A new general-purpose algorithm for mixed-integer bilevel linear programs

Matteo Fischetti [*1], Ivana Ljubić [†2], Michele Monaci [‡3], and Markus Sinnl[§4]

[1]*DEI, University of Padua, Italy.*
[2]*ESSEC Business School of Paris, France.*
[2]*DEI, University of Bologna, Italy.*
[4]*ISOR, University of Vienna, Vienna, Austria.*

Abstract

Bilevel optimization problems are very challenging optimization models arising in many important practical contexts, including pricing mechanisms in the energy sector, airline and telecommunication industry, transportation networks, optimal expansion of gas networks, critical infrastructure defense, and machine learning.

In this paper, we present a new general purpose branch-and-cut framework for the exact solution of mixed-integer bilevel linear programs (MIBLP), which constitute a very significant subfamily of bilevel optimization problems. Our framework introduces several new classes of valid inequalities to speed-up the solver, along with a very effective bilevel-specific preprocessing procedure.

A very extensive computational study is presented, where we evaluate the performance of various solution methods on a common testbed of more than 800 instances from the literature—this is by far the most extensive computational analysis ever performed for exact MIBLP solvers. Our new algorithm consistently outperforms (often by a large margin) all alternative state-of-the-art methods from the literature, including methods which exploit problem specific information for special instance classes. In particular, it allows to solve to optimality more than 300 instances previously unsolved instances from literature.

1 Introduction

In bilevel optimization, there are two decision makers, commonly denoted as the *leader* and the *follower*, and decisions are made in a hierarchical manner: first the leader makes a decision, and then the follower optimizes its objective, affected by the decisions of the leader. It is assumed that the leader can anticipate the decisions of the follower, hence the leader optimization task is a nested optimization problem that takes into consideration the follower response. Bilevel optimization problems, together with their generalization to multilevel optimization, play a fundamental role in many real-life applications, when competitive agents operate in a hierarchical way with conflicting objectives. As such, they can be interpreted as a static Stackelberg game, and find applications in many economic models. For example, it is well known that bilevel optimization is an inevitable tool for modeling pricing mechanisms in the energy sector (Zugno et al. 2013), airline and telecommunication industry (Brotcorne et al. 2008), or in transportation networks (Gilbert et al. 2015, Labbé et al. 1998). Similarly, bilevel optimization is the model of choice for capacity

[*]matteo.fischetti@unipd.it
[†]ivana.ljubic@essec.edu
[‡]michele.monaci@unibo.it
[§]markus.sinnl@univie.ac.at

planning decisions made in a competitive environment; see, e.g., Garcia-Herreros et al. (2016) for an optimal expansion of gas networks. Infrastructure planning that takes into consideration deliberate disruptions (due to sabotage or terrorist attacks) is another important example of bilevel optimization (Brown et al. 2006, Scaparra and Church 2008, Wood 2010). Recently, bilevel optimization has been used even in machine learning applications, see Kunisch and Pock (2013).

Despite this increasing interest for bilevel optimization and the fact that the seminal formulation of bilevel programs dates back to the 70's (Bracken and McGill 1973), implementation of *generic bilevel optimization solvers* has only started in recent years. This can be explained by the inherent complexity of the bilevel optimization, which is known to be NP-hard even when both leader and follower problems are linear programs (Jeroslow 1985). In the present article we propose a novel generic solver that covers a large family of bilevel optimization problems in which decisions of both, the leader and the follower, are modeled as mixed-integer linear programs.

More precisely, in this paper we address a generic Mixed-Integer Bilevel Linear Program (MIBLP), i.e., a bilevel optimization problem where all objective functions and constraints are linear, and some/all variables are required to take integer values. Note that MIBLPs are Σ_2-hard (DeNegre 2011, Jeroslow 1985). A MIBLP is defined as follows:

$$\min c_x^T x + c_y^T y \tag{1}$$
$$G_x x + G_y y \leq q \tag{2}$$
$$x_j \text{ integer}, \qquad \forall j \in J_x \tag{3}$$
$$y \in \arg\min_{y' \in \mathbb{R}^{n_2}} \{d^T y' : Ax + By' \leq b, \quad l \leq y' \leq u, \quad y'_j \text{ integer } \forall j \in J_y\} \tag{4}$$

where $x \in \mathbb{R}^{n_1}$, $y \in \mathbb{R}^{n_2}$, while c_x, c_y, G_x, G_y, q, d, A, B, b, l and u are given rational matrices/vectors of appropriate size, and sets

$$J_x \subseteq N_x := \{1, \cdots, n_1\} \text{ and } J_y \subseteq N_y := \{1, \cdots, n_2\}$$

identify the (possibly empty) indices of the integer-constrained variables in x and y, respectively. We will refer to (1) and (2)–(3) as the *leader* objective function and constraints, respectively, and to (4) as the *follower subproblem*. In case the follower subproblem has multiple optimal solutions, we assume that one with minimum leader cost among those satisfying the leader constraints is chosen—i.e., we consider the *optimistic* version of bilevel optimization; see, e.g., Loridan and Morgan (1996).

For the leader, we assume that explicit lower/upper bounds (if any) on the variables x and y are included in constraints (2). Whenever needed, however, we will refer to this subset of constraints using notation

$$x^- \leq x \leq x^+ \tag{5}$$
$$y^- \leq y \leq y^+ \tag{6}$$

where, as customary, some entries in x^+, x^-, y^-, y^- are allowed to be $\pm\infty$.

MIBLP can conveniently be restated in its *value function formulation* as:

$$\min c_x^T x + c_y^T y \tag{7}$$
$$G_x x + G_y y \leq q \tag{8}$$
$$Ax + By \leq b \tag{9}$$
$$l \leq y \leq u \tag{10}$$
$$x_j \text{ integer}, \quad \forall j \in J_x \tag{11}$$
$$y_j \text{ integer}, \quad \forall j \in J_y \tag{12}$$
$$d^T y \leq \Phi(x) \tag{13}$$

where the follower value function for a given $x^* \in \mathbb{R}^{n_1}$ is computed by solving the following *follower* Mixed-Integer Linear Program (MILP)

$$\Phi(x^*) := \min_{y \in \mathbb{R}^{n_2}} \{d^T y \ : \ By \leq b - Ax^*, \quad l \leq y \leq u, \quad y_j \text{ integer } \forall j \in J_y\}. \tag{14}$$

Dropping condition (13) from model (7)-(13) leads to the so-called *High Point Relaxation* (HPR). The latter one is a Mixed-Integer Linear Program (MILP), whose Linear Programming (LP) relaxation will be denoted by $\overline{\text{HPR}}$.

An HPR solution (x, y) will be called *bilevel infeasible* if it violates (13). A point $(x, y) \in \mathbb{R}^n$, where $n = n_1 + n_2$, will be called *bilevel feasible* if it satisfies all constraints (8)–(13).

Let A_j be the j-th column of matrix A, and A_{ij} it generic entry. In what follows we will use notation

$$J_F := \{j \in N_x : A_j \neq 0\} \tag{15}$$

to denote the index set of the leader variables x_j (not necessarily integer-constrained) appearing in the follower problem.

Fischetti et al. (2016b) recently proposed a proof-of-concept exact MIBLP solver which was in fact a Branch-and-Cut (B&C) MILP approach with non-invasive supplements needed to correctly handle bilevel optimization. An important feature of that work which distinguished it from those in previous literature, is that the MIBLP solver was built on top of a stable and powerful MILP solver taking care of all non-bilevel specific issues—including cuts, heuristics, propagations, numerical stability, effective LP parametrization, multi-threading support, etc.

In the present work, we considerably extend the results in (Fischetti et al. 2016b) and develop a more versatile and effective solver for MIBLPs. The main novel contributions of the present paper can be stated as follows.

- A new family of cuts based on formulation (7)-(13) is introduced.
- A new bilevel-specific preprocessing procedure is proposed, and its very significant impact on the performance of the MIBLP solver is demonstrated;
- New families of Intersection Cuts (ICs) for bilevel programs are presented. ICs were originally proposed by Balas (1971) for integer programs, and were exploited in the context of bilevel optimization in (Fischetti et al. 2016b) for the first time. However, in the latter approach, their application was limited to the cases where $Ax + Byb$ is integer for all HPR solutions (x, y). In the present article we present an additional class of such inequalities, based on the recent work of Xu (2012), as well as a new family of "hypercube" ICs that can be applied even when the above assumption does not hold. For all cuts, sound separation procedures are described.
- A detailed description on how the ICs can be implemented in a numerically stable way is given.
- A very extensive computational study is reported—different classes of test problems from the literature are considered, including those proposed by DeNegre and Ralphs (2009), Xu and Wang (2014), Tang et al. (2015) and Fischetti et al. (2016b). Our analysis shows that the new approach outperforms by a large margin alternative state-of-the-art methods from the literature—including the most recent ones, namely those proposed in (Xu and Wang 2014, Tang et al. 2015) and Fischetti et al. (2016b).
- The optimal solution values for hundreds of open instances from the recent literature are provided; **see the on-line Appendix (Fischetti et al. 2016a) for details.**

The paper is organized as follows. In the remainder of this section, we review the most relevant approaches to MIBLP from the literature. Section 2 introduces a family of cutting planes called *follower upper bound* cuts along with a new bilevel-specific preprocessing. In Section 3 we derive two new families of MIBLP intersection cuts. In Section 4 we introduce separation algorithms and their numerically safe implementation

within a B&C solver. Section 5 gives some details about the implementation of our B&C algorithm. The performance of our solver is evaluated in Section 6 by means of computational experiments on a very large set of instances from the literature. Finally, Section 7 draws some conclusions.

To simplify our treatment, in what follows we assume that the HPR feasible set is compact, that MIBLP has a finite optimal solution, and that the follower MILP (14) has a finite optimal solution for every HPR solution $(x, \cdot)$. The reader is referred to Fischetti et al. (2016b) for a discussion on how to deal with an unbounded HPR feasible set, and how to detect and handle unbounded/infeasible cases.

1.1 Literature Overview on MIBLP

Even though there exists a large body of literature devoted to bilevel optimization, there are relatively few generic bilevel approaches that allow for integer decision variables both in the leader and in the follower. The first generic branch-and-bound approach to MIBLP was given by Moore and Bard (1990). Their algorithm was shown to converge in two cases: either when all leader variables are integer, or when the follower subproblem is an LP. About 20 years later, building upon the ideas from Moore and Bard (1990), a MILP-based branch-and-cut algorithm was introduced by DeNegre and Ralphs (2009), DeNegre (2011). The latter approach, publicly available as `MibS` solver (Ralphs 2015), requires that both the leader and the follower are purely integer problems, as it exploits integer slacks to cut off bilevel infeasible solutions.

Only very recently, one could observe a growing number of attempts to develop generic bilevel solvers, but also specialized algorithms that address particular bilevel problem variants. In the following, we summarize these approaches. A Benders-like decomposition scheme for general MIBLPs has been proposed by Saharidis and Ierapetritou (2009). An iterative MILP approach based on multi-way branching on the slack variables on the follower constraints has been given by Xu and Wang (2014), and another multi-level branching idea has been exploited in the scheme called "the watermelon algorithm" (Xu 2012). Both approaches require the leader variables to be integer, the main difference being that the former allows the follower to be a MILP, whereas in the latter the follower contains integer variables only. Another branch-and-cut method that works for integer leader and follower variables only has been recently proposed by Caramia and Mari (2015).

For the special family of *zero-sum* bilevel problems, i.e., when the leader and the follower share the same objective but with the opposite signs, three generic solution algorithms have recently been proposed by Tang et al. (2015). Their algorithms require leader variables to be binary, whereas the follower can be a general MILP. Interdiction problems are a special family of zero-sum bilevel problems in which the leader is given a limited budget to "interdict" the action of the follower. For interdiction problems, specialized schemes have been developed, see for example, a cutting plane approach by Wood (2010), and a more recent list of references in (Tang et al. 2015).

It is also worth mentioning that, to the best of our knowledge, very few solution schemes (and almost no extensive computational studies) for more general bilevel mixed nonlinear programs have been proposed so far—see, e.g., the exact approaches by Gümüs and Floudas (2005), Mitsos (2010), and Kleniati and Adjiman (2015).

In (Fischetti et al. 2016b), the first branch-and-cut approach for MIBLP that uses intersection cuts as driving force has been given. Compared to the existing literature, this algorithm requires much less restrictions for its convergence, and allows for both mixed-integer leader and follower subproblems. An important assumption is that the leader variables that influence the follower decision are all integer. Computational results reported in (Fischetti et al. 2016b) demonstrated that this generic solver significantly outperforms the methods by DeNegre and Ralphs (2009), DeNegre (2011) and Caramia and Mari (2015). It will be therefore considered in the remainder of this article as the state-of-the-art approach and an important reference for comparing the new features of the exact MIBLP solver presented in this article.

2 An Improved Branch-and-Cut Approach

It is well known that the HPR relaxation may provide arbitrarily weak lower bounds. Hence, for enumerative methods relying on this relaxation, it is crucial to consider deep cuts that cut off infeasible solutions early

in the Branch-and-Bound (B&B) tree, so as to improve the tightness of its lower and upper bounds.

Building on the approach proposed in (Fischetti et al. 2016b), we next introduce two new features that have a significant impact on the solver performance: in Section 2.1 we introduce locally valid cuts, denoted as *Follower Upper-Bound (FUB)* cuts, that are based on an estimation of an upper bound on the optimal solution value of the follower subproblem. In Section 2.2 we describe a new preprocessing rule that allows fixing of some y values in HPR using information from the follower MILP.

2.1 Follower Upper-Bound (FUB) Cuts

Observe that valid lower bounds for a bilevel problem can be obtained by restricting the follower subproblem and, consequently, overestimating the value of $\Phi(x)$ for an arbitrary leader solution x. In the following, we exploit this fact inside of a branch-and-bound procedure, to derive valid cuts for the HPR relaxation.

Indeed, the value function reformulation (7)-(13) introduced in Section 1 is nonconvex due to the presence of constraint (13). A valid lower bound on the optimal solution value at each B&B node can be obtained by relaxing this constraint, and replacing $\Phi(x)$ with some constant overestimator for the current node. This operation, which is the basis of the branch-and-sandwich approach proposed by Kleniati and Adjiman (2015), is just trivial for the B&B nodes where all the x_j variables appearing in the follower MILP have been already fixed by branching. However, for the remaining B&B nodes, one can exploit the local bounds on the x variables (x^-, x^+) at the current node, to strengthen the HPR relaxation. We consider the following restriction of the follower subproblem: in each follower constraint, x variables are replaced by the worst possible outcome for the follower, thus resulting in an MILP on y variables only. The optimal solution of the latter MILP gives a valid overestimation of $\Phi(x)$. We have the following result.

Theorem 1. *Let (x^-, x^+) denote the bounds for the x variables at the current B&B node. Then the following* Follower Upper Bound *(FUB) cut is locally valid for the current node:*

$$d^T y \le FUB(x^-, x^+) \tag{16}$$

where $FUB(x^-, x^+)$ is the optimal solution value of the following restricted follower MILP

$$FUB(x^-, x^+) := \min d^T y \tag{17}$$

$$\sum_{j \in N_x} \max\{A_{ij} x_j^-, A_{ij} x_j^+\} + \sum_{j \in N_y} B_{ij} y_j \le b_i, \qquad i = 1, \dots, m \tag{18}$$

$$l \le y \le u \tag{19}$$

$$y_j \text{ integer}, \qquad \forall j \in J_y, \tag{20}$$

and m denotes the number of rows of matrices A and B, and $FUB(x^-, x^+) = +\infty$ in case the problem is infeasible.

Proof. It is enough to observe that, by construction, the above MILP is a restriction of the follower MILP for any x with $x^- \le x \le x^+$, which implies $\Phi(x) \le FUB(x^-, x^+)$ for any such x. The claim then follows as the FUB cut is just a relaxation of the value-function constraint (13) at the current node. □

2.2 Follower Preprocessing

Preprocessing is a very important tool in modern MILP solvers, that for many problems has a considerable impact on computing time. By design, our approach automatically exploits standard MILP-based preprocessing whenever the follower MILPs are solved. There is however a bilevel-specific preprocessing operation that is potentially very useful, in that it conveys relevant information from the follower to the HPR. In particular, any y variables that can be fixed (for whatever reason) in the follower MILP *independently of* x, can be fixed at the HPR level as well, thus potentially improving the quality of the associated lower bound. We have the following result (recall that the explicit constraints in the follower MILP (14) are in $\le$ form, namely, $Ax + By \le b$):

Theorem 2. *For every follower variable y_j ($j \in N_y$), the following fixing is correct:*

(a) if $d_j > 0$ and $B_j \geq 0$, fix y_j to its lower bound l_j by setting $y_j^+ := y_j^- := u_j := l_j$;

(b) if $d_j < 0$ and $B_j \leq 0$, fix y_j to its upper bound u_j by setting $y_j^+ := y_j^- := l_j := u_j$;

Proof. Given an $x^* \in \mathbb{R}^{n_1}$ and a follower solution y that is feasible for (14), a follower feasible solution of better cost can possibly be obtained by decreasing each variable y_j with positive cost and non-negative coefficients in all constraints, and/or by increasing each variable y_j with $d_j < 0$ and non-positive coefficients. Thus, any bilevel feasible solution will have $y_j = l_j$ and $y_j = u_j$ for each variable j in cases (a) and (b), respectively. □

Note that in *optimistic* bilevel setting, the leader is free to choose among equivalent follower solutions the one it prefers. Thus, we require the statement of Theorem 2 be valid *for any* optimal solution of the follower, yielding strict inequalities in cases (a) and (b). In case $d_j = 0$ we must preserve all equivalent optimal solutions, hence variable y_j cannot be fixed.

As shown in the computational Section 6, the simple fixing of Theorem 2 can lead to a very significant speedup when solving certain classes of instances.

3 New Families of Intersection Cuts

Intersection cuts were introduced by Balas (1971) and are widely used in the context of MILPs. As customary in a B&C context, given a bilevel-infeasible HPR point (x^*, y^*), one aims to deriving a cutting plane that will cut off this point, while keeping the bilevel-feasible points intact. For an IC to serve this purpose, one requires the definition of two sets:

(1) a cone pointed at (x^*, y^*) that contains all the bilevel feasible solutions, and

(2) a convex set S that contains (x^*, y^*) but no bilevel feasible solutions in its *interior*.

Typically, for a vertex (x^*, y^*) of $\overline{\text{HPR}}$, a suitable cone is the corner polyhedron associated with the corresponding optimal basis. In that case, the strength of the derived IC only depends on the choice of the underlying convex set S. In (Fischetti et al. 2016b) we proposed the use of the bilevel-free set:

$$S(\hat{y}) = \{(x, y) \in \mathbb{R}^n : d^T y > d^T \hat{y},\ Ax + B\hat{y} \leq b\} \tag{21}$$

defined for an arbitrary point $\hat{y} \in \mathbb{R}^{n_2}$ that satisfies (10) and (12).

In order to derive a valid IC from this set, one has to make sure that its interior does not contain any bilevel-feasible solution. To this end, the following assumption plays an important role:

Assumption 1. *$Ax + By - b$ are integer for all HPR solutions (x, y).*

Under Assumption 1, one is able to extend the set $S(\hat{y})$ by "moving apart" its facets by a non-negligible amount, resulting into the following *extended polyhedron* (let $\mathbf{1} = (1, \cdots, 1)$ denote a vector of all ones):

$$S^+(\hat{y}) = \{(x, y) \in \mathbb{R}^n : d^T y \geq d^T \hat{y},\ Ax + B\hat{y} \leq b + \mathbf{1}\} \tag{22}$$

that can safely be used to derive ICs.

3.1 Alternative Bilevel-Free Polyhedra

As indicated above, by varying the definition of the bilevel-free convex set, different families of valid ICs can be derived. Given the weak bounds of the HPR relaxation, the broader the family of ICs the better the performance of the underlying B&C solver. This was the main motivation for us to focus in this work on

alternative ways for deriving and extending bilevel-free polyhedra. We also show possible ways of enlarging these convex sets that may result in much stronger ICs.

With a little abuse of notation, in what follows we will call "facet" an inequality appearing in the outer description of a polyhedron. The bilevel-free polyhedron in Theorem 3 below was introduced by Xu (2012), where it has been used to determine branching rules in a B&B setting—while we use it derive hopefully deep ICs. As our theorem is stated in a slightly modified form, and for the sake of completeness, we also provide a short proof.

Theorem 3. *(Xu 2012) For any $\Delta\hat{y} \in \mathbb{R}^{n_2}$ such that $d^T\Delta\hat{y} < 0$ and $\Delta\hat{y}_j$ integer for all $j \in J_y$, the following polyhedron*

$$X(\Delta\hat{y}) = \{(x, y) \in \mathbb{R}^n : Ax + By + B\Delta\hat{y} \leq b,\ l \leq y + \Delta\hat{y} \leq u\} \tag{23}$$

does not contain any bilevel feasible point (not even on its frontier).

Proof. Assume by contradiction that (x, y) is bilevel feasible and that it belongs to $X(\Delta\hat{y})$. Then the point $(x, y + \Delta\hat{y})$ is a feasible solution of the follower with a strictly smaller objective value, i.e., $\Phi(x) \leq d^T(y + \Delta\hat{y}) < d^T y$, hence (x, y) does not satisfy (13) and cannot be bilevel feasible. □

Note that, contrarily to what happens with set $S(\hat{y})$ defined in (21), some facets of $X(\Delta\hat{y})$ correspond to bound constraints on the y variables.

Also in case of this polyhedron, it may happen that a bilevel-infeasible HPR solution (x^*, y^*) to be cut-off does not belong to its *interior*. Therefore, we need to extend this set as follows:

Theorem 4. *Under Assumption 1, for any $\Delta\hat{y} \in \mathbb{R}^{n_2}$ such that $d^T\Delta\hat{y} < 0$ and $\Delta\hat{y}_j$ integer for all $j \in J_y$, the following polyhedron does not contain any bilevel feasible point in its interior.*

$$X^+(\Delta\hat{y}) = \{(x, y) \in \mathbb{R}^n : \quad Ax + By + B\Delta\hat{y} \leq b + \mathbf{1},\ l - \mathbf{1} \leq y + \Delta\hat{y} \leq u + \mathbf{1}\} \tag{24}$$

Proof. To be in the interior of $X^+(\Delta\hat{y})$, a bilevel feasible (x, y) should satisfy $Ax + By + B\Delta\hat{y} < b + \mathbf{1}$ and $l - \mathbf{1} < y + \Delta\hat{y} < u + \mathbf{1}$. Because of Assumption 1, the latter condition can be replaced by $Ax + By + B\Delta\hat{y} \leq b$ and $l \leq y + \Delta\hat{y} \leq u$, hence the claim follows from Theorem 3. □

Observe that the extended bilevel-free polyhedra $X^+(\Delta\hat{y})$ and $S^+(\hat{y})$ are not directly comparable, i.e., typically it is not possible to find a $\hat{y}$ and a corresponding $\Delta\hat{y}$ such that one of the two polyhedra is a proper subset of the other—meaning that they are both of interest in our context; see Figure 1 for an illustration. Indeed, the facets of $X^+(\Delta\hat{y})$ span the whole (x, y) space, while those $S^+(\hat{y})$ only span the y space ($d^T y \geq d^T\hat{y}$) or the x space ($Ax + B\hat{y} \leq b + \mathbf{1}$). In addition, $S^+(\hat{y})$ contains the facet $d^T y \geq d^T\hat{y}$ that directly involves the follower objective function, while the role of the latter function in $X^+(\Delta\hat{y})$ is just implicit. As a matter of fact, the computational experience reported in Section 6 shows that there is no dominance between the two polyhedra in terms of IC quality.

3.2 Enlarged Bilevel-Free Polyhedra

The choice of the bilevel-free polyhedron is crucial for the computational effectiveness of the derived IC, the larger the polyhedron the better. The following arguments can be applied to any bilevel-free polyhedron to remove as many facets as possible from it, thus enlarging it and producing deeper cuts.

Theorem 5. *(Fischetti et al. 2016b) Let $S = \{(x, y) \in \mathbb{R}^n : \alpha_i^T x + \beta_i^T y \leq \gamma_i,\ i = 1, \ldots, k\}$ be any polyhedron not containing bilevel-feasible points in its interior. Then one can remove from S all its facets $i \in \{1, \ldots, k\}$ such that the half-space $\{(x, y) \in \mathbb{R}^n : \alpha_i^T x + \beta_i^T y \geq \gamma_i\}$ does not contain any bilevel-feasible solution.*

Concerning the set $S^+(\hat{y})$ defined by (22), the above property can possibly be used to remove the facet $d^T y \geq d^T\hat{y}$ if a valid lower bound of the follower problem, say FLB, is known. Indeed, if condition $d^T\hat{y} < FLB$ holds, then the facet $d^T y \geq d^T\hat{y}$ can be removed from $S^+(\hat{y})$. This property is particularly important and efficient for "zero-sum" bilevel instances where the leader and follower objective functions

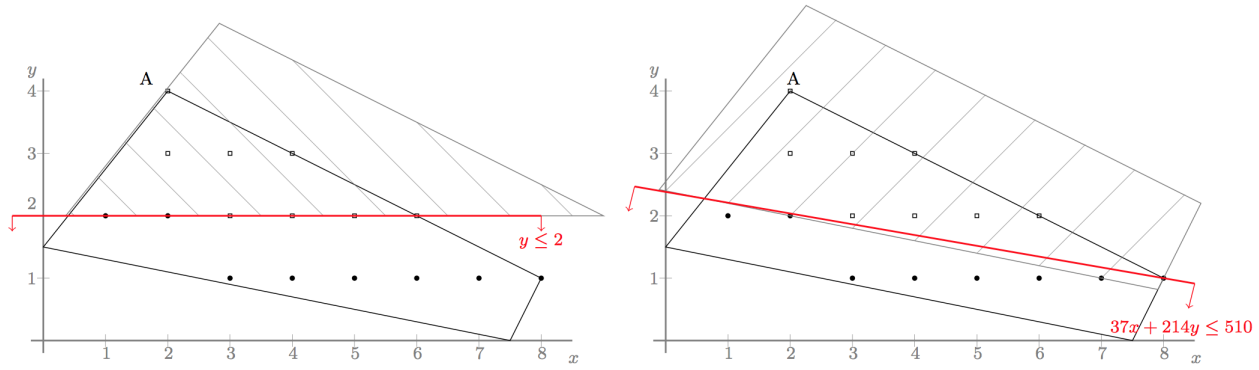


Figure 1: Intersection cuts for a notorious example from Moore and Bard (1990): the LP vertex A is cut by an IC (in red) obtained from the bilevel-free polyhedron $S^+(\hat{y})$ (left) or $X^+(\Delta\hat{y})$ (right), drawn in shaded line. No dominance exists between the two cuts.

satisfy $c_x = 0$ and $c_y = -d$ in (1). In that case, at every B&B node, the incumbent value z^* (say) is an upper bound for the leader objective function, hence $FLB = -z^*$ is a lower bound for the follower one.

Theorem 5 implies the following result:

Corollary 1. *(Fischetti et al. 2016b) Let $S = \{(x, y) \in \mathbb{R}^n : \alpha_i^T x + \beta_i^T y \le \gamma_i,\ i = 1, \dots, k\}$ be any polyhedron not containing bilevel-feasible points in its interior. Then one can remove from S all its facets $i \in \{1, \dots, k\}$ such that*

$$\sum_{j=1}^{n_1} \max\{\alpha_{ij} x_j^+, \alpha_{ij} x_j^-\} + \sum_{j=1}^{n_2} \max\{\beta_{ij} y_j^+, \beta_{ij} y_j^-\} < \gamma_i. \tag{25}$$

The above condition is exploited in Section 4.1 to derive large bilevel-free sets by solving an auxiliary MILP that maximizes the number of facets satisfying it.

Concerning the extended bilevel-free polyhedron $X^+(\Delta\hat{y})$, Theorem 5 implies that, for a given $\Delta\hat{y}$, detection of redundant facets is immediate:

Corollary 2. *Let $X^+(\Delta\hat{y})$ be the bilevel-free polyhedron of Theorem 4. Then one can remove from its definition all the facets $(Ax + By + B\Delta\hat{y})_i \le b_i + 1$ such that $(B\Delta\hat{y})_i \le 0$, along with all its facets $(y + \Delta\hat{y})_i \le u_i + 1$ with $\Delta\hat{y}_i \le 0$, and all its facets $l_i - 1 \le (y + \Delta\hat{y})_i$ with $\Delta\hat{y}_i \ge 0$.*

We observe that the above condition was also exploited in (Xu 2012) to reduce the number of children of each B&B node.

3.3 Hypercube Intersection Cuts

We finally present a very simple polyhedron (actually, a hypercube) that can be used to generate ICs even when Assumption 1 does not hold.

Theorem 6. *Assume $J_F \subseteq J_x$. For any given HPR solution (x^*, y^*), let $(\hat{x}, \hat{y})$ be an optimal bilevel-feasible solution with $\hat{x}_j = x_j^*$ for all $j \in J_F$ (if any), computed as follows:*

1. *solve the follower MILP (14) for $x = x^*$ to compute $\Phi(x^*)$;*
2. *build a restricted HPR by adding the following constraints to HPR: $x_j = x_j^*$ for all $j \in J_F$, and $d^T y \le \Phi(x^*)$;*

3. solve the restricted HPR, and let $(\hat{x}, \hat{y})$ be the optimal solution found (if any).

Then the following hypercube does not contain any bilevel-feasible solution (or any bilevel-feasible solution strictly better than $(\hat{x}, \hat{y})$, if the latter is defined) in its interior.

$$HC^+(x^*) = \{(x, y) \in \mathbb{R}^n : x_j^* - 1 \leq x_j \leq x_j^* + 1,\ \forall j \in J_F\} \tag{26}$$

Proof. Observe that the interior of $HC^+(x^*)$ only contains bilevel-feasible solutions (x, y) with $x_j = x_j^* = \hat{x}_j$ for all $j \in J_F$. By construction, among these solutions, $(\hat{x}, \hat{y})$ is a best bilevel-feasible one, hence the claim. □

Compared to the other two bilevel-free polyhedra, namely $S^+(\hat{y})$ and $X^+(\Delta\hat{y})$, this set spans only the x space. Hypercube ICs play an important role for two classes of MIBLPs for which $S^+(\hat{y})$ and $X^+(\Delta\hat{y})$ are not valid:

- if the follower subproblem is purely continuous (namely, for $J_y = \emptyset$), or
- if Assumption 1 does not hold.

In both cases, valid hypercube intersection cuts can still be derived from the polyhedron defined by (26).

4 Separation Algorithms

We now address the question of how to cut a given (integer or fractional) vertex (x^*, y^*) of $\overline{\text{HPR}}$ by using an IC. Given for granted that we use the cone associated with the current LP basis, as explained in Section 3, what remains is the choice of the bilevel-free set to be used.

Note that we accept that our separation fails in case (x^*, y^*) is not a feasible HPR solution, i.e., when (x^*, y^*) do not satisfy the integrality requirements (11)–(12). Indeed, in that case the IC is not instrumental for the correctness of our B&C, that is able to handle HPR infeasibility by standard MILP tools, namely, by cutting planes or branching.

4.1 Separation of Intersection Cuts

For hypercube ICs, the choice of the bilevel-free set is uniquely defined by the coordinates of the point (x^*, y^*). However, this is not the case for the sets $S(\hat{y})$, $S^+(\hat{y})$, $X(\Delta\hat{y})$, and $X^+(\Delta\hat{y})$ defined above. Given that for deriving a violated IC the point (x^*, y^*) has to belong to the interior of the bilevel-free set, we focus on the creation of the extended polyhedra $S^+(\hat{y})$ and $X^+(\Delta\hat{y})$. Recall that these extended polyhedra are well-defined only under Assumption 1. Without loss of generality, we also assume that (possibly after scaling) d is an integer vector so that the bilevel-infeasibility conditions of the type $d^T y < d^T y^*$ can be replaced by $d^T y \leq d^T y^* - 1$.

Bilevel-Free Polyhedron $S^+(\hat{y})$.

Given x^*, two options to define $S^+(\hat{y})$ have been proposed in (Fischetti et al. 2016b), along with the MILP models for their detection. For the sake of completeness, we briefly recall here the basic ideas behind these bilevel-free sets.

- **SEP1:** For the given $\overline{\text{HPR}}$ vertex (x^*, y^*), one solves the follower MILP to obtain its optimal solution $\hat{y}$, which eventually determines the set $S^+(\hat{y})$ according to (22). This approach aims at maximizing the distance of (x^*, y^*) from the facet $d^T y \geq d^T \hat{y}$ of $S(\hat{y})$. By construction, the point (x^*, y^*) belongs to $S(\hat{y})$, and hence to the interior of $S^+(\hat{y})$.
- **SEP2:** This is an alternative procedure that constructs $\hat{y}$ so as to have a large number of "removable facets" according to Corollary 1, i.e., of facets of the type $(Ax + B\hat{y})_i \leq b_i + 1$ with

$$\sum_{j \in N_x} \max\{A_{ij} x_j^-, A_{ij} x_j^+\} + (B\hat{y})_i \leq b_i. \tag{27}$$

When designing a separation MILP to choose such a $\hat{y}$, one has to ensure that $S^+(\hat{y})$ does not contain bilevel-feasible points in its interior, whereas (x^*, y^*) does, and that the number of removable facets is maximized.

Bilevel-Free Polyhedron $X^+(\Delta\hat{y})$.

We now address the bilevel-free polyhedron $X^+(\Delta\hat{y})$ defined in Theorem 4. Following the recipe of Xu (2012), for the given (x^*, y^*), $\Delta\hat{y}$ is defined by solving an additional "'scoop" MILP intended to produce a large number of removable facets in accordance with Corollary 2. To simplify notation, let the system

$$\tilde{A}x + \tilde{B}y \leq \tilde{b}$$

contain all the follower constraints (including the bounds on the y variables) except integrality, and let $\tilde{m}$ denote the number of rows of $\tilde{A}$. The scoop MILP then reads:

$$\textbf{SCOOP}: \quad \Delta\hat{y} \in \arg\min \sum_{i=1}^{\tilde{m}} t_i \tag{28}$$

$$d^T \Delta y \leq -1 \tag{29}$$

$$\tilde{B}\Delta y \leq \tilde{b} - \tilde{A}x^* - \tilde{B}y^* \tag{30}$$

$$\Delta y_j \text{ integer}, \qquad \forall j \in J_y \tag{31}$$

$$\tilde{B}\Delta y \leq t \text{ and } t \geq 0. \tag{32}$$

In model above, each continuous variable t_i has value 0 in case $(\tilde{B}\Delta y)_i \leq 0$, meaning that $(\tilde{A}x^* + \tilde{B}y^* + \tilde{B}\Delta\hat{y})_i \leq \tilde{b}_i + 1$ is a removable facet according to Corollary 2. On the contrary, if $(\tilde{B}\Delta y)_i > 0$, variable t_i measures the slack of solution (x^*, y^*) with respect to constraint i. So, the objective function in (28) goes into the direction of maximizing the size of the bilevel-feasible set associated with $\Delta\hat{y}$; see Xu (2012) for more details. Note that one is not allowed to increase by 1 the right-hand side of (30) as this would allow the point (x^*, y^*) to be on the frontier of the extended polyhedron $X^+(\Delta\hat{y})$.

4.2 Numerically Safe Intersection Cuts

In this section we describe how ICs can be derived from the optimal LP basis in a numerically reliable way. We follow the "disjunctive interpretation" of ICs (Glover 1974, Glover and Klingman 1976), which also reflects our actual implementation. This is a slightly more general—and numerically more stable—variant of those typically considered in the literature.

To ease exposition, in the remaining part of this subsection we will denote by $\xi = (x, y) \in \mathbb{R}^n$ the whole variable vector, while the $\overline{\text{HPR}}$ at the given B&B node will be formulated in its standard form as

$$\min\{\hat{c}^T\xi : \hat{A}\xi = \hat{b}, \xi \geq 0\}.$$

Now let ξ^* be an optimal vertex of the above LP, associated with a certain basis $\hat{B}$ (say) of $\hat{A}$, and let the bilevel-free polyhedron S of interest be defined as

$$S = \{\xi : g_i^T\xi \leq g_{0i},\ i = 1, \ldots, k\}.$$

Our order of business is to derive a valid inequality, violated by ξ^*, from the feasibility condition "ξ cannot belong to the interior of S". To this end, we observe that the latter condition can be restated as the following k-term disjunction:

$$\bigvee_{i=1}^{k} (g_i^T\xi \geq g_{i0}) \tag{33}$$

Algorithm 1: Intersection cut separation

Input : An LP vertex ξ^* along with its a associated LP basis $\hat{B}$;
the feasible-free polyhedron $S = \{\xi : g_i^T \xi \leq g_{0i},\ i = 1, \ldots, k\}$ and the associated valid disjunction $\bigvee_{i=1}^{k} (g_i^T \xi \geq g_{i0})$ whose members are violated by ξ^*;

Output: A valid intersection cut violated by ξ^*;

1 **for** $i := 1$ **to** k **do**
2 $\quad (\overline{g}_i^T, \overline{g}_{i0}) := (g_i^T, g_{i0}) - u_i^T(\hat{A}, \hat{b})$, where $u_i^T = (g_i)_{\hat{B}}^T \hat{B}^{-1}$
3 **end**
4 **for** $j := 1$ **to** n **do** $\gamma_j := \max\{\frac{\overline{g}_{ij}}{\overline{g}_{i0}} : i \in \{1, \ldots, k\}\}$;
5 **if** $\gamma \geq 0$ **then**
6 $\quad$ **for** $j := 1$ **to** n **do**
7 $\quad\quad$ **if** ξ_j *is integer constrained* **then** $\gamma_j := \min\{\gamma_j, 1\}$;
8 $\quad$ **end**
9 **end**
10 **return** *the violated cut* $\gamma^T \xi \geq 1$

where we write $\geq$ instead of $>$, as a feasible ξ can in fact belong to the frontier of S.

ICs are then obtained as in Algorithm 1. At Step 2, each violated inequality $g_i^T \xi \geq g_{i0}$ is restated in its equivalent *reduced form* $\overline{g}_i^T \xi \geq \overline{g}_{i0}$ where all basic variables ξ_j's are projected away—this is the standard operation that is applied to the objective function when computing LP reduced costs. Observe that $\overline{g}_{i0} > 0$ as the inequality is violated by ξ^* whereas, by construction, $\overline{g}_i^T \xi^* = 0$. Therefore, one can normalize it to $\frac{1}{\overline{g}_{i0}} \overline{g}_i^T \xi \geq 1$ to get a same right-hand side of 1 for all inequalities.

Step 4 relaxes each inequality so as to get a same left-hand-side coefficient for each variable in all inequalities (through the max operation), meaning that the resulting relaxed cut $\gamma^T \xi \geq 1$ is valid for each term of the disjunction and hence for the overall problem. As the max operation does not change the coefficient of each basic variable ξ_j (which is zero in all reduced-form inequalities), the resulting cut is still violated by 1 by ξ^*.

Finally, at Step 4 a simple "coefficient clipping" operation is applied in case $\gamma \geq 0$, that consists of replacing γ_j by $\min\{\gamma_j, 1\}$ whenever ξ_j is an integer-constrained variable. Validity of the resulting inequality follows from the fact that either $\xi_j = 0$ (in which case its coefficient γ_j is immaterial) or $\xi_j \geq 1$ (hence validity follows from assumptions $\gamma \geq 0$ and $\xi \geq 0$).

It is important to observe that the validity of the final IC does *not* require the vector $u_i^T = (g_i)_{\hat{B}}^T \hat{B}^{-1}$ used at Step 2 be computed with a very high numerical accuracy. This property is very important for the numerical stability of the method—numerical issues in computing u_i^T can reduce cut violation as some basic variables can have a small nonzero coefficient, but they do not have an impact on validity.

We conclude this subsection by warning the reader about the fact that, in practical cases, the LP contains bounds on the variables that are treated implicitly by the LP solver—while the above theory refers to an LP in standard form with unbounded nonnegative variables. This means that, before generating the IC, a preprocessing is required that (1) adds explicit slacks to inequality constraints, (2) complements all nonbasic variables at their upper bound, and (3) shifts all variables with nonzero lower bound. Needless to say, the final IC needs to be post-processed to undo all the above complement-and-shift operations above and to bring the cut back in the original variable space, projecting all slack variables away.

5 Implementation

In this section we describe some implementation details that play an important role in the design of an effective code. Indeed, we strongly believe that these details are worth addressing for the sake of result

reproducibility.

Our description is based on the actual MILP solver we used (IBM ILOG Cplex 12.6.3), but it extends easily to other solvers.

5.1 Branch-and-Cut scheme

The basic MILP model on which B&C is applied is HPR. Cplex's preprocessing is disabled on this model, as we need to retrieve LP bases at the various nodes (to derive ICs) and preprocessing would change the variable space by aggregating/changing variables and constraints, requiring cumbersome (if not impossible) bookkeeping mechanisms. Instead, we do apply our bilevel-specific preprocessing variable fixing described in Theorem 2.

Internal numerical-precision thresholds for integrality/cut validity tests are set to a very small value (10^{-9}) so as to guarantee a very precise overall computation.

Multi-threading opportunistic parallel mode is selected when solving HPR, so as to fully exploit the architecture in use. For thread safety, each thread works on its own copy of the follower MILP (needed to check bilevel feasibility at B&B nodes).

Internal Cplex's cuts are active in their default setting (level 0), as our approach does not require to deactivate them—contrarily to, e.g., DeNegre (2011). Similarly, our approach could handle correctly solutions produced by Cplex's internal heuristics (as shown below). Note however that each solution found by internal Cplex's heuristics requires a significant extra computation effort to determine its bilevel feasibility. Thus, in our experiments, we deactivated all Cplex's heuristics.

We implemented a bilevel-oriented branching strategy by using Cplex's branching priorities. To be specific, priority for integer-constrained variables is set to 2 (maximum) for all x_j's with $j \in J_x \cap J_F$ (i.e., appearing in the follower MILP), to 1 for x_j's with $j \in J_x \setminus J_F$, and to 0 for all other variables (i.e., for all y_j's with $j \in J_y$). Within the same priority level, Cplex is left free to choose the branching variable according to its internal "strong branching" criterion.

Each time a new integer solution (x^*, y^*) is found and is going to update the incumbent, a specific "lazyconstraint callback" function is automatically invoked by Cplex to let the user possibly discard this solution for whatever reason, and possibly add a cut that makes this solution infeasible. In this callback function, we first determine whether the solution is in fact bilevel-feasible by solving the MILP follower for $x = x^*$, thus obtaining an alternative point $(x^*, \hat{y})$ with $d^T\hat{y} = \Phi(x^*)$. If $d^T y^* > \Phi(x^*)$, the solution (x^*, y^*) is not bilevel feasible and we cut it using a suitable IC. In case heuristics were enabled and produced an infeasible candidate solution without an associated LP basis, we could just discard it. In any case, point $(x^*, \hat{y})$ is passed to an hoc-hoc feasibility-check procedure that quickly verifies its HPR feasibility (with respect to (2)) and cost, and possibly updates the incumbent. This approach is very useful as it typically produces very good heuristic solutions at the very beginning of the computation.

Immediately before branching, at each node Cplex automatically invokes a "usercut callback" function where the user can generate problem-specific (in our case, locally valid) cuts for fractional $\overline{\text{HPR}}$ solutions. Within this function, we implemented separation procedures for both FUB cuts and ICs, as described in Sections 2.1 and 4.1, respectively. Separation of FUB cuts is invoked only at the first callback call at a given node, as it does not depend on (x^*, y^*) but only on the variable bounds at the current node. Separation of ICs is instead active at each callback call, with a maximum number (say, `max_node_cuts`) of consecutive calls at the same B&B node. This is due to two considerations: first, separation procedures can be time-consuming, as they require the solution of an auxiliary MILP. Second, a known issue of ICs is that their effectiveness quickly deteriorate when applied iteratively to a same LP. In any case, in our implementation, we discard ICs with a too large dynamic (ratio between the largest and smallest nonzero coefficient, both in absolute value, greater than 10^6) as they are not considered numerically reliable. A cut is also discarded when its relative violation is very small, i.e., in case violation is small than $10^{-6} \cdot (|cut\_rhs| + 1)$.

6 Computational Results

To evaluate the performance of our improved B&C solution method, we implemented it (in C language) and run it on a large set of instances from the literature. All computational experiments are conducted on an Intel Xeon E5-2670v2 with 2.5GHz and 12GB of RAM. Computing times reported in what follows are in wall-clock seconds and refer to 4-thread runs. The time limit for each run was set to 3 600 wall-clock seconds.

6.1 Testbed

Table 1 summarizes details about the data sets that have been considered in our computational study. The considered instance sets can be divided into two groups: *general bilevel*, and *interdiction*.

While the former includes problems with no special structure, interdiction problems represent a relevant case of bilevel programs where the follower typically is a clean combinatorial optimization problem (e.g., a knapsack or an assignment problem). The leader can "interdict" a subset of elements of the follower problem (e.g., knapsack items, or edges in the assignment problem), subject to a given budget constraint. The objective of the leader is the opposite of that of the follower, which results into min-max or max-min problems. Note that, by exploiting their special structure, one can design sound methods for general interdiction problems (DeNegre 2011, Tang et al. 2015). In our computational study, instead, we treat interdiction problems as standard bilevel problems, without taking advantage of their structure.

The total number of general bilevel instances considered is 247, and the total number of interdiction instances is 567. Thus, with experiments conducted on more than 800 instances of various types and from different sources, our computational study is by far the most extensive ever reported in the MIBLP literature.

General Bilevel Instances

- Instances of class `DENEGRE` have been proposed in DeNegre (2011). They involve $n_1 \in \{5, 10, 15\}$ integer leader variables, while the number of integer follower variables n_2 is such that $n_1+n_2 = 15$ or 20. There are $m = 20$ follower constraints and no constraints at the leader level. All coefficients are integers in the range $[-50, 50]$.

- Class `MIPLIB` has been introduced in Fischetti et al. (2016b). They are based on instances of MILPLIB 3.0 (Bixby et al. 1998) containing only binary variables. These instances have been transformed into bilevel problems by considering the first $Y\%$ (rounded up) variables as follower variables, with $Y \in \{10, 50, 90\}$ and the remaining ones as leader variables. The objective function is used as the leader objective $c_x^T x + c_y^T y$ and the follower objective is set to $d^T y = -c_y^T y$. All constraints in the instances are defined to be follower constraints. The class contains 57 instances with up to about 80 000 HPR variables and 5 000 follower constraints, making them much larger (and often also much more difficult) than instances of the other classes.

- Class `XUWANG` has been proposed by Xu and Wang (2014). In these instances, we have $n_1 = n_2 \in \{1, 60, 110, \ldots, 460\}$. The leader variables are constrained to be integer, while some of the follower variables are continuous. The number of leader constraints, as well as follower constraints, is $0.4\, n_1$. All input values are integers uniformly distributed from given ranges: G_x, G_y, A, B are in $\{0, 10\}$, c_x, c_y, d are in $\{-50, 50\}$, q is in $\{30, 130\}$, and b is in $\{10, 110\}$. There are ten instances for every value of n_1. For $n_1 \in \{110, 160\}$, four additional sets of instances were created: q is increased by 10 in the first two sets, while b in increased by 10 in the last two sets. Note that all these instances have some continuous follower variables, hence Assumption (1) does not hold and only the hypercube ICs of Section 3.3 can be used by our B&C solver.

Interdiction Instances

- Class `INTER-KP` has been introduced in DeNegre (2011). The follower problem is a knapsack problem. The instances are based on bicriteria knapsack instances from the *multiple criteria decision making*

library: the first objective of the bicriteria problem is used to the define the follower objective function, while the second objective defines the interdiction budget constraint of the leader. The instances have $n_1 = n_2 \in \{10, 20, \dots 50\}$, with two additional sets with 11 and 12 items. The interdiction budget of an instance is $\lceil \sum_{i=1}^{n_1} a_i/2 \rceil$, where a_i is the cost of interdicting item i. For every number of items there are 20 instances, except for 10 items where there are 40 instances.

- Class `INTER-KP2` has been proposed by Tang et al. (2015) and also consists of knapsack interdiction instances. Instances with number of items $n_1 = n_2 \in \{20, 22, 25, 28, 30\}$ have been constructed and the interdiction budget is a cardinality constraint allowing k (say) items to be interdicted. For each value of n_1, three different values of k have been considered and, for each (n_1, k) pair, ten instances have been defined generating item weights and profits as random integers in $[1, 100]$.

- Instances of class `INTER-ASSIG` have also been introduced in DeNegre (2011). The follower problem is an assignment problem. The instances are derived from bicriteria assignment instances from the *multiple criteria decision making library* in a similar way as in class `INTER-KP`. Each instance has 25 edges (i.e., $n_1 = n_2 = 25$) and 20 follower constraints—plus the interdiction constraints stipulating that interdicted edges cannot be used by the follower. The interdiction budget was chosen as a percentage of the sum of interdiction costs.

- Class `INTER-RANDOM` are random interdiction problems proposed by DeNegre (2011). They are based on random ILPs with integer coefficients in $[-50, 50]$ of size (rows $\times$ columns) 10×10, 15×10, 20×20, and 25×20. These ILPs are transformed to interdiction instances by introducing interdiction costs for each variable. Two strategies are considered: in the first strategy, a unit cost is given, together with an interdiction budget of 3. In the second one, interdiction costs and budget are taken randomly. Instances obtained by applying the first and second strategy are denoted as *symmetric* and *asymmetric*, respectively. For each strategy, 10 instances are generated for each size.

- Class `INTER-CLIQUE` are clique interdiction problems introduced in Tang et al. (2015). The follower problem is a maximum cardinality clique problem, and the leader can interdict edges. Graphs with $\nu = \{8, 10, 12, 15\}$ nodes and edge densities $d \in \{0.7, 0.9\}$ have been used to define the instances. The interdiction budget is a cardinality constraint, and the leader can interdict at most $\lceil m/4 \rceil$ edges, where m is the number of edges in the graph. For each combination of ν and d, ten instances have been constructed, resulting in 80 instances of this class.

- Instances of class `INTER-FIRE` have been recently introduced in Baggio et al. (2016). These are randomly generated problems arising from a trilevel context in which one has to defend a given network against possible cascade failures or malicious viral attacks. This benchmark includes instances in which the network is a tree or a general graph. In both cases, the number of nodes is in {25, 50, 80}.

6.2 Computational Analysis of the Proposed Improvements

In this subsection we computationally analyze the effect of the newly presented ingredients on the performance of our bilevel B&C solver. We first consider FUB cuts, as described in Section 2.1, and then follower preprocessing (cf. Section 2.2). Afterwards, we investigate the influence of new bilevel-free sets introduced in Section 3. In this experiment, the following solver settings are considered:

- `SEP1`: our B&C solver using SEP1 formulation for separating ICs (see Section 4.1),
- `SEP2`: our B&C solver using SEP2 formulation for separating ICs (see Section 4.1),
- `XU`: our B&C solver with the extended bilevel-free set of Xu (2012) as described in Section 3.1
- `MIX`: combination of settings `SEP2` and `XU` (both ICs being separated at each separation call).

Table 1: Our testbed. The first three classes are of general bilevel type, and the latter five of interdiction type. Column #Inst reports the total number of instances in each class, column Type indicates whether the instances are binary (B), integer (I) or continuous (C), column #OptB gives the number of instances solved to optimality by previous approaches from literature, and #Opt gives the number of instances solved to optimality by the solver presented in this paper. The symbols in the column #OptB give the source of the previous best approach: † is (Fischetti et al. 2016b); + is (Xu and Wang 2014); ∗ is (Tang et al. 2015). Note that class `INTER-FIRE` has not been tested computationally before. For class `INTER-RANDOM` some results are reported in (DeNegre 2011), but **for slightly modified instances.**

Class	Source	Type	#Inst	#OptB	#Opt
`DENEGRE`	(DeNegre 2011),(Ralphs and Adams 2016)	I	50	$45^{\dagger}$	**50**
`MIPLIB`	(Fischetti et al. 2016b)	B	57	$20^{\dagger}$	27
`XUWANG`	(Xu and Wang 2014)	I,C	140	140^{+}	**140**
`INTER-KP`	(DeNegre 2011),(Ralphs and Adams 2016)	B	160	$79^{\dagger}$	138
`INTER-KP2`	(Tang et al. 2015)	B	150	$53^{\star}$	**150**
`INTER-ASSIG`	(DeNegre 2011),(Ralphs and Adams 2016)	B	25	$25^{\dagger}$	**25**
`INTER-RANDOM`	(DeNegre 2011),(Ralphs and Adams 2016)	B	80	-	**80**
`INTER-CLIQUE`	(Tang et al. 2015)	B	80	$10^{\star}$	**80**
`INTER-FIRE`	(Baggio et al. 2016)	B	72	-	**72**
total			814	372	762

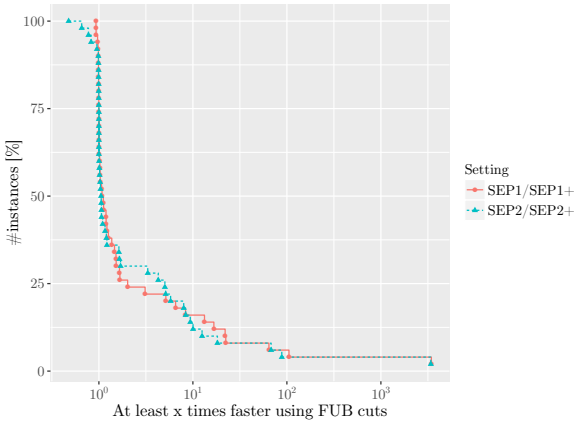
Observe that settings `SEP1` and `SEP2` have been proposed and analyzed by Fischetti et al. (2016b), where it has been shown that `SEP2` outperformed alternative approaches from the literature by a large margin. In the present article, they serve therefore as reference points for measuring the efficacy of the newly proposed features (notably: FUB cuts, preprocessing, and new ICs) exploited in our new proposals `XU` and `MIX`. For all settings, at most `max_node_cuts`=20 ICs are separated at each B&B node (including root). When one of the four settings is enhanced by FUB cuts, this is denoted by a "+" next to its name. If, in addition, follower preprocessing is turned on, this is denoted by "++".

In the following, we first analyze the effects of FUB cuts and the preprocessing, before turning these two features on for the remainder of the study.

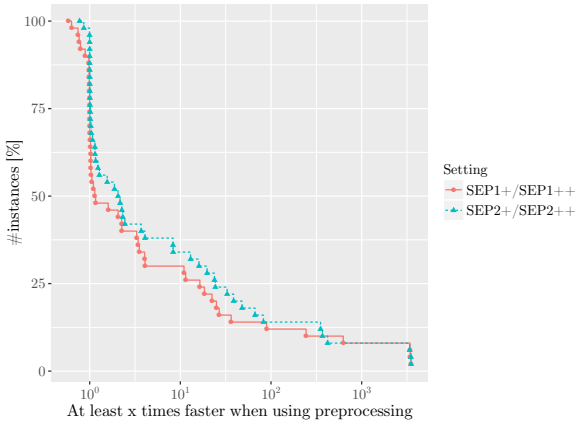
Effect of FUB Cuts.

We illustrate the importance of the FUB cuts introduced in Section 2.1 through the cumulative speed-up chart of Figure 2a. The chart shows the speed-up values for the two state-of-the-art approaches from Fischetti et al. (2016b), namely, settings `SEP1` and `SEP2`. To see the effect of FUB cuts, the follower preprocessing is turned off. The reported speed-up ratio is calculated as $\frac{t_s+t(\texttt{SEP}x)}{t_s+t(\texttt{SEP}x+)}$, where $t(\texttt{SEP}x+)$ and $t(\texttt{SEP}x)$ denote the computing time (in seconds) of a setting `SEP`x with and without FUB cuts, respectively. The time shift t_s is set to 1 second to reduce the importance of instances that are too easy in the comparison. For a given setting, each point (x, y) in this chart indicates that y% of all instances have a speed-up ratio of at least x. Notice that the values on x axis are given in log-scale. We observe that for both `SEP1` and `SEP2` a significant speed-up is achieved in the considered dataset (`DENEGRE`). In some cases, a speed-up of 2-3 orders of magnitude could be reported, thanks to the usage of FUB cuts. In very few cases, a small slow-down is observed—this usually happens for instances that could be solved within a few seconds, in which case turning FUB cuts on causes an unnecessary overhead.

Figure 2: Speed-ups achieved by FUB cuts and follower preprocessing for the instance set DENEGRE.



(a) Using FUB cuts.

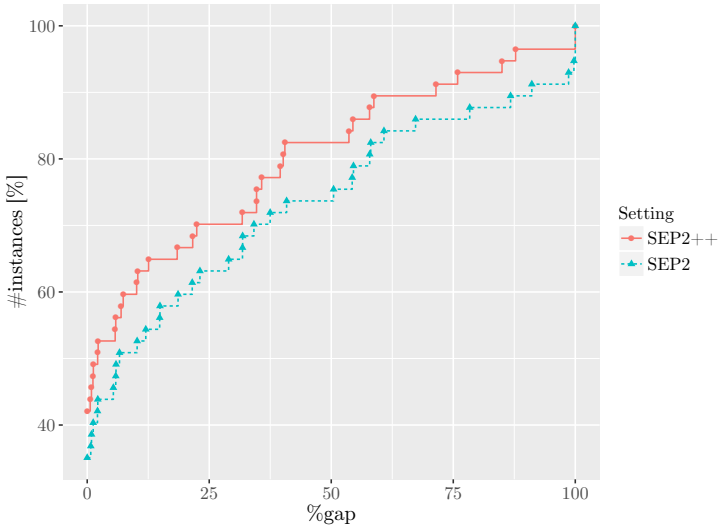


(b) Using follower preprocessing.

Effect of Follower Preprocessing.

We now demonstrate the effect of preprocessing, by continuing the above experiment: using the same set of DENEGRE instances and settings SEP1 and SEP2, we keep the FUB cuts turned on, and consider the additional speed-up that can be achieved by turning follower preprocessing on. The speed-up ratio is now calculated as $\frac{t_s+t(\texttt{SEP}x+)}{t_s+t(\texttt{SEP}x++)}$, where $t(\texttt{SEP}x++)$ and $t(\texttt{SEP}x+)$ denote the computing time (in seconds) of a setting $\texttt{SEP}x+$ with and without the follower preprocessing, respectively. The cumulative chart reporting this speed-up ratio is given in Figure 2b, where similar effects as for FUB cuts are observed: due to preprocessing, a significant number of instances could be solved within a much shorter computing time. Indeed, for more than 25% of all instances we report a speed-up of more than one order of magnitude, and in some cases the speed-up is even larger than three orders of magnitude.

Figure 3: Final gaps for settings SEP2 and SEP2++ (the latter with FUB cuts and preprocessing) for instance set MIPLIB, obtained when the time-limit of one hour is reached.



Speed-ups for other classes of benchmark instances have been also achieved, although not that large as

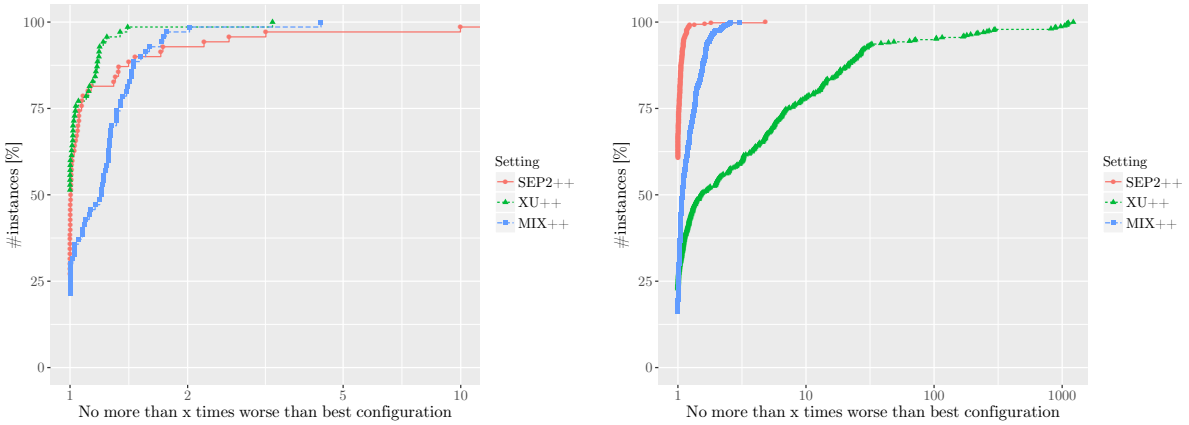
for DENEGRE, where all instances could now easily be solved to optimality (which was not the case before this work).

We next illustrate the improvement in performance of SEP2 when combined with both FUB cuts and follower preprocessing, on the most challenging set of benchmark instances in our study, namely MIPLIB. Figure 3 shows the cumulative chart for final gaps obtained after the time-limit of one hour for SEP2 and SEP2++. For each setting, each point (x, y) in this chart indicates that y% of all instances have a residual percentage gap not larger than x. In particular, the left-most point indicates the percentage of instances solved to optimality for each of the settings. These results show that a significant performance improvement could be reported: SEP2 manages to solve about 34% of the instances to optimality, and this number increases to 42% thanks to the FUB cuts and preprocessing. Furthermore, for 65% of the instances the obtained final gap is below 12% in case of SEP2++, whereas this is true only for 55% of all instances in case of SEP2.

Effect of Different Intersection Cuts.

We next compare our new approach for separating intersection cuts based on the definition of the bilevel-free set given in (24) (denoted by XU++) against the recently proposed ICs based on bilevel-free sets defined by (22) (denoted by SEP2++). In addition, we consider a mixed strategy (denoted by MIX++), in which both ICs are generated for each separation call. To allow for a fair comparison, FUB cuts and preprocessing are turned on for all three settings (hence the ++ beside the setting names). Computational performance is compared using performance profiles (PPs). In Figure 4 we report two PPs: one for the class of *general bilevel* instances (left plot), and one for the class of *interdiction* instances (right plot). Both PPs refer to the subsets of instances that could be solved to optimality by all three settings within the given time-limit of one hour. PPs are constructed following the procedure of Dolan and Moré (2002): for each setting $s \in S$ and instance $p \in P$, a *performance ratio* $r_{p,s} = \frac{t_{p,s}}{\min_{s' \in S}\{t_{p,s'}\}}$ is calculated, where $t_{p,s}$ is the time setting s needs to solve instance p. In the profiles, the cumulative distribution function of the performance ratio $\rho_s(\tau) = \frac{100}{|P|}\big|\{p \in P : r_{ps} \leq \tau\}\big|$ is displayed for each setting $s \in S$. The leftmost point of the graph for a setting s shows the percentage of instances for which s is the fastest setting.

Figure 4: Effect of different ICs on the B&C performance. Performance profiles contain instances that could be solved to optimality by all three settings. FUB cuts and preprocessing are turned on.



(a) Instances of type *general bilevel*.

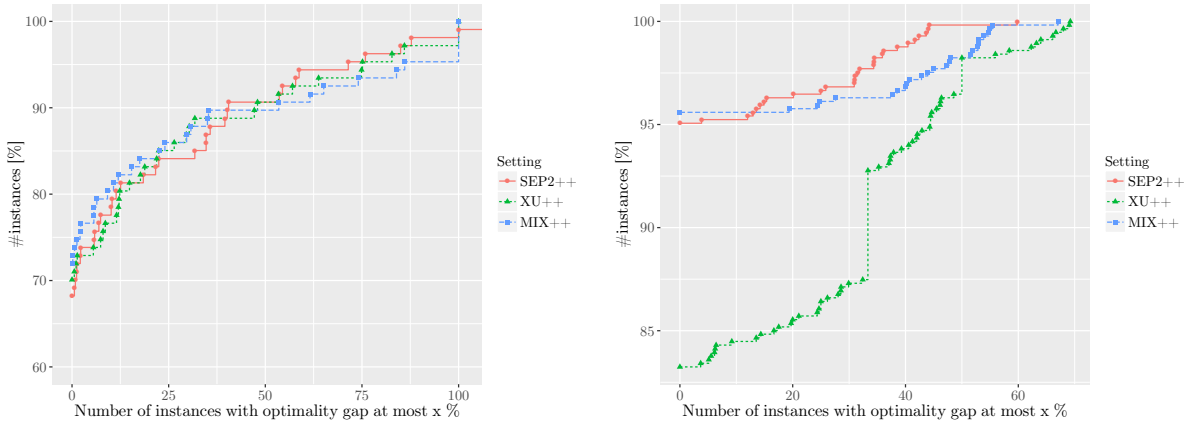
(b) Instances of type *interdiction*.

To compare the settings on the remaining instances (namely, those that were not solved by at least one of the three settings) we provide additional charts that report final gaps upon the termination of the B&C solver. Figures 5a and 5b report cumulative-gap charts for general bilevel and interdiction instances, respectively, and have the same interpretation as Figure 3.

Comparing the obtained results, we may conclude that the quality and structure of the incorporated ICs significantly influence the performance of the underlying bilevel B&C solver. It turns out that for two different instance classes, two different families of ICs seem to play a crucial role. More precisely, setting `XU++` performs better for the *general bilevel* class, while for *interdiction* instances `SEP2++` is the clear winner. A noteworthy aspect for the success of both `SEP2++` and `XU++` are the proposed FUB cuts and the preprocessing procedure.

Mixing the two families of ICs has merits in its own, but it also has some drawbacks. Indeed, `MIX++` is outperformed by `SEP2++` and `XU++` in most of the cases when the underlying instances could be solved in (relatively) short time. This is due to the unnecessary overhead of separating two ICs, when separating one of them already works very well. On the other hand, for difficult instances, `MIX++` provides the smallest final gaps, which is why we conclude that setting `MIX++` qualifies as a robust default setting for our B&C algorithm.

Figure 5: Cumulative chart for final gaps demonstrating the effect of different ICs on the B&C performance. FUB cuts and preprocessing are turned on.



(a) Instances of type *general bilevel*.

(b) Instances of type *interdiction*.

6.3 Comparison with Approaches from Literature

In this subsection we compare our default B&C code `MIX++` with other approaches from the literature, using the instance classes originally proposed by the authors of the alternative methods under comparison. **When performing this experiment, we refrained** from tuning our solver for the specific instance subclasses, even though this could produce significantly better results in many cases.

Not all instances listed in Table 1 are considered in this subsection, namely, instances from `INTER-ASSIG`, `INTER-FIRE` and `INTER-RANDOM` are left out. **Instances of class `INTER-ASSIG` are omitted for the sake of space, as they turn out to be very easy for both `MIX++` and the state-of-the-art solver for these problems—namely, `SEP2` from (Fischetti et al. 2016b). As to `INTER-FIRE` problems, no computational analysis has been carried out so far in the literature, while for `INTER-RANDOM` the only available results are from (DeNegre 2011), but for slightly modified instances.**

Instance Set `DENEGRE`.

For this class of 50 general bilevel instances introduced by DeNegre (2011), Table 2 compares the performance of two solvers: `SEP1` (in a variant with `max_node_cuts` $= 0$ that produces the best-known results for this class, as reported by Fischetti et al. (2016b)) and `MIX++` (our default B&C solver). For the former we report, for each instance, the value of the best lower and upper bounds (LB and BestSol, respectively), the associated

percentage gap (%gap), the computing time (in seconds), as well as the total number of branch-and-bound nodes. As MIX++ is able to solve to proven optimality all the instances, for this solver we only report the required computing time and number of nodes. The optimal solution value of each instance is given in column OPT. Prior to the present work, optimal solutions for five instances of this class were unknown.

We point out that with our new B&C solver, optimal solutions for all the 50 instances of this class are reported for the first time, and that the optimal solution for all but two instances is obtained within only a few seconds of computing time. Moreover, MIX++ needs much fewer branch-and-bound nodes than SEP1.

Table 2: Results for instance set DENEGRE.

			SEP1				MIX++	
instance	OPT	BestSol	LB	%gap	time [s]	nodes	time [s]	nodes
miblp-20-15-50-0110-10-10	-206	-206	-206.0000	0.00	0	628	0	10
miblp-20-15-50-0110-10-1	-388	-388	-388.0000	0.00	0	50	1	21
miblp-20-15-50-0110-10-2	-398	-398	-398.0000	0.00	17	77323	8	279
miblp-20-15-50-0110-10-3	-42	-42	-42.0000	0.00	0	2201	2	54
miblp-20-15-50-0110-10-4	-729	-729	-729.0000	0.00	0	185	1	56
miblp-20-15-50-0110-10-5	-281	-281	-281.0000	0.00	0	83	0	20
miblp-20-15-50-0110-10-6	-246	-246	-246.0000	0.00	0	233	16	205
miblp-20-15-50-0110-10-7	-260	-260	-260.0000	0.00	0	108	0	0
miblp-20-15-50-0110-10-8	-293	-293	-293.0000	0.00	0	114	0	22
miblp-20-15-50-0110-10-9	-635	-635	-635.0000	0.00	0	1061	1	16
miblp-20-20-50-0110-10-10	-441	-441	-441.0000	0.00	73	256927	2927	8068
miblp-20-20-50-0110-10-1	-359	-359	-359.0000	0.00	494	1805080	2	81
miblp-20-20-50-0110-10-2	-659	-659	-659.0000	0.00	0	939	1	17
miblp-20-20-50-0110-10-3	-618	-618	-618.0000	0.00	1	9456	0	52
miblp-20-20-50-0110-10-4	-604	-604	-695.0000	15.07	3600	7479668	1	51
miblp-20-20-50-0110-10-5	-1003	-1003	-1003.0000	0.00	0	20	0	13
miblp-20-20-50-0110-10-6	-731	-707	-858.3704	21.41	3600	6244669	10	511
miblp-20-20-50-0110-10-7	-683	-683	-683.0000	0.00	2788	7420465	0	54
miblp-20-20-50-0110-10-8	-667	-667	-667.0000	0.00	3	8116	15	232
miblp-20-20-50-0110-10-9	-256	-256	-256.0000	0.00	4	42945	0	71
miblp-20-20-50-0110-15-10	-251	-251	-251.0000	0.00	0	400	0	14
miblp-20-20-50-0110-15-1	-450	-420	-511.6744	21.83	3600	4313453	0	16
miblp-20-20-50-0110-15-2	-645	-645	-744.0000	15.35	3600	14175981	0	6
miblp-20-20-50-0110-15-3	-593	-593	-593.0000	0.00	838	1420792	3	43
miblp-20-20-50-0110-15-4	-441	-441	-487.6710	10.58	3600	5448638	5	131
miblp-20-20-50-0110-15-5	-379	-334	-518.0000	55.09	3600	6169959	1392	5466
miblp-20-20-50-0110-15-6	-596	-596	-596.0000	0.00	3260	5955753	50	483
miblp-20-20-50-0110-15-7	-471	-471	-471.0000	0.00	246	787848	0	23
miblp-20-20-50-0110-15-8	-370	-370	-838.3617	126.58	3600	11797237	0	3
miblp-20-20-50-0110-15-9	-584	-584	-584.0000	0.00	1	2027	0	8
miblp-20-20-50-0110-5-10	-340	-340	-340.0000	0.00	0	45	0	38
miblp-20-20-50-0110-5-11	-426	-426	-426.0000	0.00	0	9	0	11
miblp-20-20-50-0110-5-12	-854	-854	-854.0000	0.00	0	43	0	21
miblp-20-20-50-0110-5-13	-519	-519	-519.0000	0.00	116	947138	0	11
miblp-20-20-50-0110-5-14	-923	-923	-923.0000	0.00	0	109	0	58
miblp-20-20-50-0110-5-15	-617	-617	-617.0000	0.00	157	1031098	1	197
miblp-20-20-50-0110-5-16	-833	-833	-833.0000	0.00	0	2535	0	44
miblp-20-20-50-0110-5-17	-944	-944	-944.0000	0.00	0	3580	0	19
miblp-20-20-50-0110-5-18	-386	-386	-386.0000	0.00	0	2	0	0
miblp-20-20-50-0110-5-19	-431	-431	-431.0000	0.00	3	25762	0	95
miblp-20-20-50-0110-5-1	-548	-548	-548.0000	0.00	1	6981	0	21
miblp-20-20-50-0110-5-20	-438	-438	-438.0000	0.00	0	3918	0	32
miblp-20-20-50-0110-5-2	-591	-591	-591.0000	0.00	1558	6053523	0	49
miblp-20-20-50-0110-5-3	-477	-477	-477.0000	0.00	0	53	0	50
miblp-20-20-50-0110-5-4	-753	-753	-753.0000	0.00	0	142	0	71
miblp-20-20-50-0110-5-5	-392	-392	-392.0000	0.00	0	51	0	31
miblp-20-20-50-0110-5-6	-1061	-1061	-1061.0000	0.00	5	79502	0	92
miblp-20-20-50-0110-5-7	-547	-547	-547.0000	0.00	0	80	0	16
miblp-20-20-50-0110-5-8	-936	-936	-936.0000	0.00	0	69	0	91
miblp-20-20-50-0110-5-9	-877	-877	-877.0000	0.00	0	112	0	62

Instance Set XUWANG.

In Xu and Wang (2014), a generic solver for MIBLPs has been introduced, along with a set of 140 instances, comprising integer leader variables, and continuous follower variables. In the following, we perform a comparison of our B&C solver with the solver of Xu and Wang (2014). As Assumption (1) does not hold for the

follower subproblem, our solver relies on the separation of the hypercube ICs of Subsection 3.3 only. FUB cuts and preprocessing remain valid in this case, and are turned on in our experiment.

Both approaches solve all 140 instances from this class to optimality. Table 3 reports the required computing times.

Each table row corresponds to a set of ten instances and computing times of our solver are given per instance (for $i = 1, \ldots, 10$) and on average (column *avg*). Column *avg-XU* gives instead the average computing time as reported in Xu and Wang (2014), and refers to a "desktop computer with 2.4 GHz" which is therefore 2 to 5 time slower than our hardware. The table shows that, after taking the hardware differences into account, our approach remains orders of magnitude faster that the one presented in Xu and Wang (2014), in particular for the largest instances. Indeed, for the instances with $n_1 = 460$ our solver has an average computing time of 23.1 seconds, against 6 581.4 seconds reported in Xu and Wang (2014). Moreover, for instances of class B2-160, which seem the hardest in this testbed, the speed-up is even more pronounced with average computing times of 37.4 versus 22 999.7 seconds.

Table 3: Results for the instance set `XUWANG`. Computing times (in seconds) given for ten instances ($i = 1, \ldots, 10$) for fixed n_1. Column avg-XU gives the average computing times as reported in Xu and Wang (2014).

n_1	$i=1$	$i=2$	$i=3$	$i=4$	$i=5$	$i=6$	$i=7$	$i=8$	$i=9$	$i=10$	avg	avg-XU
10	3	3	3	3	2	3	2	3	2	3	2.6	1.4
60	2	0	0	1	1	1	1	1	2	2	0.9	45.6
110	2	1	2	2	1	2	1	2	2	12	2.8	111.9
160	2	2	3	2	3	1	4	1	1	3	2.1	177.9
210	2	3	1	1	3	3	3	2	5	3	2.6	1224.5
260	3	4	3	6	3	5	6	2	7	11	5.0	1006.7
310	5	10	11	14	7	16	15	8	5	3	9.4	4379.3
360	17	28	11	13	11	15	7	19	9	14	14.4	2972.4
410	19	10	29	8	21	10	9	15	23	42	18.7	4314.2
460	22	10	22	35	21	21	32	22	23	23	23.1	6581.4
B1-110	0	0	0	0	0	1	0	1	0	9	1.3	132.3
B1-160	1	1	3	1	2	1	3	0	0	2	1.3	184.4
B2-110	16	2	2	8	1	25	15	5	1	122	19.7	4379.8
B2-160	8	38	21	91	34	4	40	3	12	123	37.4	22999.7

Instance Sets `INTER-KP2` and `INTER-CLIQUE`.

Another generic solver for zero-sum MIBLPs has been recently proposed in Tang et al. (2015), along with two families of benchmark instances of *interdiction* type on which the solver is tested: knapsack-interdiction and clique-interdiction, denoted by `INTER-KP2` and `INTER-CLIQUE` in the following. We compare our default setting for the B&C solver, namely `MIX++`, with the best results reported in Tang et al. (2015), which have been obtained "on a PC with 3.30 GHz using CPLEX 12.5". The results of this comparison are summarized in Table 4. Each row reports average results over a subset of ten instances that share the same (n_1, k) pair (left part of the table, `INTER-KP2` instances) or the same graph-parameters (right part, associated with `INTER-CLIQUE` instances). For the solver of Tang et al. (2015), we report the average value of the best known upper bound (BestSol), the average computing time (in seconds), the average final lower bound (obtained after one hour of computing) and the total number of unsolved instances out of ten (denoted by *#unsol*). For `MIX++` we only report the average computing time in seconds, as it turns out that our solver manages to solve all 150 + 80 instances to optimality (compared to 53 + 10 for the best solver from Tang et al. (2015)). For `INTER-KP2` the average computing times of `MIX++` vary between a fraction of a second to five minutes. The set `INTER-CLIQUE` turns out to be much easier, as in most of the cases, optimal solutions are found within a fraction of a second. Only for the largest instances with 15 nodes and graph density of 0.9, our solver requires about 12 seconds on average. We finally point out that the solver of Tang et al. (2015) is specifically tailored for problems with binary leader variables and with the zero-sum problem structure, whereas our solver handles these instances as general bilevel problems and does not explicitly exploit these properties.

Table 4: Results for the instance sets INTER-KP2 (left) and INTER-CLIQUE (right). Each row contains average results for ten instances. All instances are solved to proven optimality by MIX++.

			Tang et al. (2015)				MIX++
n_1	k	OPT	t[s]	BestSol	LB	#unsol	t[s]
20	5	388.5	721.4	388.5	388.5	0	5.4
20	10	163.7	2992.6	159.9	104.2	3	1.7
20	15	31.4	129.5	31.4	31.4	0	0.2
22	6	382.7	1281.2	382.7	141.5	6	10.3
22	11	161.0	3601.8	162.4	0.0	10	2.3
22	17	29.2	248.2	29.2	29.2	0	0.2
25	7	436.2	3601.4	437.2	0.0	10	33.6
25	13	191.5	3602.3	195.6	0.0	10	8.0
25	19	41.8	1174.6	41.8	41.8	0	0.4
28	7	516.1	3601.0	516.5	0.0	10	97.9
28	14	223.4	3602.5	226.8	0.0	10	22.6
28	21	46.2	3496.9	46.4	7.0	8	0.5
30	8	536.3	3601.0	537.4	0.0	10	303.0
30	15	230.0	3602.3	230.8	0.0	10	31.8
30	23	47.5	3604.5	48.5	0.0	10	0.6

			Tang et al. (2015)				MIX++
ν	d	OPT	t[s]	BestSol	LB	#unsol	t[s]
8	0.7	2.2	373.0	2.2	2.2	0	0.1
8	0.9	3.0	3600.0	3.0	0.0	10	0.2
10	0.7	2.9	3600.1	3.0	0.0	10	0.3
10	0.9	3.0	3600.2	4.0	0.0	10	0.7
12	0.7	3.0	3600.3	3.6	0.0	10	0.8
12	0.9	3.0	3600.4	5.0	0.0	10	1.9
15	0.7	3.0	3600.3	4.6	0.0	10	2.2
15	0.9	3.0	3600.2	6.3	0.0	10	12.6

Instance Set MIPLIB.

Table 5 compares MIX++ with the previous-best code from the literature (namely, SEP2 by Fischetti et al. (2016b)) on the very hard MIPLIB class. Recall that this class contains some instances with up to 80 000 HPR variables, hence in many cases the optimal solutions are still unknown. For the two settings and for each instance, Table 5 reports the best obtained upper bound (BestSol), the best obtained lower bound (LB), the final percentage gap, the computing time (in seconds), and the number of B&C nodes. Whereas SEP2 solves 20 instances of this class to optimality, MIX++ manages to provide optimal solutions in 27 cases. Furthermore, for all but three instances, the final lower bounds is strictly (and, very often, significantly) better by using our new MIX++ algorithm.

7 Conclusions

Mixed-Integer Bilevel Linear Programs are very important and challenging optimization models arising in many important practical contexts, including pricing mechanisms in the energy sector, airline and telecommunication industry, transportation networks, optimal expansion of gas networks, critical infrastructure defense, and machine learning.

In the present paper we have presented a new branch-and-cut algorithm for the exact solution of such problems. We have described an effective bilevel-specific preprocessing procedure. In addition, new classes of valid linear inequalities have been introduced, along with the corresponding separation procedures.

The computational performance of our method has been evaluated on a very large set of test problems from the literature—with its 800+ instances of various types, our computational study is by far the most extensive ever reported in the literature. Our new algorithm consistently outperformed (often by a large margin) all alternative state-of-the-art methods from the literature, including methods which exploit problem specific information for special instance classes, and was also able to provide the optimal solution for hundreds of open instances from the recent literature.

Future work should investigate how our approach can be specialized to better deal with bilevel optimization problems with a specific structure—notably, interdiction-type problems. The extension of our solution scheme to more general (possibly nonlinear) settings is also an interesting topic for future research.

Acknowledgements

This research was funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-014. The work of M. Fischetti and M. Monaci was also supported by the University of Padova (Progetto

di Ateneo "Exploiting randomness in Mixed Integer Linear Programming"), and by MiUR, Italy (PRIN project "Mixed-Integer Nonlinear Optimization: Approaches and Applications"). The work of I. Ljubić and M. Sinnl was also supported by the Austrian Research Fund (FWF, Project P 26755-N19).

References

A. Baggio, M. Carvalho, A. Lodi, and A. Tramontani. Multilevel approaches for the critical node problem. working paper, cole Polytechnique de Montral, 2016.

E. Balas. Intersection cuts–a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39, 1971.

R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. P Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

J. Bracken and J. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21:37–44, 1973.

Luce Brotcorne, Martine Labb, Patrice Marcotte, and Gilles Savard. Joint design and pricing on a network. *Operations Research*, 56(5):1104–1115, 2008.

Gerald Brown, Matthew Carlyle, Javier Salmern, and Kevin Wood. Defending critical infrastructure. *Interfaces*, 36 (6):530–544, 2006.

M. Caramia and R. Mari. Enhanced exact algorithms for discrete bilevel linear problems. *Optimization Letters*, 9(7): 1447–1468, 2015.

S. DeNegre. *Interdiction and Discrete Bilevel Linear Programming*. PhD thesis, Lehigh University, 2011.

S. DeNegre and T. K. Ralphs. A branch-and-cut algorithm for integer bilevel linear programs. In *Operations research and cyber-infrastructure*, pages 65–78. Springer, 2009.

E. D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. Appendix of: A new general-purpose algorithm for mixed-integer bilevel linear programs. 2016a. URL `http://www.dei.unipd.it/~fisch/appendix.pdf`.

Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. Intersection cuts for bilevel optimization. In Quentin Louveaux and Martin Skutella, editors, *Integer Programming and Combinatorial Optimization: 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings, Springer International Publishing*, pages 77–88. extended version submitted for publication, 2016b. URL `http://www.dei.unipd.it/~fisch/papers/intersection_cuts_for_bilevel_optimization.pdf`.

Pablo Garcia-Herreros, Lei Zhang, Pratik Misra, Erdem Arslan, Sanjay Mehta, and Ignacio E. Grossmann. Mixed-integer bilevel optimization for capacity planning with rational markets. *Computers & Chemical Engineering*, 86:33 – 47, 2016.

Franois Gilbert, Patrice Marcotte, and Gilles Savard. A numerical study of the logit network pricing problem. *Transportation Science*, 49(3):706–719, 2015.

F. Glover. Polyhedral convexity cuts and negative edge extensions. *Zeitschrift für Operations Research*, 18(5):181–186, 1974.

F. Glover and D. Klingman. Improved convexity cuts for lattice point problems. *Journal of Optimization Theory and Applications*, 19(2):283–291, 1976.

Zeynep H. Gümüs and Christodoulos A. Floudas. Global optimization of mixed-integer bilevel programming problems. *Computational Management Science*, 2(3):181–212, 2005.

Robert G Jeroslow. The polynomial hierarchy and a simple model for competitive analysis. *Mathematical programming*, 32(2):146–164, 1985.

P.-M. Kleniati and C. S. Adjiman. A generalization of the branch-and-sandwich algorithm: From continuous to mixed-integer nonlinear bilevel problems. *Computers & Chemical Engineering*, 72:373 – 386, 2015.

Karl Kunisch and Thomas Pock. A bilevel optimization approach for parameter learning in variational models. *SIAM Journal on Imaging Sciences*, 6(2):938–983, 2013.

Martine Labbé, Patrice Marcotte, and Gilles Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management science*, 44(12-part-1):1608–1622, 1998.

P. Loridan and J. Morgan. Weak via strong Stackelberg problem: new results. *Journal of Global Optimization*, 8: 263–297, 1996.

A. Mitsos. Global solution of nonlinear mixed-integer bilevel programs. *Journal of Global Optimization*, 47(4): 557–582, 2010.

J. Moore and J. Bard. The mixed integer linear bilevel programming problem. *Operations Research*, 38(5):911–921, 1990.

T. K. Ralphs. MibS, 2015. `https://github.com/tkralphs/MibS`.

T. K. Ralphs and E. Adams. Bilevel instance library, 2016. `http://coral.ise.lehigh.edu/data-sets/bilevel-instances/`.

G. K. Saharidis and M. G. Ierapetritou. Resolution method for mixed integer bi-level linear problems based on decomposition technique. *Journal of Global Optimization*, 44(1):29–51, 2009.

Maria P. Scaparra and Richard L. Church. A bilevel mixed-integer program for critical infrastructure protection planning. *Computers & Operations Research*, 35(6):1905 – 1923, 2008.

Yen Tang, Jean-Philippe P. Richard, and J. Cole Smith. A class of algorithms for mixed-integer bilevel min–max optimization. *Journal of Global Optimization*, pages 1–38, 2015.

R. Kevin Wood. *Bilevel Network Interdiction Models: Formulations and Solutions*. John Wiley & Sons, Inc., 2010.

P. Xu. *Three essays on bilevel optimization algorithmsand applications*. PhD thesis, Iowa State University, 2012.

P. Xu and L. Wang. An exact algorithm for the bilevel mixed integer linear programming problem under three simplifying assumptions. *Computers & Operations Research*, 41:309–318, 2014.

Marco Zugno, Juan Miguel Morales, Pierre Pinson, and Henrik Madsen. A bilevel model for electricity retailers' participation in a demand response market environment. *Energy Economics*, 36:182–197, 2013.

Table 5: Results for instance set MIPLIB. *Instances solved to proven optimality by MIX++ using a more aggressive setting: air03-0.1 (optimal value 376936), p0201-0.1 (opt. 12225), and p0282-0.5 (opt. 272659).

	SEP2					MIX++				
instance	BestSol	LB	%gap	time [s]	nodes	BestSol	LB	%gap	time [s]	nodes
air03-0.1 *	382822	343658.2857	10.23	3600	146125	379800	344940.6202	9.18	3600	92677
air03-0.5	505172	344292.2835	31.85	3600	85478	512698	360653.0000	29.66	3600	76005
air03-0.9	823130	344977.6032	58.09	3600	44697	770100	358130.7506	53.50	3600	42757
air04-0.1	56563	56150.9887	0.73	3600	55921	56399	56291.5472	0.19	3600	61419
air04-0.5	60131	56152.5000	6.62	3600	35826	60076	56224.6292	6.41	3600	33459
air04-0.9	84993	55961.5987	34.16	3600	3752	73759	56035.8472	24.03	3600	6658
air05-0.1	26801	26472.0000	1.23	3600	101047	26577	26577.0000	0.00	401	10168
air05-0.5	32497	26448.1580	18.61	3600	92234	31290	26469.7692	15.41	3600	75980
air05-0.9	44567	26358.2362	40.86	3600	82050	40558	26350.2340	35.03	3600	63300
cap6000-0.1	-	-2451433.7522	-	3600	1980	-1967015	-1967015.0000	0.00	587	48281
cap6000-0.5	-	-2451320.1481	-	3600	1481	-	-2400881.8508	-	3600	1115
cap6000-0.9	-259599	-2449416.5638	843.54	3600	9709	-	-2372513.7387	-	3600	328
enigma-0.1	0	0.0000	0.00	0	990	0	0.0000	0.00	0	739
enigma-0.5	0	0.0000	0.00	4	13842	0	0.0000	0.00	6	10531
enigma-0.9	0	0.0000	0.00	46	2670	0	0.0000	0.00	186	2966
fast0507-0.1	12562	172.3615	98.63	3600	604	12484	12484.0000	0.00	2	0
fast0507-0.5	61516	173.5238	99.72	3600	7767	61439	61439.0000	0.00	2	0
fast0507-0.9	109916	109916.0000	0.00	8	2	109916	109916.0000	0.00	1	0
l152lav-0.1	4722	4722.0000	0.00	2	367	4722	4722.0000	0.00	2	363
l152lav-0.5	4866	4760.4318	2.17	3600	311915	4868	4758.5000	2.25	3600	258223
l152lav-0.9	5090	4789.6571	5.90	3600	211309	5072	4785.3277	5.65	3600	171722
lseu-0.1	1120	1120.0000	0.00	0	15	1120	1120.0000	0.00	0	19
lseu-0.5	2525	1154.0895	54.29	3600	13333	2313	2036.3864	11.96	3600	12840
lseu-0.9	5838	5838.0000	0.00	24	299	5838	5838.0000	0.00	65	357
mitre-0.1	122310	115155.2394	5.85	3600	20791	122235	115310.6145	5.66	3600	41872
mitre-0.5	146730	115155.0000	21.52	3600	15611	-	115492.2852	-	3600	19004
mitre-0.9	168885	115155.9467	31.81	3600	13066	-	115493.3824	-	3600	10099
mod010-0.1	6554	6554.0000	0.00	8	739	6554	6554.0000	0.00	4	9
mod010-0.5	6692	6551.2005	2.10	3600	117241	6618	6573.4891	0.67	3600	164755
mod010-0.9	7448	6554.2000	12.00	3600	158667	7355	6565.6667	10.73	3600	111883
nw04-0.1	17066	17066.0000	0.00	820	2884	17066	17066.0000	0.00	1140	2842
nw04-0.5	23914	16985.0000	28.97	3600	18519	24100	16689.4706	30.75	3600	8472
nw04-0.9	43374	18271.4453	57.87	3600	12282	52290	18241.2202	65.12	3600	6631
p0033-0.1	3089	3089.0000	0.00	0	0	3089	3089.0000	0.00	0	0
p0033-0.5	3095	3095.0000	0.00	0	2	3095	3095.0000	0.00	0	0
p0033-0.9	4679	4679.0000	0.00	0	7	4679	4679.0000	0.00	0	6
p0201-0.1 *	12465	7793.2074	37.48	3600	5092	12555	9724.8337	22.54	3600	5837
p0201-0.5	13650	11615.0000	14.91	3600	649100	13635	13635.0000	0.00	1113	71052
p0201-0.9	15025	15025.0000	0.00	1	150	15025	15025.0000	0.00	1	157
p0282-0.1	260785	258489.8687	0.88	3600	371989	260781	260781.0000	0.00	4	272
p0282-0.5 *	273069	258437.0000	5.36	3600	998732	272659	267014.5318	2.07	3600	120899
p0282-0.9	627411	285137.0000	54.55	3600	2075980	616034	398553.0624	35.30	3600	175290
p0548-0.1	11301	8691.0000	23.10	3600	54071	11051	9115.1458	17.52	3600	102504
p0548-0.5	22197	8701.0000	60.80	3600	5121	-	11358.4606	-	3600	11943
p0548-0.9	49235	16109.8084	67.28	3600	293986	49509	19026.7330	61.57	3600	17003
p2756-0.1	14444	3124.0000	78.37	3600	36718	12862	3338.0000	74.05	3600	37599
p2756-0.5	23565	3124.0000	86.74	3600	58203	25384	4077.6628	83.94	3600	18777
p2756-0.9	35087	3124.0000	91.10	3600	13687	33623	4685.3819	86.06	3600	9263
seymour-0.1	486	413.8447	14.85	3600	231	476	469.9485	1.27	3600	48178
seymour-0.5	836	414.0236	50.48	3600	564	807	807.0000	0.00	2	18
seymour-0.9	1251	1251.0000	0.00	9	2	1251	1251.0000	0.00	1	0
stein27-0.1	18	18.0000	0.00	22	983	18	18.0000	0.00	0	528
stein27-0.5	19	19.0000	0.00	7	336	19	19.0000	0.00	0	5
stein27-0.9	24	24.0000	0.00	0	0	24	24.0000	0.00	0	0
stein45-0.1	30	30.0000	0.00	1899	12549	30	30.0000	0.00	3	2999
stein45-0.5	32	32.0000	0.00	658	18613	32	32.0000	0.00	0	14
stein45-0.9	40	40.0000	0.00	0	0	40	40.0000	0.00	0	0