'})^\top(q+Mz^*_{N'})+(1-\alpha)\sum_{j\in I'_0}z^*_{N',j}+(1-\alpha)\sum_{j\in I'_1}z^*_{N',j}\\<\;&\alpha(z^*_{N'})^\top(q+Mz^*_{N'})+(1-\alpha)\sum_{j\in\mathcal{I}}\min\left\{z^*_{N',j},1-z^*_{N',j}\right\}.\end{aligned}$$

This implies

$$\sum_{j\in I'_0}\left(z^*_{N',j}-\min\left\{z^*_{N',j},1-z^*_{N',j}\right\}\right)+\sum_{j\in I'_1}\left(1-z^*_{N',j}-\min\left\{z^*_{N',j},1-z^*_{N',j}\right\}\right)<0,$$

which is a contradiction by definition. Hence,

$$f(z^*)\leq\min\left\{f_N(z^*_N)\colon N=(I_0,I_1)\text{ with }I_0\,\dot\cup\,I_1=I\right\}$$

holds and the claim follows. □

As a second result, we show that Lemma 2 is also valid when simple cuts are used in the branch-and-bound method.

Lemma 4. *Let $N'=(I'_0,I'_1)$ be a successor of some node $N=(I_0,I_1)$ in the branching tree, i.e., $I_0\subseteq I'_0$ and $I_1\subseteq I'_1$ holds. Further, let $z^*_N,z^*_{N'}$ be optimal solutions of nodes N and N', respectively, when simple cuts are used. Then,*

$$f_N(z^*_N)\leq f_{N'}(z^*_{N'})$$

holds.

Proof. By definition, we have

$$\begin{aligned}f_{N'}(z^*_{N'})&=\alpha(z^*_{N'})^\top(q+Mz^*_{N'})+(1-\alpha)\sum_{j\in I'_0}z^*_{N',j}+(1-\alpha)\sum_{j\in I'_1}(1-z^*_{N',j})\\&=\alpha(z^*_{N'})^\top(q+Mz^*_{N'})+(1-\alpha)\sum_{j\in I_0}z^*_{N',j}+(1-\alpha)\sum_{j\in I_1}(1-z^*_{N',j})\\&\quad+(1-\alpha)\sum_{j\in I'_0\setminus I_0}z^*_{N',j}+(1-\alpha)\sum_{j\in I'_1\setminus I_1}(1-z^*_{N',j})\\&\geq\alpha(z^*_{N'})^\top(q+Mz^*_{N'})+(1-\alpha)\sum_{j\in I_0}z^*_{N',j}+(1-\alpha)\sum_{j\in I_1}(1-z^*_{N',j})\\&=f_N(z^*_{N'})\geq f_N(z^*_N).\end{aligned}$$

Note that the first inequality holds since $z^*_{N',j}\geq 0$ and $(1-z^*_{N',j})\geq 0$ are valid on the feasible set. The last inequality holds because the feasible sets of the nodes are nested in the sense, that the feasible set of node N' is a subset of the feasible set of node N. Hence, every feasible point of N' is also feasible for N. □

Theorem 2. *Algorithm 1 remains correct when simple cuts*

$$z_j\leq 0.5\text{ for all }j\in I_0,\quad z_j\geq 0.5\text{ for all }j\in I_1$$

are added at any node $N=(I_0,I_1)$.

Proof. From Lemma 3, we know that the optimal solution of Problem (6) is the optimal solution of a leaf node. From Lemma 4, we know that the objective value of every ancestor node of a leaf yields a lower bound for the objective value of this leaf. Hence, if we have a feasible point z^*_{inc} of Problem (6) and some node N for which

$$f(z^*_{\text{inc}})\leq f_N(z^*_N)$$

holds, we know that z^*_{inc} is a solution that is as good as every solution that any leaf being a successor of N can yield. Thus, we can prune the subtree rooted in N. The same applies for the case in which a node problem becomes infeasible due to

the introduction of cuts. Hence, Algorithm 1 remains correct when simple cuts are used. □

4.4.2. *Optimality Cuts.* The second class of inequalities that we introduce are so-called optimality cuts. In order to define them, we use the necessary optimality conditions for Problem (6); see, e.g., Corollary 3.68 in Beck (2017). Let $z^* \in Z$ be an optimal solution of Problem (6), then $g \in \partial f(z^*)$ exists such that

$$g^\top (z - z^*) \geq 0 \text{ for all } z \in Z.$$

Hence, if we find a point z^* during our branch-and-bound search that does not fulfill this inequality for any known feasible point $z \in Z$, we can cut off z^*. In particular, we derive the valid inequality

$$g^\top z' \geq g^\top z,$$

with $z' \in Z$ being some fixed feasible solution. Furthermore, for any $\bar{g}, \tilde{g} \in \partial f(z)$ such that $\bar{g}^\top z' \geq g^\top z'$ and $\tilde{g}^\top z \leq g^\top z$ holds, the following inequality is also valid:

$$\bar{g}^\top z' \geq \tilde{g}^\top z.$$

This will be necessary to convexify the valid inequality.

Lemma 5. *Let $z' \in Z$ be a feasible solution and let $N = (I_0, I_1)$. Then,*

$$\begin{aligned} &\alpha z^\top (q + 2Mz) + (1-\alpha) \sum_{j \in I_0} z_j + (1-\alpha) \sum_{j \in I_1} (1 - z_j) - (1-\alpha)|I \setminus I_0| \\ &\leq \alpha (z')^\top (q + 2Mz) + (1-\alpha) \sum_{j \in I \setminus I_1} z'_j \end{aligned}$$

is a valid inequality for the subtree rooted at node N.

Proof. Let $z' \in Z$, $z \in Z$, and $g \in \partial f(z)$ be given. We need to underestimate $g^\top z$ and overestimate $g^\top z'$. The ith component of $g \in \partial f(z)$ is given by

$$g_i = \alpha q_i + \alpha \sum_{j \in [n]} 2M_{i,j} z_j \begin{cases} +(1-\alpha), & \text{for } z_i < 0.5,\ i \in I, \\ -(1-\alpha), & \text{for } z_i > 0.5,\ i \in I, \\ +(1-\alpha) y_i, & \text{for } z_i = 0.5,\ i \in I, \\ +0, & \text{for } i \notin I, \end{cases}$$

for some $y_i \in [-1, 1]$.

We can then underestimate $g^\top z$ as follows:

$$\begin{aligned} g^\top z &= \alpha z^\top (q + 2Mz) + (1-\alpha) \left(\sum_{\substack{i \in I: \\ z_i < 0.5}} z_i - \sum_{\substack{i \in I: \\ z_i > 0.5}} z_i + \sum_{\substack{i \in I: \\ z_i = 0.5}} y_i^g z_i \right) \\ &\geq \alpha z^\top (q + 2Mz) + (1-\alpha) \left(\sum_{\substack{i \in I: \\ z_i < 0.5}} z_i - \sum_{\substack{i \in I: \\ z_i > 0.5}} z_i - \sum_{\substack{i \in I: \\ z_i = 0.5}} z_i \right) \\ &= \alpha z^\top (q + 2Mz) + (1-\alpha) \left(\sum_{\substack{i \in I: \\ z_i < 0.5}} z_i + \sum_{\substack{i \in I: \\ z_i \geq 0.5}} (1 - z_i) - \sum_{\substack{i \in I: \\ z_i \geq 0.5}} 1 \right) \\ &\geq \alpha z^\top (q + 2Mz) + (1-\alpha) \sum_{i \in I} \min\{z_i, 1 - z_i\} - (1-\alpha)|I| \\ &\geq \alpha z^\top (q + 2Mz) + (1-\alpha) \sum_{i \in I_0} z_i + (1-\alpha) \sum_{i \in I_1} (1 - z_i) - (1-\alpha)|I|. \end{aligned}$$

Note that the term $|I|$ can be replaced by $|I \setminus I_0|$ if simple cuts are included. On the other hand, we can overestimate $g^\top z'$ as follows:

$$\begin{aligned}
g^\top z' &= \alpha(z')^\top(q+2Mz) + (1-\alpha)\left(\sum_{\substack{i\in I:\\ z_i<0.5}} z'_i - \sum_{\substack{i\in I:\\ z_i>0.5}} z'_i + \sum_{\substack{i\in I:\\ z_i=0.5}} y^g_i z'_i\right)\\
&\le \alpha(z')^\top(q+2Mz) + (1-\alpha)\left(\sum_{\substack{i\in I:\\ z_i<0.5}} z'_i - \sum_{\substack{i\in I:\\ z_i>0.5}} z'_i + \sum_{\substack{i\in I:\\ z_i=0.5}} z'_i\right)\\
&= \alpha(z')^\top(q+2Mz) + (1-\alpha)\left(\sum_{\substack{i\in I:\\ z_i\le 0.5}} z'_i - \sum_{\substack{i\in I:\\ z_i>0.5}} z'_i\right)\\
&\le \alpha(z')^\top(q+2Mz) + (1-\alpha)\sum_{i\in I\setminus I_1} z'_i - (1-\alpha)\sum_{\substack{i\in I_1:\\ z_i\neq 0.5}} z'_i\\
&\le \alpha(z')^\top(q+2Mz) + (1-\alpha)\sum_{i\in I\setminus I_1} z'.
\end{aligned}$$

The combination of the two inequalities yields the lemma. □

5. Numerical Results

We start with describing the software and hardware setup of our numerical tests and discuss the test set. We implemented the penalty branch-and-bound method presented in Section 3 in Python 3.7. All node problems are solved with the QP solver of Gurobi 9.1.2 and all the tests were run on an Intel Xeon CPU E5-2699 v4 @ 2.20 GHz (88 cores) with 756 GB RAM. In this section, we refer to the implementation of Algorithm 1 as MILCP-PBB. For our tests, we consider instances that we randomly generated as follows. The matrices $M \in \mathbb{R}^{n\times n}$ have been created using the sprandsym function of MATLAB for sizes $n \in \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$. Details on the spectra and the densities of the matrices can be found in Appendix A. We then built vectors $q \in \mathbb{R}^n$ in four different ways, each reflecting a certain "degree of feasibility" in the resulting instance. Let $z^* \in \mathbb{R}^n$ be a solution of an instance of Problem (3). Then, it satisfies

(i) Feasibility w.r.t. Z: $z^* \in Z$,
(ii) Integrality: $z^*_i \in \{0, 1\}$ for all $i \in I$,
(iii) Complementarity: $(z^*)^\top(q + Mz^*) = 0$.

The vectors q have been created to satisfy at least one of the conditions above. More precisely, we built instances for which $z \in \mathbb{R}^n$ exists so that

(a) only Condition (i) is guaranteed to be satisfied,
(b) only Conditions (i) and (ii) are guaranteed to be satisfied,
(c) only Conditions (i) and (iii) are guaranteed to be satisfied,
(d) all Conditions (i)–(iii) are guaranteed to be satisfied.

We created 10 instances for every size n and the types (a)–(c), yielding 300 different instances in total. Type (d) appeared to be very easy to solve, which is why we exclude these instances from the test set. Again, more details on how the test set has been built can be found in Appendix A. The instances are available at the following link: https://github.com/m-schmidt-math-opt/milcp-penalty-bnb-instance-data.

For the comparisons presented in this section we use logarithmic performance profiles in the sense of Dolan and Moré (2002) as well as tables with the most

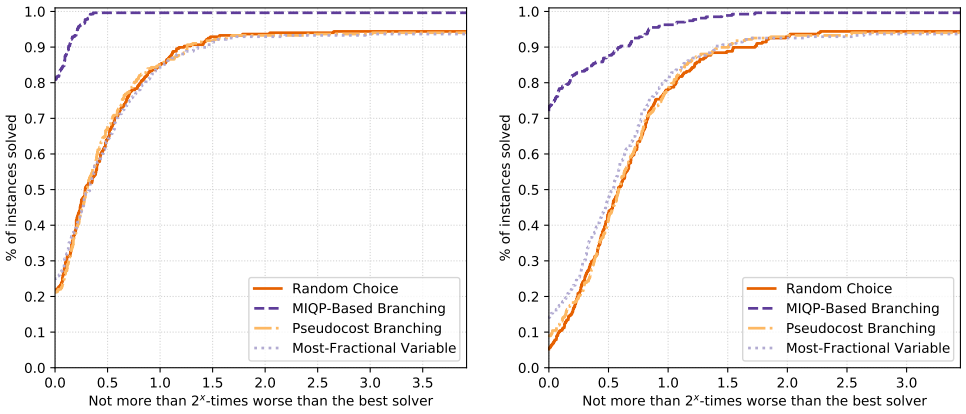

FIGURE 1. Performance profiles for the number of branch-and-bound nodes (left) and the running time (right) of all branching rules

important statistical measures. For the tables, we aggregated all instances that have been solved by all parameter settings or solution approaches for the specific test w.r.t. the instance size. The first column always states the dimension n of the problem. The second column contains the arithmetic mean of node counts resp. running times for all instances solved by every parameterization. The next columns contain the median, the minimum, and the maximum value of the data set. The sixth and seventh column contain the 0.25-quantile, i.e., the node count or running time after which 25 % of instances were solved, as well as the 0.75-quantile. The next two columns contain the geometric mean and the geometric shifted mean. The shift is 100 for the node counts and 10 for the running times. The last column contains the percentage of instances solved to global optimality for the parameterization and instance size. The best value for every measure and instance size among all tables for that test is printed bold. The table of the winning setting, i.e., the best performing parameterization, is included in this section whereas the tables of the other settings are included in Appendix B.

In Section 4, we discussed a number of ways to enhance Algorithm 1. To determine the best setting for MILCP-PBB, we study (i) the impact of different branching rules, (ii) different node selection strategies, (iii) the use of warmstarts, and (iv) the use of valid inequalities. This is done in Sections 5.1–5.4, respectively, where we set a time limit of 1 h. In Section 5.5, the best parameterization of MILCP-PBB is then compared with two other approaches. The first one is proposed in Gabriel, Conejo, Ruiz, et al. (2013), where the authors reformulate the MILCP as an MILP using additional binary variables and big-M constraints to re-model the complementarity constraints. The second approach is inspired by this model but uses an MIQP instead of an MILP reformulation of the given MILCP.

5.1. **Comparison of Different Branching Rules.** We now compare the performance of MILCP-PBB when equipped with the four different branching rules described in Section 4.2. For these tests, the node selection strategy is set to breadth-first search, warmstarts are disabled, and no valid inequalities are added. For the pseudocost branching strategy, we set $\mu = 0.5$. We exclude 32 instances from the test set since no parameterization is able to solve them within the time limit. Figure 1 displays the performance profiles w.r.t. the required number of branch-and-bound nodes (left figure) and running times (right figure). One can see that the running time and the number of nodes for the random branching rule, the pseudocost

branching strategy, and the branching strategy based on the most fractional variable do not differ much. However, the MIQP-based branching rule yields a significant improvement in terms of the required number of nodes, the running time, and also in terms of the overall number of solved instances. This improvement is especially true for the number of nodes as our MIQP-based approach visits significantly fewer nodes for the vast majority of the instances, while also solving the overall largest number of instances to global optimality. The improvement regarding the running times is a little less significant. This is to be expected since the ordering of branching priorities during the presolve phase is more expensive compared to the computational effort required by the other branching strategies. However, the advantage regarding the number of nodes overcompensates this disadvantage and the MIQP-based branching rule also dominates all other strategies w.r.t. running times as well.

Similar conclusions can be drawn from the statistical measures as displayed in the Table 1 (and Tables 8–10 in the appendix). In comparison of all tables one sees that, except for the minimum running time, the MIQP-based branching rule outperforms the other approaches w.r.t. almost every other measure and every instance size.

TABLE 1. Aggregated node counts and running times for the branching rule test with MIQP-based branching

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	13.9	9.0	**3.0**	**37.0**	**5.0**	**21.0**	13.4	**100**
100	**17.5**	**13.0**	**3.0**	73.0	**7.0**	**21.0**	**16.7**	**100**
150	**38.5**	**33.0**	**7.0**	**139.0**	**14.0**	**44.5**	**35.4**	**100**
200	**60.1**	**44.0**	**15.0**	235.0	**31.0**	**82.5**	**55.3**	**100**
250	**116.5**	**96.0**	**5.0**	**353.0**	**68.5**	**132.5**	**100.7**	**100**
300	**273.9**	**155.0**	**15.0**	1199.0	**95.5**	**292.0**	**203.3**	**100**
350	**549.8**	**331.0**	**7.0**	**2705.0**	**80.0**	**729.0**	**333.0**	**100**
400	**426.7**	**248.0**	**47.0**	**1245.0**	**76.5**	**750.5**	**289.7**	**80**
450	**408.3**	**349.0**	**51.0**	**1713.0**	**139.0**	**511.0**	**305.2**	**67**
500	**544.1**	**543.0**	**71.0**	**1043.0**	**342.0**	**734.0**	**444.9**	**40**
50	0.3	0.3	**0.1**	**0.5**	0.2	0.3	0.3	**100**
100	**2.4**	1.7	0.6	**9.4**	1.1	**2.9**	**2.2**	**100**
150	**15.1**	**12.6**	3.3	**54.0**	**5.2**	**18.2**	**12.7**	**100**
200	**45.5**	**31.5**	13.2	**147.6**	**24.0**	**58.1**	**38.8**	**100**
250	**149.8**	**112.8**	16.3	**504.9**	**75.4**	**180.4**	**110.1**	**100**
300	**448.8**	**291.2**	48.5	**1918.0**	**162.0**	**550.5**	**306.8**	**100**
350	**889.3**	**574.3**	54.2	**2853.7**	**211.5**	**1575.2**	**559.2**	**100**
400	**821.3**	**726.8**	**191.3**	**2429.7**	**302.9**	**1260.8**	**620.7**	**80**
450	**909.3**	**941.1**	260.4	**2325.1**	**469.1**	**1176.4**	**766.6**	**67**
500	**1106.9**	**1197.4**	355.0	**1666.5**	**870.6**	**1394.2**	**1000.5**	**40**

5.2. **Comparison of Different Node Selection Strategies.** We now compare the three node selection strategies described in Section 4.1. To this end, we use the MIQP-based branching strategy, while warmstarts and valid inequalities are disabled. We exclude 54 instances from the set since no parameterization of our method is able to solve them within the time limit. Based on Figure 2, one can notice that the node selection strategies only have a minor impact on the performance of the overall method both in terms of the number of nodes and the running time. Especially regarding the required number of branch-and-bound nodes, no parameterization seems to have an advantage. Regarding the running time, the lower-bound-push

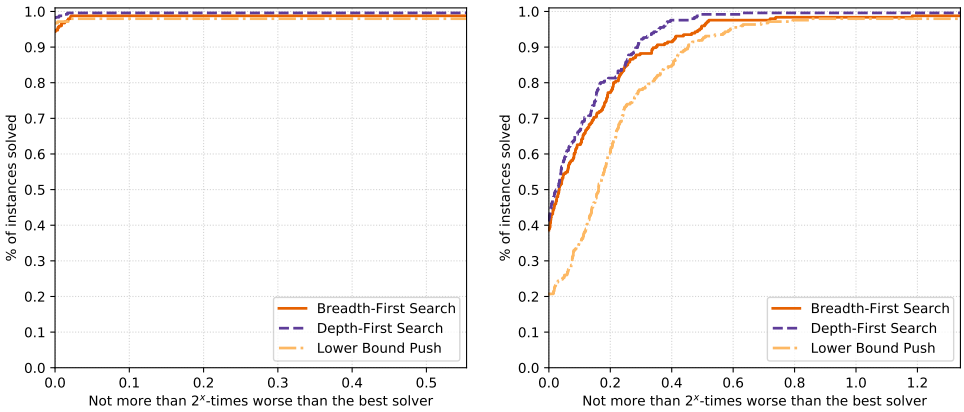

FIGURE 2. Performance profiles on the number of branch-and-bound nodes (left) and the running time (right) of all node selection strategies.

strategy seems to be slightly worse, while breadth-first and depth-first search are very close in comparison with a slight advantage for the depth-first search. Again, this is due to the higher computational cost for the ordering of the nodes. As the depth-first search also solves slightly more instances, we choose it for our "best-setting" implementation of MILCP-PBB.

The statistical measures we present in Tables 2, 11, and 12 support these conclusions. For most measures the depth-first search strategy performs best, followed by the breadth-first search strategy. But nevertheless, for all measures and instance sizes, the differences are rather small.

5.3. **The Benefits of Warmstarts.** We now compare the performance of MILCP-PBB with and without warmstarts. To this end, we use the MIQP-based approach branching rule, the breadth-first search node selection strategy, and avoid the use of any valid inequalities. We tried two different techniques within Gurobi to warm start the node problems. First, we used the Gurobi attributes VBasis and CBasis, i.e., we started every node problem with the optimal basis of its parent node. Second, we used the attributes PStart and DStart, where the optimal basis vector of the parent node is computed from the optimal solution. In case that warmstarts are used, we need to solve the node problems using the primal simplex method within Gurobi. However, this leads to some numerical instabilities that we detected during our preliminary testing. Thus, we implemented a backup strategy that disables warmstarts in the case of numerical troubles and then allows that Gurobi chooses any other method for solving the node problems. We exclude 46 instances from the set as no parameterization is able to solve them within the time limit. As expected, warmstarts significantly help to reduce the running time; see Figure 3 (right). Especially the use of parameters VBasis and CBasis have a big impact, which is expected as it is not needed to compute the basis vector first. Let us finally comment on the surprising result that using warmstarts or not leads to a different number of branch-and-bound nodes required to solve the problems; see Figure 3 (left). This is due to the occurrence of node problems with non-unique optimal solutions. In such a case, using warmstarts or not might lead to different solutions of the node problems, which, in turn, effects the overall search tree. The same can be seen in Tables 3, 13, and 14. For the node counts, differences are not remarkably large with a slight advantage for the warmstarted methods on most instances. With

TABLE 2. Aggregated node counts and running times for the node selection test with depth-first search

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	**35.6**	**24.0**	**7.0**	**139.0**	**13.5**	**42.5**	**32.6**	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	**54.9**	**100**
250	**112.3**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	**193.3**	**141.0**	**15.0**	**741.0**	**95.0**	**254.5**	**161.7**	**100**
350	**277.0**	**144.0**	**7.0**	**735.0**	**79.0**	**479.0**	**203.1**	**77**
400	**351.2**	**271.0**	**47.0**	**813.0**	**78.0**	**613.0**	**259.3**	**70**
450	314.7	345.0	**51.0**	**647.0**	**126.5**	**473.5**	**263.7**	**47**
500	461.3	519.0	**71.0**	**923.0**	**265.50**	546.0	381.8	**23**
50	**0.3**	**0.2**	**0.1**	**0.4**	**0.2**	**0.3**	**0.3**	**100**
100	**4.5**	**2.6**	1.4	24.1	2.0	5.4	**4.0**	**100**
150	27.1	**18.1**	6.4	97.0	10.6	32.6	22.1	**100**
200	91.9	**65.1**	30.4	299.9	**49.8**	**111.6**	**78.5**	**100**
250	249.5	221.9	37.5	**788.4**	156.2	**286.80**	198.9	**100**
300	**566.8**	**427.2**	132.4	**1889.0**	**266.2**	769.5	**454.1**	**100**
350	**1064.1**	**543.2**	**102.3**	**2983.5**	401.0	**1623.7**	**716.8**	**77**
400	1457.7	**1146.4**	309.1	**3215.6**	**419.6**	2474.4	**1060.7**	**70**
450	**1540.1**	**1468.0**	**369.0**	**2823.5**	**654.3**	2489.2	**1243.7**	**47**
500	2098.4	2272.3	553.1	**3178.9**	1553.9	2817.7	1821.9	**23**

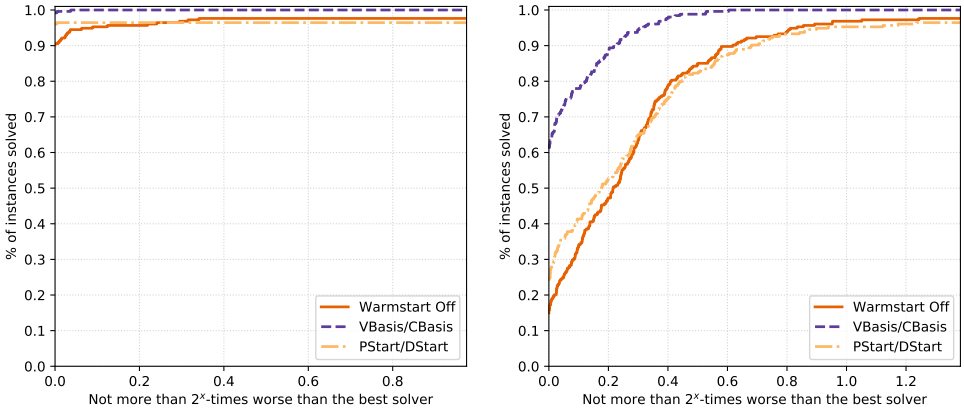

FIGURE 3. Performance profiles for the number of branch-and-bound nodes (left) and the running time (right) of the warmstart test.

respect to running times, the warmstarting strategy using VBasis and CBasis gives a significant advantage for all measures and instance sizes.

5.4. **Computational Analysis of the Valid Inequalities.** We tested different types of valid inequalities as described in Section 4.4. Unfortunately, incorporating the optimality cuts (see Section 4.4.2) results in severe numerical troubles for Gurobi. Possible reasons for that might be that these cuts are both quadratic second-order-cone constraints and very dense. We also tried different relaxations of these cuts to obtain sparser cuts but this did not resolve the numerical troubles. We also

TABLE 3. Aggregated node counts and running times for the warm-start test using VBasis/CBasis

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	**35.3**	**24.0**	**7.0**	**139.0**	**12.0**	**42.5**	**32.3**	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	**54.8**	**100**
250	**112.2**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	**193.3**	**141.0**	**15.0**	**741.0**	**95.0**	**254.5**	**161.6**	**100**
350	**355.4**	**215.0**	**7.0**	**1107.0**	**79.0**	**679.0**	**249.9**	**87**
400	**417.6**	**279.0**	**47.0**	**1245.0**	**81.0**	**719.0**	**300.3**	**77**
450	**312.4**	**345.0**	**41.0**	**649.0**	**126.5**	**473.5**	**258.6**	**50**
500	**476.6**	**543.0**	**61.0**	**1043.0**	**189.0**	**547.0**	**366.6**	**33**
50	**0.3**	**0.3**	**0.1**	**0.8**	**0.2**	**0.3**	**0.3**	**100**
100	**3.3**	**2.2**	**0.6**	**11.8**	**1.6**	**3.9**	**3.1**	**100**
150	**22.4**	**14.0**	**4.0**	**76.0**	9.4	28.3	**18.2**	**100**
200	**82.2**	**56.7**	**24.3**	**284.7**	**43.7**	**104.3**	**69.1**	**100**
250	**213.9**	**191.8**	30.8	**624.4**	**136.7**	**260.3**	**169.0**	**100**
300	**439.1**	**381.3**	**91.9**	**1387.5**	**224.9**	**521.4**	**360.3**	**100**
350	**1079.0**	720.5	**94.8**	**3339.2**	**324.2**	**1721.9**	**718.9**	**87**
400	**1421.9**	1497.6	**284.9**	**3090.9**	403.4	**2060.3**	**1040.5**	**77**
450	**1291.5**	1534.2	**301.1**	**2426.3**	**629.0**	**1838.2**	**1064.9**	**50**
500	2119.2	2200.6	**503.5**	**3396.8**	1265.3	3229.8	1730.6	**33**

tested a linearized version of this quadratic cut. This resolved almost all numerical issues but, on the other hand, lead to the fact that we do not cut off any points anymore. Thus, making these optimality cuts work in a practical implementation is still subject to future work. Consequently, we only consider simple cuts. Note that these cuts can be set up at no computational cost and that it is to be expected that they only have a minimal impact on the computational time required to solve the node problems since they are merely variable bounds. We compare a version of MILCP-PBB in which all possible simple cuts are added in every node with a version of MILCP-PBB in which no simple cuts are added. For this test, the branching rule is set to the MIQP-based branching rule, the node selection strategy is set to depth-first search, and warmstarts are disabled. Let us quickly comment on why warmstarts are disabled for this test even though they have a positive impact on the performance. For technical reasons, Gurobi needs both parameters VBasis and CBasis to warmstart a node problem, which contain the variable basis vector and the constraint basis vector. When cuts are added, the constraint basis vector needed to warmstart the problem is of higher dimension than the constraint basis vector of the parent node, which is why the use of VBasis/CBasis is mutually exclusive with the use of cuts. It would be possible, to use the parameters PStart/DStart, as Gurobi only needs a primal start pointing, which is available even with added cuts. However, as the difference between a warmstart with PStart/DStart and no warmstart is not significant, we choose to disable the warmstart here for simplicity. No instances are excluded for this test. As can be seen in Figure 4, incorporating the simple cuts has a great impact both on the number of branch-and-bound nodes as well as on the running time. This is also obvious from the results in Tables 4 and 15. For almost all measures and instance sizes, the approach with the simple cuts significantly outperforms the method without the cuts both w.r.t. the node

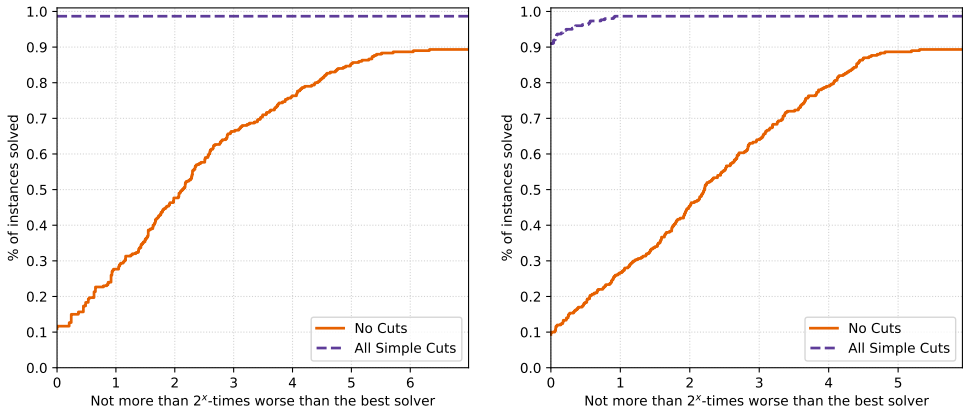

FIGURE 4. Performance profiles for the number of branch-and-bounds nodes (left) and the running time (right) for variants with all possible simple cuts and without any.

counts and the running times. Moreover, we see in Table 4 that we can solve almost all instances of the entire test.

TABLE 4. Aggregated node counts and running times for the valid inequalities test with all simple cuts

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**7.7**	**7.0**	**3.0**	**15.0**	**5.0**	**11.0**	**7.7**	**100**
100	**8.6**	**9.0**	**3.0**	**15.0**	**5.5**	**11.0**	**8.6**	**100**
150	**12.4**	**13.0**	**7.0**	**23.0**	**9.5**	**13.0**	**12.3**	**100**
200	**15.8**	**16.0**	**9.0**	**27.0**	**13.0**	**17.0**	**15.7**	**100**
250	**18.5**	**19.0**	**5.0**	**29.0**	**17.0**	**21.0**	**18.4**	97
300	**23.0**	**23.0**	**11.0**	**37.0**	**19.5**	**27.0**	**22.9**	**100**
350	**30.0**	**31.0**	**7.0**	**47.0**	**21.0**	**37.0**	**29.7**	**97**
400	**37.3**	**33.0**	**17.0**	**81.0**	**22.0**	**50.0**	**36.4**	**100**
450	**32.4**	**28.0**	**23.0**	**63.0**	**27.0**	**31.0**	**32.1**	**93**
500	**35.4**	**35.0**	**23.0**	**51.0**	**32.0**	**37.0**	**35.2**	**100**
50	**0.2**	0.3	**0.1**	**0.3**	**0.2**	**0.3**	**0.2**	**100**
100	**1.2**	**1.2**	**0.7**	**2.2**	**1.1**	**1.3**	**1.2**	**100**
150	**4.5**	**4.4**	**2.8**	**6.7**	**3.9**	**4.9**	**4.4**	**100**
200	**10.8**	**10.7**	**7.9**	**14.5**	**9.5**	**11.9**	**10.7**	**100**
250	**20.5**	**20.6**	**13.5**	**26.9**	**17.2**	**23.4**	**20.2**	97
300	**34.5**	**32.9**	**27.0**	**45.6**	**29.8**	**39.2**	**34.1**	**100**
350	**57.2**	**56.1**	**33.1**	**80.4**	**48.8**	**64.4**	**56.2**	**97**
400	**85.0**	**78.8**	**61.9**	**150.2**	**69.6**	**92.6**	**82.9**	**100**
450	**104.6**	**101.6**	**84.9**	**167.8**	**91.4**	**112.8**	**103.2**	**93**
500	**124.9**	**128.3**	**74.5**	**155.0**	**116.5**	**134.3**	**123.0**	**100**

5.5. **Comparing MILCP-PBB with Other Approaches.** Our preliminary numerical tests reveal that the best parameterization of MILCP-PBB uses the MIQP-based branching rule and adds all possible simple cuts at every node and warmstarts are disabled. As mentioned before, we choose depth-first search as our node selection strategy.

In order to compare MILCP-PBB with other approaches from the literature, we consider what is proposed in Gabriel, Conejo, Ruiz, et al. (2013). There, the authors reformulate the MILCP problem as an MILP with additional binary variables and big-M constraints to re-model complementarity constraints. Note that we use the notation B for the large constants to avoid confusion with the LCP's matrix M. The respective MILP then reads as follows:

$$\begin{aligned}
\min_{z,z',z'',\rho,\sigma} \quad & \alpha \sum_{i=1}^{n} \rho_i + (1-\alpha) \sum_{i \in I} \sigma_i && \text{(9a)}\\
\text{s.t.} \quad & z \geq 0, \quad q + Mz \geq 0, && \text{(9b)}\\
& z \leq Bz' + \rho, && \text{(9c)}\\
& q + Mz \leq B(1 - z') + \rho, && \text{(9d)}\\
& 0 \leq z_I \leq z'' + \sigma, && \text{(9e)}\\
& z'' - \sigma \leq z_I \leq 1, && \text{(9f)}\\
& z \in \mathbb{R}^n, \quad z' \in \{0,1\}^n, \quad z'' \in \{0,1\}^I, && \text{(9g)}\\
& \sigma \in \mathbb{R}^I_{\geq 0}, \quad \rho \in \mathbb{R}^n_{\geq 0}. && \text{(9h)}
\end{aligned}$$

Similar to Formulation (5), ρ_i is used to bound the violation of each complementarity constraints, while σ_i bounds the violation of the binary constraints. The variables z'_i are indicator variables that decide if for index i the corresponding variable z_i or $(q + Mz)_i$ is as close as possible to 0. Analogously, the indicator variables z''_i, $i \in I$, decide if the corresponding variable z_i is as close as possible to 0 or to 1. Note that this formulation will result in different optimal objective function values compared to our approach as the violation of the complementarity constraint is penalized in a different way. Furthermore note that Model (9) requires a significantly larger set of $3n + 2|I|$ variables.

Besides the significantly larger number of variables, one additional drawback of Model (9) is that it requires to determine sufficiently large big-B constraints. However, we can actually modify Problem (9) to get rid of these big-Bs and to measure the violation of the complementarity constraints using the same term as in our approach. This comes at the price of considering a quadratic instead of a linear problem. This resulting MIQP is given by

$$\begin{aligned}
\min_{z,z',\sigma} \quad & \alpha z^\top (q + Mz) + (1-\alpha) \sum_{i \in I} \sigma_i && \text{(10a)}\\
\text{s.t.} \quad & z \geq 0, \quad q + Mz \geq 0, && \text{(10b)}\\
& 0 \leq z_I \leq z' + \sigma, && \text{(10c)}\\
& z' - \sigma \leq z_I \leq 1, && \text{(10d)}\\
& z \in \mathbb{R}^n, \quad z' \in \{0,1\}^I, && \text{(10e)}\\
& \sigma \in \mathbb{R}^I_{\geq 0}. && \text{(10f)}
\end{aligned}$$

Instead of using variables ρ_i for bounding the violation of the complementarity, we use the direct penalization via the corresponding quadratic term. The violation of the binary constraints is still measured in the same way as in the MILP (9), with z'_i being the corresponding indicator variables as before.

Note that the MIQP (10) only has $|I|$ additional binary variables z' and $|I|$ additional continuous variables σ_i when compared to the original MILCP. Thus, the number of additionally required auxiliary variables is significantly reduced compared to the MILP reformulation (9). This makes a huge difference in practice: Gurobi is able to solve the MIQP (10) in significantly less time compared to what is required

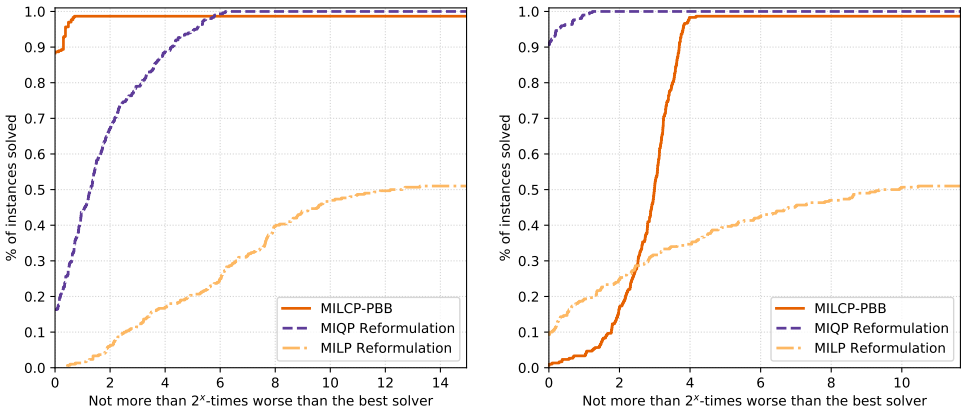

FIGURE 5. Performance profiles for the number of branch-and-bound nodes (left) and the running time (right) for the MIQP reformulation, the MILP reformulation and MILCP-PBB.

for solving the MILP (9); see Figure 5. When (9) and (10) are solved using Gurobi, all presolve techniques and heuristics have been disabled. For obtaining a fair comparison, we further restrict both the MIQP solver of Gurobi and the QP solver of Gurobi used for solving the nodes within MILCP-PBB to only use a single thread.

For a first comparison we use the same instances as before and no instances are excluded. Figure 5 shows the performance profiles of MILCP-PBB and Gurobi for both the MILP and the MIQP formulation w.r.t. the number of nodes and running times. It can be seen that MILCP-PBB needs significantly fewer nodes, while still needing more running time. The increased running time can probably be attributed to inefficiencies from our Python implementation.

More details can be found in Tables 5, 6, and 16. It is evident that our approach clearly outperforms the two benchmark approaches w.r.t. the node count for all measures and sizes. For the running time it is evident that the MIQP approach outperforms both our approach and the MILP formulation.

TABLE 5. Aggregated node counts and running times for the first benchmark test for MILCP-PBB

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**7.7**	7.0	**3.0**	**15.0**	5.0	**11.0**	**7.7**	**100**
100	**8.6**	**9.0**	**3.0**	**15.0**	5.5	**11.0**	**8.6**	**100**
150	**12.4**	**13.0**	7.0	**23.0**	**9.5**	**13.0**	**12.3**	**100**
200	**15.8**	**16.0**	**9.0**	**27.0**	**13.0**	**17.0**	**15.7**	**100**
250	**19.3**	**19.0**	11.0	**29.0**	**17.0**	**21.5**	**19.2**	97
300	**26.5**	**26.0**	13.0	**37.0**	**23.5**	**33.0**	**26.3**	**100**
50	0.2	0.3	0.1	0.3	0.2	0.3	0.2	**100**
100	1.2	1.2	0.7	2.2	1.1	1.3	1.2	**100**
150	4.5	4.4	2.8	6.7	3.9	4.9	4.4	**100**
200	10.8	10.7	7.9	14.5	9.5	11.9	10.7	**100**
250	20.8	20.7	14.1	26.9	17.6	23.5	20.6	97
300	38.5	38.8	29.6	45.6	35.5	42.2	38.2	**100**

As the MIQP reformulation has no failures and MILCP-PBB only has four, we increased the difficulty of the test set to have a further comparison on a harder test

TABLE 6. Aggregated node counts and running times for the first benchmark test for the MIQP reformulation

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	9.9	**6.0**	**3.0**	27.0	**4.0**	15.0	9.6	**100**
100	11.3	9.5	**3.0**	39.0	**5.2**	13.8	11.0	**100**
150	21.4	16.5	**5.0**	61.0	10.0	26.2	20.5	**100**
200	35.4	28.0	12.0	75.0	21.0	50.5	34.2	**100**
250	65.4	66.0	**7.0**	175.0	35.8	71.5	60.4	**100**
300	128.0	77.0	**8.0**	447.0	61.5	142.8	102.6	**100**
50	**0.1**	**0.1**	**0.1**	**0.2**	**0.1**	**0.2**	**0.1**	**100**
100	**0.3**	**0.3**	**0.1**	**0.4**	**0.2**	**0.3**	**0.3**	**100**
150	**0.7**	**0.7**	0.5	**1.0**	**0.6**	**0.8**	**0.7**	**100**
200	**1.4**	**1.4**	**1.2**	**1.7**	**1.4**	**1.5**	**1.4**	**100**
250	**2.2**	**2.2**	**1.7**	**2.9**	**2.0**	**2.4**	**2.2**	**100**
300	**3.5**	**3.0**	**2.5**	**6.0**	**3.0**	**3.6**	**3.4**	**100**

set for the MIQP reformulation and MILCP-PBB. As the other two methods clearly outperform the MILP formulation, we do not compare the MILP formulation on the more difficult set. We built a second test set of 300 random instances as before but double both the instance sizes as well as the fraction of the integer variables. For this test, we also tripled the time limit and now consider as failures only those instances that are not solved within 3 h.

The comparison of the methods applied to these instances is shown in Figure 6 and Tables 7 and 17, where we excluded 86 instances since no method is able to solve them. We can notice that, again, the number of nodes needed by MILCP-PBB is significantly smaller than the one needed by Gurobi. MILCP-PBB is also faster and has significantly less unsolved instances. Thus, it turns out to be more robust as well. MILCP-PBB and Gurobi have 19 as well as 49, respectively, failures on instances of Type (a), 34 and 53 failures on instances of Type (b), as well as 40 and 61 failures on instances of Type (c). A possible explanation for these results is the difference in the size of the respective branching trees. As the size of a branch-and-bound tree roughly grows exponentially with the number of binary variables, the larger number of nodes in the tree of the MIQP reformulation becomes even larger for the more difficult instances. While Gurobi needs less time per node and probably also finds the optimal solution, the sheer size of the tree prevents it from proving optimality within the time limit. Extensive tables including node counts, running times, and optimality gaps for all instances including the instances not solved by both solvers can be found in Appendix H.

6. CONCLUSION

We presented, analyzed, enhanced, and tested a novel penalty branch-and-bound method for solving MILCPs. Here, "solving" means that we indeed compute a solution if one exists or that we compute an approximate solution that minimizes an infeasibility measure based on the violation of the integrality and complementarity conditions of the problem. Together with further MILCP-tailored enhancements such as valid inequalities, node selection strategies, branching rules, and warmstarting techniques, this leads to a method that significantly outperforms two benchmark approaches that we compared our method with.

Despite these contributions, many interesting research questions remain open from which we finally want to sketch four:

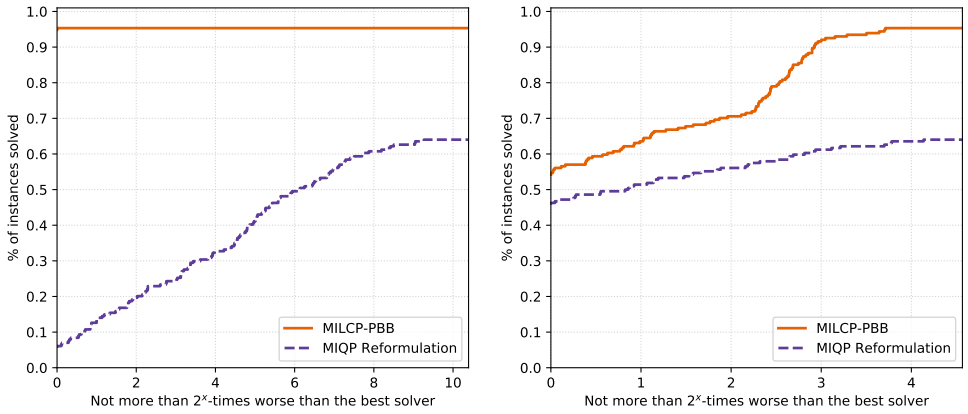

FIGURE 6. Performance profiles for the number of branch-and-bound nodes (left) and the running time (right) for the MIQP reformulation and our algorithm (for the second test set).

TABLE 7. Aggregated node counts and running times for the second benchmark test for MILCP-PBB

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
100	**17.1**	**17.0**	**9.0**	**31.0**	**13.5**	**21.0**	**17.0**	**100**
200	**43.1**	**39.0**	**21.0**	**107.0**	**30.5**	**49.5**	**42.1**	93
300	**96.6**	**76.0**	**35.0**	**259.0**	**67.5**	**123.5**	**91.0**	93
400	**246.8**	**183.0**	**37.0**	**779.0**	**139.0**	**309.0**	**205.6**	87
500	**296.5**	**297.0**	**113.0**	**685.0**	**131.0**	**385.0**	**262.0**	**93**
600	**183.0**	**185.0**	**163.0**	**201.0**	**174.0**	**193.0**	**182.6**	**80**
100	2.4	2.4	1.7	3.1	2.2	2.6	2.4	**100**
200	21.7	19.5	13.4	52.9	17.3	24.6	21.0	93
300	99.6	92.9	49.0	**215.8**	70.9	123.2	93.5	93
400	**370.0**	**336.0**	134.3	**955.4**	**235.2**	**379.4**	**319.4**	87
500	**483.9**	**421.7**	**154.6**	**881.9**	**385.2**	**479.0**	**441.9**	**93**
600	**662.2**	**698.5**	**512.0**	**775.9**	**605.3**	**737.2**	**652.4**	**80**

(1) Is it possible to extend our penalty branch-and-bound method to MILCPs with indefinite matrices M? In this case, the QPs to be solved in the nodes of our branch-and-bound tree become nonconvex so that one either needs to exploit global optimization techniques in these nodes or to modify the bounding step applied during the search in the tree.
(2) Are there further techniques as used in classic branch-and-bound for MILPs such as presolve or heuristics as well as additional valid inequalities, etc. that are explicitly tailored to the setting of MILCPs? In particular, the question remains open on how the optimality cuts can be modified and implemented so that they are useful for practical computations.
(3) It would be interesting to see specific applications of MILCPs solved with our method. Moreover, a further interesting aspect would be the problem-specific development of techniques such as valid inequalities etc. to additionally speed up the solution process.
(4) Finally, further algorithmic developments for MILCPs would clearly benefit from test sets of real-world instances for testing and comparing algorithms.

As we wrote in the introduction, there are many potential real-world applications of MILCPs. However, up to now, these problems have not been widely studied due to a lack of existing solution methods. This is exactly the gap in the literature that we want to (start to) fill with this paper and we thus had to resort to a randomly generated test set. Whether the instances are representative for the hardness of potential real-world applications is, consequently, hard to say and a collection of real-world instances would clearly propel the development and testing of further methods for solving MILCPs.

Acknowledgments

The first author acknowledges support within the project RM120172A2970290, which has received funding from Sapienza, University of Rome. The third author thanks the DFG for their support within project A05 and B08 in the "SFB TRR 154 Mathematical Modelling, Simulation and Optimization using the Example of Gas Networks". The last author has been supported by the German Research Foundation (DFG) within the Research Training Group 2126: "Algorithmic Optimization".

Appendix A. Detailed Description of the Test Set

In order to build a proper test set of MILCP instances, we created matrices $M \in \mathbb{R}^{n\times n}$ using the sprandsym function of MATLAB for sizes $n \in \{50, 100, 150, 200, 250, 300, 350, 400, 450, 500\}$. The corresponding matrix densities have been chosen so that they roughly follow the sigmoid-like function

$$d(n) := \frac{1}{1 + e^{\frac{1}{50}n-5}}.$$

Moreover, we obtain a random non-negative spectrum with an upper bound of 100 for the eigenvalues.

The set of integral variables I has been chosen as a random sample of size

$$r(n) := \frac{1}{5\left(1 + e^{\frac{1}{80}n-3}\right)}.$$

Finally, we built vectors $q \in \mathbb{R}^n$ for the four different "degrees of feasibility"; see Section 5. In order to build instances of Type (a), for which only feasibility with respect to Z is guaranteed (i.e., Condition (i) is satisfied), we set $q = x - Mz$ starting from two random vectors $x, z \in \mathbb{R}^n$ such that $x \geq 0$, $z \geq 0$, and $z_I \in [0, 1]^I$. Note that it is possible that this process yields instances for which the integrality or complementarity constraints are satisfied as well—although this is rather unlikely. Instances of Type (b), for which feasibility with respect to Z (Condition (i)) and integrality (Condition (ii)) are guaranteed, have been built by setting $q = x - Mz$ with $x, z \in \mathbb{R}^n$ being randomly generated so that, besides $x \geq 0$ and $z \geq 0$, also $z_I \in \{0, 1\}^I$ holds. In order to build instances of Type (c), for which feasibility w.r.t. Z (Condition (i)) and the complementarity constraint (Condition (iii)) are fulfilled, we set $q = -Mz$ with $z \in \mathbb{R}^n$ being a randomly created point with $z \geq 0$ and $z_I \in [0, 1]^I$. Note that this is the same procedure as for the first test set. Instances of Type (d), for which all three conditions are fulfilled, have been built by setting $q = -Mz$ with $z \in \mathbb{R}^n$ being a randomly created point with $z \geq 0$ and $z_I \in \{0, 1\}^I$ (as we did for the instances of Type (b)). The "degree of feasibility" of the instance clearly has a significant impact on its difficulty; see Figure 7, where a comparison of the performances of MILCP-PBB with different branching rules on the instances is reported.

Instances of Type (d) that have been created to be feasible both for the complementarity as well as the integrality conditions, turned out to be very easy. Most of them have been solved in the root node of the corresponding branch-and-bound tree. Thus, we decided to exclude them from our computational analysis. Instances of Type (a) and (b) that not have been forced to be feasible w.r.t. the complementarity conditions and which are either forced to be integer-feasible or not are also solved rather quickly. The most complicated

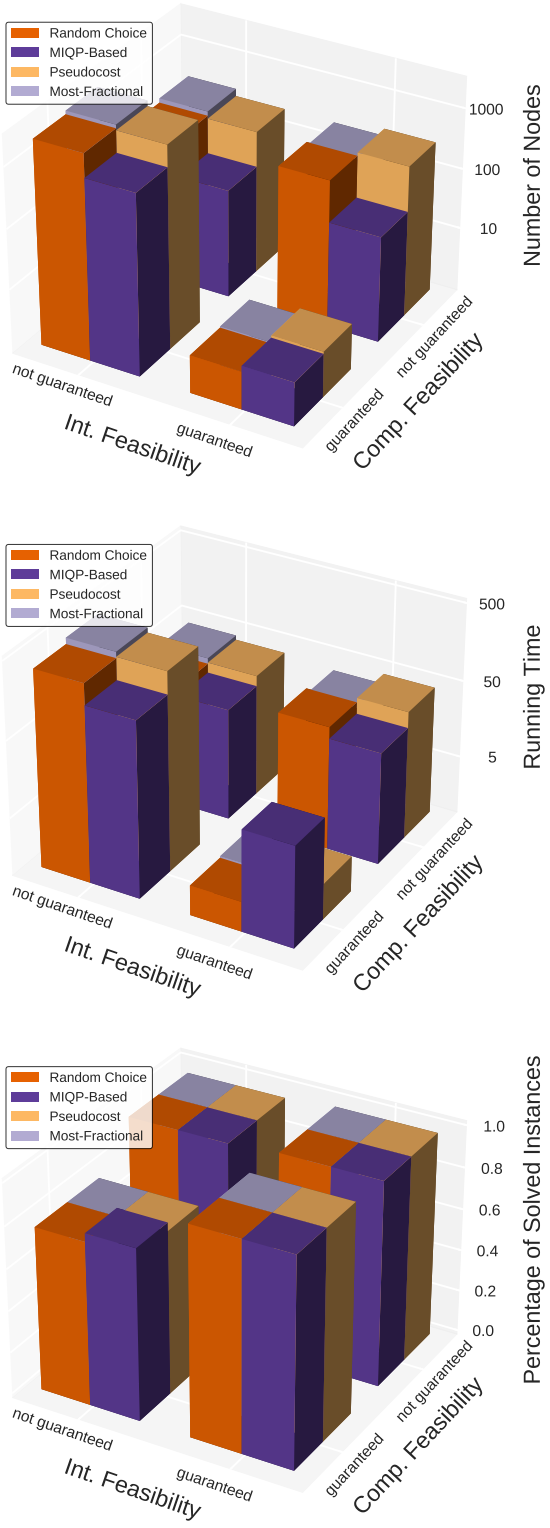

FIGURE 7. Test of different branching rules in dependence of the "degree of feasibility" of the instance

instances are those of Type (c) which are forced to be feasible w.r.t. the complementarity conditions but which are not forced to be integer feasible. This is possibly related to the difference in which the violation of the complementarity constraint and the violation of the integrality constraints are penalized along the nodes of MILCP-PBB. While the term penalizing the violation of the complementarity constraint is added to every node problem, we are penalizing the violation of all integrality constraints only at the leaf nodes. Hence, the lower bound for instances that are complementarity feasible but that are not forced to be integer feasible will stay closer to zero for longer.

Due to these preliminary tests and experiments, we decided to construct matrices with 5 % density and we further adjusted the fraction of integer variables to make the instances of Type (a)–(c) comparably difficult. Instances of Type (a) have 8 % integer variables, instances of Type (b) have 4 % integer variables, and instances of Type (c) have 10 % integer variables.

Appendix B. Branching Rule Test

Here and in what follows, we include all tables for the aggregated running times and node counts of the settings not reported in Section 5.

Table 8. Aggregated node counts and running times for the branching rule test with random choice

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.8**	**8.0**	**3.0**	39.0	**5.0**	**21.0**	**13.3**	**100**
100	18.1	14.0	**3.0**	83.0	**7.0**	26.0	17.2	**100**
150	40.5	37.0	**7.0**	155.0	17.0	44.5	37.1	**100**
200	69.3	60.0	**15.0**	269.0	34.0	95.5	63.7	**100**
250	142.3	118.0	9.0	399.0	79.5	196.5	122.0	**100**
300	335.5	224.0	17.0	1281.0	159.0	357.5	261.5	**100**
350	704.7	435.0	**7.0**	3499.0	128.0	927.0	427.0	93
400	620.4	418.0	65.0	2221.0	115.5	1061.0	407.0	70
450	630.5	473.0	73.0	2399.0	203.0	946.0	464.2	53
500	791.9	715.0	105.0	1479.0	471.0	1151.0	644.6	27
50	0.4	0.2	**0.1**	1.2	**0.1**	0.4	0.4	**100**
100	2.7	1.6	0.6	19.4	**0.9**	3.1	2.4	**100**
150	24.0	19.6	2.4	100.0	6.6	25.5	18.0	**100**
200	72.6	56.6	**13.1**	248.0	37.4	105.3	60.1	**100**
250	213.6	184.3	12.6	561.1	126.1	279.8	160.7	**100**
300	552.3	442.0	**45.0**	**1494.0**	300.4	653.0	424.0	**100**
350	1097.6	685.5	38.0	3240.6	326.9	1690.4	717.6	93
400	1105.3	1052.3	224.6	2848.0	386.3	1768.1	823.7	70
450	1327.8	1160.7	261.9	3215.6	636.7	1878.1	1074.1	53
500	1502.6	1609.6	394.5	2167.7	1201.0	1972.1	1339.9	27

TABLE 9. Aggregated node counts and running times for the branching rule test with pseudocost branching

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	15.1	9.0	**3.0**	43.0	**5.0**	26.5	14.5	**100**
100	18.5	16.0	**3.0**	**63.0**	**7.0**	25.0	17.7	**100**
150	41.3	38.0	**7.0**	147.0	15.5	**44.0**	37.8	**100**
200	73.4	51.0	17.0	287.0	34.0	97.0	66.0	**100**
250	142.9	104.0	11.0	421.0	75.5	194.0	120.0	**100**
300	345.7	235.0	19.0	**1171.0**	147.5	390.0	264.0	**100**
350	704.0	389.0	9.0	3041.0	115.0	958.0	422.5	97
400	658.7	395.0	67.0	2513.0	86.5	1090.5	394.9	67
450	620.5	377.0	53.0	2329.0	244.0	916.0	453.6	53
500	750.4	591.0	107.0	1395.0	526.0	1054.0	625.3	23
50	0.3	0.2	**0.1**	0.9	0.2	0.4	0.3	**100**
100	2.5	**1.5**	**0.4**	15.0	**0.9**	3.1	**2.2**	**100**
150	24.0	19.8	**2.3**	98.4	6.5	25.7	18.1	**100**
200	72.9	55.3	14.5	250.1	33.3	95.6	58.6	**100**
250	211.8	180.3	18.0	519.1	113.6	273.7	155.3	**100**
300	594.2	462.7	60.2	1607.5	310.0	767.7	450.4	**100**
350	1178.3	997.9	40.9	3177.6	312.5	1862.6	741.4	97
400	1185.6	914.0	251.3	3495.7	311.3	1879.6	817.2	67
450	1258.5	901.8	**223.1**	3084.2	598.5	1902.2	1009.1	53
500	1473.1	1581.2	**352.4**	2133.3	1257.9	1864.7	1302.6	23

TABLE 10. Aggregated node counts and running times for the branching rule test with most-fractional variable branching

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	13.9	**8.0**	**3.0**	**37.0**	**5.0**	22.5	13.4	**100**
100	18.6	16.0	**3.0**	81.0	**7.0**	23.0	17.8	**100**
150	41.3	**33.0**	**7.0**	151.0	**14.0**	46.0	37.8	**100**
200	67.9	58.0	23.0	**233.0**	35.0	90.5	62.9	**100**
250	144.5	126.0	7.0	403.0	77.5	163.0	124.3	**100**
300	354.7	228.0	19.0	1385.0	139.0	391.0	263.2	**100**
350	742.2	439.0	**7.0**	3641.0	124.0	1001.0	445.6	90
400	631.0	434.0	51.0	2343.0	119.0	1038.5	412.1	63
450	648.9	539.0	93.0	2265.0	224.0	908.0	483.8	57
500	711.9	683.0	119.0	1375.0	429.0	974.0	585.0	27
50	**0.2**	**0.1**	**0.1**	0.8	**0.1**	**0.2**	**0.2**	**100**
100	2.6	**1.5**	**0.4**	16.5	**0.9**	3.1	2.3	**100**
150	23.4	18.3	2.8	82.8	6.2	25.7	17.8	**100**
200	68.1	51.8	22.4	214.4	31.5	80.8	57.1	**100**
250	221.7	214.7	**12.4**	584.8	128.8	258.4	167.0	**100**
300	592.6	456.9	53.9	1763.6	270.0	728.6	437.6	**100**
350	1224.6	956.6	**32.2**	3573.5	352.4	2046.3	786.4	90
400	1202.0	1106.9	197.3	3345.7	401.1	1506.8	882.5	63
450	1343.3	1298.9	329.6	3057.6	551.8	2039.7	1085.8	57
500	1390.6	1469.6	443.9	2127.8	1097.6	1749.0	1243.3	27

Appendix C. Node Selection Test

Table 11. Aggregated node counts and running times for the node selection test with breadth-first search

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	**35.6**	**24.0**	**7.0**	**139.0**	**13.5**	**42.5**	**32.6**	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	**54.9**	**100**
250	**112.3**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	200.3	142.0	**15.0**	743.0	**95.0**	**254.5**	164.4	**100**
350	277.1	**144.0**	**7.0**	**735.0**	**79.0**	**479.5**	203.2	**77**
400	351.3	**271.0**	**47.0**	**813.0**	**78.0**	**613.0**	259.4	67
450	315.1	345.0	**51.0**	651.0	127.0	**473.5**	264.1	**47**
500	461.0	519.0	**71.0**	**923.0**	**265.5**	544.5	381.6	20
50	**0.3**	**0.2**	0.2	0.7	**0.2**	**0.3**	**0.3**	**100**
100	4.7	3.0	0.8	23.8	2.0	**4.7**	4.1	**100**
150	**25.9**	19.0	6.0	**90.3**	10.3	**30.6**	**21.1**	**100**
200	**91.2**	73.2	30.4	**272.7**	54.4	115.7	79.3	**100**
250	**249.2**	228.0	37.1	807.3	**148.4**	295.5	**198.8**	**100**
300	595.8	475.8	**101.4**	1993.4	281.5	826.4	467.3	**100**
350	1103.9	576.6	110.1	3052.6	**364.4**	1647.0	742.1	**77**
400	**1452.9**	1173.7	**304.9**	3312.1	515.5	**2336.5**	1077.2	67
450	1599.1	1707.9	418.3	2893.0	763.8	2440.0	1328.6	**47**
500	**2000.0**	**2069.0**	**535.6**	3257.8	1530.5	**2555.6**	**1742.3**	20

Table 12. Aggregated node counts and running times for the node selection test with lower-bound-push

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	**35.6**	**24.0**	**7.0**	**139.0**	**13.5**	**42.5**	**32.6**	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	**54.9**	**100**
250	**112.3**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	195.5	142.0	**15.0**	**741.0**	**95.0**	**254.5**	162.6	**100**
350	**277.0**	**144.0**	**7.0**	**735.0**	**79.0**	**479.0**	**203.1**	73
400	351.3	**271.0**	**47.0**	**813.0**	**78.0**	**613.0**	259.4	63
450	**314.6**	**343.0**	**51.0**	**647.0**	127.0	**473.5**	**263.7**	**47**
500	**459.3**	**516.0**	**71.0**	**923.0**	**265.5**	**540.0**	**380.3**	20
50	**0.3**	**0.2**	0.2	0.6	**0.2**	**0.3**	**0.3**	**100**
100	4.6	2.7	**1.2**	**20.6**	**1.9**	5.0	4.1	**100**
150	29.0	20.5	**5.7**	97.2	**10.2**	34.3	23.1	**100**
200	99.3	75.6	**25.7**	329.1	54.0	131.7	83.9	**100**
250	267.2	**217.4**	**34.8**	872.2	158.8	307.6	208.1	**100**
300	630.8	495.9	116.9	2252.5	350.4	**761.6**	501.2	**100**
350	1233.5	606.3	105.0	3389.2	410.1	1977.7	816.9	73
400	1594.7	1605.4	337.1	3521.9	556.6	2717.4	1179.8	63
450	1573.3	1729.7	399.1	2829.8	783.4	**2206.1**	1322.0	**47**
500	2161.2	2345.8	649.4	3359.0	**1346.4**	3027.0	1862.0	20

Appendix D. Warmstart Test

Table 13. Aggregated node counts and running times for the warmstart test without any warmstart

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	35.6	**24.0**	**7.0**	**139.0**	13.5	**42.5**	32.6	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	54.9	**100**
250	**112.3**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	**193.3**	**141.0**	**15.0**	**741.0**	**95.0**	**254.5**	161.7	**100**
350	356.3	**215.0**	**7.0**	1119.0	**79.0**	**679.0**	251.1	**87**
400	418.3	**279.0**	**47.0**	**1245.0**	**81.0**	**719.0**	301.1	70
450	314.7	**345.0**	51.0	**647.0**	**126.5**	**473.5**	263.7	47
500	478.6	**543.0**	71.0	**1043.0**	**189.0**	**547.0**	372.3	23
50	**0.3**	**0.3**	0.2	1.2	**0.2**	**0.3**	**0.3**	**100**
100	4.5	2.4	1.1	20.6	1.9	5.4	4.0	**100**
150	26.8	19.3	6.5	111.9	10.0	31.4	21.5	**100**
200	90.4	62.8	28.7	295.1	50.6	116.2	77.5	**100**
250	241.2	209.7	32.3	762.3	161.0	309.2	190.7	**100**
300	533.8	451.4	112.5	1764.6	260.8	655.1	429.0	**100**
350	1256.3	787.4	105.2	3360.6	369.7	2311.8	826.9	**87**
400	1550.7	1401.0	300.9	3522.8	**396.9**	2497.7	1125.9	70
450	1460.0	**1505.9**	345.3	2608.6	751.1	2112.5	1223.4	47
500	**1995.4**	2206.2	584.4	**3375.5**	1055.1	**2755.9**	**1663.8**	23

Table 14. Aggregated node counts and running times for the warmstart test using PStart/DStart

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	**13.9**	**8.0**	**3.0**	**37.0**	**5.0**	**21.0**	**13.3**	**100**
100	**16.1**	**10.0**	**3.0**	**73.0**	**7.0**	**21.0**	**15.3**	**100**
150	**35.3**	**24.0**	**7.0**	**139.0**	**12.0**	**42.5**	**32.3**	**100**
200	**59.5**	**43.0**	**15.0**	**235.0**	**31.0**	**82.5**	**54.8**	**100**
250	**112.2**	**96.0**	**5.0**	**353.0**	**67.5**	**131.0**	**97.6**	**100**
300	**193.3**	**141.0**	**15.0**	**741.0**	**95.0**	**254.5**	**161.6**	**100**
350	**355.4**	**215.0**	**7.0**	**1107.0**	**79.0**	**679.0**	**249.9**	83
400	**417.6**	**279.0**	**47.0**	**1245.0**	**81.0**	**719.0**	**300.3**	70
450	**312.4**	**345.0**	**41.0**	**649.0**	**126.5**	**473.5**	**258.6**	47
500	**476.6**	**543.0**	**61.0**	**1043.0**	**189.0**	**547.0**	**366.6**	17
50	**0.3**	**0.3**	**0.1**	1.2	**0.2**	**0.3**	**0.3**	**100**
100	4.0	2.4	1.3	15.7	1.8	4.5	3.6	**100**
150	25.1	17.9	4.5	84.9	**9.3**	**27.7**	20.1	**100**
200	86.2	63.2	25.9	296.0	48.5	116.0	73.5	**100**
250	245.0	207.1	**26.5**	754.8	145.4	309.0	189.6	**100**
300	549.8	450.0	103.4	1586.1	299.5	745.0	442.9	**100**
350	1324.4	**708.9**	113.3	3361.2	383.7	2404.4	857.4	83
400	1544.5	**1039.5**	292.4	3201.7	460.1	2421.4	1136.5	70
450	1500.2	1591.4	406.3	2919.3	719.5	2102.1	1262.7	47
500	2119.3	**2194.8**	838.8	3386.6	**1015.6**	3160.8	1822.6	17

Appendix E. Valid Inequalities Test

Table 15. Aggregated node counts and running times for the valid inequalities test with no cuts

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	13.9	8.0	**3.0**	37.0	**5.0**	21.0	13.3	**100**
100	16.1	10.0	**3.0**	73.0	7.0	21.0	15.3	**100**
150	35.6	24.0	**7.0**	139.0	13.5	42.5	32.6	**100**
200	59.5	43.0	15.0	235.0	31.0	82.5	54.9	**100**
250	113.1	99.0	**5.0**	353.0	67.0	131.0	98.0	**100**
300	193.3	141.0	15.0	741.0	95.0	254.5	161.7	**100**
350	532.9	309.0	7.0	2701.0	81.0	735.0	334.5	**100**
400	860.0	579.0	47.0	3929.0	109.0	1056.0	493.6	90
450	593.0	402.0	51.0	1713.0	163.5	834.5	416.3	67
500	816.1	923.0	71.0	1523.0	519.0	1107.0	663.9	37
50	**0.2**	**0.2**	**0.1**	0.5	**0.2**	**0.3**	**0.2**	**100**
100	2.3	1.6	0.8	10.2	1.2	2.1	2.2	**100**
150	13.8	10.0	3.3	50.6	5.8	16.0	11.7	**100**
200	43.4	31.3	13.0	150.8	24.2	58.1	37.4	**100**
250	136.4	111.7	14.2	444.8	74.5	179.0	102.4	**100**
300	293.8	244.4	45.6	851.7	150.1	391.9	235.9	**100**
350	788.0	530.1	64.5	1962.8	212.7	1342.9	527.6	**100**
400	1250.2	1140.1	192.5	3514.5	364.3	1476.5	867.6	90
450	1215.6	1032.2	252.7	2872.7	445.4	1917.3	934.0	67
500	1545.5	1397.8	363.5	2646.2	1043.1	2135.5	1345.9	37

Appendix F. A First Benchmark Test

Table 16. Aggregated node counts and running times for the first benchmark test for the MILP reformulation

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
50	36.9	24.0	5.0	142.0	15.0	46.8	33.7	**100**
100	330.6	193.0	13.0	1416.0	69.5	462.5	227.6	**100**
150	866.5	921.0	26.0	2928.0	167.8	1179.8	582.3	**100**
200	6122.2	2985.0	122.0	78700.0	1454.8	4958.8	2755.0	**100**
250	30406.6	11944.0	287.0	185996.0	5482.0	29671.0	12745.4	83
300	36647.2	45035.0	2065.0	71988.0	21157.5	46958.8	24581.9	27
50	**0.1**	**0.1**	0.1	**0.2**	**0.1**	**0.2**	**0.1**	**100**
100	0.7	0.4	0.2	1.7	0.3	0.9	0.7	**100**
150	2.9	3.2	**0.3**	7.3	0.9	4.2	2.8	**100**
200	64.1	28.3	1.3	941.0	10.7	46.1	28.8	**100**
250	599.2	239.2	2.6	3169.1	106.1	572.8	252.7	83
300	1434.6	1305.0	56.4	3529.4	809.4	1797.3	926.1	27

Appendix G. A Second Benchmark Test

Table 17. Aggregated node counts and running times for the second benchmark test for the MIQP reformulation

n	Avg.	Med.	Min.	Max.	$Q(0.25)$	$Q(0.75)$	GSM	Perc.
100	45.1	33.5	11.0	209.0	18.5	63.0	41.0	**100**
200	639.1	408.5	34.0	2856.0	207.5	615.8	416.9	**100**
300	2978.9	2270.0	283.0	16147.0	1948.2	2760.5	2269.2	**100**
400	40317.4	16547.0	897.0	160445.0	8167.0	57017.0	17853.9	**97**
500	45637.3	30480.0	5734.0	165419.0	13164.0	52819.0	28969.2	47
600	88273.7	85589.0	38908.0	140324.0	62248.5	112956.5	77615.3	13
100	**0.4**	**0.4**	**0.2**	**0.7**	**0.3**	**0.4**	**0.4**	**100**
200	**5.8**	**3.5**	**2.1**	**24.0**	**3.0**	**4.9**	**5.1**	**100**
300	**69.2**	**51.7**	**5.0**	242.0	**40.2**	**60.3**	**54.3**	**100**
400	1597.7	471.8	**39.3**	6670.1	349.0	2664.9	721.6	**97**
500	2556.7	1768.2	409.5	6946.8	1039.5	4008.1	1845.4	47
600	8217.1	9612.1	4431.1	10608.0	7021.6	10110.0	7674.2	13

Appendix H. Full Results for Benchmark Test on the Harder Set

Here, we present running times, node counts, and optimality gaps for the second test in Section 5.5. The tables are split among the different types of feasibility. Note that the instances in Table 18 have a matrix density of 5 % and 16 % integer variables, instances in Table 19 have a matrix density of 5 % and 20 % integer variables, and instances in Table 20 have a matrix density of 5 % and 8 % integer variables.

Table 18: Full table of results for the benchmark test with feasibility type (a)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
0	100	27	3.1159	0.0000	68	0.5751	0.0000
1	100	15	2.3551	0.0000	33	0.1886	0.0000
2	100	21	2.6895	0.0000	103	0.7351	0.0000
3	100	17	2.2424	0.0000	15	0.3592	0.0001
4	100	15	2.3921	0.0000	44	0.1839	0.0000
5	100	19	2.2394	0.0000	34	0.6797	0.0000
6	100	11	2.3276	0.0000	18	0.3701	0.0000
7	100	15	2.1230	0.0000	23	0.4201	0.0000
8	100	11	2.1795	0.0000	20	0.3384	0.0000
9	100	13	1.8813	0.0000	14	0.3545	0.0000
10	200	35	17.5621	0.0000	532	4.0627	0.0001
11	200	27	14.9239	0.0000	279	3.0247	0.0000
12	200	39	18.7982	0.0000	399	3.4994	0.0000
13	200	43	19.6466	0.0000	191	2.5610	0.0000
14	200	27	16.4442	0.0000	127	2.6382	0.0000
15	200	35	17.0882	0.0000	213	3.1579	0.0000
16	200	35	19.3562	0.0000	345	3.5871	0.0001
17	200	29	18.7861	0.0000	110	2.5195	0.0001
18	200	37	18.8028	0.0000	565	3.8783	0.0000
19	200	14	11.0052	0.0027	42	2.3146	0.0000

Continued on next page

Table 18: Full table of results for the benchmark test with feasibility type (a)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
20	300	109	99.8787	0.0000	7509	189.4373	0.0001
21	300	41	52.1754	0.0000	2507	51.3995	0.0001
22	300	63	71.4824	0.0000	2413	51.9769	0.0001
23	300	73	71.2369	0.0000	2807	54.0534	0.0001
24	300	43	64.3237	0.0000	1856	31.6593	0.0001
25	300	72	78.4768	0.0014	1253	22.1835	0.0001
26	300	63	69.7968	0.0000	1979	40.4144	0.0001
27	300	73	78.0971	0.0000	2036	39.3973	0.0001
28	300	49	61.9105	0.0000	2352	47.4422	0.0001
29	300	143	123.7982	0.0000	2258	56.8950	0.0001
30	400	69	155.2925	0.0000	3991	164.9751	0.0001
31	400	98	186.8513	0.0177	16683	638.9243	0.0001
32	400	215	354.8989	0.0000	8855	348.2817	0.0001
33	400	43	134.2780	0.0000	1387	62.4837	0.0001
34	400	139	240.8656	0.0000	12310	426.6363	0.0001
35	400	161	244.0987	0.0000	12252	442.3353	0.0000
36	400	65	166.9921	0.0000	2175	87.9427	0.0001
37	400	183	235.1562	0.0000	16547	431.5883	0.0001
38	400	485	462.7339	0.0000	10324	414.0996	0.0001
39	400	65	152.6845	0.0000	1518	63.0660	0.0001
40	500	335	608.6910	0.0000	265002	10814.2415	0.0014
41	500	167	388.2661	0.0000	16672	1232.6245	0.0001
42	500	117	466.1749	0.0000	10873	681.7911	0.0001
43	500	131	472.0715	0.0000	11418	1039.4614	0.0001
44	500	207	279.1084	0.0000	5734	409.5402	0.0001
45	500	123	154.5519	0.0000	14787	1087.2545	0.0001
46	500	321	479.0112	0.0000	40024	1768.1539	0.0001
47	500	134	287.6528	0.0090	203450	10816.5233	0.0023
48	500	313	390.9182	0.0000	165419	6946.8199	0.0001
49	500	297	380.2324	0.0000	101750	4942.9432	0.0001
50	600	293	899.3240	0.0000	120917	10824.2120	0.0003
51	600	327	696.3595	0.0000	110087	10829.7680	0.0002
52	600	185	512.0450	0.0000	38908	4431.0990	0.0001
53	600	543	1262.1285	0.0000	140118	10829.3220	0.0004
54	600	129	794.9032	0.0000	156342	10823.6857	0.0012
55	600	163	698.5480	0.0000	85589	9612.0795	0.0001
56	600	201	775.8917	0.0000	140324	10608.0138	0.0001
57	600	223	736.2904	0.0000	138024	10827.5670	0.0003
58	600	607	1019.5381	0.0000	121674	10827.8138	0.0013
59	600	677	1418.2227	0.0000	134749	10823.6026	0.0016
60	700	249	975.6561	0.0000	70611	10836.4064	0.0002
61	700	163	1091.6707	0.0000	80347	10830.6276	0.0004
62	700	303	1174.9706	0.0000	54001	10835.4704	0.0004
63	700	241	1058.5091	0.0000	79913	10838.8376	0.0012
64	700	837	1907.9259	0.0000	88157	10833.0279	0.0247
65	700	1015	2113.7192	0.0000	71434	10831.3456	0.0056
66	700	301	1187.2362	0.0000	73266	10839.0607	0.0011
67	700	369	1465.1541	0.0000	75375	10841.1106	0.0037
68	700	1181	2456.5842	0.0000	67206	10829.3712	0.0013
69	700	213	1729.2047	0.0000	64898	10836.1695	0.0002

Continued on next page

Table 18: Full table of results for the benchmark test with feasibility type (a)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
70	800	209	2894.0162	0.0000	50340	10851.0105	0.0009
71	800	805	3542.0355	0.0049	49637	10853.6512	0.0039
72	800	359	1987.7595	0.0000	45985	10849.3627	0.0005
73	800	1099	4571.4084	0.0000	45138	10847.8649	0.0005
74	800	126	1556.4838	0.0025	48760	10847.7848	0.0014
75	800	905	2953.2680	0.0000	43308	10846.2972	0.0032
76	800	129	1381.9814	0.0000	52954	10846.8718	0.0004
77	800	246	1621.6197	0.0042	64855	10844.3243	0.0029
78	800	3069	6670.6305	0.0000	47909	10848.3922	0.0019
79	800	2215	6693.6248	0.0000	51677	10843.3809	0.0012
80	900	256	3563.1181	0.0008	24062	10865.4706	0.0007
81	900	2718	10801.2767	0.0017	39963	10861.3130	0.0015
82	900	1947	6118.9939	0.0000	40548	10857.4068	0.0010
83	900	2067	10179.3084	0.0000	28530	10862.6133	0.0011
84	900	1463	7062.3731	0.0000	33852	10861.5166	0.0006
85	900	1434	10800.9542	0.0010	31292	10871.1387	0.0007
86	900	401	4306.1255	0.0000	26750	10863.4494	0.0016
87	900	1399	10801.3957	0.0011	29690	10858.3488	0.0007
88	900	641	4740.7997	0.0000	32362	10857.8563	0.0009
89	900	2391	10458.6046	0.0000	30177	10844.8206	0.0009
90	1000	1163	10802.2212	0.0032	21296	10868.1840	0.0022
91	1000	2982	10801.7772	0.0050	21011	10866.1448	0.0040
92	1000	1508	10801.4001	0.0012	19055	10859.4321	0.0009
93	1000	1335	9664.4435	0.0000	20940	10861.2169	0.0016
94	1000	830	10802.3529	0.0013	19697	10870.9180	0.0009
95	1000	2758	10800.8066	0.0153	25807	10863.2771	0.0128
96	1000	2781	10530.2989	0.0000	23170	10864.4643	0.0008
97	1000	1283	10801.6599	0.0024	25515	10858.4379	0.0019
98	1000	1250	10801.9894	0.0046	17429	10864.9515	0.0034
99	1000	2745	10801.4159	0.0037	21093	10854.0908	0.0033

Table 19: Full table of results for the benchmark test with feasibility type (b)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
0	100	11	1.8736	0.0000	11	0.2401	0.0000
1	100	17	2.5071	0.0000	35	0.4127	0.0000
2	100	11	1.8778	0.0000	14	0.3880	0.0000
3	100	31	3.0990	0.0000	209	0.6444	0.0000
4	100	15	2.5030	0.0000	51	0.3522	0.0000
5	100	15	2.5308	0.0000	27	0.3500	0.0000
6	100	11	1.7119	0.0000	12	0.3445	0.0000
7	100	21	2.8848	0.0000	72	0.5284	0.0000
8	100	9	1.7975	0.0000	11	0.2022	0.0000
9	100	11	1.9970	0.0000	16	0.2678	0.0000
10	200	39	20.1349	0.0000	478	3.4626	0.0001
11	200	75	27.5083	0.0000	363	3.3318	0.0000

Continued on next page

Table 19: Full table of results for the benchmark test with feasibility type (b)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
12	200	51	23.1963	0.0000	2451	17.3827	0.0001
13	200	31	17.3348	0.0000	320	2.7709	0.0001
14	200	53	24.3716	0.0000	435	3.5231	0.0000
15	200	21	13.3523	0.0000	48	2.0762	0.0001
16	200	47	20.3192	0.0000	418	3.3012	0.0001
17	200	21	14.0000	0.0000	34	2.1945	0.0001
18	200	57	25.2956	0.0000	2856	19.3368	0.0001
19	200	29	17.1288	0.0000	86	2.2520	0.0000
20	300	75	85.2808	0.0000	2745	57.3262	0.0001
21	300	39	50.5451	0.0000	341	7.5386	0.0001
22	300	149	131.9792	0.0000	2090	46.8724	0.0000
23	300	35	49.0304	0.0000	283	5.0292	0.0001
24	300	25	45.2144	0.0025	2597	39.8905	0.0001
25	300	71	74.3649	0.0000	1588	27.9635	0.0001
26	300	259	215.7792	0.0000	1756	42.2825	0.0000
27	300	95	89.7939	0.0000	3977	55.6581	0.0001
28	300	105	110.0118	0.0000	16147	241.9878	0.0001
29	300	125	108.0865	0.0000	1106	21.9187	0.0001
30	400	42	158.0413	0.0046	3202	178.1554	0.0001
31	400	255	335.9703	0.0000	32060	1002.8782	0.0001
32	400	309	361.5803	0.0000	34792	1298.5175	0.0001
33	400	75	206.8250	0.0000	10523	471.8054	0.0001
34	400	227	364.7588	0.0000	8167	433.7521	0.0001
35	400	92	210.3800	0.0012	62434	2256.8009	0.0001
36	400	37	138.3447	0.0000	897	39.3350	0.0001
37	400	243	310.8748	0.0000	57017	2005.0877	0.0001
38	400	165	287.8678	0.0000	7969	349.0208	0.0001
39	400	357	395.5973	0.0020	267946	7095.4798	0.0001
40	500	113	385.2338	0.0000	13164	806.6624	0.0001
41	500	511	845.0595	0.0000	30480	2447.1081	0.0001
42	500	769	1563.5448	0.0000	204209	10813.9150	0.0004
43	500	1009	1436.5816	0.0000	171660	10815.6210	0.0003
44	500	685	881.8793	0.0000	48103	4008.1435	0.0001
45	500	2393	3237.3454	0.0000	157972	10816.5673	0.0021
46	500	385	421.6556	0.0000	52819	3227.4129	0.0001
47	500	485	746.2523	0.0000	82042	4638.5376	0.0001
48	500	543	532.2371	0.0000	187584	10814.5764	0.0007
49	500	155	227.8922	0.0007	34045	2399.3100	0.0001
50	600	1221	2966.6930	0.0020	115289	10824.2649	0.0012
51	600	539	1488.9798	0.0000	80734	10823.4649	0.0005
52	600	377	781.4358	0.0000	120840	10824.6150	0.0002
53	600	473	1174.9950	0.0000	120541	10824.3100	0.0010
54	600	411	1146.9656	0.0000	123993	10823.1460	0.0003
55	600	531	1376.3102	0.0000	135998	10823.9850	0.0003
56	600	443	862.3373	0.0000	108494	10822.6135	0.0017
57	600	347	708.2811	0.0000	134048	10824.2348	0.0010
58	600	143	722.6545	0.0007	15769	1731.9034	0.0001
59	600	355	1161.8243	0.0000	125770	10824.3285	0.0003
60	700	883	2755.3962	0.0000	86915	10832.8358	0.0030
61	700	981	3975.7980	0.0000	60690	10834.3674	0.0003

Continued on next page

Table 19: Full table of results for the benchmark test with feasibility type (b)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
62	700	165	1288.9324	0.0000	59965	10834.6514	0.0002
63	700	383	1461.5977	0.0000	66430	10835.0290	0.0002
64	700	1031	2620.0381	0.0000	71492	10837.0440	0.0007
65	700	3183	5449.0948	0.0000	77091	10834.8594	0.0012
66	700	1219	4465.0251	0.0030	64723	10832.1042	0.0019
67	700	308	1820.4327	0.0031	68971	10830.7743	0.0014
68	700	801	2834.0616	0.0000	57584	10831.4539	0.0020
69	700	494	2710.5223	0.0016	64278	10833.1984	0.0009
70	800	1367	4188.1333	0.0000	48056	10844.6034	0.0006
71	800	1394	6301.3453	0.0011	35039	10843.7623	0.0007
72	800	2682	10800.9745	0.0030	38116	10846.3990	0.0023
73	800	319	2445.0687	0.0027	54200	10847.9241	0.0015
74	800	3698	10801.9776	0.0020	41462	10850.7178	0.0015
75	800	1635	4952.7012	0.0000	37047	10850.4457	0.0006
76	800	206	2589.2310	0.0019	42891	10847.8842	0.0013
77	800	1565	7296.9954	0.0000	32957	10845.7794	0.0015
78	800	5283	10801.1113	0.0016	42228	10841.0241	0.0011
79	800	1399	5201.9637	0.0000	56617	10843.3123	0.0019
80	900	1068	10803.4404	0.0018	26577	10856.2667	0.0012
81	900	1539	10800.0682	0.0019	23554	10853.1402	0.0016
82	900	3585	10801.5707	0.0014	30402	10866.2853	0.0010
83	900	2131	10801.4520	0.0011	31559	10857.8554	0.0008
84	900	996	10801.3616	0.0036	26331	10860.5435	0.0026
85	900	1157	10803.2434	0.0033	1	192.6932	inf
86	900	2086	10801.6023	0.0013	22924	10859.6405	0.0010
87	900	3121	10800.1144	0.0010	27014	10855.2115	0.0008
88	900	2281	10755.1982	0.0000	32147	10855.9983	0.0015
89	900	1440	8768.8400	0.0012	29748	10841.0772	0.0007
90	1000	1267	10686.1705	0.0000	18266	10865.9510	0.0006
91	1000	505	7181.2388	0.0025	20744	10864.9113	0.0018
92	1000	387	5606.9840	0.0018	18491	10863.1418	0.0014
93	1000	1073	10800.9104	0.0027	18227	10859.5245	0.0021
94	1000	1420	10802.2677	0.0019	18885	10852.9994	0.0015
95	1000	1877	10800.4012	0.0019	16198	10853.8927	0.0014
96	1000	244	6376.3823	0.0015	18063	10858.1245	0.0011
97	1000	50	5808.5997	0.0011	20066	10862.6262	0.0008
98	1000	1419	10800.7780	0.0018	23980	10858.5239	0.0013
99	1000	1506	10800.1857	0.0014	20533	10859.9718	0.0011

Table 20: Full table of results for the benchmark test with feasibility type (c)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
0	100	17	2.2032	0.0000	69	0.3876	0.0000
1	100	17	2.2020	0.0000	35	0.3816	0.0000
2	100	23	2.7185	0.0000	46	0.3951	0.0000
3	100	17	2.4266	0.0000	25	0.3482	0.0000

Continued on next page

Table 20: Full table of results for the benchmark test with feasibility type (c)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
4	100	21	2.5839	0.0000	73	0.1972	0.0000
5	100	17	2.3956	0.0000	60	0.1806	0.0000
6	100	17	2.3847	0.0000	31	0.2105	0.0000
7	100	25	2.7450	0.0000	93	0.3988	0.0000
8	100	21	2.5308	0.0000	26	0.3163	0.0000
9	100	23	2.6372	0.0000	64	0.3479	0.0000
10	200	29	14.2188	0.0000	142	3.0308	0.0000
11	200	33	18.1685	0.0000	768	6.0798	0.0000
12	200	47	25.6820	0.0000	873	7.5518	0.0000
13	200	49	26.3865	0.0000	451	4.4537	0.0000
14	200	53	28.1232	0.0000	1041	13.0250	0.0000
15	200	71	33.9903	0.0000	1057	8.7275	0.0000
16	200	47	22.6940	0.0000	220	3.0421	0.0000
17	200	33	18.3735	0.4603	183	2.9801	0.0000
18	200	41	21.0298	0.0000	444	4.0270	0.0000
19	200	107	52.9182	0.0000	2650	23.9550	0.0000
20	300	123	135.1874	0.0000	2729	87.4795	0.0000
21	300	75	109.0742	0.0000	2432	53.4491	0.0000
22	300	171	160.6897	0.0000	4016	148.4997	0.0000
23	300	103	122.9437	0.0000	2876	119.3457	0.0000
24	300	125	137.8550	0.0000	1090	30.7861	0.0000
25	300	77	97.9103	0.0000	1993	56.0017	0.0000
26	300	79	86.1865	0.0000	2139	50.6660	0.0000
27	300	203	168.5056	0.0000	5906	203.0584	0.0000
28	300	69	66.8797	0.0000	2282	69.0258	0.0000
29	300	71	95.9877	0.0000	2197	50.7216	0.0000
30	400	779	847.8113	0.0001	121573	4819.6968	0.0000
31	400	339	519.3352	0.0000	51856	3256.8336	0.0001
32	400	519	779.5839	0.0000	107926	3578.9797	0.0001
33	400	171	341.3093	0.0000	62027	3165.3430	0.0001
34	400	267	444.8971	0.0066	245500	10808.8307	0.1056
35	400	615	955.3639	0.0000	70852	2664.9435	0.0000
36	400	261	379.4113	0.0001	160445	6670.0696	0.0000
37	400	165	278.4771	0.0000	29917	1243.3834	0.0001
38	400	163	343.1481	0.0000	47667	1712.7171	0.0001
39	400	423	651.1013	0.0000	134887	4788.1721	0.0001
40	500	1059	2464.7738	0.0000	163745	10815.2826	0.2403
41	500	2059	4223.0813	0.0000	165779	10818.6912	0.1847
42	500	763	1678.4753	0.0000	140830	10817.5013	0.2403
43	500	285	765.3047	0.0000	198463	10817.2753	0.1256
44	500	923	885.9161	0.0043	136841	10815.9276	0.1945
45	500	407	647.9789	0.0000	126879	10816.3697	0.0875
46	500	265	284.3640	0.0000	128834	10816.3495	0.3436
47	500	665	1058.7074	0.0000	140357	10815.7690	0.0676
48	500	485	571.5811	0.0000	174696	10816.4255	0.2317
49	500	629	706.5074	0.0000	189088	10815.0503	0.2593
50	600	10808	10800.2888	0.9262	79144	10826.8733	0.4985
51	600	3943	3818.3468	0.0000	100441	10828.4613	0.5088
52	600	683	1024.9046	0.0000	107480	10828.8295	0.4322
53	600	631	2597.7218	0.0067	97200	10826.0636	0.3061

Continued on next page

Table 20: Full table of results for the benchmark test with feasibility type (c)

		MILCP-PBB			MIQP Reformulation		
Inst.	n	Nodecount	Time	Gap	Nodecount	Time	Gap
54	600	413	1049.1375	0.0002	94083	10825.6469	0.4607
55	600	653	2568.5341	0.0000	104809	10828.5226	0.4150
56	600	54	609.9500	1.0000	97417	10826.1439	0.4225
57	600	1453	2971.0164	0.0000	102075	10829.7080	0.4732
58	600	9	232.5856	0.9680	74238	10828.7163	0.4997
59	600	678	1588.2402	0.8213	91260	10830.4672	0.4883
60	700	249	1128.5292	0.9734	41533	10849.1465	0.6142
61	700	273	1297.1256	0.9276	57176	10837.0305	0.5865
62	700	4066	10801.5454	0.6834	60318	10835.0205	0.5784
63	700	13	444.0867	1.0000	50058	10841.8503	0.4474
64	700	805	1967.7534	0.9671	45885	10844.9726	0.6096
65	700	3	423.5964	1.0000	58474	10842.7989	0.5818
66	700	8193	10800.4775	0.9730	46798	10842.8968	0.6575
67	700	449	1834.5219	0.9831	56457	10840.2731	0.6207
68	700	2154	10801.2486	0.9716	59715	10843.3221	0.5869
69	700	815	4528.8139	0.0000	58624	10847.4065	0.5740
70	800	8	1325.5795	0.9868	37286	10858.9890	0.5818
71	800	1412	10802.3115	1.0000	32379	10865.9849	0.6106
72	800	1263	9918.1721	0.0002	33698	10861.6950	0.6170
73	800	283	2182.9504	1.0000	26555	10859.3335	0.7463
74	800	2787	10800.9629	0.9007	32880	10860.2008	0.6965
75	800	5184	10801.2334	1.0000	33831	10860.9934	0.6802
76	800	2973	10800.1716	0.9940	33833	10853.8966	0.6885
77	800	711	4456.3225	0.9571	24822	10862.4918	0.7351
78	800	29	1585.9883	0.9929	29949	10863.7911	0.6905
79	800	5	1418.6417	1.0000	1	234.0652	inf
80	900	346	4289.8207	0.9899	22578	10869.1079	0.7215
81	900	1004	10802.9810	0.9772	21926	10859.6112	0.7038
82	900	1527	10800.4423	1.0000	17300	10863.7038	0.7632
83	900	539	3108.7699	0.9583	20772	10861.9778	0.7353
84	900	921	10803.1083	0.9328	19133	10856.5056	0.7033
85	900	1443	10801.9199	1.0000	19178	10859.8103	0.7166
86	900	135	3343.6416	1.0000	17451	10858.2922	0.7607
87	900	1154	10802.4067	1.0000	26547	10849.0477	0.7456
88	900	80	2913.2712	0.9839	20462	10845.4186	0.7423
89	900	1	1271.1297	1.0000	19289	10844.5248	0.7457
90	1000	163	5233.4671	1.0000	14795	10859.0219	0.7851
91	1000	2678	10800.5171	1.0000	17441	10856.4719	0.8033
92	1000	73	4245.0406	1.0000	15919	10858.5450	0.7977
93	1000	862	10802.7972	0.9878	16530	10871.7120	0.7967
94	1000	1326	10801.8961	0.9750	18380	10868.6293	0.7768
95	1000	76	4242.0261	0.9999	17985	10866.4091	0.8162
96	1000	1	2617.2543	1.0000	22577	10864.8447	0.7729
97	1000	1045	10801.3138	1.0000	19917	10861.5392	0.7658
98	1000	68	3136.0200	1.0000	24022	10850.9295	0.7608
99	1000	30	3088.7299	1.0000	19310	10854.8429	0.7707

References

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.

Beck, A. (2017). *First-order methods in optimization*. Vol. 25. SIAM. DOI: 10.1137/1.9781611974997.

Benichou, M., J. M. Gauthier, P. Girodet, G. Hentges, G. Ribiere, and O. Vincent (1971). "Experiments in mixed-integer linear programming." In: *Mathematical Programming* 1.1, pp. 76–94. DOI: 10.1007/BF01584074.

Chandrasekaran, R., S. N. Kabadi, and R. Sridhar (1998). "Integer Solution for Linear Complementarity Problem." In: *Mathematics of Operations Research* 23.2, pp. 390–402. DOI: 10.1287/moor.23.2.390.

Cottle, R. W., J.-S. Pang, and R. E. Stone (2009). *The Linear Complementarity Problem*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898719000.

Cunningham, W. H. and J. F. Geelen (1998). "Integral Solutions of Linear Complementarity Problems." In: *Mathematics of Operations Research* 23.1, pp. 61–68. DOI: 10.1287/moor.23.1.61.

De Santis, M., S. Lucidi, and F. Rinaldi (2013). "A new class of functions for measuring solution integrality in the Feasibility Pump approach." In: *SIAM Journal on Optimization* 23.3, pp. 1575–1606. DOI: 10.1137/110855351.

Dolan, E. D. and J. J. Moré (2002). "Benchmarking optimization software with performance profiles." In: *Mathematical programming* 91.2, pp. 201–213. DOI: 10.1007/s101070100263.

Dubey, D. and S. K. Neogy (2018). "Total dual integrality and integral solutions of the linear complementarity problem." In: *Linear Algebra and its Applications* 557, pp. 359–374. DOI: 10.1016/j.laa.2018.08.004.

Fomeni, F. D., S. A. Gabriel, and M. F. Anjos (2019a). "An RLT approach for solving the binary-constrained mixed linear complementarity problem." In: *Computers & Operations Research* 110, pp. 48–59. DOI: 10.1016/j.cor.2019.05.008.

– (2019b). "Applications of logic constrained equilibria to traffic networks and to power systems with storage." In: *Journal of the Operational Research Society* 70.2, pp. 310–325. DOI: 10.1080/01605682.2018.1438761.

Gabriel, S. A. (1998). "A hybrid smoothing method for mixed nonlinear complementarity problems." In: *Computational Optimization and Applications* 9.2, pp. 153–173. DOI: 10.1023/A:1018311004565.

– (2017). "Solving discretely constrained mixed complementarity problems using a median function." In: *Optimization and Engineering* 18.3, pp. 631–658. DOI: 10.1007/s11067-012-9182-2.

Gabriel, S. A., A. J. Conejo, J. D. Fuller, B. F. Hobbs, and C. Ruiz (2012). *Complementarity Modeling in Energy Markets*. Vol. 180. 1. Springer Science & Business Media. DOI: 10.1007/978-1-4419-6123-5.

Gabriel, S. A., A. J. Conejo, C. Ruiz, and S. Siddiqui (2013). "Solving discretely constrained, mixed linear complementarity problems with applications in energy." In: *Computers & Operations Research* 40.5, pp. 1339–1350. DOI: 10.1016/j.cor.2012.10.017.

Gabriel, S. A., M. Leal, and M. Schmidt (2021). "Solving Binary-Constrained Mixed Complementarity Problems Using Continuous Reformulations." In: *Computers & Operations Research* 131. Online first. DOI: 10.1016/j.cor.2020.105208.

Gabriel, S. A. and J. J. Moré (1997). "Smoothing of mixed complementarity problems." In: *Complementarity and Variational Problems: State of the Art*, pp. 105–116.

Gabriel, S. A., S. A. Siddiqui, A. J. Conejo, and C. Ruiz (2013). "Solving discretely-constrained Nash–Cournot games with an application to power markets." In: *Networks and Spatial Economics* 13.3, pp. 307–326. DOI: 10.1007/s11067-012-9182-2.

Giannessi, F. and F. Tardella (1998). "Connections between nonlinear programming and discrete optimization." In: *Handbook of combinatorial optimization*. Springer, pp. 149–188. DOI: 10.1007/978-1-4613-0303-9_3.

Lucidi, S. and F. Rinaldi (2010). "Exact penalty functions for nonlinear integer programming problems." In: *Journal of optimization theory and applications* 145.3, pp. 479–488. DOI: 10.1007/s10957-010-9700-7.

Pardalos, P. M. (1988). "Linear complementarity problems solvable by integer programming." In: *Optimization* 19.4, pp. 467–474. DOI: 10.1080/02331938808843365.

– (1994). "The Linear Complementarity Problem." In: *Advances in Optimization and Numerical Analysis*. Ed. by S. Gomez and J.-P. Hennart. Vol. 275. Dordrecht: Springer Netherlands, pp. 39–49. DOI: 10.1007/978-94-015-8330-5_3.

– (1996). "Continuous Approaches to Discrete Optimization Problems." In: *Nonlinear Optimization and Applications*. Ed. by G. Di Pillo and F. Giannessi. Boston, MA: Springer US, pp. 313–325. DOI: 10.1007/978-1-4899-0289-4_22.

Pardalos, P. M. and A. Nagurney (1990). "The integer linear complementarity problem." In: *International Journal of Computer Mathematics* 31.3-4, pp. 205–214. DOI: 10.1080/00207169008803803.

Pardalos, P. M. and J. Rosen (1988). "Global Optimization Approach to the Linear Complementarity Problem." In: *SIAM Journal on Scientific and Statistical Computing* 9.2, pp. 341–353. DOI: 10.1137/0909022.

Rinaldi, F. (2009). "New results on the equivalence between zero-one programming and continuous concave programming." In: *Optimization Letters* 3.3, pp. 377–386. DOI: 10.1007/s11590-009-0117-x.

Sumita, H., N. Kakimura, and K. Makino (2018). "Total dual integrality of the linear complementarity problem." In: *Annals of Operations Research* 274.1–2. DOI: 10.1007/s10479-018-2926-8.

Val, P. D. (1940). "The Unloading Problem for Plane Curves." In: *American Journal of Mathematics* 62.1, pp. 307–311. DOI: 10.2307/2371454.

Weinhold, R. and S. A. Gabriel (2020). "Discretely constrained mixed complementary problems: Application and analysis of a stylised electricity market." In: *Journal of the Operational Research Society* 71.2, pp. 237–249. DOI: 10.1080/01605682.2018.1561163.

Zhu, W. X. (2003). "Penalty parameter for linearly constrained 0–1 quadratic programming." In: *Journal of Optimization Theory and Applications* 116.1, pp. 229–239. DOI: 10.1023/A:1022174505886.

(M. de Santis) Sapienza Università di Roma, Department of Computer, Control, and Management Engineering, Via Ariosto 25, 00185 Roma, Italy
Email address: marianna.desantis@uniroma1.it

(S. de Vries, M. Schmidt, L. Winkel) Trier University, Department of Mathematics, Universitätsring 15, 54296 Trier, Germany
Email address: devries@uni-trier.de
Email address: martin.schmidt@uni-trier.de
Email address: winkell@uni-trier.de