Presolving Linear Bilevel Optimization Problems

Thomas Kleinert, Julian Manns, Martin Schmidt, Dieter Weninger

Abstract. Linear bilevel optimization problems are known to be strongly NP-hard and the computational techniques to solve these problems are often motivated by techniques from single-level mixed-integer optimization. Thus, during the last years and decades many branch-and-bound methods, cutting planes, or heuristics have been proposed. On the other hand, there is almost no literature on presolving linear bilevel problems although presolve is a very important ingredient in state-of-the-art mixed-integer optimization solvers. In this paper, we carry over standard presolve techniques from single-level optimization to bilevel problems and show that this needs to be done with great caution since a naive application of well-known techniques does often not lead to correctly presolved bilevel models. Our numerical study shows that presolve can also be very beneficial for bilevel problems but also highlights that these methods have a more heterogeneous effect on the solution process compared to what is known from single-level optimization. As a side result, our numerical experiments reveal that there is an urgent need for better and more heterogeneous test instance libraries to further propel the field of computational bilevel optimization.

1. Introduction

In the last years and decades, bilevel optimization has received increasing attention. In particular, the progress of computational mixed-integer bilevel optimization has been significant. This progress resembles the development in mixed-integer single-level optimization for two reasons. First, many algorithms for (mixed-integer) linear bilevel problems rely on highly-developed mixed-integer solvers as the working horse. Second, game-changing developments from mixed-integer optimization are also used in bilevel optimization such as tailored branch-and-bound methods [6, 13, 29], cutting planes [4, 15, 21, 34], and heuristics [13, 16, 23]—to name only a few references for each of the fields. These developments allow to tackle bilevel problems of significant size today that have been far out of reach one or two decades ago. Nevertheless, one main driver of the success of mixed-integer solvers has hardly seen any attention in the bilevel literature so far: *presolve*.

As usual, presolve for bilevel problems should be applied, in general, to reduce the size of the problem in terms of variables and constraints as well as to obtain a tighter formulation of the problem. For bilevel problems, however, it is particularly important to reduce the number of follower constraints since classic reformulations such as the one using the Karush–Kuhn–Tucker (KKT) conditions lead to harder problems if the number of follower constraints is large.

To the best of our knowledge, [14] is the only work that computationally analyzes a general-purpose presolve method for bilevel problems. The authors show that applying duality fixing in the lower level of a mixed-integer bilevel problem can be very effective. We will later numerically confirm this result for linear bilevel problems as well. One reason for the sparse literature on bilevel presolve might be the inherent nonconvex structure of bilevel problems. This structure yields some surprising properties that render presolving bilevel problems a very challenging task. In [25], it is discussed that bilevel

Date: July 14, 2021.

2010 *Mathematics Subject Classification.* 90-08, 90C11, 90C30, 90C26, 90C46.

Key words and phrases. Linear Bilevel Optimization, Presolve, Computational Analysis.

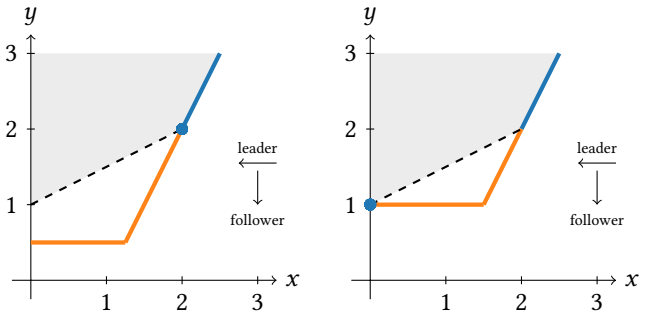

FIGURE 1. Feasible set and optimal solution without (left) and with (right) bound tightening applied. The dashed black line corresponds to the upper-level coupling constraint, the orange lines correspond to the lower-level constraints, the gray area is the joint feasible set w.r.t. the upper- and lower-level constraints, and the blue lines denote the bilevel feasible set with the dot marking the optimal solution.

problems do not posses the independence of irrelevant constraints (IIC) property known from single-level optimization. Essentially, the consequence is that adding inequalities to the lower-level problem of a bilevel problem that are inactive at the optimal bilevel solution may change the bilevel feasible set and, ultimately, the optimal solution as well. In [12], it is shown that under certain assumptions on the bilevel problem, a global solution remains locally optimal when adding irrelevant constraints. The two results render cutting-plane algorithms, which subsequently develop the lower-level problem, infeasible. This is in contrast to single-level optimization, where a solution of a relaxed problem (in which, e.g., some constraints are left out) that is also feasible for the omitted constraints is the global optimum. Additionally, the results also indicate that applying common mixed-integer presolve techniques such as, e.g., bound strengthening, to bilevel problems may not be without obstacles. To shed some light on this, let us consider the linear bilevel problem

$$\min_{x,y\in\mathbb{R}} \quad x \quad \text{s.t.} \quad y \geq 0.5x + 1,\ x \geq 0,\ y \in \arg\min_{\bar{y}\in\mathbb{R}} \{\bar{y}\colon \bar{y} \geq 2x - 2,\ \bar{y} \geq 0.5\},$$

with optimal solution $(2, 2)$; see Figure 1 (left). When strengthening the bound $\bar{y} \geq 0.5$ in the lower-level problem using the constraint $y \geq 0.5x + 1$ of the upper-level problem, one finds that the minimum value of $0.5x + 1$ is 1 due to $x \geq 0$, which increases the bound of $\bar{y}$ to $\bar{y} \geq 1$. This yields the problem

$$\min_{x,y\in\mathbb{R}} \quad x \quad \text{s.t.} \quad y \geq 0.5x + 1,\ x \geq 0,\ y \in \arg\min_{\bar{y}\in\mathbb{R}} \{\bar{y}\colon \bar{y} \geq 2x - 2,\ \bar{y} \geq 1\},$$

having the optimal solution $(0, 1) \neq (2, 2)$; see Figure 1 (right). See also the thesis [26] for further examples. It can thus not be expected that presolve techniques known from single-level mixed-integer linear optimization can be applied directly to bilevel problems.

In this paper we formally and computationally analyze presolve methods for the easiest variant of bilevel problems, i.e., problems with a linear upper and lower level. To this end, we mainly carry over classic presolve techniques from mixed-integer optimization to the bilevel setting, which has—to the best of our knowledge—not been done before in the literature with the only exception being a single technique discussed in [14]. We introduce the relevant notation and theory in Section 2. We then introduce several presolve methods for linear bilevel problems in Section 3 by carrying over classic presolve ideas from (single-level) linear and mixed-integer optimization to the field of bilevel problems. Afterward, we evaluate these methods in a computational study in Section 4. These numerical experiments reveal two main insights. First, we observe that presolving linear

bilevel problems can be very beneficial for some instances whereas it is rather harming for the solution process of other instances. Second, although we have a test set of more than 2500 instances, only very few of them are affected by some of the presolve methods discussed in this paper. We conclude, thus, that there is an urgent need for more realistic and more heterogeneous test instance libraries to further propel the field of computational bilevel optimization; see Section 5.

2. Notation and Theoretical Background

In this paper, we consider linear bilevel problems of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n, y\in\mathbb{R}^m} \quad & c^\top x + d^\top y \\ \text{s.t.} \quad & Ax + By \geq a, \\ & \underline{x}^L \leq x \leq \bar{x}^L,\ \underline{y}^L \leq y \leq \bar{y}^L, \\ & y \in \mathcal{S}(x), \end{aligned} \tag{1}$$

where $\mathcal{S}(x)$ denotes the set of optimal solutions or the rational reaction set of the so-called follower (or lower-level) problem

$$\min_{y\in\mathbb{R}} \quad f^\top y \quad \text{s.t.} \quad Dy \geq b - Cx,\ \underline{y}^F \leq y \leq \bar{y}^F, \tag{2}$$

with $c \in \mathbb{R}^n$, $d, f \in \mathbb{R}^m$, $A \in \mathbb{R}^{k\times n}$, $B \in \mathbb{R}^{k\times m}$, $a \in \mathbb{R}^k$, $C \in \mathbb{R}^{\ell\times n}$, $D \in \mathbb{R}^{\ell\times m}$, and $b \in \mathbb{R}^\ell$. Further, we denote variable bounds by $\underline{x}^L, \bar{x}^L \in \mathbb{R}^n$ and $\underline{y}^L, \bar{y}^L \in \mathbb{R}^m$ as well as $\underline{y}^F, \bar{y}^F \in \mathbb{R}^m$. Note that we explicitly distinguish leader and follower bounds on the follower variables y. We exploit these bounds later in various presolve techniques.

The formulation (1) establishes the optimistic (or cooperative) solution, i.e., in case the set of optimal follower solutions $\mathcal{S}(x)$ is not a singleton, the reaction $y \in \mathcal{S}(x)$ is chosen in favor of the objective function of the so-called leader (or upper-level) problem (1); see, e.g., [11]. We denote the shared constraint set and its projection onto the decision space of the leader by

$$\Omega := \left\{(x,y)\colon Ax + By \geq a,\ Cx + Dy \geq b,\ \underline{x}^L \leq x \leq \bar{x}^L,\ \underline{y}^L \leq y \leq \bar{y}^L,\ \underline{y}^F \leq y \leq \bar{y}^F\right\}$$

and

$$\Omega_L := \{x\colon \exists y \text{ with } (x,y) \in \Omega\},$$

respectively. Finally, the bilevel feasible set, often also called the inducible region, is denoted by

$$\mathcal{F} := \left\{(x,y)\colon Ax + By \geq a,\ \underline{x}^L \leq x \leq \bar{x}^L,\ \underline{y}^L \leq y \leq \bar{y}^L,\ y \in \mathcal{S}(x)\right\}.$$

It is well known that the set $\{(x,y)\colon x \in \mathbb{R}^n,\ y \in \mathcal{S}(x)\}$ is nonconvex as well as that the bilevel feasible set $\mathcal{F}$ is nonconvex and, in general, disconnected; see, e.g., [5] or [11]. This is also illustrated in Figure 1 (right). Consequently, even linear bilevel problems are intrinsically nonconvex and nonsmooth. It is shown in [20] that linear bilevel problems are strongly NP-hard and, in [35], that even checking local optimality of a given point is NP-hard.

These unpleasant properties of linear bilevel problems are well known and it is thus clear that one faces different challenging aspects when solving bilevel problems. However, it is still surprising that applying classic presolve techniques like, e.g., bound tightening, fails for bilevel problems; see Figure 1. We highlight that it is in particular invalid to exploit the so-called high-point relaxation

$$\min_{x,y} \quad c^\top x + d^\top y \quad \text{s.t.} \quad (x,y) \in \Omega \tag{3}$$

to presolve the original bilevel problem (1). As briefly discussed in the introduction, this is related to the independence of irrelevant constraints property. For single-level

problems, this property states that adding any constraint that is satisfied at the global optimal solution does not change the global optimum. In [25], the IIC property for linear bilevel problems of the form (1) is defined as follows.

Definition 1. *Let Ω^* be the set of optimal solutions of a linear bilevel problem P of the form (1). Further, let $\tilde{P} := P(u, v, w)$ be the modified problem, in which the inequality $u^\top x + v^\top y \geq w$ is added to the follower problem of P and let $\tilde{\Omega}^*$ be its set of optimal solutions. P is called* independent of irrelevant constraints, *if for any $(u, v, w) \in \mathbb{R}^{n+m+1}$ with $u^\top x^* + v^\top y^* \geq w$ it holds*

$$(x^*, y^*) \in \tilde{\Omega}^*$$

for every $(x^*, y^*) \in \Omega^*$.

It is shown in [25] that bilevel problems can only possess the IIC property if the solution of the high-point relaxation (3) is also a solution of the bilevel problem (1). Consequently, most practical bilevel problems for which the objectives of the leader and the follower are not aligned, lack the IIC property.

On the other hand, Definition 1 of the IIC property for bilevel problems is not entirely in line with the IIC property known for single-level problems. In order to see this, we consider the equivalent single-level reformulation

$$\min_{x,y} \quad c^\top x + d^\top y \tag{4a}$$

$$\text{s.t.} \quad (x, y) \in \Omega, \tag{4b}$$

$$f^\top y \leq \varphi(x), \tag{4c}$$

of Problem (1), which makes use of the optimal-value function

$$\varphi(x) := \min_{y \in \mathbb{R}^m} \left\{ f^\top y \colon Dy \geq b - Cx,\ \underline{y}^F \leq y \leq \bar{y}^F \right\}$$

of the follower problem. We denote the optimal solution of this problem by (x^*, y^*). It is clear that any inequality valid for (x^*, y^*) can safely be added to Problem (4), i.e., the single-level problem (4) possesses the (single-level) IIC property. Adding such a constraint to Problem (4) however corresponds to adding an irrelevant constraint to the leader problem of the bilevel problem (1). On the contrary, adding a constraint that is valid for the bilevel optimal solution to the follower problem of (1) *may* change the rational reaction set $\mathcal{S}(x)$, respectively, the optimal-value function $\varphi(x)$. In the point of view of Problem (4), adding such a constraint to the follower problem thus not only adds this constraint to the set Ω but *may* also change the right-hand side of the existing Constraint (4c). In this case, the concept of the (single-level) IIC property is not applicable. Detecting such constraints that do not affect Constraint (4c) is a delicate task and will play a crucial role in the presolve of linear bilevel problems.

In general, the goal of presolving optimization problems is to transform the given problem into an easier-to-solve problem that has the following properties:

(i) It is infeasible (unbounded) if and only if the original problem is infeasible (unbounded).
(ii) Every feasible solution of the presolved problem can be transformed into a feasible solution of the original problem.
(iii) The objective value of every optimal solution of the presolved problem matches (at least after possibly required post-processing steps) the optimal objective value of the original problem.

This gives rise to two different presolve strategies. First, we can modify Problem (1) in a way that leaves the bilevel feasible set $\mathcal{F}$ unchanged, i.e., we can apply a feasibility-based presolve; see, e.g., [7, 8] for single-level problems. Second, we can apply an optimality-based presolve by modifying the bilevel feasible set $\mathcal{F}$ in a way to ensure that at least one

bilevel optimal solution remains feasible for the reduced problem; see, e.g., [32] for single-level problems or the removal of dominated columns in mixed-integer programming [18]. In the following section, we introduce several such presolve methods. We also evaluate these methods in a computational study in Section 4.

3. Presolve Methods

In this section, we present presolve methods for linear bilevel problems.

3.1. Bound Strengthening.

A well-known presolve technique for single-level optimization is bound strengthening [2, 33]. This approach tries to strengthen variable bounds by domain propagation without changing the set of feasible solutions.

We now consider bound strengthening for bilevel problems of Type (1). Leader bounds $\underline{x}_i^L$ and $\bar{x}_i^L$ can be tightened by exploiting the shared constraint set Ω and, thus, by solving

$$\min\{x_i : (x,y) \in \Omega\} \quad \text{and} \quad \max\{x_i : (x,y) \in \Omega\}, \tag{5}$$

respectively, for $i = 1, \dots, n$. Tight follower bounds $\underline{y}^L$ and $\bar{y}^L$ located in the leader problem can be derived analogously. We emphasize that this computation of tighter bounds is computationally costly. For more details on this approach applied to single-level problems we refer to [19]. In practice, there exist different approaches to compute tight bounds efficiently for single-level optimization problems and, usually, there is a trade-off between the tightness of the derived bounds and computation times. In the simplest case, one considers each constraint separately and tries to derive tighter bounds for the variables occurring in it, which is doable in linear time; see [2, 27]. Approaches that consider more than one constraint simultaneously, see, e.g., [1, 10], usually yield better bounds than approaches that consider only one constraint. However, an increased running time is to be expected. Naturally, this procedure can be continued until all constraints are considered, which results in the best possible bounds. Obviously, the computational effort involved is also the largest. In this work, we focus on the analysis of the effectiveness of bound strengthening by using the best available bounds and, thus, we refrain from using other methods as to solve (5) to compute these bounds.

The strengthening of follower bounds $\underline{y}^F$ and $\bar{y}^F$ located in the follower problem is not that straightforward. We already illustrated in the introduction that a naive strengthening of follower bounds that exploits the shared constraint set Ω may result in wrong "solutions" of Problem (1). The reason is that the shared constraint set also contains the leader constraints. This suggests to tighten follower bounds by disregarding the leader constraints, i.e., by solving

$$\min_{x,y} \quad y_i \quad \text{s.t.} \quad Cx + Dy \geq b,\ \underline{y}^F \leq y \leq \bar{y}^F \tag{6}$$

and

$$\max_{x,y} \quad y_i \quad \text{s.t.} \quad Cx + Dy \geq b,\ \underline{y}^F \leq y \leq \bar{y}^F, \tag{7}$$

for $i = 1, \dots, m$. These problems can be further tightened by considering follower-independent leader constraints that only depend on leader variables x, i.e., leader constraints with $B_{i\cdot} = 0$ or leader bounds $\underline{x}^L \leq x \leq \bar{x}^L$. We show in the following theorem that follower-independent leader constraints can be moved to the follower problem without changing the bilevel feasible set $\mathcal{F}$.

Theorem 1. *Consider a leader constraint $u^\top x \geq w$ with $u \in \mathbb{R}^n$ and $w \in \mathbb{R}$ of the bilevel problem (1). Further, consider the bilevel problem obtained from Problem (1) by moving the constraint $u^\top x \geq w$ to the follower problem. We denote the feasible set of the latter problem by $\mathcal{F}^M$. Then, $\mathcal{F} = \mathcal{F}^M$ holds.*

Proof. The shared constraint set for both problems is given by

$$\tilde{\Omega} = \Omega \cap \{(x,y) : u^\top x \geq w\}.$$

Further, we denote the rational reaction set of Problem (1), for which the constraint is moved to the follower, by $\mathcal{S}^M$. We then have

$$\mathcal{F}^M = \tilde{\Omega} \cap \{(x,y)\colon x \in \mathbb{R}^n, y \in \mathcal{S}^M(x)\}.$$

Thus, for any point (x,y) in $\mathcal{F}^M$ it holds $u^\top x \geq w$. Further, for $x \in \mathbb{R}^n$ it holds

$$\mathcal{S}^M(x) = \begin{cases} \mathcal{S}(x), & \text{if } u^\top x \geq w, \\ \emptyset, & \text{if } u^\top x < w. \end{cases}$$

Consequently,

$$\mathcal{F}^M = \tilde{\Omega} \cap \{(x,y)\colon x \in \mathbb{R}^n, y \in \mathcal{S}^M(x)\} = \tilde{\Omega} \cap \{(x,y)\colon x \in \mathbb{R}^n, y \in \mathcal{S}(x)\} = \mathcal{F}. \qquad \square$$

This theorem can be exploited to tighten the Problems (6) and (7).

Observation 1. *We can add all follower-independent leader constraints to Problem (6) and (7). In particular, this includes the bounds $\underline{x}^L \leq x \leq \bar{x}^L$ on the leader variables.*

Let us also note that we can handle tightened bounds also slightly different. In practice, linear bilevel problems (1) are mostly solved by a single-level reformulation that replaces the follower problem with its KKT conditions or with the strong-duality condition. In this view, it makes sense to move follower bounds to the leader instead of tightening them. This results in a smaller single-level reformulation. For instance, in the case that the KKT reformulation is used, moving bounds to the leader results in fewer KKT complementarity conditions. However, it is well known that moving constraints between the leader and the follower problem is not without obstacles and, in general, it may change the optimal solution. Nevertheless, we make the following observation.

Observation 2. *Consider a follower variable y_i with a lower bound $\underline{y}_i^F$ that can be tightened, i.e., the objective value $\underline{y}_i^*$ of Problem (6) is strictly larger than $\underline{y}_i^F$. Instead of tightening the bound, we can move the original bound $\underline{y}_i^F$ from the follower to the leader problem. Since the bound is implied by the follower constraints anyway, this does not change the rational reaction set $\mathcal{S}(x)$ of the follower. The same holds for upper bounds.*

Note that this is in contrast two general lower-level constraints (i.e., constraints that are more involved than simple bounds for the lower-level variables), which can be added to the leader without problems but not moved from the follower to the leader without altering the bilevel-feasible set $\mathcal{F}$. We finally note that one can also simply remove the (anyway redundant) bound $\underline{y}_i^F$ from the follower problem without adding it to the leader problem, or one can add the updated bound $\underline{y}_i^*$ to the leader problem. From single-level optimization it is known to be unclear, which strategy "helps" the solver and which one is rather harmful to the solution process. Thus, these different strategies need to be evaluated numerically.

3.2. **Parallel Columns.** We first briefly explain the presolve of parallel columns in the single-level context; see also, e.g., [2, 3]. In [2] it is both shown that parallel columns are common in public and commercial single-level problems and that their treatment helps for solving the instances more efficiently. To this end, consider the single-level follower problem (2) for a fixed leader decision $x = \bar{x}$ as well as two follower variables y_i and y_j for some $i \neq j$, $i,j \in \{1,\dots,m\}$. The two columns $D_{\cdot i}$ and $D_{\cdot j}$ are called parallel if $D_{\cdot i} = \mu D_{\cdot j}$ holds for some $\mu \neq 0$. An algorithm for determining parallel columns with runtime $O(\sum_i l_i \log l_i)$, where l_i represents the number of non-zeros in column $D_{\cdot i}$, is given in [9].

If, in addition, $f_i = \mu f_j$ holds, we can merge, or "crush", the two variables y_i and y_j into a new variable y_{new} by setting

$$y_{\text{new}} := y_j + \mu y_i. \tag{8}$$

Consequently, we obtain a presolved version of Problem (2) by removing the two original variables y_i and y_j and by instead adding the variable y_{new} with

$$f_{\text{new}} = f_j, \quad D_{\cdot\text{new}} = D_{\cdot j}$$

as well as bounds

$$\underline{y}^F_{\text{new}} = \begin{cases} \underline{y}^F_j + \mu \underline{y}^F_i & \text{for } \mu > 0, \\ \underline{y}^F_j + \mu \bar{y}^F_i & \text{for } \mu < 0, \end{cases} \quad \bar{y}^F_{\text{new}} = \begin{cases} \bar{y}^F_j + \mu \bar{y}^F_i & \text{for } \mu > 0, \\ \bar{y}^F_j + \mu \underline{y}^F_i & \text{for } \mu < 0. \end{cases} \tag{9}$$

Now, let y^* be an optimal solution of the presolved problem. Then, we can post-process, or "uncrush", the value y^*_{new} into values y^*_i and y^*_j by using Equation (8) and by paying attention to the bounds (9). It holds

$$D_{\text{new}} y^*_{\text{new}} = D_{\cdot j}(y^*_j + \mu y^*_i) = D_{\cdot j} y^*_j + \mu D_{\cdot j} y^*_i = D_{\cdot j} y^*_j + D_{\cdot i} y^*_i$$

so that the uncrushed solution is feasible for the original problem (2) if and only if y^* is feasible for the presolved problem. In addition, we have

$$f_{\text{new}} y^*_{\text{new}} = f_j(y^*_j + \mu y^*_i) = f_j y^*_j + \mu f_j y^*_i = f_j y^*_j + f_i y^*_i.$$

Thus, the uncrushed solution is optimal for Problem (2) if and only if y^*_{new} is optimal for the presolved problem.

From now on, we consider the bilevel setting again. A straightforward idea is to apply the approach stated above only to the parametric follower problem (2). In other words, we have two follower variables y_i and y_j for some $i \neq j$, $i, j \in \{1, \dots, m\}$, with parallel columns in the follower problem. However, the corresponding leader columns need not necessarily be parallel, i.e., we might have $B_{\cdot i} \neq \mu B_{\cdot j}$. In this setting, we can crush the two variables in the follower problem according to (8). This removes the two variables y_i and y_j from the follower problem such that they only appear in the leader problem. In order to still respect the (crushed) optimal follower solution in the leader problem, we add the constraint $y_{\text{new}} = y_j + \mu y_i$ to the leader problem. This leaves the bilevel feasible set, after uncrushing, unchanged.

A similar approach can be applied if two follower columns are parallel in both the leader and the follower problem, i.e.,

$$d_i = \mu d_j, \quad f_i = \mu f_j, \quad B_{\cdot i} = \mu B_{\cdot j}, \quad D_{\cdot i} = \mu D_{\cdot j}$$

for $i \neq j$, $i, j \in \{1, \dots, m\}$, and some $\mu \neq 0$. We can now crush y_i and y_j into y_{new}. For the follower, we proceed according to (8) and (9), For the leader we apply

$$d_{\text{new}} = d_j, \quad B_{\cdot\text{new}} = B_{\cdot j}$$

and

$$\underline{y}^L_{\text{new}} = \begin{cases} \underline{y}^L_j + \mu \underline{y}^L_i & \text{for } \mu > 0, \\ \underline{y}^L_j + \mu \bar{y}^L_i & \text{for } \mu < 0, \end{cases} \quad \bar{y}^L_{\text{new}} = \begin{cases} \bar{y}^L_j + \mu \bar{y}^L_i & \text{for } \mu > 0, \\ \bar{y}^L_j + \mu \underline{y}^L_i & \text{for } \mu < 0. \end{cases}$$

As above, we argue that an uncrushed solution is feasible for the follower problem (2) if and only if the crushed solution is feasible. Also, following the discussion for the single-level case above, the crushed and uncrushed solutions yield the same objective value for the follower problem. Furthermore, with the same arguments as for the follower-only case, the uncrushed solution is feasible for the leader if and only if the crushed solution is feasible for the leader. Consequently, the uncrushed solution is bilevel feasible if and only if the crushed solution is bilevel feasible. Finally, the crushed and uncrushed solution have the same leader objective value, which means that the uncrushed solution is bilevel optimal if and only if the crushed solution is bilevel optimal.

We now consider two parallel leader columns in Problem (1), i.e.,

$$c_i = \mu c_j, \quad A_{\cdot i} = \mu A_{\cdot j}, \quad C_{\cdot i} = \mu C_{\cdot j}$$

for $i \neq j$, $i, j \in \{1, \dots, n\}$, and some $\mu \neq 0$. We can crush x_i and x_j into x_{new} across all leader and follower constraints and the two objectives, similarly to the parallel follower

TABLE 1. Implications of parallel rows depending on μ and the right-hand sides $\tilde{a}_q$ and $\tilde{a}_r$.

	$\tilde{a}_q < \mu\tilde{a}_r$	$\tilde{a}_q = \mu\tilde{a}_r$	$\tilde{a}_q > \mu\tilde{a}_r$
$\mu > 0$	inequality with index q is irrelevant	both inequalities are equivalent	inequality with index r is irrelevant
$\mu < 0$	both inequalities are relevant		infeasible model

case. Now, let x^* be part of an optimal solution of the presolved, i.e., crushed, problem. We uncrush this solution by setting $x^*_{\text{new}} = x^*_j + \mu x^*_i$. This does not affect the right-hand sides $b - Cx$ of the follower constraints. In addition, the uncrushed solution is feasible for the leader constraints if and only if the crushed solution is feasible. Thus, the uncrushed solution is bilevel feasible if and only if the crushed solution is bilevel feasible for presolved bilevel problem.

Up to now, we only considered the case in which either both columns are leader columns or both columns are follower columns. Unfortunately, we cannot presolve a leader column that is parallel to a follower column, as the following example shows.

Example 1. *Consider the linear bilevel problem*

$$\min_{x\in\mathbb{R}, y\in\mathbb{R}^2} \quad x + y_1 \quad \text{s.t.} \quad 0 \le x \le 1,\ y \in \arg\max_{\bar{y}\in\mathbb{R}^2} \{\bar{y}_2 \colon \bar{y}_2 \le x + \bar{y}_1,\ 0 \le \bar{y}_1 \le 1\},$$

which has the unique optimal solution $x^* = 0$, $y^*_1 = 1$, and $y^*_2 = 1$. Note that the columns of x and y_1 are parallel with $\mu = 1$. If we crush them into a leader variable, we obtain

$$\min_{x_{\text{new}}, y_2\in\mathbb{R}} \quad x_{\text{new}} \quad \text{s.t.} \quad 0 \le x_{\text{new}} \le 2,\ y_2 \in \arg\max_{\bar{y}_2\in\mathbb{R}} \{\bar{y}_2 \colon \bar{y}_2 \le x_{\text{new}}\},$$

which has the unique optimal solution $x^*_{\text{new}} = 0$ and $y^*_2 = 0$. By uncrushing $x^*_{\text{new}} = x^* + y_1$, we obtain $x^* = 0$ and $y_1 = 0$. Contrary, if we crush the two variables into a follower variable, we obtain

$$\min_{y_{\text{new}}, y_2\in\mathbb{R}} \quad y_{\text{new}} \quad \text{s.t.} \quad (y_{\text{new}}, y_2) \in \arg\max_{\bar{y}_{\text{new}}, \bar{y}_2\in\mathbb{R}} \{\bar{y}_2 \colon \bar{y}_2 \le \bar{y}_{\text{new}},\ 0 \le \bar{y}_{\text{new}} \le 2\},$$

which has the unique optimal solution $y^*_{\text{new}} = 2$ and $y^*_2 = 2$. Uncrushing yields $x^* = 1$ and $y^*_1 = 1$.

3.3. **Parallel Rows.** This presolve technique is also known from single-level optimization. We briefly explain it based on the high-point relaxation (3) and use the notation

$$\tilde{A} = \begin{bmatrix} A \\ C \end{bmatrix}, \quad \tilde{B} = \begin{bmatrix} B \\ D \end{bmatrix}, \quad \tilde{a} = \begin{pmatrix} a \\ b \end{pmatrix}.$$

We call two inequalities with indices $q, r \in \{1, \dots, k + \ell\}$ parallel, if $\tilde{A}_{q\cdot} = \mu\tilde{A}_{r\cdot}$ and $\tilde{B}_{q\cdot} = \mu\tilde{B}_{r\cdot}$ hold for $\mu \neq 0$. The sign of μ and the right-hand sides $\tilde{a}_q$ and $\tilde{a}_r$ can be used to determine infeasibility of the entire problem or that one of the two constraints is redundant. More precisely, we distinguish the cases shown in Table 1. Detailed descriptions of this procedure for single-level optimization are given in [2, 3, 27]. Although the occurrence of parallel rows can be seen as bad modeling, it is shown in [2] that parallel rows occur rather frequently in real-world single-level instances and that their treatment contributes to a more efficient solution process for these instances. Note that the same algorithm as in [9] to detect parallel columns can also be used to detect parallel rows.

We now turn to the linear bilevel problem (1). The two inequalities can now either be two leader inequalities, two follower inequalities, or one leader and one follower inequality. In the first two cases, we can directly apply the rules given in Table 1. The third case requires a more detailed discussion. We assume w.l.o.g. that q is an index of a leader

inequality and r is an index of a follower inequality. In case $\mu < 0$, the rules in Table 1 still apply, because for $\tilde{a}_q > \mu\tilde{a}_r$ the model is infeasible and for $\tilde{a}_q \leq \mu\tilde{a}_r$ no deduction can be made. However, in case $\mu > 0$, we obtain different rules:

(i) If $\tilde{a}_q < \mu\tilde{a}_r$, then $\tilde{A}_q x + \tilde{B}_q y \geq \tilde{a}_q$ is implied by $y \in \mathcal{S}(x)$ and the leader inequality with index q is redundant.
(ii) If $\tilde{a}_q > \mu\tilde{a}_r$, then the follower constraint may only be active for a point (x, y) that violates the leader constraint. Thus, removing this constraint only changes the rational reaction $y \in \mathcal{S}(x)$ for leader decisions $x \notin \Omega_L$. Consequently, the follower inequality with index r must be inactive for every bilevel feasible point and we can safely remove it.
(iii) If $\tilde{a}_q = \mu\tilde{a}_r$, then the inequalities are equivalent. For the follower inequality with index r, the argument of Case (ii) does not hold, because it can be active at a bilevel feasible point. Thus, removing the follower constraint might change the response of the follower in a way that renders the follower response infeasible for the leader constraints. However, the leader inequality with index q is redundant and we can remove it.

In summary, this means that we can detect parallel rows using the shared constraint set Ω as long as we pay attention to Case (iii).

Until now, we only covered parallel inequalities. Equality constraints can, in principle, be reformulated as two inequalities, such that we may apply the techniques above. From a practical point of view, however, it is often not a good idea to divide an equation into two inequalities. Keeping the equation results in further cases that have to be distinguished; see, e.g., [2].

3.4. **Duality Fixing.** This well-known single-level presolve method tries to detect variables that can be fixed to certain values based on optimality conditions. Duality fixing is often very efficient in practice and can be applied to both continuous and integer variables. Details on this method are given in [2, 27].

In the following, we briefly describe duality fixing for single-level problems by considering the follower problem (2) for a fixed leader decision $x = \bar{x}$. We now consider variables y_i that fulfill one of the two conditions:

(i) $f_i \geq 0$ and $D_{ji} \leq 0$ for all $j \in \{1, \dots, \ell\}$,
(ii) $f_i \leq 0$ and $D_{ji} \geq 0$ for all $j \in \{1, \dots, \ell\}$.

Assume that the index i fulfills Condition (i) and that $\underline{y}_i^F > -\infty$ holds. Then we can fix $y_i = \underline{y}_i^F$. For $f_i > 0$, this variable fixing is given directly as a necessary optimality condition. For $f_i = 0$, there exists at least one optimal solution that satisfies this variable fixing. If we assume $\underline{y}_i^F = -\infty$ instead of $\underline{y}_i^F > -\infty$, we have to distinguish two cases. For $f_i > 0$, the problem is infeasible or unbounded and for $f_i = 0$, the variable y_i and all constraints, for which y_i has a nonzero coefficient, can be removed from the problem. Condition (ii) can be tackled analogously. If $\bar{y}_i^F < \infty$, we fix $y_i = \bar{y}_i^F$. For $\bar{y}_i^F = \infty$ we distinguish the same cases as for Condition (i).

As shown in [14], this single-level duality fixing can be applied almost directly to the follower problem (2) without taking into account the leader decision x.

Theorem 2 ([14]). *For every follower variable y_i, $i \in \{1, \dots, m\}$, the following fixing is correct:*

(i) *If $f_i > 0$ and $D_{ji} \leq 0$ for all $j \in \{1, \dots, \ell\}$, fix $y_i = \underline{y}_i^F$,*
(ii) *If $f_i < 0$ and $D_{ji} \geq 0$ for all $j \in \{1, \dots, \ell\}$, fix $y_i = \bar{y}_i^F$.*

Note that in Theorem 2 the case $f_i = 0$ is not taken into account. The reason might be that this case is somehow problematic. For $f_i = 0$, the optimal choice of y_i by the follower might be ambiguous. Under the optimistic assumption, the leader is free to choose among the optimal follower solutions the one that she prefers. Thus, fixing y_i must preserve this

optimistic solution. Since the case $f_i = 0$ does hardly appear in our instance set that we use in the computational study in Section 4, we do not further investigate this issue.

Until now, we have applied duality fixing to the follower problem. We now present an example that illustrates the occurring difficulties in the application of duality fixing to the leader problem or to the leader and follower problem simultaneously.

Example 2. *Consider the bilevel problem*

$$\min_{x,y} \quad x - y \quad \text{s.t.} \quad -3x + y \geq -3,\ x \geq 0,\ y \in \arg\min_{\bar{y}} \{\bar{y} \colon -2x + \bar{y} \geq 0\}. \tag{10}$$

For all leader and follower constraints, the coefficient of the leader variable x is negative. In addition, the coefficient in the upper-level objective function is positive. If we naively apply duality fixing as we would do in single-level optimization, we would fix x to its lower bound, i.e., $x = 0$. This yields the point $(0, 0)$ with objective function value 0.

However, for every leader decision x, the optimal solution of the follower is given by $y = 2x$. Thus, we can substitute y with $2x$ in the leader problem of Problem (10) to obtain

$$\min_{x} \quad -x \quad \text{s.t.} \quad -x \geq -3,\ x \geq 0.$$

This reformulated problem has the optimal solution $x = 3$, such that we obtain the optimal solution $(3, 6)$ of Problem (10) with objective function value -3. Hence, the point $(0, 0)$ we obtained by fixing $x = 0$ is not an optimal solution of Problem (10).

The reason for the observation in the example is that we omit the optimality of the follower in the fixing step. In other words, we applied duality fixing to the high-point relaxation (3), which is not sufficient to guarantee correctness.

3.5. **Optimality-Based Presolve.** Up to now, we presented feasibility-based presolve methods that do not change the bilevel-feasible set $\mathcal{F}$. In order to further simplify the solution of Problem (1), it might be desirable to tighten $\mathcal{F}$ in a way that retains at least one bilevel optimal solution. In particular, it would be beneficial to exclude bilevel-feasible points that can be proven to be not bilevel optimal. In general, this requires to respect bilevel optimality, i.e., to deal with the nonconvexity of the problem. One the one hand, optimality-based presolve can thus be expected to be a very difficult task. On the other hand, one might be able to identify non-optimal feasible points without knowledge of the bilevel-optimal solution. In this section, we present a first step in this direction.

In Section 3.1, we discussed to move constraints to tighten the linear problems used for the bound strengthening. To be specific, we moved follower-independent leader constraints to the follower problem. This is a feasible approach because in this special case, the bilevel feasible set $\mathcal{F}$ remains unchanged. However, it is well-known that moving constraints between the two levels changes the bilevel feasible set in general. In the following paragraphs we propose a criterion that can be used to detect follower constraints that can be safely moved to the leader problem without changing the set of bilevel-optimal points—although the bilevel-feasible set $\mathcal{F}$ might be changed. We illustrate this approach in Figure 2, in which upper-level constraints correspond to dashed lines and lower-level constraints to solid lines. The specific problem can be found on Page 33 of [26]. In addition, the shared constraint set Ω is colored gray, the bilevel feasible set is colored in blue, and the set of optimal follower solutions lifted to the x-y-space is the union of the blue and orange lines. Figure 2 (left) shows the original problem, in which the bilevel solution is attained at the point $(0, 1)$. We see that the lower-level constraints c_1, c_2, and c_3 are not active in $(0, 1)$. In Figure 2 (right), we moved these constraints to the leader. This changes the optimal reaction of the follower and, thus, also changes the bilevel feasible set. However, the bilevel optimal solution $(0, 1)$ is retained. We formalize this in the following theorem.

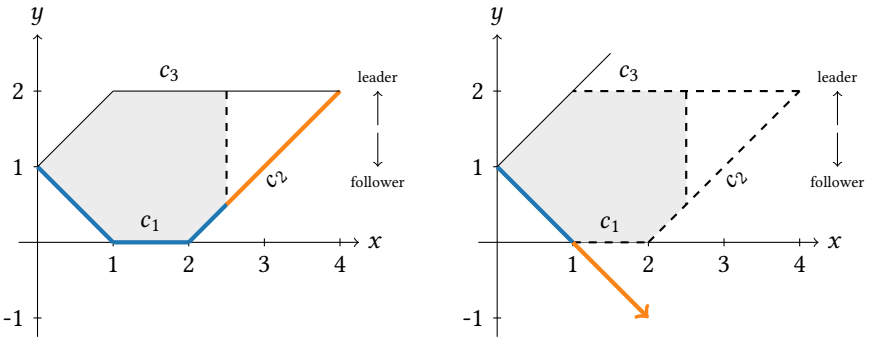

FIGURE 2. Moving follower constraints c_1, c_2, and c_3 (left) that are inactive at the bilevel optimal solution $(0, 1)$ to the leader (right) changes the reaction of the follower (blue and orange) and the bilevel feasible set (blue), but retains the optimal solution.

Theorem 3. *Let (x^*, y^*) be an optimal solution of Problem (1) and consider a follower constraint with index $i \in \{1, \dots, \ell\}$ such that $C_i.x^* + D_i.y^* > b_i$ holds. Then, (x^*, y^*) is also optimal for the problem that we obtain by moving $C_i.x + D_i.y \geq b_i$ from the follower to the leader.*

Proof. We denote the bilevel feasible set of Problem (1) by $\mathcal{F}$ and the bilevel feasible set after moving the constraint by $\mathcal{F}_M$. We prove the theorem by showing that

(i) the optimal solution (x^*, y^*) remains bilevel feasible when moving the constraint, i.e., $(x^*, y^*) \in \mathcal{F}_M$, and
(ii) we do not obtain additional bilevel feasible points by moving the constraint, i.e., $(x, y) \in \mathcal{F}_M$ implies $(x, y) \in \mathcal{F}$.

We first show (i). Since $C_i.x^* + D_i.y^* > b_i$, the reaction y^* of the follower must still be optimal for the leader decision x^* after removing the constraint from the follower problem. In addition, the point (x^*, y^*) is feasible for all original leader constraints as well as for the constraint $C_i.x + D_i.y \geq b_i$. Thus, $(x^*, y^*) \in \mathcal{F}_M$ holds.

We now turn to (ii). Let $(\bar{x}, \bar{y}) \in \mathcal{F}_M$, i.e., $\bar{y}$ is optimal for the relaxed follower problem (2) for fixed $x = \bar{x}$, in which the ith follower constraint is removed. Since $(\bar{x}, \bar{y}) \in \mathcal{F}_M$, $C_i.\bar{x} + D_i.\bar{y} \geq b_i$ holds and $\bar{y}$ is also optimal for Problem (2) with fixed $x = \bar{x}$. Thus, $(\bar{x}, \bar{y}) \in \mathcal{F}$. □

In order to turn Theorem 3 into a presolve method, it is crucial to detect such inactive constraints. One approach might be to exploit the high-point relaxation in special cases. For example, in min-max problems the objective functions of the leader and the follower point into opposite directions. Follower constraints that are binding for the solution of the high-point relaxation are canonical candidates to be inactive at the bilevel-optimal solution—although, of course, easy examples can be constructed for which this does not hold. It remains an open question and subject of future research whether this technique can be carried out efficiently.

4. Computational Experiments

In this section, we try to shed some light on the effectiveness of the proposed presolve techniques. To this end, we consider the following single-level reformulation

$$\min_{x,y,\underline{\lambda},\bar{\lambda},\lambda} \quad c^\top x + d^\top y \tag{11a}$$
$$\text{s.t.} \quad (x,y) \in \Omega, \tag{11b}$$
$$\lambda, \underline{\lambda}, \bar{\lambda} \geq 0, \tag{11c}$$
$$D^\top \lambda + \underline{\lambda} - \bar{\lambda} = f, \tag{11d}$$
$$\lambda^\top (Cx + Dy - b) = 0, \tag{11e}$$
$$\underline{\lambda}^\top (y - \underline{y}^F) = 0, \tag{11f}$$
$$\bar{\lambda}^\top (\bar{y}^F - y) = 0. \tag{11g}$$

of Problem (1). This well-known reformulation has been proposed in [17] and can be derived by replacing the lower-level problem (2) with its Karush–Kuhn–Tucker (KKT) conditions, i.e., primal feasibility that is contained in (11b), nonnegativity of the dual variables (11c), stationarity (11d), and complementarity (11e)–(11g). Due to the latter conditions, Problem (11) is a mathematical problem with complementarity constraints (MPCC). The complementarity constraints (11e) can be replaced using the mixed-integer reformulation

$$u \in \{0,1\}^\ell, \quad \lambda \leq Mu, \quad Cx + Dy - b \leq M(1-u);$$

see [17] as well. The complementarity constraints (11f) and (11g) can be treated in the same way. This formulation requires additional binary variables u and a sufficiently large value M. It is pointed out in [30] that this approach is not without obstacles. Choosing M too large may cause numerical instabilities and choosing it too small may result in suboptimal solutions of Problem (1). Further, in [22], it is shown that verifying the correctness of a given value M is as hard as solving the original bilevel problem. Still, this approach is by far the most frequently used approach to solve linear bilevel problems in practice, which is why we test our presolve techniques for this approach.[1] Throughout all tests, we set $M = 10^6$. Although this might exclude some bilevel-optimal points from the feasible set of the bilevel problem's reformulation (11), it still allows for a proper computational evaluation of the speed-ups obtained by applying presolve techniques, which is what we do in the following.

4.1. Computational Setup and Test Sets.

The big-M based single-level reformulation as well as all presolve techniques have been implemented in C++-11 and have been compiled with GCC 7.5.0. We solved all mixed-integer problems with Gurobi 9.0.2. The computational experiments have been executed on a compute cluster using compute nodes with Intel Xeon E3-1240 v6 CPUs with 4 cores, 3.7 GHz, and 32 GB RAM; see [31] for more details. Note the we limited the number of threads to 1 in all computations.

Our initial test set contains more than 2500 instances and mainly consists of the mixed-integer linear bilevel instances that are used in [23]. This set consists of mixed-integer bilevel instances from the literature, for which the integrality conditions are relaxed to obtain continuous bilevel problems. From this set, we removed all infeasible instances and all instances that we solve in less than 1 s and that we thus consider as too easy. The resulting test set only contains roughly 600 instances, on which we test the proposed presolve methods in the following. Let us already comment that—although we started with a rather large instance set that contains (to the best of our knowledge) all available

[1] Note that the presolve techniques evaluated in this section would also have a comparably beneficial effect on other solution approaches such as classic branch-and-bound for LP-LP bilevel problems [5] due to, e.g., a reduced number of lower-level constraints.

TABLE 2. Running times for parallel row presolve. All times are given in seconds.

		Running Time				# Rows	
Instance i	Ref.	t_i^{wo}	t_i^{w}	t_i^p	s_i	w/o	w
neos-1109824	[23, 24]	63.51	1.06	0.11	59.92	28979	9979
gmu-35-40	[23, 24]	1.28	0.86	0.01	1.49	424	419
acc-tight5	[23, 24]	12.37	9.55	0.03	1.30	3052	3045
rocl-4-11	[23, 28]	3.29	2.91	0.04	1.13	10883	10663
neos-4647030-tutaki	[23, 28]	206.08	209.87	3.08	-1.02	8382	8381
unitcal_7	[23, 24]	82.29	84.94	0.14	-1.03	48939	48936
bab5	[23, 24]	17.21	28.45	0.14	-1.65	4964	4943
gmu-35-50	[23, 28]	2.00	3.30	0.01	-1.65	435	427

(mixed-integer) linear bilevel instances from the literature—most presolve methods can only be applied to a very small amount of instances. The reason is that not too many instances indeed have, e.g., parallel rows or columns. We will discuss this test-library specific problem later on again in our conclusion in Section 5. Since the number of relevant instances is rather small, we specify the instances that we use for each presolve method along with a reference to its origin in the literature in each of the following sections.

4.2. **Parallel Rows.** We detected parallel rows in only 8 out of the roughly 600 instances. In Table 2, we compare the running times t_i^{wo} for solving an instance i without presolve with the running time t_i^{w} obtained if parallel-rows presolve is applied; see Section 3.3. In addition, we also specify the time t_i^p that is needed for detecting and removing parallel rows. Note that t_i^{w} specifies the total running time including t_i^p. The effect of the presolve method is measured by the speed-up factor

$$s_i = \begin{cases} t_i^{\text{wo}}/t_i^{\text{w}}, & \text{if } t_i^{\text{wo}} \geq t_i^{\text{w}}, \\ -t_i^{\text{w}}/t_i^{\text{wo}}, & \text{else.} \end{cases}$$

We see that removing parallel rows has a very significant effect on the neos-1109824 instance, which is solved almost 60 times faster if parallel-rows presolve is applied. The reason for this is an enormous reduction of constraints: around one third of all constraints can be removed; see the "# Rows" column in Table 2. On the other instances, presolving has mixed effects. On some instances (gmu-35-40, acc-tight5, and rocl-4-11), we observe a moderate reduction in running time and of the number of constraints. However, on other instances, removing parallel rows has none or a rather negative effect. The latter is hard to explain on the data basis we have—especially because different aspects seem to be the reason for the observed results. For example, the instance bab5 is solved by a root-node heuristic if no presolve is applied, which is not the case if presolve is applied, whereas the instance gmu-35-50 requires less branch-and-bound nodes to be solved if parallel-rows presolve is not used.

4.3. **Parallel Columns.** We detected parallel columns in 7 instances. The results are similar to the case of parallel-rows presolve; see Table 3. Detecting parallel columns has a significantly positive effect on the instance tanglegram1, on which we observe a speed-up of over 8, which is due to 126 presolved columns. On all other instances, we observe only moderate or no effects. This is surprising for some instances. For example, we remove more than 40 % of the variables of the instance gmu-35-50. Still, the speed-up of 1.13 is rather small. Moreover, we observed that the root node model after Gurobi's own presolve is smaller if we apply our parallel-columns presolve beforehand. However, this is not reflected in significant speed-ups for the running time.

TABLE 3. Numerical results for parallel columns presolve. All times are given in seconds.

		Running Time				# Columns	
Instance i	Ref.	t_i^{wo}	t_i^{w}	t_i^p	s_i	w/o	w
tanglegram1	[23, 24]	42.54	5.19	0.31	8.20	34759	34633
gmu-35-50	[23, 28]	2.00	1.77	0.02	1.13	1919	1106
unitcal_7	[23, 24]	82.49	81.42	0.20	1.01	25755	24747
eilB101	[23, 24]	23.84	23.88	0.03	-1.00	2818	2817
istanbul-no-cutoff	[23, 28]	24.18	24.36	0.08	-1.01	5282	5278
tanglegram2	[23, 24]	1.12	1.16	0.04	-1.04	4714	4680
neos13	[23, 24]	1.37	1.49	0.23	-1.09	1827	1826

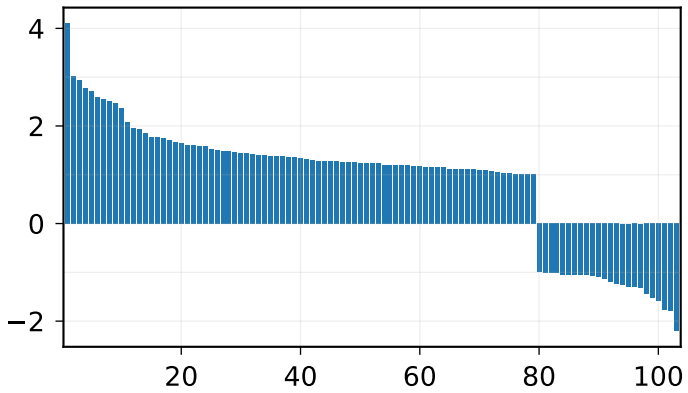

FIGURE 3. Speed-up factors s_i for duality fixing.

4.4. **Duality Fixing.** We detected variables that can be fixed via duality fixing in 103 instances. Thus, duality fixing can be applied by far to the largest subset of instances of our test set. This is in line with the positive results reported in [14], in which duality fixing is applied as well. Due to the large number of instances, we illustrate the speed-up factors s_i in a more condensed way in Figure 3. For the sake of completeness, we also denote running times and the number of variables in Table 6 in the appendix. We see that for the majority of instances, duality fixing is beneficial, and for many of them, the speed-up can be considered significant. For example, for 26 instances, we observe a speed-up of 1.5 or above, and for 55 instances, we observe a speed-up of 1.2 or above. In contrast, there are only very few instances, for which duality fixing has a negative impact. Only 5 instances have a speed-up factor of -1.5 or below.

4.5. **Bound Strengthening and Bound Moving.** Finally, we consider bound strengthening (see Section 3.1) and bound moving (see Remark 2). These methods are applicable to 7 instances; see Table 4 and Table 5. For both methods, the detection of implied tighter bounds is implemented in a computationally costly way by solving Problem (6) and (7). This could certainly be improved, e.g., by using internal solver information that is not accessible when using commercial solvers. In order to measure the speed-up provided by the presolve method, we denote cleaned speed-up factors $\tilde{s}_i$ with respect to $\tilde{t}_i^{\text{w}} = t_i^{\text{w}} - p_i$ instead of t_i^{w}, i.e., we disregard the presolve time when computing the speed-up factors.

For bound strengthening, we observe positive effects for the majority of instances. Comparing the two tables, we see that bound strengthening clearly outperforms bound moving. One reason for this might be that for bound strengthening we may compute

TABLE 4. Numerical results for bound strengthening. All times are given in seconds.

		Running Time				
Instance i	Ref.	t_i^{wo}	t_i^{w}	t_i^p	$\tilde{s}_i$	# Bounds
gmu-35-50	[23, 28]	2.00	2.24	1.33	2.20	41
gmu-35-40	[23, 24]	1.28	1.35	0.56	1.62	17
2AP05-21	[13]	49.68	40.85	0.01	1.22	1
n4-3	[23, 24]	1.76	8.14	6.65	1.18	1637
mcsched	[23, 24]	1.61	3.10	1.50	1.01	24
rococoB10-011000	[23, 28]	1.78	8.51	6.74	1.01	385
bnatt500	[23, 28]	1.07	18.65	17.57	-1.01	1055

TABLE 5. Numerical results for bound moving. All times are given in seconds.

		Running Time				
Instance i	Ref.	t_i^{wo}	t_i^{w}	t_i^p	$\tilde{s}_i$	# Bounds
gmu-35-40	[23, 24]	1.28	1.30	0.53	1.66	12
2AP05-21	[13]	49.68	38.09	0.01	1.30	1
rococoB10-011000	[23, 28]	1.78	10.96	9.20	1.01	385
bnatt500	[23, 28]	1.07	21.17	20.10	-1.00	1055
n4-3	[23, 24]	1.76	6.36	4.59	-1.01	13
mcsched	[23, 24]	1.61	3.88	1.48	-1.49	24
gmu-35-50	[23, 28]	2.00	4.79	1.29	-1.75	36

implied bounds even for follower variables that are initially unbounded. In contrast, bound moving only shifts existing redundant bounds from the follower to the leader. Thus, bound strengthening affects more bounds as indicated in the final column of the two tables.

5. Conclusion

In this paper, we systematically studied presolve methods for linear bilevel problems. Due to the fact that the IIC property does not hold for bilevel problems in general, this is a delicate task and the application of standard presolve methods from single-level optimization thus has to be done with great caution. Fortunately, we are able to carry over bound strengthening, the handling of parallel rows and columns, duality fixing, and a special type of an optimality-based presolve to the field of bilevel optimization.

Our numerical results indicate that the studied presolve methods can both be very beneficial for some instances but can also harm the solution process for other problems of our test set. However, the number of instances to which, e.g., parallel-rows or parallel-columns, presolve could be applied in our numerical experiments is too small to discuss the general impact of presolve for linear bilevel problems. This, in particular, reveals that there is an urgent need of better test instance libraries to further propel the field of computational bilevel optimization. The instance sets used in computational (mixed-integer) linear bilevel optimization are mostly interdiction instances that (i) have a very special structure and that (ii) are too easy if their inherent integrality conditions are relaxed so that they can be used to test techniques to solve continuous bilevel problems. In particular, (i) is most likely the reason that all interdiction instances of our general test set do not play any role for the numerical experiments carried out in this paper. Moreover, there are almost no real-world test instances publicly available that usually possess much more structures such as parallel columns or rows. In our opinion—and we are rather convinced that the

numerical study in this paper clearly reveals this aspect—the availability of better test instances is of crucial importance for the further development of the field. Finally, the field would also benefit from publicly available open-source solver frameworks for bilevel optimization so that one is not restricted to use commercial solvers as black-boxes.

Acknowledgments

This research has been performed as part of the Energie Campus Nürnberg and is supported by funding of the Bavarian State Government. We also thank the Deutsche Forschungsgemeinschaft for their support within project A05, B08, and Z01 in the "Sonderforschungsbereich/Transregio 154 Mathematical Modelling, Simulation and Optimization using the Example of Gas Networks".

Appendix: Detailed Results for Duality Fixing.

Here, we list detailed results for all of the 103 instances to which we applied duality fixing.

Table 6: Numerical results for duality fixing. All times are given in seconds.

		Running Time				# Variables	
Instance i	Ref.	t_i^{wo}	t_i^{w}	t_i^{p}	s_i	w/o	w
tanglegram1	[23, 24]	42.57	10.36	0.07	4.11	34759	17968
xuLarge900-1	[14]	70.74	23.41	0.26	3.02	1800	1355
bmilplib_310_1	[36]	2.15	0.73	0.04	2.95	620	448
bmilplib_360_9	[36]	3.19	1.15	0.05	2.77	720	550
xuLarge800-4	[14]	7.25	2.67	0.21	2.72	1600	1227
xuLarge500-9	[14]	6.12	2.35	0.09	2.60	1000	762
xuLarge1000-8	[14]	107.76	42.13	0.32	2.56	2000	1476
bmilplib_360_7	[36]	2.01	0.80	0.05	2.51	720	547
bmilplib_460_6	[36]	5.68	2.31	0.07	2.46	920	683
xuLarge900-4	[14]	70.74	29.97	0.26	2.36	1800	1341
bmilplib_410_4	[36]	1.35	0.65	0.08	2.08	820	609
xuLarge600-1	[14]	13.76	7.05	0.12	1.95	1200	887
xuLarge500-4	[14]	8.40	4.36	0.08	1.93	1000	739
bmilplib_310_2	[36]	1.88	1.01	0.04	1.86	620	458
xuLarge800-5	[14]	5.60	3.14	0.21	1.78	1600	1189
xuLarge800-9	[14]	20.68	11.60	0.21	1.78	1600	1197
xuLarge700-5	[14]	9.27	5.30	0.16	1.75	1400	1027
xuLarge800-1	[14]	49.98	29.00	0.21	1.72	1600	1215
xuLarge900-5	[14]	21.15	12.70	0.26	1.67	1800	1342
bmilplib_460_9	[36]	3.73	2.25	0.07	1.66	920	675
bmilplib_360_2	[36]	4.25	2.65	0.05	1.60	720	546
bmilplib_310_4	[36]	2.58	1.61	0.04	1.60	620	455
xuLarge1000-4	[14]	6.14	3.86	0.32	1.59	2000	1479
xuLarge600-5	[14]	8.11	5.14	0.12	1.58	1200	877
xuLarge1000-3	[14]	99.36	64.99	0.32	1.53	2000	1461
xuLarge700-10	[14]	24.49	16.27	0.16	1.51	1400	1038
xuLarge700-1	[14]	3.28	2.20	0.16	1.49	1400	1035
xuLarge900-6	[14]	19.63	13.18	0.26	1.49	1800	1366
bmilplib_410_8	[36]	2.02	1.37	0.06	1.47	820	609
xuLarge1000-5	[14]	35.67	24.72	0.32	1.44	2000	1488

bmilplib_460_2	[36]	5.57	3.86	0.07	1.44	920	680
bmilplib_260_2	[36]	1.14	0.80	0.03	1.42	520	394
xuLarge1000-9	[14]	18.40	13.15	0.32	1.40	2000	1494
bmilplib_310_6	[36]	1.36	0.97	0.04	1.40	620	456
xuLarge600-3	[14]	5.31	3.83	0.12	1.39	1200	893
xuLarge900-8	[14]	47.64	34.32	0.26	1.39	1800	1332
xuLarge700-8	[14]	19.37	13.90	0.16	1.39	1400	1049
bmilplib_410_9	[36]	5.50	4.03	0.06	1.36	820	620
bmilplib_310_8	[36]	1.21	0.89	0.04	1.36	620	446
bmilplib_360_5	[36]	2.37	1.77	0.05	1.34	720	548
bmilplib_460_8	[36]	7.64	5.78	0.08	1.32	920	682
xuLarge800-7	[14]	7.85	5.98	0.21	1.31	1600	1199
bmilplib_310_7	[36]	1.38	1.07	0.04	1.29	620	485
xuLarge600-8	[14]	6.41	4.98	0.12	1.29	1200	917
bmilplib_310_10	[36]	3.20	2.48	0.04	1.29	620	474
bmilplib_410_2	[36]	3.08	2.41	0.06	1.28	820	597
xuLarge700-3	[14]	56.34	44.84	0.16	1.26	1400	1051
xuLarge900-3	[14]	55.40	43.87	0.30	1.26	1800	1333
xuLarge600-2	[14]	7.60	6.02	0.12	1.26	1200	881
xuLarge800-2	[14]	100.68	80.77	0.21	1.25	1600	1206
xuLarge500-5	[14]	7.63	6.17	0.09	1.24	1000	759
bmilplib_360_4	[36]	4.57	3.70	0.05	1.24	720	536
bmilplib_360_3	[36]	2.66	2.17	0.05	1.23	720	549
bmilplib_460_4	[36]	3.67	3.06	0.07	1.20	920	689
bmilplib_460_1	[36]	4.85	4.03	0.08	1.20	920	699
xuLarge600-10	[14]	6.13	5.14	0.12	1.19	1200	895
xuLarge600-6	[14]	7.51	6.33	0.12	1.19	1200	914
xuLarge500-1	[14]	10.97	9.24	0.09	1.19	1000	755
xuLarge1000-1	[14]	23.32	19.78	0.32	1.18	2000	1482
xuLarge800-6	[14]	28.41	24.11	0.21	1.18	1600	1205
xuLarge500-7	[14]	6.83	5.93	0.08	1.15	1000	753
bmilplib_410_1	[36]	4.79	4.16	0.06	1.15	820	625
xuLarge500-10	[14]	3.10	2.69	0.09	1.15	1000	739
bmilplib_310_9	[36]	1.97	1.71	0.04	1.15	620	471
xuLarge700-7	[14]	12.97	11.58	0.16	1.12	1400	1040
bmilplib_410_6	[36]	2.41	2.15	0.06	1.12	820	609
bmilplib_410_3	[36]	1.72	1.55	0.06	1.11	820	614
gmu-35-50	[23, 28]	2.00	1.80	0.01	1.11	1919	1769
xuLarge600-7	[14]	15.23	13.72	0.12	1.11	1200	895
xuLarge500-8	[14]	4.05	3.68	0.09	1.10	1000	759
bmilplib_460_3	[36]	3.32	3.04	0.07	1.09	920	695
wachplan	[23, 28]	1.32	1.22	0.03	1.08	3361	2870
cvs16r128-89	[23, 28]	15.27	14.41	0.01	1.06	3472	3160
bmilplib_360_10	[36]	2.17	2.10	0.05	1.03	720	541
xuLarge700-4	[14]	21.16	20.54	0.16	1.03	1400	1059
xuLarge700-6	[14]	11.23	11.04	0.16	1.02	1400	1040
bmilplib_360_1	[36]	2.36	2.31	0.05	1.02	720	533
bmilplib_410_5	[36]	2.49	2.46	0.06	1.01	820	600
bmilplib_360_8	[36]	3.73	3.71	0.05	1.01	720	543
tanglegram2	[23, 24]	1.12	1.12	0.01	-1.00	4714	2581
bmilplib_410_10	[36]	9.56	9.62	0.06	-1.01	820	631
xuLarge800-10	[14]	24.10	24.39	0.21	-1.01	1600	1200

bmilplib_460_5	[36]	10.60	10.68	0.07	-1.01	920	702
gmu-35-40	[23, 24]	1.28	1.35	0.01	-1.05	1205	1055
xuLarge800-3	[14]	92.04	97.73	0.21	-1.06	1600	1195
xuLarge900-9	[14]	51.79	54.74	0.26	-1.06	1800	1330
xuLarge600-9	[14]	12.76	13.54	0.12	-1.06	1200	891
bmilplib_460_7	[36]	2.94	3.11	0.08	-1.06	920	684
xuLarge700-2	[14]	26.89	29.05	0.16	-1.08	1400	1061
satellites1-25	[23, 24]	59.16	64.37	0.03	-1.09	9013	9012
xuLarge1000-10	[14]	48.11	54.28	0.33	-1.13	2000	1516
xuLarge600-4	[14]	6.54	7.86	0.12	-1.20	1200	919
xuLarge500-6	[14]	7.12	8.86	0.09	-1.24	1000	756
xuLarge500-3	[14]	4.67	5.84	1.89	-1.25	1000	749
xuLarge1000-2	[14]	13.10	16.95	0.32	-1.29	2000	1513
xuLarge900-7	[14]	89.33	116.28	0.27	-1.30	1800	1361
xuLarge500-2	[14]	12.64	16.58	0.09	-1.31	1000	761
xuLarge800-8	[14]	15.86	22.97	0.21	-1.45	1600	1181
xuLarge1000-7	[14]	116.16	177.57	0.32	-1.53	2000	1460
xuLarge900-2	[14]	58.69	92.74	0.26	-1.58	1800	1341
xuLarge1000-6	[14]	18.48	32.71	0.32	-1.77	2000	1488
xuLarge900-10	[14]	123.22	221.17	0.26	-1.79	1800	1338
bmilplib_410_7	[36]	2.90	6.41	0.06	-2.21	820	619

References

[1] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. "Multi-Row Presolve Reductions in Mixed Integer Programming." In: *Proceedings of the Twenty-Sixth RAMP Symposium* (Tokyo). Ed. by T. Hosei University. Oct. 16–17, 2014, pp. 181–196. URL: http://www.orsj.or.jp/ramp/2014/paper/4-4.pdf.

[2] T. Achterberg, R. E. Bixby, Z. Gu, E. Rothberg, and D. Weninger. "Presolve Reductions in Mixed Integer Programming." In: *INFORMS Journal on Computing* 32.2 (2020), pp. 473–506. DOI: 10.1287/ijoc.2018.0857.

[3] E. D. Andersen and K. D. Andersen. "Presolving in linear programming." In: *Mathematical Programming* 71 (2 1995), pp. 221–245. DOI: 10.1007/BF01586000. URL: http://dx.doi.org/10.1007/BF01586000.

[4] C. Audet, G. Savard, and W. Zghal. "New Branch-and-Cut Algorithm for Bilevel Linear Programming." In: *Journal of Optimization Theory and Applications* 134.2 (2007), pp. 353–370. DOI: 10.1007/s10957-007-9263-4.

[5] J. F. Bard. *Practical bilevel optimization: algorithms and applications*. Vol. 30. Springer Science & Business Media, 1998. DOI: 10.1007/978-1-4757-2836-1.

[6] J. F. Bard and J. T. Moore. "A Branch and Bound Algorithm for the Bilevel Programming Problem." In: *SIAM Journal on Scientific and Statistical Computing* 11.2 (1990), pp. 281–292. DOI: 10.1137/0911017.

[7] P. Belotti, S. Cafieri, J. Lee, and L. Liberti. *On feasibility based bounds tightening*. Tech. rep. Preprint. Optimization Online, 2012. URL: http://www.optimization-online.org/DB_HTML/2012/01/3325.html.

[8] P. Belotti, S. Cafieri, J. Lee, and L. Liberti. "Feasibility-Based Bounds Tightening via Fixed Points." In: *Combinatorial Optimization and Applications*. Ed. by W. Wu and O. Daescu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 65–76.

[9] R. E. Bixby and D. K. Wagner. "A note on detecting simple redundancies in linear systems." In: *Operations Research Letters* 6.1 (1987), pp. 15–17. DOI: 10.1016/0167-6377(87)90004-6.

[10] W.-K. Chen, P. Gemander, A. Gleixner, L. Gottwald, A. Martin, and D. Weninger. "Two-row and two-column mixed-integer presolve using hashing-based pairing methods." In: *EURO Journal on Computational Optimization* 8.3 (2020), pp. 205–240. DOI: 10.1007/s13675-020-00129-6.

[11] S. Dempe. *Foundations of Bilevel Programming*. Springer, 2002. DOI: 10.1007/b101970.

[12] S. Dempe and S. Lohse. *Dependence Of Bilevel Programming On Irrelevant Data*. Tech. rep. Preprint. TU Bergakademie Freiberg, 2011.

[13] S. T. DeNegre. "Interdiction and discrete bilevel linear programming." PhD Thesis. Lehigh University, 2011. URL: http://coral.ie.lehigh.edu/~ted/files/papers/ScottDeNegreDissertation11.pdf.

[14] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. "A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs." In: *Operations Research* 65.6 (2017), pp. 1615–1637. DOI: 10.1287/opre.2017.1650.

[15] M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. "On the use of intersection cuts for bilevel optimization." In: *Mathematical Programming* 172.1 (2018), pp. 77–103. DOI: 10.1007/s10107-017-1189-5.

[16] M. Fischetti, M. Monaci, and M. Sinnl. "A dynamic reformulation heuristic for Generalized Interdiction Problems." In: *European Journal of Operational Research* 267.1 (2018), pp. 40–51. DOI: 10.1016/j.ejor.2017.11.043.

[17] J. Fortuny-Amat and B. McCarl. "A Representation and Economic Interpretation of a Two-Level Programming Problem." In: *The Journal of the Operational Research Society* 32.9 (1981), pp. 783–792. DOI: 10.1057/jors.1981.156.

[18] G. Gamrath, T. Koch, A. Martin, M. Miltenberger, and D. Weninger. "Progress in presolving for mixed integer programming." In: *Mathematical Programming Computation* 7 (2015), pp. 367–398. DOI: 10.1007/s12532-015-0083-5.

[19] A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge. "Three enhancements for optimization-based bound tightening." In: *Journal of Global Optimization* 67.4 (2017), pp. 731–757. DOI: 10.1007/s10898-016-0450-4. URL: https://doi.org/10.1007/s10898-016-0450-4.

[20] P. Hansen, B. Jaumard, and G. Savard. "New branch-and-bound rules for linear bilevel programming." In: *SIAM Journal on Scientific and Statistical Computing* 13.5 (1992), pp. 1194–1217. DOI: 10.1137/0913069.

[21] T. Kleinert, M. Labbé, F. Plein, and M. Schmidt. "Closing the Gap in Linear Bilevel Optimization: A New Valid Primal-Dual Inequality." In: *Optimization Letters* 15 (2021), pp. 1027–1040. DOI: 10.1007/s11590-020-01660-6.

[22] T. Kleinert, M. Labbé, F. Plein, and M. Schmidt. "Technical Note—There's No Free Lunch: On the Hardness of Choosing a Correct Big-*M* in Bilevel Optimization." In: *Operations Research* 68.6 (2020), pp. 1716–1721. DOI: 10.1287/opre.2019.1944.

[23] T. Kleinert and M. Schmidt. "Computing Feasible Points of Bilevel Problems with a Penalty Alternating Direction Method." In: *INFORMS Journal on Computing* (2020). DOI: 10.1287/ijoc.2019.0945. Online first.

[24] T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. "MIPLIB 2010." In: *Mathematical Programming Computation* 3.2 (2011), pp. 103–163. DOI: 10.1007/s12532-011-0025-9.

[25] C. M. Macal and A. P. Hurter. "Dependence of bilevel mathematical programs on irrelevant constraints." In: *Computers & Operations Research* 24.12 (1997), pp. 1129–1140. DOI: 10.1016/S0305-0548(97)00025-7.

[26] J. Manns. "Presolve of Linear Bilevel Programs." MA thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 2020. URL: https://opus4.kobv.de/opus4-trr154/frontdoor/index/index/docId/320.

[27] A. Martin. "General Mixed Integer Programming: Computational Issues for Branch-and-Cut Algorithms." In: *Computational combinatorial optmization*. Ed. by D. Naddef and M. Jünger. Vol. 2241. Berlin: Springer, 2001, pp. 1–25. DOI: 10.1007/3-540-45586-8_1.

[28] *MIPLIB 2017*. http://miplib.zib.de/. Accessed: 2021-03-05. 2018.

[29] J. T. Moore and J. F. Bard. "The Mixed Integer Linear Bilevel Programming Problem." In: *Operations Research* 38.5 (1990), pp. 911–921. DOI: 10.1287/opre.38.5.911.

[30] S. Pineda and J. M. Morales. "Solving Linear Bilevel Problems Using Big-Ms: Not All That Glitters Is Gold." In: *IEEE Transactions on Power Systems* 34.3 (2019), pp. 2469–2471. DOI: 10.1109/TPWRS.2019.2892607.

[31] Regionales Rechenzentrum Erlangen. *Woodcrest Cluster*. URL: https://www.anleitungen.rrze.fau.de/hpc/woody-cluster/ (visited on 11/30/2020).

[32] H. S. Ryoo and N. V. Sahinidis. "A branch-and-reduce approach to global optimization." In: *Journal of Global Optimization* 8.2 (1996), pp. 107–138. DOI: 10.1007/BF00138689. URL: https://doi.org/10.1007/BF00138689.

[33] M. W. P. Savelsbergh. "Preprocessing and probing techniques for mixed integer programming problems." In: *ORSA Journal on Computing* 6 (1994), pp. 445–454.

[34] S. Tahernejad, T. K. Ralphs, and S. T. DeNegre. "A branch-and-cut algorithm for mixed integer bilevel linear optimization problems and its implementation." In: *Mathematical Programming Computation* (2020), pp. 1–40. DOI: 10.1007/s12532-020-00183-6.

[35] L. Vicente, G. Savard, and J. Júdice. "Descent approaches for quadratic bilevel programming." In: *Journal of Optimization Theory and Applications* 81.2 (1994), pp. 379–399. DOI: 10.1007/BF02191670.

[36] P. Xu and L. Wang. "An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions." In: *Computers & Operations Research* 41 (2014), pp. 309–318. DOI: 10.1016/j.cor.2013.07.016.

(T. Kleinert) (A) Friedrich-Alexander-Universität Erlangen-Nürnberg, Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany; (B) Energie Campus Nürnberg, Fürther Str. 250, 90429 Nürnberg, Germany

Email address: thomas.kleinert@fau.de

(J. Manns) Friedrich-Alexander-Universität Erlangen-Nürnberg, Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany

Email address: julian.manns@fau.de

(M. Schmidt) Trier University, Department of Mathematics, Universitätsring 15, 54296 Trier, Germany

Email address: martin.schmidt@uni-trier.de

(D. Weninger) Friedrich-Alexander-Universität Erlangen-Nürnberg, Discrete Optimization, Cauerstr. 11, 91058 Erlangen, Germany

Email address: dieter.weninger@fau.de