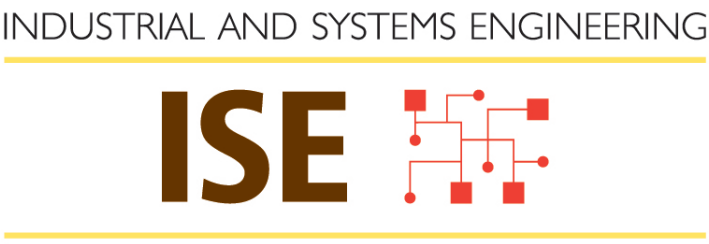


Valid Inequalities for Mixed Integer Bilevel Linear Optimization Problems

Sahar Tahernejad

SimCorp, Copenhagen, Denmark

Ted K. Ralphs

Department of Industrial and System Engineering, Lehigh University, Bethlehem, PA

COR@L Technical Report 20T-013-R1

Valid Inequalities for Mixed Integer Bilevel Linear Optimization Problems

Sahar Tahernejad*[1] and Ted K. Ralphs†[2]

[1]SimCorp, Copenhagen, Denmark
[2]Department of Industrial and System Engineering, Lehigh University, Bethlehem, PA

Original Publication: October 30, 2020
Last Revised: September 25, 2025

Abstract

Despite the success of branch-and-cut methods for solving mixed integer bilevel linear optimization problems (MIBLPs) in practice, there are still gaps in both the theory and practice surrounding these methods. In the first part of this paper, we lay out a basic theory of valid inequalities and cutting-plane methods for MIBLPs that parallels the existing theory for mixed integer linear optimization problems (MILPs). We provide a general scheme for classifying valid inequalities and illustrate how the known classes of valid inequalities fit into this categorization, as well as generalizing several existing classes.

In the second part of the paper, we assess the computational effectiveness of these valid inequalities and discuss the myriad challenges that arise in integrating methods of dynamically generating inequalities valid for MIBLPs into a branch-and-cut algorithms originally designed for solving MILPs. Although branch-and-cut methods for solving for MIBLPs are in principle straightforward generalizations of those used for MILP, there are subtle but important differences and there remain many unanswered questions regarding how to suitably modify control mechanisms and other algorithmic details in order to ensure performance in the MIBLP setting. We demonstrate that performance of version `1.2` of the open-source solver `MibS` was substantially improved over that of version `1.1` through a variety of improvements to the previous implementation.

1 Introduction

Bilevel optimization (and multilevel optimization, more generally) provides a framework for modeling and solution of optimization problems in which decisions are made in multiple time stages by multiple (possibly competing) decision-makers (DMs). Such optimization problems arise in a wide

*satn@simcorp.com
†ted@lehigh.edu

array of applications, with ever more coming to light as computational tools for solution of such problems become more widely available. For an overview of bilevel optimization (the case in which the number of decision stages is two) and its applications, we refer the reader to Dempe [2002].

The focus of this paper is on solution of *mixed integer bilevel linear optimization problems* (MIBLPs) in which some of the variables must take on integer values. The first algorithm proposed for solution of MIBLPs was the LP-based branch-and-bound algorithm of Bard and Moore [1990]. Building on this basic framework, DeNegre and Ralphs [2009] showed that it could be successfully used as the basis for a branch-and-cut algorithm much like those that revolutionized the solution of *mixed integer linear optimization problems* (MILPs). Just as in the MILP case, approximation of the convex hull of feasible solutions by valid inequalities has proved to be an effective means by which to improve bounds obtained by solving the (typically very weak) linear optimization problem (LP) relaxation employed in the original branch-and-bound algorithm.

The open-source solver `MibS` [DeNegre, Ralphs, and Tahernejad (2024)] was built on top of an existing branch-and-cut framework intended for solving MILPs called `BLIS` [Xu and Ralphs (2022)]. The original goal of `MibS` was to enable testing of algorithmic ideas related to the various components of the branch-and-cut approach in a "white box" framework. The initial version of the solver, which targeted the pure integer case, was described by DeNegre and Ralphs [2009]. Since then, `MibS` has continued to evolve and is now a full-featured solver capable of handling general MIBLPs. The implementation of version `1.1` was described in detail by Tahernejad et al. [2020]. This paper is a follow-on that discusses details of the implementation of the newest version `1.2`, which features an improved control mechanisms tuned specifically for the MIBLP setting and improved strategies for generation of valid inequalities.

The rest of the paper is divided into two parts. In the first part, a basic theory of valid inequalities for MIBLPs is developed. A primary theme is that much of the theory that has been developed for the MILP case and has been exploited so successfully in the development of MILP solvers can be generalized to the MIBLP case. However, while the techniques for generating valid inequalities for MILPs have been well-systematized, with a rich and well-developed theory underpinning them, the same cannot be said in the MIBLP case. As far as we know, there is, for example, no generalization of Meyer's Theorem [Meyer (1974)] for MIBLPs. In Section 3, we attempt to fill some of the existing gaps by laying out a theory of valid inequalities paralleling the one that already exists in the MILP case. Then in Section 4, we propose a systematization that encompasses the existing classes of valid inequalities and provides an overview of the known techniques for generation of valid inequalities.

In the second part of the paper, we first discuss details of the practical implementation of a branch-and-cut algorithm for MIBLPs in Section 5, focusing on the integration of practical methods of separating solutions to the LP relaxation from the convex hull of feasible solutions by the dynamic generation of strong valid inequalities [Gomory (1958); Balas, Ceria, et al. (1993); Cornuéjols (2008a); Wolter (2006)]. From a complexity-theoretic standpoint, the principle of equivalence of optimization and separation tells us that cut generation is necessarily much more challenging in the case of MIBLPs in the worst case. Our empirical analysis shows that this is also true from the standpoint of practical computation.

Finally, in Section 6, we conclude that substantial progress in improving effectiveness of practical algorithms has been made. We describe computational experiments carried out with `MibS 1.2` aimed at assessing the overall effectiveness of this class of algorithms on solving the instances in a

set of standard benchmarks from the literature, the progress that has been made over time, and the relative effectiveness of the various classes of inequalities on different classes of problems. We also discuss the importance of various parameters and how details of the cut generation strategy and other components of the algorithm (particularly branching) impact the effectiveness of the valid inequalities.

2 Mixed Integer Bilevel Optimization

Bilevel optimization problems are difficult to solve, both in an empirical and theoretical sense, even in the simple case of (continuous) bilevel linear optimization problems (BLPs), in which the constraint and objective functions are linear and the variables are continuous. MIBLPs are a specific class of bilevel optimization problems in which the constraints and objective functions must be linear, but some variables may also be required to take on integer values. While BLPs are hard for the complexity class NP, MIBLPs are hard for the class Σ_2^{P}, one level higher in the so-called polynomial time hierarchy [Stockmeyer (1976)].

2.1 Formulation

To formally define the class of optimization problems considered in this study, let $x \in X$ denote the set of variables controlled by the *first-level* DM or *leader* and $y \in Y$ denote the set of variables controlled by the *second-level* DM or *follower*, where $X = \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1-r_1}$ and $Y = \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2-r_2}$. The general form of an MIBLP is

$$\min_{x \in X} cx + \Xi(x), \tag{MIBLP}$$

where the function Ξ is a *risk function* that encodes the part of the objective value of x that depends on the response to x in the second level.

The function Ξ may have different forms, depending on the precise variant of the bilevel problem being solved. In this paper, we consider deterministic MIBLPs and focus on the so-called *optimistic* case [Loridan and Morgan (1996)], in which

$$\Xi(x) = \min \left\{ d^1 y \;\middle|\; y \in \mathcal{P}_1(x), y \in \operatorname{argmin}\{d^2 y \;\middle|\; y \in \mathcal{P}_2(x) \cap Y\} \right\}, \tag{RF}$$

where

$$\mathcal{P}_1(x) = \left\{ y \in \mathbb{R}_+^{n_2} \;\middle|\; G^1 y \geq b^1 - A^1 x \right\}$$

is a parametric family of polyhedra containing points satisfying the linear constraints of the first-level problem with respect to a given $x \in \mathbb{R}^{n_1}$ and

$$\mathcal{P}_2(x) = \left\{ y \in \mathbb{R}_+^{n_2} \;\middle|\; G^2 y \geq b^2 - A^2 x \right\}$$

is a second parametric family of polyhedra containing points satisfying the linear constraints of the second-level problem with respect to a given $x \in \mathbb{R}^{n_1}$. The input data is $A^1 \in \mathbb{Q}^{m_1 \times n_1}$, $G^1 \in \mathbb{Q}^{m_1 \times n_2}$, $b^1 \in \mathbb{Q}^{m_1}$, $c \in \mathbb{Q}^{n_1}$, $d^1, d^2 \in \mathbb{Q}^{n_2}$, $A^2 \in \mathbb{Q}^{m_2 \times n_1}$, $G^2 \in \mathbb{Q}^{m_2 \times n_2}$ and $b^2 \in \mathbb{Q}^{m_2}$. We note that the formulation allows participation of the second-level variables in the first-level constraints. The matrix G^1 can be taken to be a matrix of zeros (with appropriate dimensions) in the case in which the first-level constraints are independent of the second-level variables.

In the optimistic case, the evaluation of Ξ is a lexicographic optimization problem, with the choice among alternative optima to the inner optimization problem made according to the first-level objective function, as specified formally in (RF). In this case, (MIBLP) can be reformulated, in principle, as a (single-level) mathematical optimization problem by considering the *value function* of the second-level problem instead of the risk function. This formulation is given by

$$\min \left\{ cx + d^1 y \;\middle|\; x \in X, y \in \mathcal{P}_1(x) \cap \mathcal{P}_2(x) \cap Y, d^2 y \leq \phi(b^2 - A^2 x) \right\}, \tag{MIBLP-VF}$$

where ϕ represents the value function of the second-level problem and is defined as

$$\phi(\beta) = \min \left\{ d^2 y \;\middle|\; G^2 y \geq \beta, y \in Y \right\} \quad \forall \beta \in \mathbb{R}^{m_2}. \tag{VF}$$

The function ϕ yields the optimal value of the second-level problem corresponding to a given $\beta \in \mathbb{R}^{m_2}$. For a fixed $\beta \in \mathbb{R}^{m_2}$, the problem of evaluating ϕ is called either *the follower's problem* or the *second-level problem*.

A wide variety of special cases of MIBLPs have been studied in the literature. *Interdiction problems* are one of the most important classes among these special cases. In these problems, a subset of first-level variables indexed by $L = \{1, \ldots, k_1\}$ ($k_1 \leq \min\{r_1, r_2\}$), are the *interdiction variables*, binary variables that are each associated with a corresponding second-level variable whose value must be zero if the associated interdiction variable's value is one. A standard formulation for instances in this class of problem is

$$\min \left\{ dy \;\middle|\; x \in \mathcal{P}_1^{INT} \cap X, x_L \in \mathbb{B}^L, y \in \operatorname{argmax}\{ dy \;|\; y \in \mathcal{P}_2^{INT}(x) \cap Y \} \right\}, \tag{MIPINT}$$

where

$$\mathcal{P}_1^{INT} = \left\{ x \in \mathbb{R}^{n_1}_+ \;\middle|\; Ax \geq b \right\},$$
$$\mathcal{P}_2^{INT}(x) = \left\{ y \in \mathbb{R}^{n_2}_+ \;\middle|\; Gy \geq g, y_L \leq \operatorname{diag}(u)(e - x_L) \right\}.$$

Here, $A \in \mathbb{Q}^{m_1 \times n_1}$, $b \in \mathbb{Q}^{m_1}$, $d \in \mathbb{Q}^{n_2}$, $G \in \mathbb{Q}^{m_2 \times n_2}$, $g \in \mathbb{Q}^{m_2}$, $u \in \mathbb{R}^L_+$ represents the (variable) upper bound vector for the second-level variables indexed by L, and e is the $|L|$-dimensional vector of ones. In Section 4, we describe specialized inequalities that are valid for this more general class. `MibS` has specialized methods for solving (MIPINT) and other special classes of MIBLPs.

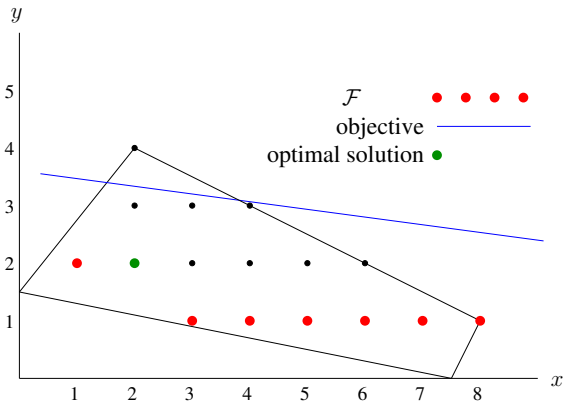
Finally, we introduce some additional sets that will be needed for discussing the relaxations used in the remainder of the paper. Dropping the integrality constraints and the optimality constraint of the second-level problem from (MIBLP-VF) results in the *LP relaxation*, with feasible region

$$\mathcal{P} = \left\{ (x, y) \in \mathbb{R}^{n_1 + n_2}_+ \;\middle|\; y \in \mathcal{P}_1(x) \cap \mathcal{P}_2(x) \right\}.$$

This set includes all $(x, y) \in \mathbb{R}^{n_1+n_2}_+$ that satisfy the linear constraints of the first- and second-level problems. The subset of $\mathcal{P}$ containing points that also satisfy the integrality constraints in both levels is

$$\mathcal{S} = \mathcal{P} \cap (X \times Y).$$

and forms the feasible region of the *MILP relaxation*.



$$\begin{aligned} \min_{x \in \mathbb{Z}_+} \quad & -x - 10y \\ \text{s.t.} \quad & y \in \operatorname{argmin} \{ y : \\ & -5x + 4y \leq 6 \\ & x + 2y \leq 10 \\ & 2x - y \leq 15 \\ & 2x + 10y \geq 15 \\ & y \in \mathbb{Z}_+ \} \end{aligned}$$

Figure 1: The feasible region and optimal solution of the the example from [Moore and Bard (1990)].

2.2 Bilevel Feasible Region and Certificates of Infeasibility

Because feasibility conditions are more complex and more difficult to verify for a given point than in the MILP case, it is important to formally define what is meant by the feasible region in this case. With respect to a given $x \in \mathbb{R}^{n_1}_+$, the *rational reaction set* is defined as

$$\mathcal{R}(x) = \operatorname{argmin} \left\{ d^2 y \mid y \in \mathcal{P}_2(x) \cap Y \right\}$$

and contains all $y \in Y$ that are optimal to the second-level problem arising from fixing the first-level variables to x. Note that (x, y) may not be bilevel feasible, even if $y \in \mathcal{R}(x)$ if either $x \notin X$ or $y \notin \mathcal{P}_1(x)$. As such, the conditions for bilevel feasibility of $(x, y) \in \mathbb{R}^{n_1+n_2}_+$ can be stated as

Feasibility Condition 1. $x \in X$.

Feasibility Condition 2. $y \in \mathcal{P}_1(x) \cap \mathcal{R}(x)$.

Based on these conditions, the *bilevel feasible region* is

$$\mathcal{F} = \{(x, y) \in X \times Y \mid y \in \mathcal{P}_1(x) \cap \mathcal{R}(x)\}.$$

The problem (MIBLP) can thus be re-cast as the optimization problem

$$\min_{(x,y) \in \mathcal{F}} cx + d^1 y. \tag{MIBLP-F}$$

The bilevel feasible region and optimal solution of the well-known example from [Moore and Bard (1990)], shown in Figure 1, illustrates these concepts.

In the context of MIBLPs, the infeasible solutions that arise as a result of solving the LP relaxation can violate either Feasibility Condition 1 or 2 (or both). Identifying violations of Condition 1 is easy, but in contrast with MILPs, identifying violations of Condition 2 may not be easy. Condition 2 can in fact be broken down into three sub-conditions, as follows.

Feasibility Condition 2a $y \in \mathcal{P}_1(x)$;

Feasibility Condition 2b $y \in Y$; and

Feasibility Condition 2c $d^2 y \leq \phi(b^2 - A^2 x)$.

Like Condition 1, Conditions 2a and 2b are easy to check, but verifying condition 2c requires solving an MILP. When condition 2c is *not* satisfied, the proof is a *certificate* that generally takes one of two forms.

Definition 1 (Certificate of Infeasibility)**.** *With respect to* $(\hat{x}, \hat{y}) \in \mathcal{P} \setminus \mathcal{F}$*,*

- *An* improving solution *is* $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$ *such that* $d^2\hat{y} > d^2 y^*$.
- *An* improving direction *is* $\Delta y \in Y$ *such that* $d^2 \Delta y < 0$ *and* $y + \Delta y \in \mathcal{P}_2(x)$.

The certificate proves the infeasibility of the given point and this information is used directly in generating valid inequalities that separate the point from $\mathcal{F}$, as described later in Section 4. These two certificates are theoretically equivalent for points in $\mathcal{S} \setminus \mathcal{F}$ and it is easy to convert one to the other (importantly, this is not the case for points in $\mathcal{P} \setminus \mathcal{S}$). Despite the theoretical equivalence—each can be generated by solving an MILP—the two approaches to proving infeasibility are quite different in practice.

2.3 Assumptions

In the rest of the paper, we make the following assumptions.

Assumption 1. $\mathcal{P}$ *is bounded.*

This assumption ensures the boundedness of (MIBLP), but it is made primarily for ease of presentation and can be straightforwardly relaxed.

Assumption 2. $\left\{ r \in \mathbb{R}^{n_2}_+ \;\middle|\; G^2 r \geq 0, d^2 r < 0 \right\} = \emptyset$.

This second assumption prevents the unboundedness of the second-level problem (regardless of the first-level solution) and can be checked in a pre-processing step.

Assumption 3. *All first-level variables with at least one non-zero coefficient in the second-level problem (the* linking variables*) are integer, i.e.,*

$$L = \left\{ i \in \{1, \ldots, n_1\} \;\middle|\; A^2_i \neq 0 \right\} \subseteq \{1, ..., r_1\},$$

where A^2_i *represents the* i^{th} *column of matrix* A^2.

This third assumption guarantees that the optimal solution value of (MIBLP) is attainable whenever the optimal solution value is finite [Vicente et al. (1996)]. The linking variables are structurally important and recognition of this can streamline the process of solving MIBLPs in several ways. Proposition 1 formally states the special role played by the linking variables.

Proposition 1. *For the vectors x^1 and $x^2 \in \mathbb{R}^{n_1}_+$ with $x^1_L = x^2_L \in \mathbb{Z}^L$, we have*

$$\phi(b^2 - A^2x^1) = \phi(b^2 - A^2x^2),$$

where x^1_L and x^2_L represent the subvector of x^1 and x^2, respectively, corresponding to the linking variables.

This theorem states formally that in the optimistic setting, when the linking variables are fixed, (MIBLP) (which is a non-linear optimization problem in general, due to the non-linearity of the optimality constraint of the second-level problem) becomes an MILP. This is because $\phi(b^2 - A^2x)$ is a constant whenever the linking variables are fixed. Corollary 1 states this more explicitly.

Corollary 1. *For $\gamma \in \mathbb{Z}^L$, we have*

$$\min\left\{cx + d^1y \;\middle|\; (x,y) \in \mathcal{F}, x_L = \gamma\right\} = \min\left\{cx + d^1y \;\middle|\; (x,y) \in \mathcal{S}, d^2y \leq \phi(b^2 - A^2x), x_L = \gamma\right\}. \tag{UB}$$

Hence, the best bilevel feasible solution $(x,y) \in \mathcal{F}$ with $x_L = \gamma \in \mathbb{Z}^L$ can be obtained by solving the problem (UB), which is an MILP. As already observed, Corollary 1 holds only in the optimistic setting and in this setting, the corollary makes it clear that the non-linking first-level variables can in fact all be assumed w.l.o.g to be part of the lower-level problem. It seems unlikely that this observation has important algorithmic implications, however.

3 Theory of Valid Inequalities for MIBLPs

The theory of valid inequalities for polyhedra and other closed convex sets is well-known and supported by deep foundations developed over several decades (see, e.g., Grötschel et al. [1993a]). Although much of the theory was developed specifically with the case of MILPs in mind, it can be applied in other contexts, such as MIBLPs. The main tools used in applying this theory to MILPs are *convexification* and the well-known result of Grötschel et al. [1993a] that optimization over a closed convex set is polynomially equivalent to the so-called *separation problem* associated with the set. Essentially, we optimize over the convex hull of feasible solutions rather than the original feasible region, thereby transforming the original non-convex problem into an equivalent convex one.

3.1 Convexification

Convexification considers a conceptual reformulation of the problem obtained by taking the convex hull of $\mathcal{F}$, as shown Figure 2. It may not be immediately obvious that convexification and separation can be employed in the context of MIBLPs, so we first show that MIBLPs can, in theory, be solved by a so-called *cutting-plane method*. Showing this formally involves showing that convexifying the feasible region does not change the optimal solution value. This can be done in several steps.

Proposition 2. *Under Assumption 3, $\mathcal{F}$ is closed.*

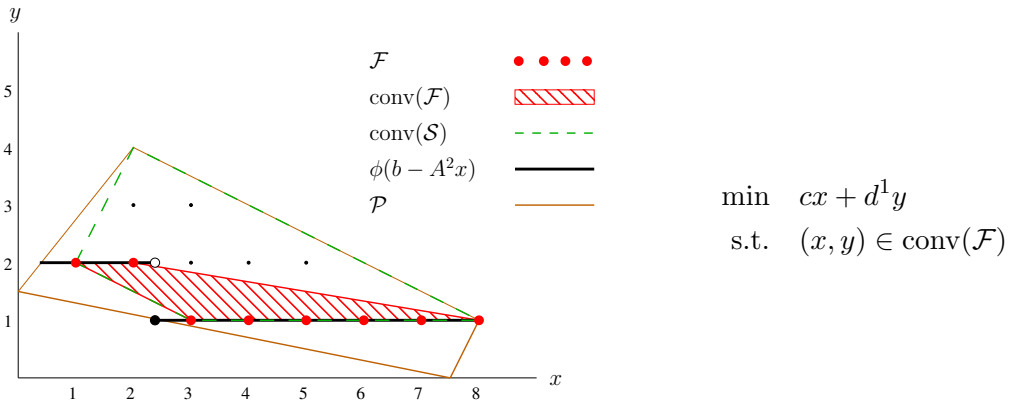


Figure 2: The polyhedral reformulation of the example from [Moore and Bard (1990)].

Proof. Let $\mathcal{S}_\gamma$ be the feasible region of the problem (UB) for $\gamma \in \mathbb{Z}^L$. Under Assumption 3, $\mathcal{F}$ is the union of (possibly infinite) disjoint sets $\mathcal{S}_\gamma$ for $\gamma \in \mathcal{F}_{x_L} = \text{proj}_{x_L}(\mathcal{F})$, i.e.,

$$\mathcal{F} = \bigcup_{\gamma \in \mathcal{F}_{x_L}} \mathcal{S}_\gamma. \tag{1}$$

Furthermore, under Assumption 3, for all $(\hat{x}, \hat{y}) \in \mathbb{R}^{n_1+n_2}$, there is at least one neighborhood that intersects at most one of the sets $\mathcal{S}_\gamma$ with $\gamma \in \mathcal{F}_{x_L}$. The radius $\bar{r}$ of this neighborhood can be defined as $0 < \bar{r} < \min\left\{||x_L - \hat{x}_L||^2 \;\middle|\; x_L \neq \hat{x}_L, x_L \in \mathcal{F}_{x_L}\right\}$ (such $\bar{r}$ exists due to Assumption 3). Therefore, the collection of sets $\mathcal{S}_\gamma$ for $\gamma \in \mathcal{F}_{x_L}$ is a locally finite collection of $\mathbb{R}^{n_1+n_2}$ under Assumption 3. From this and (1), it follow that [Munkres (2014)]

$$\text{cl}(\mathcal{F}) = \text{cl}\left(\bigcup_{\gamma \in \mathcal{F}_{x_L}} \mathcal{S}_\gamma\right) = \bigcup_{\gamma \in \mathcal{F}_{x_L}} \text{cl}\left(\mathcal{S}_\gamma\right). \tag{2}$$

We have $\text{cl}\left(\mathcal{S}_\gamma\right) = \mathcal{S}_\gamma$ since this set is the feasible region of (UB), which is an MILP. Therefore, from (2), we have

$$\text{cl}(\mathcal{F}) = \bigcup_{\gamma \in \mathcal{F}_{x_L}} \mathcal{S}_\gamma = \mathcal{F}.$$

Closedness of $\mathcal{F}$ follows. □

Theorem 1. *Under Assumptions 1 and 3,* $\text{conv}(\mathcal{F})$ *is a rational polyhedron.*

Proof. Basu et al. [2018] showed that when input data are rational, the closure of $\mathcal{F}$ can be written as the finite union of MILP-representable sets (with rational data). Since $\mathcal{F}$ is closed by Proposition 2, we have that

$$\mathcal{F} = \text{cl}\left(\mathcal{F}\right) = \bigcup_{i=1}^{k} L\left(\mathcal{S}_i\right), \tag{3}$$

where k is a scalar, L denotes a linear transformation (specifically projection) and $\mathcal{S}_i$ is the feasible region of an MILP for $i = 1, ..., k$.

From the fundamental theorem of integer programming [Meyer (1974)] and (3), it follows that

$$\begin{aligned}\operatorname{conv}(\mathcal{F}) &= \operatorname{conv}\left(\bigcup_{i=1}^{k} L(\mathcal{S}_i)\right) = \operatorname{conv}\left(\bigcup_{i=1}^{k} \operatorname{conv}(L(\mathcal{S}_i))\right) \\ &= \operatorname{conv}\left(\bigcup_{i=1}^{k} L(\operatorname{conv}(\mathcal{S}_i))\right) = \operatorname{conv}\left(\bigcup_{i=1}^{k} \mathcal{Q}_i\right),\end{aligned} \tag{4}$$

where $\mathcal{Q}_i = L(\operatorname{conv}(\mathcal{S}_i))$ is a rational polyhedron for $i = 1, ..., k$.

Furthermore, since $\mathcal{Q}_i$ is bounded for $i = 1, ...k$ by Assumption 1, the result follows from the fact that the convex hull of the union of a finite number of bounded polyhedra is a polyhedron [Balas (1985); Balas (1998)]. □

Theorem 2. *Under Assumption 3, we have that*

$$\min_{(x,y)\in\mathcal{F}} cx + d^1 y = \min_{(x,y)\in\operatorname{conv}(\mathcal{F})} cx + d^1 y. \tag{5}$$

Proof. Since the objective function is linear, we have [Conforti et al. (2014)]

$$\inf_{(x,y)\in\mathcal{F}} cx + d^1 y = \inf_{(x,y)\in\operatorname{conv}(\mathcal{F})} cx + d^1 y$$

and the infimum of $cx + d^1 y$ is attained over $\mathcal{F}$ if and only if it is attained over $\operatorname{conv}(\mathcal{F})$. This follows the result because the infimum of $cx + d^1 y$ is attained over $\mathcal{F}$ under Assumption 3. □

The problem (5) would be an LP in principle, if we knew a complete description of $\operatorname{conv}(\mathcal{F})$. In such case, a solution of the MIBLP could be obtained by producing an extremal solution to this LP by, e.g., the simplex algorithm. Since we generally do not know a complete description of $\operatorname{conv}(\mathcal{F})$ and cannot construct one efficiently (this would be at least as difficult as solving the original optimization problem), the cutting-plane method is to employ the well-known technique of generating an approximation of $\operatorname{conv}(\mathcal{F})$ by solving the separation problem to generate valid inequalities, as described next.

3.2 Cutting-plane Method

Although they are well-known, we first review several standard definitions and results before describing the cutting-plane method in broad outline.

Definition 2. *A* valid inequality *for the set $\mathcal{F}$ is a triple $(\alpha^x, \alpha^y, \beta)$, where $(\alpha^x, \alpha^y) \in \mathbb{Q}^{n_1+n_2}$ is the* coefficient vector *and $\beta \in \mathbb{Q}$ is a* right-hand side*, such that*

$$\mathcal{F} \subseteq \left\{(x, y) \in \mathbb{R}^{n_1\times n_2} \;\middle|\; \alpha^x x + \alpha^y y \geq \beta\right\}.$$

It is easy to see that an inequality is valid for $\mathcal{F}$ if and only if it is valid for the convex hull of $\mathcal{F}$. The so-called *separation problem* for $\operatorname{conv}(\mathcal{F})$ is to either generate an inequality valid for $\operatorname{conv}(\mathcal{F})$, but violated by a given vector $(\hat{x}, \hat{y}) \notin \operatorname{conv}(\mathcal{F})$ or to show that the vector is actually a member of $\operatorname{conv}(\mathcal{F})$. Formally, we define the problem as follows.

Definition 3. *The* separation problem *for* $\text{conv}(\mathcal{F})$ *with respect to a given* $(\hat{x}, \hat{y}) \in \mathbb{R}^{n_1+n_2}$ *is to determine whether or not* $(\hat{x}, \hat{y}) \in \text{conv}(\mathcal{F})$ *and if not, to produce an inequality* $(\alpha^x, \alpha^y, \beta) \in \mathbb{Q}^{n_1+n_2+1}$ *valid for* $\text{conv}(\mathcal{F})$ *and for which* $\alpha^x \hat{x} + \alpha^y \hat{y} < \beta$.

The process of solving an MIBLP by a standard cutting-plane method is initiated by solving a convex relaxation of the original problem (MIBLP). In contrast to MILPs, the problem obtained by removing integrality constraints on all variables (at both levels) is not a relaxation (its feasible region does not necessarily contain $\mathcal{F}$). On the other hand, we have that $\mathcal{F} \subseteq \mathcal{S} \subseteq \mathcal{P}$, so the problem of optimizing over either $\mathcal{P}$ or $\mathcal{S}$ is a valid relaxation. In `MibS`, $\mathcal{P}$ is used as the feasible region of that starting relaxation and the relaxation problem is

$$\min_{(x,y)\in\mathcal{P}} cx + d^1 y. \tag{LR}$$

In the remainder of the paper, we take this as the initial relaxation.

A valid inequality for $\mathcal{F}$ that is violated by a point in the feasible region of the relaxation (typically an extreme point) is called a *cutting plane* or *cut*.

Definition 4. *A* cut *is an inequality valid for* $\mathcal{F}$*, but violated by some* $(\hat{x}, \hat{y}) \in \mathcal{P} \setminus \mathcal{F}$ *(typically an optimal solution to a relaxation defined with respect to* $\mathcal{P}$*).*

The term "cut" is often used interchangeably with the term "valid inequality," but they are not technically synonymous. Nevertheless, we will use them interchangeably to some extent in the remainder of the paper in order to be consistent with existing terminology.

Let (x^0, y^0) be an extremal optimal solution of (LR). Then the cutting-plane method consists of the following loop beginning with $t = 0$.

1. **Determine whether $(x^t, y^t) \in \text{conv}(\mathcal{F})$ or not.** Determining whether a given arbitrary point is in $\text{conv}(\mathcal{F})$ is a Σ_2^{P}-hard problem in general, but just as in the MILP setting, we can exploit the fact that (x^t, y^t) is an extremal member of $\mathcal{P}$. Such a point must be a member of $\mathcal{F}$ in order to be a member of $\text{conv}(\mathcal{F})$. Therefore, we need only determine whether it satisfies the constraints that were relaxed. In this case, we must check whether $(x^t, y^t) \in X \times Y$ and $y^t \in \mathcal{R}(x^t)$. If so, then $(x^t, y^t) \in \mathcal{F}$ (and (x^t, y^t) is an optimal solution), so the solution process is terminated. Otherwise $(x^t, y^t) \notin \mathcal{F}$ and we move to Step 2. As such, the problem of checking whether $(x^t, y^t) \in \text{conv}(\mathcal{F})$ is in NP when (x^t, y^t) is an extreme point of $\mathcal{P}$.

2. **Separate $(x^t, y^t) \notin \mathcal{F}$ from $\text{conv}(\mathcal{F})$.** To do so, we construct an inequality $(\alpha^x, \alpha^y, \beta)$ that is valid for $\text{conv}(\mathcal{F})$ but violated by (x^t, y^t). This can be done by the procedures discussed in Section 4.

3. **Iterate.** Add the constraint $\alpha^x x + \alpha^y y \geq \beta$ to the relaxation and re-solve to obtain (x^{t+1}, y^{t+1}). Set $t = t + 1$ and go to Step 1.

As usual, we iterate until either the method terminates or until one of a specified set of termination criteria are met, e.g., the number of iterations exceeds some pre-defined limit. For the pure integer

case ($n_1 = r_1$, $n_2 = r_2$), a finite cutting-plane algorithm for MIBLP was described by DeNegre and Ralphs [2009].

Whether or not this method converges finitely depends on exactly how the valid inequalities are generated and what properties they are guaranteed to have. In the case of MILPs, finite cutting-plane algorithms for the pure integer and general cases under mild assumptions were, respectively, given by Gomory [1958] and Del Pia and Weismantel [2012] (see [Gade and Küçükyavuz (2011)] for more detailed discussion on the convergence of the cutting-plane algorithm). Since the feasible region $\mathcal{P}$ of our assumed initial relaxation is a polyhedron, the bounding problem (LR) is an LP and can be solved by standard algorithms. Any extreme point of $\mathcal{P}$ is either a member of $\mathcal{F}$ or is *not* contained in $\text{conv}(\mathcal{F})$ and can be separated from it by an inequality valid for $\text{conv}(\mathcal{F})$, as described above. Hence, MIBLPs can, in principle, be solved by a cutting-plane method.

3.3 Improving Valid Inequalities

Traditionally, cutting-plane methods have been described theoretically as generating only inequalities valid for the entire feasible set. In practice, however, it is well-known that the addition of inequalities removing feasible solutions is permitted, as long as this does not change the *optimal solution value* of the original problem. In the case of (MIBLP-VF), inequalities removing subsets of $\mathcal{F}$ are used routinely and we thus formally define a notion of valid inequality that allows this.

When $(x^*, y^*) \in \mathcal{F}$ and we have that $cx^* + d^1y^* \leq \min_{(x,y)\in(\mathcal{G}\cap\mathcal{F})} cx + d^1y$ for some set $\mathcal{G} \subseteq \mathbb{R}^{n_1+n_2}$, then we have that

$$\min_{(x,y)\in\text{conv}(\mathcal{F})} cx + d^1y = \min\left\{cx^* + d^1y^*, \min_{(x,y)\in\text{conv}(\mathcal{F})\setminus\mathcal{G}} cx + d^1y\right\}.$$

In this case, although $\mathcal{G} \cap \text{conv}(\mathcal{F})$ may contain a subset of the feasible region, it can be removed because it does not include any *improving solutions* relative to (x^*, y^*). Based on this discussion, the definition of valid inequality can be generalized as follows.

Definition 5 (Improving Valid Inequality)**.** *An* improving valid inequality *for the set $\mathcal{F}$ with respect to an incumbent $(x^*, y^*) \in \mathcal{F}$, is a triple $(\alpha^x, \alpha^y, \beta) \in \mathbb{R}^{n_1+n_2+1}$ such that*

$$\left\{(x, y) \in \mathcal{F} \;\middle|\; cx + d^1y < cx^* + d^1y^*\right\} \subseteq \left\{(x, y) \in \mathbb{R}^{n_1\times n_2} \;\middle|\; \alpha^x x + \alpha^y y \geq \beta\right\}.$$

In the remainder of paper, the term "valid inequality" is taken to mean "improving valid inequality" unless otherwise stated.

3.4 Methods for Construction

In this section, we discuss general recipes for obtaining valid inequalities for MIBLPs. Each recipe is generalized from a standard framework used to derive valid inequalities in the MILP case. Note that these frameworks have close connections to each other and most classes of inequalities can be developed by application of more than one of these frameworks. Although the general classes employed in solving generic MILPs were conceived specifically for that purpose, the theoretical basis for many of the classes does not actually depend on any specific properties of MILP and

they can thus be easily employed in other settings. The reader is referred to Marchand et al. [2002], Wolter [2006], and Cornuéjols [2008b] for broad overviews of these approaches and the relationships between the various classes. We describe each of these classes here at a high level in the context of MIBLPs and then discuss specific instantiations of these construction methods in Section 4.

3.4.1 Disjunctive Procedure

Disjunctive programming is both a modeling paradigm and a set of algorithmic techniques introduced by Balas [1979] based on the concept of what we refer to as a *valid disjunction*.

Definition 6 (Valid Disjunction)**.** *A collection of disjoint sets $X_i \subseteq \mathbb{R}^{n_1+n_2}$ for $i = 1, ..., k$ represents a* valid disjunction *for $\mathcal{F}$ if*

$$\mathcal{F} \subseteq \bigcup_{i=1}^{k} X_i.$$

In this context, the collection $\{X_i\}_{1 \leq i \leq k}$ is called a *disjunctive set*. The branch-and-cut algorithm depends crucially on the identification of valid disjunctions that are violated by the solution to some relaxation. The identified disjunctions are used both for branching and cutting, two essential elements of the branch-and-cut algorithm.

Most algorithms for solving MILPs exploit in some way the fact that points in $\mathcal{P} \setminus \mathcal{S}$ must violate one of the trivial disjunctions arising from integrality requirements on the variables. Most algorithms for solving MIBLPs exploit the fact that any point $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$ must violate the more complex valid disjunction

$$\begin{pmatrix} A^1 x \geq b^1 - G^1 y^* \\ A^2 x \geq b^2 - G^2 y^* \\ d^2 y \leq d^2 y^* \end{pmatrix} \quad \text{OR} \quad \begin{pmatrix} A^1 x \ngeq b^1 - G^1 y^* \\ \text{OR} \\ A^2 x \ngeq b^2 - G^2 y^*, \end{pmatrix} \tag{6}$$

where $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{R}(\hat{x})$ is an improving solution with respect to an incumbent $\hat{y}$ (recall Definition 1). Note that such a $y^* \neq \hat{y}$ must exist. Although this disjunction can be written compactly using the "$\ngeq$" operator, there is no compact way to express it using linear inequalities, which is an indication of what makes separation difficult in the MIBLP case.

Just as with valid inequalities, the basic notion of valid disjunction can be modified to allow for the possibility that the disjunction does not contain the entire set $\mathcal{F}$, but possibly eliminates some solutions known to be suboptimal. This yields the concept of an *improving valid disjunction*.

Definition 7 (Improving Valid Disjunction)**.** *A collection of disjoint sets $X_i \subseteq \mathbb{R}^{n_1+n_2}$ for $i = 1, ..., k$ represents an* improving valid disjunction *for $\mathcal{F}$ with respect to an incumbent $(x^*, y^*) \in \mathcal{F}$ if*

$$\left\{(x, y) \in \mathcal{F} \;\middle|\; cx + d^1 y < cx^* + d^1 y^*\right\} \subseteq \bigcup_{i=1}^{k} X_i.$$

In the remainder of paper, the term "valid disjunction" is taken to mean "improving valid disjunction" unless otherwise stated.

Definition 8 (Disjunctive Inequality)**.** *A* disjunctive (valid) inequality *for the set $\mathcal{F}$ with respect to $\mathcal{Q} \supseteq \mathcal{F}$ and a valid disjunction $\{X_i\}_{1 \leq i \leq k}$ is a triple $(\alpha^x, \alpha^y, \beta) \in \mathbb{R}^{n_1+n_2+1}$ such that*

$$\mathcal{Q} \bigcap \left(\bigcup_{i=1}^{k} X_i \right) \subseteq \left\{ (x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; \alpha^x x + \alpha^y y \geq \beta \right\}.$$

A subclass of the disjunctive inequalities arises from the disjunctions known as *split disjunctions* that have only two terms and are defined as follows.

Definition 9 (Split Disjunction)**.** *Let $(\pi^x, \pi^y, \pi_0) \in \mathbb{Z}^{n_1+n_2+1}$ be such that $\pi_i^x = 0$ for $i \geq r_1 + 1$ and $\pi_i^y = 0$ for $i \geq r_2 + 1$ (the coefficients of the continuous variables in both first and second levels are zero). Then when*

$$X_1 = \left\{ (x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; \pi^x x + \pi^y y \leq \pi_0 - 1 \right\} \text{ and } X_2 = \left\{ (x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; \pi^x x + \pi^y y \geq \pi_0 \right\}, \quad (7)$$

$\{X_1, X_2\}$ is a valid disjunction for $\mathcal{F}$ called a split disjunction*.*

The validity of the above disjunction arises from the fact that the inner product of any member of $\mathcal{F}$ with the coefficient vector is an integer and thus all members of $\mathcal{F}$ must belong to either X_1 or X_2. The disjunctive inequalities derived from such a disjunction are known as *split inequalities*.

Definition 10 (Split Inequality)**.** *A* split inequality *for the set $\mathcal{F}$ with respect to $\mathcal{Q} \supseteq \mathcal{F}$ and a split disjunction $\{X_1, X_2\}$ is a disjunctive inequality with respect to $\mathcal{Q}$ and the disjunction $\{X_1, X_2\}$.*

3.4.2 Generalized Chvátal Procedure

The usual Chvátal inequalities, as defined in the theory of MILPs (see [Conforti et al. (2014)]), are a subclass of the split inequalities in which $X_1 \cap \mathcal{Q} = \emptyset$ (as defined in Definition 10). We introduce here a class we refer to as *generalized Chvátal inequalities* that differ slightly in form, but are similar in spirit to the usual Chvátal inequalities.

Definition 11 (Generalized Chvátal Inequality)**.** *Let a split disjunction $\{X_1, X_2\}$ for set $\mathcal{F}$ be given such that $X_1 \cap \mathcal{Q} = \{(\hat{x}, \hat{y})\}$ and $(\hat{x}, \hat{y}) \notin \mathcal{F}$ for some given $\mathcal{Q} \supseteq \mathcal{F}$. Then (π^x, π^y, π_0) in (7) is itself a valid inequality for the set $\mathcal{F}$, known as a* generalized Chvátal inequality *with respect to set $\mathcal{Q}$.*

The connection to the Chvátal inequalities in MILP should be clear. In the MILP case, the usual Chvátal inequalities are split inequalities for which we have a proof that X_1 does not contain any feasible solutions to the LP relaxation of the MILP (and hence does not contain any solutions to the MILP itself). We can thus easily conclude that the MILP feasible region is contained in X_2 and derive the associated valid inequality. Here, we extend this idea to allow that X_1 may contain a (single) solution to the relaxation (whose feasible region we can think of as being $\mathcal{Q}$), but that we have an independent proof that the single solution to the relaxation is not contained in the feasible region $\mathcal{F}$ of the MIBLP itself. We thus have a similar proof that $\mathcal{F}$ is contained in the set X_2, yielding a valid inequality as in the MILP case. Note that we could further extend this definition to include cases where X_1 contains a set of feasible solutions to the relaxation, all of which can be

proven not to be in $\mathcal{F}$. However, as we currently have no practical application of such a definition, we use the simpler one here.

An obvious question that we address later is how to obtain a split disjunction (π^x, π^y, π_0) satisfying the requirements of the above definition in a practical way. In MILP, a relevant split disjunction can be derived from the tableau using the procedure of Gomory [Gomory (1960)]. One way of viewing this procedure is that we first derive an inequality valid for the feasible region of the LP relaxation by taking a combination of the inequalities in the original formulation, then ensure that the coefficients of the left-hand side satisfy the integrality requirements for a split disjunction either by rounding or in some other fashion, and finally round the right-hand side to obtain a valid inequality (see Gomory [1960], for more details).

A very similar procedure is possible in the case of MIBLP by taking $\mathcal{Q}$ in Definition 11 to be the LP relaxation $\mathcal{P}$ of (MIBLP) and $(\hat{x}, \hat{y}) \in (X \times Y) \setminus \mathcal{F}$ to be an extremal optimum to the LP relaxation. By taking a combination of the constraints binding at $(\hat{x}, \hat{y})$, we can derive a split disjunction satisfying Definition 11, as detailed later in Theorems 3 and 4.

3.4.3 Intersection Inequalities

Another well-known methodology for generating valid inequalities for MILPs that can be generalized to the MIBLP setting is that of generating so-called *intersection inequalities*, which we refer to by their more common name *intersection cuts* (ICs). This method of constructing inequalities was originally proposed by Balas [1971] and has already been extended to more general classes of optimization problems by Bienstock et al. [2016] and was extended to MIBLPs by Fischetti et al. [2018] and Fischetti et al. [2017a].

The first step in constructing an IC is to identify a so-called *bilevel-free* set. The identification of such sets underlies not only the generation of ICs, but also many other classes of inequalities.

Definition 12. *A* bilevel-free set *(BFS) is a closed, convex set* $\mathcal{C} \subseteq \mathbb{R}^{n_1+n_2}$ *such that* $\text{int}(\mathcal{C}) \cap \mathcal{F} = \emptyset$.

If $\mathcal{C}$ is a BFS, then inequalities valid for $\text{conv}(\overline{\text{int}(\mathcal{C})} \cap \mathcal{P})$[1] are also valid for $\mathcal{F}$. Such inequalities are called *intersection cuts* (ICs).

Given a point $(\hat{x}, \hat{y}) \notin \mathcal{F}$ such that $(\hat{x}, \hat{y})$ is an extreme point of $\mathcal{P}$ and a BFS $\mathcal{C}$ containing $(\hat{x}, \hat{y})$ in its interior, an IC violated by $(\hat{x}, \hat{y})$ can easily be generated using the following general recipe.

1. Let $\mathcal{V}(\hat{x}, \hat{y})$ be a simplicial radial cone with vertex $(\hat{x}, \hat{y})$ described by any set of $n_1 + n_2$ linearly independent inequalities of $\mathcal{P}$ that are binding at $(\hat{x}, \hat{y})$.
2. Let the triple $(\alpha^x, \alpha^y, \beta) \in \mathbb{R}^{n_1+n_2+1}$ be such that the set $\{(x, y) \in \mathbb{R}^{n_1+n_2} \mid \alpha^x x + \alpha^y y = \beta\}$ is the unique hyperplane containing the points of intersection of $\mathcal{C}$ with the extreme rays of $\mathcal{V}(\hat{x}, \hat{y})$.
3. Then $(\alpha^x, \alpha^y, \beta)$ is an inequality valid for $\mathcal{F}$ and violated by $(\hat{x}, \hat{y})$.

[1] The notation $\overline{\text{int}(\mathcal{C})}$ denotes the complement of the interior of $\mathcal{C}$

Typically, the point to be separated is a basic feasible solution to an LP relaxation and so the simplicial cone can be easily obtained by considering an optimal basis.

The challenge in generating ICs is primarily in the identification of the BFS. Fortunately, a BFS is easily generated when one of the two certificates of infeasibility from Definition 1 has already been generated. Given $(\hat{x}, \hat{y}) \in \mathcal{P}$ and an associated improving solution $y^* \in Y$, then

$$\mathcal{C}(y^*) = \left\{(x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; d^2 y \geq d^2 y^*, A^2 x \geq b^2 - G^2 y^* - 1\right\}. \tag{IS-BFS}$$

is a BFS. Similarly, given an associated improving direction $\Delta y \in Y$, we have that

$$\mathcal{C}(\Delta y) = \left\{(x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; A^2 x + G^2 y \geq b^2 - G^2 \Delta y - 1, y + \Delta y \geq -1\right\} \tag{ID-BFS}$$

is also a BFS. Note that the size of the defined set $\mathcal{C}$ can impact the strength of the generated IC (in general, bigger is better). Since the size is determined by the particular certificate of infeasibility generated and the goal is to find the largest set $\mathcal{C}$ possible, we must take into account how big the resulting BFS will be when solving the problem of generating the certificate of infeasibility. We discuss this more later.

3.4.4 Benders Cuts

A final class of inequalities that are not typically considered as a generic class in the theory of MILP, but are of particular importance in solving MIBLPs are the Benders cuts. Benders cuts arise from first applying a Benders reformulation, which involves projecting out a subset of the variables. This reformulation introduces optimality conditions that ensure the variables projected out must take optimal values with respect to the values of the remaining variables. Valid inequalities can then be derived by relaxing these optimality conditions. The general theory is described in detail by Bolusani and Ralphs [2022].

In the particular case of MIBLPs, the second-level optimality condition is captured by the constraint

$$d^2 y \leq \psi(x) := \phi(b^2 - A^2 x), \tag{8}$$

which can be seen as arising from a Benders-style projection operation. Inequalities of the form

$$d^2 y \leq \bar{\psi}(x), \tag{9}$$

where $\bar{\psi} : \mathbb{R}^{n_1} \to \mathbb{R}$ is such that $\bar{\psi}(x) \geq \psi(x)$ for all $x \in \mathbb{R}^{n_1}$ are therefore generally called *Benders cuts*. We introduce two classes of such inequalities in the next section.

4 Valid Inequalities for MIBLPs

As with MILPs, the main aim of generating a valid inequality is to remove from the feasible region of the relaxation a solution which is not feasible for the original problem, along with as much of the surrounding region as possible. In the case of MILPs, all such solutions that arise in a typical cutting-plane method violate integrality conditions and known classes primarily exploit violations

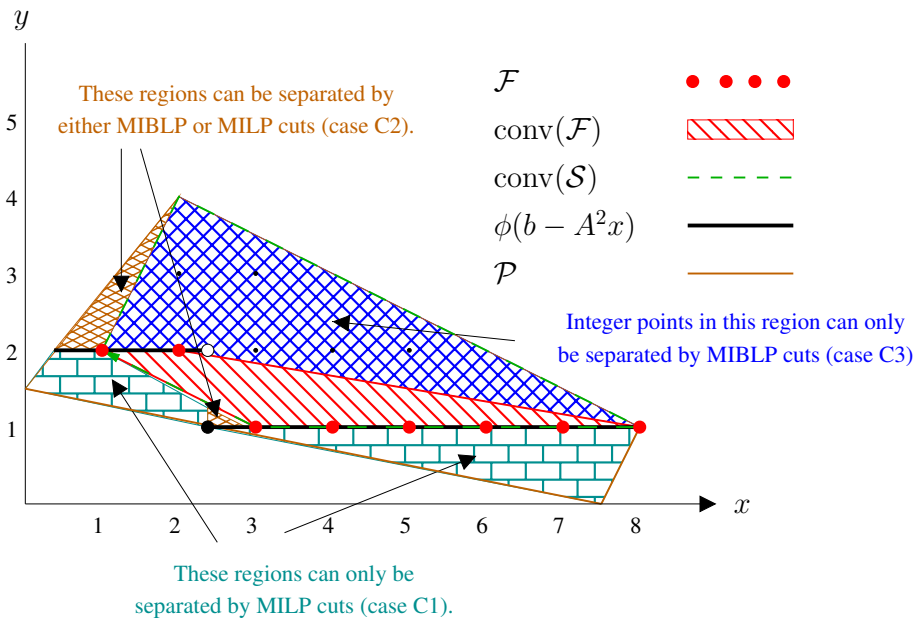


Figure 3: Types of infeasible points in the example from [Moore and Bard (1990)].

of simple disjunctions associated with single variables. In MIBLP, the situation is more complex, as we have several different kinds of points we are trying to separate and the methodologies for separating vary substantially in both strength and computational expense.

In general, solutions to (LR) can be categorized as follows.

C1. An extreme point $(\hat{x}, \hat{y})$ of $\mathcal{P}$ such that $(\hat{x}, \hat{y}) \notin X \times Y$ and $d^2\hat{y} \leq \phi(b^2 - A^2\hat{x})$.

C2. An extreme point $(\hat{x}, \hat{y})$ of $\mathcal{P}$ such that $(\hat{x}, \hat{y}) \notin X \times Y$ and $d^2\hat{y} > \phi(b^2 - A^2\hat{x})$.

C3. An extreme point $(\hat{x}, \hat{y})$ of $\mathcal{P}$ such that $(\hat{x}, \hat{y}) \in X \times Y$ and $d^2\hat{y} > \phi(b^2 - A^2\hat{x})$.

Figure 3 illustrates the distinction between these types of points. Distinguishing between these cases is important, but not easy. Determining when $(x, y) \notin X \times Y$ is straightforward, but further distinguishing between the cases C1 and C2 is difficult. This can have important impacts in practice, since distinguishing these cases is required to determine what classes of inequality are needed to separate a given point. Case C3 is unique to MIBLPs and involves removing a point that is integral. However, in this case, we at least know that we are restricted to inequalities specific to MIBLPs.

With respect to the goals of generating valid inequalities stated above, the set of applicable valid inequalities for MIBLPs can be classified roughly into three categories, as detailed in the next three paragraphs, in which U is taken to be a global upper bound on the optimal solution value. Note that the presented classification is just to provide a rough idea about different classes of valid inequalities and the classes are not entirely distinct.

Integrality Cuts. These are inequalities that enforce integrality and are violated by an extreme point of $\mathcal{P}$ for which $(x, y) \notin X \times Y$ (cases C1 and C2). They are valid for

$$\text{conv}\left(\left\{(x, y) \in \mathcal{S} \;\middle|\; cx + d^1 y < U\right\}\right).$$

This set includes all inequalities valid for $\mathcal{S}$, the feasible region of the MILP relaxation

$$\min_{(x,y)\in\mathcal{S}} cx + d^1 y.$$

These are the very same inequalities that are used in solving MILPs and include all general families, as well as any specialized inequalities valid for set $\mathcal{S}$ with particular structure. Since `MibS` utilizes the COIN-OR Cut Generation Library (CGL) [Forrest (2025)], all available separation routines in this package can also be employed in `MibS`. Most of these inequalities are themselves derived based on the principles described earlier in Section 3.4 and fall into those general categories described there. The disjunctions utilized are typically split disjunctions, often those involving a single variable.

Optimality Cuts. These are inequalities that enforce optimality conditions of the second-level problems and are violated by extreme points of $\mathcal{P}$ of the form described in cases C2 and C3. They are valid for

$$\text{conv}\left(\left\{(x, y) \in \mathcal{F} \;\middle|\; cx + d^1 y < U\right\}\right).$$

Whereas the integrality cuts are meant to enforce the requirement that $(x, y) \in \mathcal{S}$, the optimality cuts are meant to approximately enforce the optimality condition

$$d^2 y \leq \phi(x).$$

for the second-level problem. It is these optimality conditions that are relaxed (in addition to the integrality conditions) in order to obtain a tractable bounding problem. Optimality cuts are so named exactly because they approximate the above non-linear inequality with linear inequalities (the name is to evoke the optimality cuts arising in Benders decomposition, which are closely related). Note that because ϕ is non-convex and non-concave in general, this constraint cannot be approximated exactly by linear inequalities (all discussed optimality cuts in this section are linear). When combined with branching in a branch-and-cut algorithm, however, we can restrict the feasible region to areas in which this function is convex.

Projected Optimality Cuts. These are inequalities that are valid for

$$\text{conv}\left(\left\{(x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; x \in \text{proj}_x(\mathcal{F}), cx + \Xi(x) < U\right\}\right).$$

where Ξ is as defined in (RF). As mentioned earlier, such inequalities can be generated to remove regions containing only non-improving solutions, after solving (UB).

In the remainder of the section, we describe the optimality cuts and projected optimality cuts implemented in `MibS`.

4.1 Generalized Chvátal Inequality

4.1.1 General Chvátal Inequality

Assumptions. None.

Theorem 3. *Let $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$ be a basic feasible solution to (LR) and let $H \subseteq \mathbb{N}^N$ consist of the indices of a set of $n_1 + n_2$ linearly independent inequalities (including non-negativity) in the description of $\mathcal{P}$ that are binding at $(\hat{x}, \hat{y})$ (describe the polyhedral cone $\mathcal{V}(\hat{x}, \hat{y})$), where $N = m_1 + m_2 + n_1 + n_2$. Let $u \in \mathbb{Q}^N$ be such that $u_i = 0$ for $i \notin H$, $u_i > 0$ for $i \in H$, and such that $(\alpha^x, \alpha^y, \beta)$ is a split disjunction where*

$$(\alpha^x, \alpha^y) = u \begin{bmatrix} A^1 & G^1 \\ A^2 & G^2 \\ I & 0 \\ 0 & I \end{bmatrix}, \qquad \beta = u \begin{bmatrix} b^1 \\ b^2 \\ 0 \\ 0 \end{bmatrix} + 1.$$

Then

$$\alpha^x x + \alpha^y y \geq \beta \quad \forall (x, y) \in \mathcal{F}.$$

Furthermore, we have

$$\alpha^x \hat{x} + \alpha^y \hat{y} = \beta - 1,$$

so the inequality is violated by $(\hat{x}, \hat{y})$.

Proof. The inequality $(\alpha^x, \alpha^y, \beta - 1)$ is valid for $\mathcal{P}$, since it is derived as a positive combination of inequalities in the description of $\mathcal{P}$. Furthermore, $(\alpha^x, \alpha^y, \beta)$ is a split disjunction for which

$$\left\{(x, y) \in \mathbb{R}^{n_1+n_2} \;\middle|\; \alpha^x x + \alpha^y y \leq \beta - 1\right\} \cap \mathcal{P} = \{(\hat{x}, \hat{y})\}.$$

If not, then the face $\{(x, y) \in \mathcal{P} \mid \alpha^x x + \alpha^y y = \beta - 1\}$ contains a non-trivial face of $\mathcal{P}$, which leads to a contradiction of the fact that the set of inequalities indexed by H are linearly independent. Hence,

$$\left\{(x, y) \in \mathbb{R}^{n_1+n_2} \;\middle|\; \alpha^x x + \alpha^y y \leq \beta - 1\right\} \cap \mathcal{F} = \emptyset$$

and $(\alpha^x, \alpha^y, \beta)$ is valid for $\mathcal{F}$. □

Discussion. This inequality is a form of the generalized Chvátal inequality introduced in Definition 11. In the stated form, it is not clear how to choose H and how to generate the weight vector u. Choosing H is not difficult, as it can be derived from an optimal basis in the typical case in which $(\hat{x}, \hat{y})$ is an optimal basic feasible solution to the LP relaxation. Choosing u is more difficult. The integer no-good inequality introduced next uses $u_i = 1 \; \forall i \in H$. A more general way to generate such inequalities systematically is through the use of an auxiliary optimization problem similar to the cut-generating LP used in the case of lift-and-project for MILPs [Balas, Ceria, et al.

(1993)], as follows.

$$\begin{aligned}
\min_{u \in \mathbb{Q}^{m_1+m_2+n_1+n_2}} \quad & \alpha^x \hat{x} + \alpha^y \hat{y} - \beta \\
\alpha^x &= \begin{bmatrix} (A^1)^\top & (A^2)^\top & I & 0 \end{bmatrix} u \\
\alpha^y &= \begin{bmatrix} (G^1)^\top & (G^2)^\top & 0 & I \end{bmatrix} u \\
\beta &= \begin{bmatrix} (b^1)^\top & (b^2)^\top & 0 & 0 \end{bmatrix} u \\
u_i &\geq \epsilon (> 0) \quad \forall i \in H \\
u_i &= 0 \quad \forall i \notin H \\
\alpha^x &\in \mathbb{Z}^{r_1} \times \{0\}^{n_1 - r_1} \\
\alpha^y &\in \mathbb{Z}^{r_2} \times \{0\}^{n_2 - r_2}
\end{aligned}$$

As written, the above problem is unbounded, so some appropriate normalization is also needed. The general version of this cut has not been implemented in `MibS`, since it seems clear that it would not be as effective as the intersection cuts, which have the ability to eliminate infeasible integer points and can also be used to separate non-integer points.

4.1.2 Integer No-good Cut

Assumptions.

- $r_1 = n_1$ and $r_2 = n_2$.
- Vectors b^1 and b^2 and all matrices A^1, A^2, G^1, and G^2 are integer.

Theorem 4 (DeNegre and Ralphs [2009])**.** *Let $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$ be a basic feasible solution to (LR) and $H \subseteq \mathbb{N}^N$ consist of the indices of a set of $n_1 + n_2$ linearly independent inequalities (including non-negativity) in the description of $\mathcal{P}$ that are binding at $(\hat{x}, \hat{y})$, as in Theorem 3. Then, under the stated assumptions, we have*

$$\alpha^x x + \alpha^y y \geq \beta \quad \forall (x, y) \in \mathcal{F},$$

where

$$(\alpha^x, \alpha^y) = \mathbf{1}^N \begin{bmatrix} A^1 & G^1 \\ A^2 & G^2 \\ I & 0 \\ 0 & I \end{bmatrix}, \qquad \beta = \mathbf{1}^N \begin{bmatrix} b^1 \\ b^2 \\ 0 \\ 0 \end{bmatrix} + 1,$$

where $\mathbf{1}^N$ is a row vector of dimension $N = m_1 + m_2 + n_1 + n_2$ withe all entries equal to one. Furthermore, we have

$$\alpha^x \hat{x} + \alpha^y \hat{y} = \beta - 1,$$

so the inequality is violated by $(\hat{x}, \hat{y})$.

Proof. This is a special case of Theorem 5 in which $u = \mathbf{1}^N$ □

Discussion. Since we require α^x, α^y and β to integer-valued, this inequality is a disjunctive inequality for $\mathcal{F}$ with respect to the split disjunction

$$X_1 = \left\{(x, y) \in \mathbb{R}^{n_1+n_2} \;\middle|\; \alpha^x x + \alpha^y y \leq \beta - 1\right\} \qquad X_2 = \left\{(x, y) \in \mathbb{R}^{n_1+n_2} \;\middle|\; \alpha^x x + \alpha^y y \geq \beta\right\}.$$

Since $\mathcal{P} \cap X_1 = (\hat{x}, \hat{y})$ and $(\hat{x}, \hat{y}) \notin \mathcal{F}$, it is also a generalized Chvátal inequality for $\mathcal{F}$ with respect to $\mathcal{P}$ and the above split disjunction. The utilized split disjunction is defined by considering a combination of constraints of $\mathcal{P}$ binding at $(\hat{x}, \hat{y})$, as described earlier. The combination is guaranteed to yield a split disjunction because of our assumption that the constraint matrix is integral. Essentially, this is a simple case in which a vector u satisfying the conditions of Theorem 5 can be easily derived.

This cut can be applied to remove an optimal solution $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$ to (LR) under the above assumptions, but it does not eliminate any other bilevel infeasible members of $\mathcal{S}$. It is easy to generate such inequalities because all required information for its generation can be obtained from the optimal tableau obtained when solving (LR).

Example. The red line in Figure 4 shows the generated integer no-good cut for removing the bilevel infeasible extreme point $(2, 4)$ of $\mathcal{P}$ in the example shown in Figure 1. The inequality is obtained in two steps, as described above. We first sum the first two inequalities in the formulation (which are the ones binding at $(2, 4)$) to obtain the inequality $-2x + 3y \leq 8$, valid for $\mathcal{P}$. We then subtract one from the right-hand side to obtain the final inequality $-2x + 3y \leq 7$. As one can observe, this inequality does not remove any integer points from $\mathcal{P}$ except $(2, 4)$. It is not difficult to see that a deeper cut can be obtained by using general weights to combine the inequalities. Combining the inequalities with weight vector $u = [14, 70]^\top$ yields the inequality $y \leq 4$, valid for $\mathcal{P}$. Rounding, we obtain the valid inequality $y \leq 3$, which clearly dominates the original integer no good.

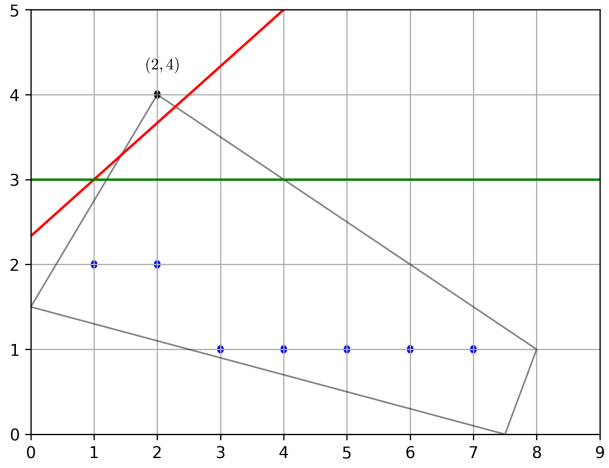


Figure 4: The generated integer no-good cut for the example shown in Figure 1

4.2 Benders Cuts

MibS includes two classes of Benders cuts, one that applies to problems with binary linking variables and a stronger class that applies only to interdiction problems.

4.2.1 Benders Binary Cut

Assumptions.

- $(x, y) \in \mathcal{F} \Rightarrow x_L \in \mathbb{B}^L$

Theorem 5. *Let $(\hat{x}, \hat{y}) \in \mathcal{P}$ such that $\hat{x}_L \in \mathbb{B}^L$ and $d^2\hat{y} > d^2y^*$ for some $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{P}_2(\hat{x}) \cap Y$. Further let*

- $L^- = \left\{i \in L \;\middle|\; A_i^2 \leq 0\right\}$, *and*
- $L^+ = \left\{i \in L \;\middle|\; A_i^2 \geq 0\right\}$.

Then, under the assumption, we have

$$d^2y - M\left(\sum_{i \in L \setminus L^- :\, \hat{x}_i = 1} (1 - x_i) + \sum_{i \in L \setminus L^+ :\, \hat{x}_i = 0} x_i\right) \leq d^2y^* \quad \forall (x, y) \in \mathcal{F}, \tag{10}$$

where $M \geq \max\{d^2y \mid (x, y) \in \mathcal{F}\} - d^2y^*$. Furthermore, this inequality is violated by all $(x, y) \in \mathcal{P}$ for which $x_L = \hat{x}_L$ and $d^2y > d^2y^*$. More specifically, it is violated by all $(x, y) \in \mathcal{S} \setminus \mathcal{F}$ such that $x_L = \hat{x}_L$.

Proof. Note that L^+ are the indices of linking variables for which an increase in value enlarges the second-level feasible region. Similarly, L^- are the indices of linking variables for which a decrease in value enlarges the second-level feasible region.

Let $(\tilde{x}, \tilde{y}) \in \mathcal{F}$, $(\hat{x}, \hat{y}) \in \mathcal{P}$, and $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{P}_2(\hat{x}) \cap Y$ be given as in the statement of the theorem. We show that the inequality derived from $(\hat{x}, y^*)$ is satisfied by $(\tilde{x}, \tilde{y})$. We consider two cases.

(i) Assume $\tilde{x}_i = 0$ for $i \in L \setminus L^+$ such that $\hat{x}_i = 0$ and $\tilde{x}_i = 1$ for $i \in L \setminus L^-$ such that $\hat{x}_i = 1$, so that

$$\sum_{i \in L \setminus L^- :\, \hat{x}_i = 1} (1 - \tilde{x}_i) + \sum_{i \in L \setminus L^+ :\, \hat{x}_i = 0} \tilde{x}_i = 0 \tag{11}$$

Then we first claim that

$$G^2y^* \geq b^2 - A\hat{x} \geq b^2 - A\tilde{x},$$

where the first inequality in the chain follows from $y^* \in \mathcal{P}_2(\hat{x})$ and the second inequality follows from our assumption and the definition of L^- and L^+. Then we have $y^* \in \mathcal{P}_2(\tilde{x}) \cap Y$. It follows that we must have

$$d^2\tilde{y} - M\left(\sum_{i \in L \setminus L^- :\, \hat{x}_i = 1} (1 - \tilde{x}_i) + \sum_{i \in L \setminus L^+ :\, \hat{x}_i = 0} \tilde{x}_i\right) = d^2\tilde{y} \leq d^2y^*.$$

Hence, $(\tilde{x}, \tilde{y})$ satisfies the inequality (10) in this case.

(ii) Assume that either for some $i \in L \setminus L^+$, we have $\tilde{x}_i = 1$, $\hat{x}_i = 0$, or for some $i \in L \setminus L^-$, we have $\tilde{x}_i = 0$, $\hat{x}_i = 1$. Then

$$d^2\tilde{y} \leq d^2y^* + M \leq d^2y^* + M \left(\sum_{i \in L \setminus L^- \colon \hat{x}_i = 1} (1 - \tilde{x}_i) + \sum_{i \in L \setminus L^+ \colon \hat{x}_i = 0} \tilde{x}_i \right).$$

Hence, $(\tilde{x}, \tilde{y})$ satisfies the inequality (10) in this case.

Since $(\tilde{x}, \tilde{y})$ was arbitrary, we have that the inequality (10) is valid for $\mathcal{F}$.

Finally, let $(\bar{x}, \bar{y}) \in \mathcal{P}$ be given with $\bar{x}_L = \hat{x}_L$ and assume that

$$d^2\bar{y} > d^2y^* \tag{12}$$

so that $(\bar{x}, \bar{y}) \notin \mathcal{F}$. Then by the same logic as (11), we have that

$$d^2\bar{y} - M \left(\sum_{i \in L \setminus L^- \colon \bar{x}_i = 1} (1 - \bar{x}_i) + \sum_{i \in L \setminus L^+ \colon \bar{x}_i = 0} \bar{x}_i \right) = d^2\bar{y} > d^2y^*$$

and the inequality (10) is violated by $(\bar{x}, \bar{y})$. Since $(\bar{x}, \bar{y})$ was chosen arbitrarily, we have that the inequality is violated by all $(x, y) \in \mathcal{P}$ with $x_L = \hat{x}_L$ and $d^2y > d^2y^*$. □

Discussion. This inequality is easily seen to be a disjunctive inequality for $\mathcal{F}$ with respect to the valid disjunction

$$X_1 = \left\{ (x, y) \in \mathbb{R}^{n_1+n_2} \,\middle|\, \sum_{i \in L \setminus L^- \colon \hat{x}_i = 1} (1 - x_i) + \sum_{i \in L \setminus L^+ \colon \hat{x}_i = 0} x_i = 0,\ d^2y \leq d^2\hat{y} \right\}$$
$$X_2 = \left\{ (x, y) \in \mathbb{R}^{n_1+n_2} \,\middle|\, \sum_{i \in L \setminus L^- \colon \hat{x}_i = 1} (1 - x_i) + \sum_{i \in L \setminus L^+ \colon \hat{x}_i = 0} x_i \geq 1 \right\}.$$

Note that this inequality can be used to eliminate infeasible solutions $(\hat{x}, \hat{y}) \in \mathcal{P} \setminus \mathcal{F}$ to the relaxation (LR) with $\hat{x}_L \in \mathbb{B}^L$ even when $\hat{y} \notin Y$, provided that we can produce a point $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{P}_2(\hat{x}) \cap Y$ such that $d^2\hat{y} > d^2y^*$. Such y^* can be obtained, e.g., by solving the second-level problem associated with $\hat{x}$ (the existence of $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{P}_2(\hat{x}) \cap Y$ such that $d^2\hat{y} > d^2y^*$ is only guaranteed, however, when $\hat{y} \in Y$).

It is also important to note that this inequality is closely related to the generalized no-good cut introduced later, which may be a preferable alternative, due to the possible difficulty of computing an appropriate value of M (we solve an integer program to find the big-M and it is likely that computing a valid big-M is formally NP-hard). The (possible) advantage of this inequality is that in cases where some columns of the matrix A^2 have uniform sign for first-level variables, this inequality may be stronger.

4.2.2 Benders Interdiction Cut

Assumptions. We assume the given problem is of the form (MIPINT).

In what follows, G_i is the i^{th} column of G and $\mathcal{P}^{INT} = \{(x, y) \in \mathbb{R}^{n_1+n_2} \mid x \in \mathcal{P}_1^{INT}, y \in \mathcal{P}_2^{INT}(x)\}$ the feasible region of the relaxation of (MIPINT) equivalent to (LR), $\mathcal{R}(x)$ is the rational reaction set associated with $x \in \mathcal{P}_1^{INT}$ and $\mathcal{F}$ is the set of bilevel feasible solutions, as usual.

Theorem 6. *Let $(\hat{x}, \hat{y}) \in \mathcal{P}^{INT}$ such that $\hat{x}_L \in \mathbb{B}^L$ and $d\hat{y} > dy^*$ for some $y^* \in \mathcal{P}_2^{INT}(\hat{x}) \cap Y$. Further, let*

- $L^- = \{i \in L \mid G_i \leq 0\}$, *and*
- $L^+ = \{i \in L \mid G_i \geq 0\}$.

Then, under the assumptions, we have

$$dy + \sum_{i \in L^-} (d_i y_i^*) x_i - \sum_{i \in L^+ :\, y_i^* = 0} d_i y_i - M \left(\sum_{i \in L \setminus L^- :\, y_i^* > 0} x_i \right) \leq dy^* \quad \forall (x, y) \in \mathcal{F}, \tag{13}$$

where $M \geq \max\{dy \mid (x, y) \in \mathcal{F}\} - d\hat{y}$. Furthermore, this inequality is violated by all $(x, y) \in \mathcal{P}^{INT}$ such that $x_L = \hat{x}_L$ and $dy > dy^*$ and more specifically, it is violated by all $(x, y) \in \mathcal{S} \setminus \mathcal{F}$ such that $x_L = \hat{x}_L$.

Proof. The sets L^+ and L^- play a role similar here to the role they play in the Benders binary cut. However, in interdiction problems, it is the bounds on the second-level variables that are parametric in the first-level variables rather than the right-hand side of the second-level constraints. This is why these sets are defined directly in terms of the matrix G rather than the matrix A. As such, L^+ (resp., L^-) represent indices of second-level variables that can be increased (resp., decreased) in value without affecting feasibility of y^*.

Let $(\tilde{x}, \tilde{y}) \in \mathcal{F}$, $(\hat{x}, \hat{y}) \in \mathcal{P}^{INT}$, and $y^* \in \mathcal{P}_2^{INT}(\hat{x}) \cap Y$ be given as in the statement of the theorem. We show that the inequality derived from y^* is satisfied by $(\tilde{x}, \tilde{y})$. We consider two cases.

(i) Assume $\tilde{x}_i = 0$ for $i \in L \setminus L^-$ such that $y_i^* > 0$. This means that only variables whose value can be decreased without affecting feasibility are fixed to value zero by the choice of $\tilde{x}$. Then we construct $y' \in Y$

$$y'_i = \begin{cases} 0 & \text{if } i \in L^- \text{ and } \tilde{x}_i = 1, \\ \tilde{y}_i & \text{if } i \in L^+ \text{ and } y_i^* = 0, \\ y_i^* & \text{otherwise .} \end{cases}$$

Then we first claim that $Gy' \geq g$, since we have

- $Gy^* \geq g$,
- $y'_i \geq y_i^*$ for all $i \in L^+$,
- $y'_i \leq y_i^*$ for all $i \in L^-$, and

– $y'_i = y^*_i$ otherwise.

We also have $\tilde{x}_i = 1 \Rightarrow y'_i = 0$. Hence, $y' \in \mathcal{P}_2^{INT}(\tilde{x}) \cap Y$ and since $\tilde{y} \in \mathcal{R}(\tilde{x})$, we have

$$\begin{aligned} d\tilde{y} \leq dy' &= dy^* - \sum_{i \in L^-} (d_i y^*_i)\tilde{x}_i + \sum_{i \in L^+ \colon y^*_i = 0} d_i \tilde{y}_i \\ &= dy^* - \sum_{i \in L^-} (d_i y^*_i)\tilde{x}_i + \sum_{i \in L^+ \colon y^*_i = 0} d_i \tilde{y}_i + M \left(\sum_{i \in L \setminus L^- \colon y^*_i > 0} \tilde{x}_i \right), \end{aligned}$$

since $\sum_{i \in L \setminus L^- \colon y^*_i > 0} \tilde{x}_i = 0$. It follows that $(\tilde{x}, \tilde{y})$ satisfies the inequality (13).

(ii) Now assume $\tilde{x}_i = 1$ for some $i \in L \setminus L^-$ such that $y^*_i > 0$. Then $\sum_{i \in L \setminus L^- \colon y^*_i > 0} \tilde{x}_i \geq 1$ and we have

$$d\tilde{y} \leq dy' + M \leq dy^* - \sum_{i \in L^-} (d_i y^*_i)\tilde{x}_i + \sum_{i \in L^+ \colon y^*_i = 0} d_i \tilde{y}_i + M \left(\sum_{i \in L \setminus L^- \colon y^*_i > 0} \tilde{x}_i \right)$$

It follows that $(\tilde{x}, \tilde{y})$ satisfies the inequality (13).

Since $(\tilde{x}, \tilde{y})$ was arbitrary, it follows that the inequality is valid for $\mathcal{F}$.

Moreover, since $y^* \in \mathcal{P}_2^{INT}(\hat{x})$, we have $y^*_i = 0$ for all $\{i \in L \mid \hat{x}_i = 1\}$. It follows that

$$dy^* = dy^* - \sum_{i \in L^-} (d_i y^*_i)\hat{x}_i + \sum_{i \in L^+ \colon y^*_i = 0} d_i y^*_i + M \left(\sum_{i \in L \setminus L^- \colon y^*_i > 0} \hat{x}_i \right) \tag{14}$$

Finally, let $(\bar{x}, \bar{y}) \in \mathcal{P}^{INT}$ with $\bar{x}_L = \hat{x}_L$ and

$$d\bar{y} > dy^* \tag{15}$$

be given. Then by (14) and (15), the inequality (13) is violated by $(\bar{x}, \bar{y})$. Since $(\bar{x}, \bar{y})$ was arbitrary, the inequality is violated by all $(x, y) \in \mathcal{P}^{INT}$ with $x_L = \hat{x}_L$ such that $dy > dy^*$. □

Discussion. Caprara et al. [2016] proposed a special case of this inequality for knapsack interdiction problems, though it was not found to be useful in their algorithm. A generalized version, valid for problems of the form (MIPINT), was released in `MibS 1.0` [DeNegre, Ralphs, and Tahernejad (2015)], as discussed by Ralphs [2015]. Similar general versions later appeared in Tahernejad [2019] and Fischetti et al. [2019]. What is presented here is a further generalization that does not make any assumptions on, e.g., monotonicity. This inequality is a disjunctive inequality for $\mathcal{F}$ with respect to the valid disjunction

$$X_1 = \left\{ (x, y) \in \mathbb{R}^{n_1+n_2} \,\middle|\, \sum_{i \in L \colon \hat{x}_i = 0} x_i + \sum_{i \in L \colon \hat{x}_i = 1} (1 - x_i) = 0, dy \leq d\hat{y} \right\}$$

$$X_2 = \left\{ (x, y) \in \mathbb{R}^{n_1+n_2} \,\middle|\, \sum_{i \in L \colon \hat{x}_i = 0} x_i + \sum_{i \in L \colon \hat{x}_i = 1} (1 - x_i) \geq 1, dy + \sum_{i \in L^-} (d_i \hat{y}_i) x_i - \sum_{i \in L^+ \colon \hat{y}_i = 0} d_i y_i \leq d\hat{y} \right\}.$$

As with the Benders binary cut, the Benders interdiction cut can be exploited to eliminate infeasible solutions $(\hat{x}, \hat{y}) \in \mathcal{P}^{INT} \setminus \mathcal{F}$ to the relaxation (LR) even when $\hat{y} \notin Y$, provided that we can produce a point $y^* \in \mathcal{P}_2^{INT}(\hat{x}) \cap Y$ such that $d\hat{y} > dy^*$ (see earlier discussion in the previous section on the Benders binary cut).

4.3 Intersection Cuts

`MibS` includes several known classes of ICs that are closely related, differing only in the form of the BFS used to generate each of them. The first two classes are covered in this section and use closely related BFSs. The first of these is derived from the existence of an improving feasible *solution* to the second-level problem, while the second is derived based on the existence of improving feasible *direction* to the second-level problem. The third class is based on an entirely different type of BFS and is discussed in Section 4.3.3.

4.3.1 Improving Solution Intersection Cuts

Assumptions.

- $A^2x + G^2y - b^2 \in \mathbb{Z}^{m_2}$ for all $(x, y) \in \mathcal{S}$.
- $d^2 \in \mathbb{Z}^{n_2}$ (Type II only).

Theorem 7 (Fischetti et al. [2018], [Fischetti et al. (2017a)])**.** *Let $(\hat{x}, \hat{y}) \in \mathcal{P} \setminus \mathcal{F}$ be an optimal solution of (LR) such that $d^2\hat{y} > d^2y^*$ for some $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$. Then, under the stated assumptions, we have*

$$\alpha^x x + \alpha^y y \geq \beta \quad \forall (x, y) \in \mathcal{F}, \tag{16}$$

where the inequality (16) is the IC associated with the sets $\mathcal{V}(\hat{x}, \hat{y})$ and

$$\mathcal{C} = \left\{(x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; d^2y \geq d^2y^*, A^2x \geq b^2 - G^2y^* - 1\right\}, \tag{17}$$

Furthermore, the inequality (16) is violated by $(\hat{x}, \hat{y})$.

Discussion. Two distinct procedures can be used can to derive violated valid inequalities from the above formula, by computing y^* in two different ways. ***Type I*** ISICs are obtained simply by solving the second-level problem to determine $y^* \in \mathcal{R}(\hat{x})$ (though any $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$ such that $d^2y^* < d^2\hat{y}$ will suffice). In ***Type II*** ISICs, y^* is taken to be an optimal solution of the following MILP, which is constructed with an objective function designed to make the resulting BFS as large as possible.

$$\begin{aligned} y^* \in \operatorname{argmin} & \sum_{i=1}^{m_2} w_i \\ & d^2y \leq \lceil d^2\hat{y} - 1 \rceil \\ & G^2y + (A^2\hat{x} - L)w \geq b^2 - L \\ & y \in Y \\ & w \in \{0, 1\}^{m_2}, \end{aligned} \tag{18}$$

where $L_i = \sum_{j=1}^{n_1} \min\{A^2_{ij} l_{x_j}, A^2_{ij} u_{x_j}\}$ for $i = 1, ..., m_2$ and A^2_{ij} represents the element of i^{th} row and j^{th} column of A^2. Due to constraint (18), when $w^*_i = 0$ (where w^* represents the optimal value of w), we can conclude that $g^2_i y^* \geq b^2_i - a^2_i x$ for all bilevel feasible solutions. Hence, this inequality can be removed from the definition of set $\mathcal{C}$ in (17) in this case. By utilizing such property, the defined set $\mathcal{C}$ for an ISIC of type II is hopefully larger than the one for type I and may provide a stronger cut. However, it should also be noted that finding y^* for ISICs of type II requires solving an MILP, while for ISICs of I, no additional effort is needed if the feasibility check is already being done.

It is important to point out that these two types of ISICs make no requirement on integrality of $(\hat{x}, \hat{y})$. We require that $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$, but y^* need not be a member of $\mathcal{R}(\hat{x})$ (i.e., optimal to the second-level problem arising from fixing $\hat{x}$). Type I ISICs are cheaply and conveniently generated when we've already generated a y^* satisfying the desired conditions either as a by-product of a primal heuristic or a feasibility check. When such y^* is not already available, it must be generated as part of the cut generation. Whether and when it is worthwhile to do this is an important empirical question. Type II ISICs always require the solution of an auxiliary subproblem, making them potentially more expensive.

With both types of inequalities, the separation procedure can fail in two different ways: either $\mathcal{P}$ may be entirely contained in the BFS or the subproblem for generating y^* may be infeasible. The former case is detected when the rays of $\mathcal{V}(\hat{x}, \hat{y})$ do not intersect the boundary of the BFS. This situation is obviously unlikely if $(\hat{x}, \hat{y})$ is a solution the initial LP relaxation, but when separation is done for a subproblem deeper in the branch-and-bound tree, it can (and does) happen. The latter case arises if either $d^2\hat{y} \leq \phi(b^2 - A^2\hat{x})$ or $\mathcal{R}(\hat{x}) = \emptyset$. When $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$, we must have $d^2\hat{y} > \phi(b^2 - A^2\hat{x})$ and $\mathcal{R}(\hat{x}) \neq \emptyset$. When $(\hat{x}, \hat{y}) \notin \mathcal{S}$, we may be in case C1 (the optimality conditions for the second-level problem that were relaxed in (LR) are already satisfied by $(\hat{x}, \hat{y})$) and we cannot use MIBLP inequalities to separate $(\hat{x}, \hat{y})$. As a simple example, consider what happens in the example from Figure 1 if the first-level objective is changed to minimizing the value of $3x - y$. Then the optimal solution to the bilevel problem is $(1, 2)$ and the solution to the relaxation (LR) is $(0, \frac{3}{2})$. Since $\phi(0) = +\infty$, it is obvious that no MIBLP cut can be generated (obviously, we have the option of separating $(\hat{x}, \hat{y})$ with inequalities valid for $\mathcal{S}$ (MILP cuts)). If $\hat{y} \in Y$ but $\hat{x} \notin X$, then separation fails only if $\hat{y} \in \mathcal{R}(\hat{x})$ (there is no improving solution). More analysis of the prevalence of failures in generating ICs is given later in Section 6.4.4.

Example. Figure 5 shows the generated ISIC (the red line) for removing the extreme point (2,4) of $\mathcal{P}$ in the example shown in Figure 1. The blue cone and the green dotted region show $\mathcal{V}(2, 4)$ and set $\mathcal{C}$, respectively. Note that in this case, the generated inequalities and sets $\mathcal{C}$ are the same for both described types of ISIC.

4.3.2 Improving Direction Intersection Cuts

Assumptions.

- $A^2x + G^2y - b^2 \in \mathbb{Z}^{m_2}$ for all $(x, y) \in \mathcal{S}$.
- $d^2 \in \mathbb{Z}^{n_2}$.

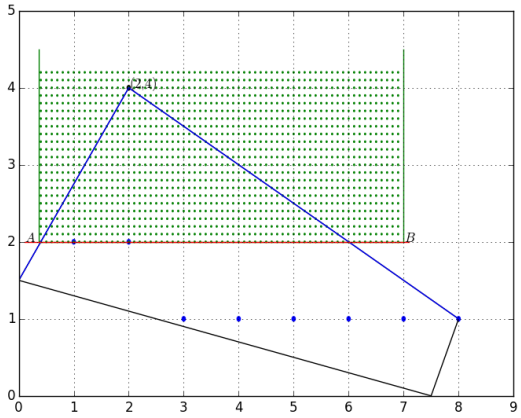


Figure 5: The generated ISIC for the example shown in Figure 1

Theorem 8 (Fischetti et al. [2017a]). *Let* $(\hat{x}, \hat{y}) \in \mathcal{P} \setminus \mathcal{F}$ *be an optimal solution to* (LR) *and* $\Delta\hat{y} \in \mathbb{Z}^{r_2} \times \mathbb{R}^{n_2 - r_2}$ *satisfy these conditions:*

- $d^2 \Delta\hat{y} < 0$.
- $\hat{y} + \Delta\hat{y} \in \mathcal{P}_2(\hat{x})$.

Then, under the stated assumptions, we have

$$\alpha^x x + \alpha^y y \geq \beta \quad \forall (x, y) \in \mathcal{F}, \tag{19}$$

where the inequality (19) *is the IC generated from the cone* $\mathcal{V}(\hat{x}, \hat{y})$ *and*

$$\mathcal{C} = \left\{ (x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; A^2 x + G^2 (y + \Delta\hat{y}) \geq b^2 - 1, y + \Delta\hat{y} \geq -1 \right\}. \tag{20}$$

Furthermore, the inequality (19) *is violated by* $(\hat{x}, \hat{y})$.

Discussion. Generating an IDIC requires finding an improving direction $\Delta\hat{y} \in \mathbb{Z}^{r_2} \times \mathbb{R}^{n_2 - r_2}$ for the second-level problem, as in Definition 1 and Theorem 8. Given an improving direction, an IC can be derived from the associated BFS defined in (IS-BFS). As with ISICs of type II, the aim is

to find the largest possible convex set and this is attempted by solving the following subproblem.

$$\begin{aligned}\Delta\hat{y} \in \text{argmax} & \sum_{i=1}^{m_2} w_i + \sum_{i=1}^{n_2} v_i \\ & d^2\Delta y \leq -1 \\ & G^2\Delta y \geq b^2 - A^2\hat{x} - G^2\hat{y} \\ & \Delta y \geq -\hat{y} \\ & G^2\Delta y \geq w \\ & \Delta y \geq v \\ & \Delta y \in \mathbb{Z}^{r_2} \times \mathbb{R}^{n_2 - r_2} \\ & w \leq 0 \\ & v \leq 0.\end{aligned} \tag{21}$$

The optimal values of the vectors w and v can be employed to drop some of the inequalities in the definition of set (20), enlarging our convex set further. $w_i^* = 0$ ($v_i^* = 0$, resp.) means that $g_i^2(y + \Delta\hat{y}) \geq b_i^2 - a_i^2 x$ ($y_i + \Delta\hat{y}_i \geq 0$, resp.) for all bilevel feasible solutions, so this inequality can be removed from the definition of set (20).

As with ISICs of types I and II, the IDIC makes no assumption about the point to be separated and it can thus be used to separate points $(\hat{x}, \hat{y})$ for which $\hat{x} \notin X$ and/or $\hat{y} \notin Y$. Also as with ISICs, separation can fail in the same two ways: $\mathcal{P}$ may be entirely contained in the BFS or (21) may be infeasible. In the latter case, the infeasibility can be a result of any one of *three* conditions:

1. $\mathcal{R}(\hat{x}) = \emptyset$,
2. $d^2\hat{y} \leq \phi(b^2 - A^2\hat{x})$ (we are in case C1), or
3. $d^2\hat{y} > \phi(b^2 - A^2\hat{x})$ but there is no improving direction satisfying the integrality conditions of (21).

When $\hat{y} \in Y \setminus \mathcal{R}(\hat{x})$, then (21) must be feasible.

To illustrate these concepts, it's useful to consider again what happens in the example of Figure 1 if the first-level objective is changed to minimizing the value of $3x - y$. As before, the optimal solution to the MIBLP is $(1, 2)$ and the solution to the relaxation (LR) is $(0, \frac{3}{2})$. In this case, it is obvious that no cut can be generated simply because it is clear that no improving direction exists, which means (21) is infeasible. The generation of improving directions when $(\hat{x}, \hat{y}) \notin \mathcal{S}$ will play an important role in the computational experiments later.

The close connection between ISICs and IDICs is evident in the fact that if $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$ is an improving solution, then $y^* - \hat{y}$ is an improving direction. Importantly, however, it can be the case that $y^* - \hat{y} \notin \mathbb{Z}^{r_2} \times \mathbb{R}^{n_2 - r_2}$ and this is the reason why some points can be separated by ISICs but not by IDICs. In fact, neither class of inequalities strictly dominates the other, as one can observe in Figures 5 and 6.

Example. Figure 6 shows the generated IDIC (the red line) for removing the extreme point (2,4) of $\mathcal{P}$ in the example shown in Figure 1. The blue cone and the green dotted region show $\mathcal{V}(2,4)$ and set $\mathcal{C}$, respectively. Note that the points in the triangular region that is neither in the BFS nor in conv($\mathcal{F}$) are the points that can be separated by an ISIC in this case, but not an IDIC.

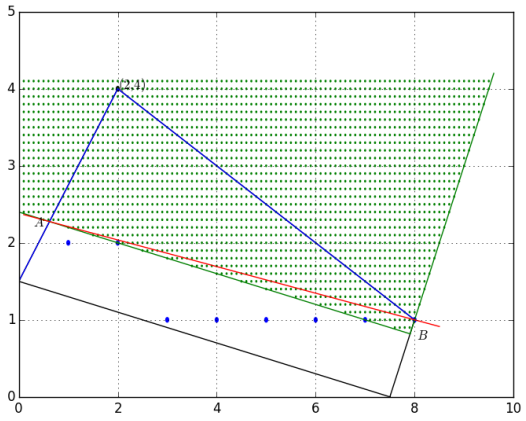


Figure 6: The generated IDIC for the example shown in Figure 1

4.3.3 Hypercube Intersection Cut

Assumptions.

- None.

Theorem 9 (Fischetti et al. [2017a])**.** *Let $(\hat{x}, \hat{y}) \in \mathcal{P}$ be an optimal solution of (LR) with $\hat{x}_L \in \mathbb{Z}^L$. Then, under the stated assumption, we have*

$$\alpha^x x + \alpha^y y \geq \beta \quad \forall (x,y) \in \mathcal{F} \text{ such that } x_L \neq \hat{x}_L, \tag{22}$$

where the inequality (22) is the IC generated from the cone $\mathcal{V}(\hat{x}, \hat{y})$ and convex set

$$\mathcal{C} = \left\{ (x,y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; \hat{x}_i - 1 \leq x_i \leq \hat{x}_i + 1 \quad \forall i \in L \right\}.$$

Furthermore, the inequality (22) is violated by $(\hat{x}, \hat{y})$.

Discussion. Since this inequality may eliminate bilevel feasible solutions for which $\gamma = \hat{x}_L$, the problem (UB) must first be solved with $\gamma = \hat{x}_L$ prior to generating such a cut. In this way, we ensure that all removed bilevel feasible solutions will be non-improving. As with previous classes, we can attempt to generate such a cut even when $\hat{y} \notin Y$. Even when (UB) is infeasible, which can happen, the inequality is still valid.

Example. Figure 7 shows the generated hypercube IC (the red line) for removing the extreme point (2,4) of $\mathcal{P}$ in the example shown in Figure 1. The blue cone and the green dotted region show $\mathcal{V}(2,4)$ and set $\mathcal{C}$, respectively.

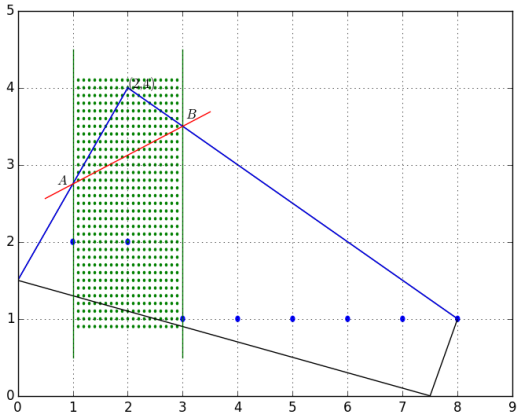


Figure 7: The generated hypercube IC for the example shown in Figure 1

4.4 Generalized No-good Cut

Assumptions.

- $x_L \in \mathbb{B}^L$.

Theorem 10. *Let $\gamma \in \mathbb{B}^L$. Then, under the desired assumption, we have*

$$\sum_{i \in L:\gamma_i=0} x_i + \sum_{i \in L:\gamma_i=1} (1 - x_i) \geq 1 \quad \forall (x, y) \in \mathcal{F} \text{ such that } x_L \neq \gamma. \tag{23}$$

Furthermore, the inequality (23) is violated by all $(x, y) \in \mathcal{P}$ with $x_L = \gamma$.

Proof. When $x_L = \gamma$, we have $\sum_{i \in L:\gamma_i=0} x_i = 0$ and $\sum_{i \in L:\gamma_i=1}(1 - x_i) = 0$, which follows that

$$x_L = \gamma \Rightarrow \sum_{i \in L:\gamma_i=0} x_i + \sum_{i \in L:\gamma_i=1} (1 - x_i) = 0. \tag{24}$$

Furthermore, when $x_L \neq \gamma$, at least one of the following happens

- $\exists i \in L$ such that $\gamma_i = 0$ and $x_i = 1 \Rightarrow \sum_{i \in L:\gamma_i=0} x_i \geq 1$.
- $\exists i \in L$ such that $\gamma_i = 1$ and $x_i = 0 \Rightarrow \sum_{i \in L:\gamma_i=1}(1 - x_i) \geq 1$.

Hence, we have

$$x_L \neq \gamma \Rightarrow \sum_{i \in L:\gamma_i=0} x_i + \sum_{i \in L:\gamma_i=1} (1 - x_i) \geq 1. \tag{25}$$

The result follows from (24) and (25). □

Discussion. The generalized no-good cut is the only projected optimality cut currently in `MibS` and is a generalization of the no-good cut suggested by DeNegre [2011]. The cut proposed in DeNegre [2011] removes all solutions with the same first-level values, while the new version strengthens the former one by separating the solutions with the same linking part. Moreover, since the no-good cut is valid only for the problems in which all first-level variables are binary, the assumption of the generalized one is less strict.

As with the hypercube IC, the corresponding problem (UB) should be solved prior to generating a generalized no-good cut. In contrast with the hypercube ICs, it is guaranteed that the generalized no-good cut eliminates all solutions with the target linking values. Unlike the generalized no-good cut, however, the hypercube IC is applicable for general MIBLPs.

The generalized no-good cut can be seen as a disjunctive cut arising from the disjunction

$$x_L = \gamma \qquad \text{OR} \qquad \sum_{i \in L:\gamma_i = 0} x_i + \sum_{i \in L:\gamma_i = 1} (1 - x_i) \geq 1.$$

If we have already solved the problem (UB) with the linking variables fixed to γ, we can conclude that the disjunctive term on the left above can contain no *improving* solutions. Therefore, we can impose the term on the right as a valid inequality. Technically speaking, this is then actually a generalized Chvátal inequality with the nonzero weights being on the bounds of the linking variables.

4.5 Stronger Valid Inequalities

As in the MILP case, there is a limit to the strength of inequalities that can be obtained through "simple" procedures, such as the ones described in the last section. In the case of MILPs, cut generation procedures are usually limited to those with running times that are polynomial in the encoding size of the given instance (generally at most solving an LP), though it is clear from the equivalence of optimization and separation [Grötschel et al. (1993b)] that inequalities generated by such procedures will necessarily be relatively weak in general. Of course, recursive application can obviously lead to much stronger inequalities (those of higher *rank*).

Since MIBLPs are hard for the complexity class Σ_2^{P}, little can be expected from separation procedures with polynomial running times. Most of the procedures described in this paper require instead the solution of problems that are $\mathsf{NP}-$hard (solving an MILP subproblem). Once again, because of the equivalence of optimization and separation, even procedures involving solving subproblems that are MILPs cannot, in general, be expected to produce strong inequalities, except through recursive procedures that yield inequalities of higher rank. Even inequalities generated by solving MILP subproblems can be weak because the relaxation (LR) is weak to begin with. The weakness of the relaxation (LR) comes mainly from relaxing the optimality conditions for the second-level problem that require

$$d^2 y \leq \phi(b^2 - A^2 x) \; \forall (x, y) \in \mathcal{F}. \tag{26}$$

Thus, all valid inequalities that are cuts with respect to the relaxation (LR) can be thought of as somehow approximating these optimality conditions by imposing a collection of weaker (linear) inequalities.

To obtain inequalities stronger than the ones discussed herein, it is necessary to consider subproblems more difficult (in principle) than simple MILPs. The most straightforward approach is to focus on Benders inequalities of the form (9) in which the value function ϕ is replaced by an upper approximation (a so-called *primal function* (see, e.g., [Bolusani, Coniglio, et al. (2020)]). Tahernejad [2019] investigated the generation of one such class of inequalities in which d^2y is bounded above by a constant. This requires the solution of a (simpler) bilevel optimization problem, which can be solved approximately to obtain a parametric class of inequalities that can be strengthened after branching. Another class of Benders inequalities utilized in a pure Benders algorithm is described in Bolusani and Ralphs [2022]. These inequalities require the recursive approximation of the full value function. In general, the question of how to generate such stronger inequalities efficiently is very much an open one.

4.6 Comparing Classes Analytically

Having now discussed in detail the specific classes of inequalities implemented in `MibS`, we summarize by grouping and comparing the various classes discussed. A rough comparison can be done by considering the size of the full set of bilevel infeasible solutions that are guaranteed to be violated by a given inequality. Obviously, this is a rough way of comparing, but we will see that this rough guide does in fact align well with the empirical results reported in Section 6. Roughly speaking,

- Generalized Chvátal cuts are guaranteed to remove only a single point $(x, y) \in \mathcal{S} \setminus \mathcal{F}$
- HICs and Generalized no-good cuts remove all $(\hat{x}, y) \in \mathcal{S}$ (feasible or not) for some $\hat{x} \in X$ (once $\Xi(\hat{x})$ is computed by solving (UB)), i.e., all $(\hat{x}, y) \in \mathcal{F}$ for **fixed** $\hat{x}$.
- Benders cuts and ISICs remove all $(x, y) \in \mathcal{P}$ such that $y^* \in \mathcal{P}_2(x)$ and $d^2y > d^2y^*$, i.e.,, all (x, y^*) for which a **fixed** y^* provides a certificate of infeasibility.
- IDICs remove all $(x, y) \in \mathcal{P}$ such that Δy is an improving feasible direction for y, given x, i.e., all $(x, y) \in \mathcal{P}$ for which **a fixed** Δy provides a certificate of infeasibility.

Very roughly speaking, we may expect that the set of solutions removed by IDICs should be larger, given that the set includes solutions without either x or y fixed to a particular value. The intuition is in fact borne out in practice, as we show in the second part of the paper.

5 Implementation Details

`MibS` is built on top of the `BLIS` [Xu and Ralphs (2022)] MILP solver, which is in turn built on the `CHiPPS` [Ralphs et al. (2004)] framework. `BLIS` provides the standard control mechanisms and features of a modern implementation of branch-and-cut for solution of MILPs. Most importantly, this includes parameters for managing the LP relaxation (primarily, controlling the addition and removal of cuts). There are, however, some significant differences between the mechanisms for managing the generation of valid inequalities in the MILP and MIBLP cases. One of the key takeaways of the results presented here is that the performance of these inequalities is intricately tied to and affected by details of the overall solution strategy (particularly branching) of the underlying

MILP solver to a greater extent than in the MILP case. For this reason, it appears to be much more difficult to tease out the effect on performance of the individual components of the algorithms, especially cut generation. In the remainder of this section, we briefly discuss important control parameters that all must be tuned to achieve performance. We defer until Section 6 detailed discussion of empirical results that lead to the current set of default parameters.

5.1 Cut Generation Parameters

Standard Control Parameters. The underlying solver framework `BLIS` provides a complete set of control parameters similar to what one would find in other MILP solvers. We mention here only the most important ones for our purposes. Specifically, there are parameters that

- control whether inequalities of a particular class should be generated at all and specify strategy for generation (various parameters `cut*Strategy`);
- limit the number of cut passes (`cutPass`);
- limit the number of cuts that can be added to the relaxation in total (`cutFactor`, relative to the original number of constraints);
- control whether cuts get added based on their dynamism (`scaleConFactor`; see Dey and Molinaro [2018]);
- control whether cuts are added based on their density (`denseConFactor`); and
- control *tailoff detection* (`tailOff`).

More will be said about the specific effect of these parameters later.

MIBLP-specific Control Parameters. In addition to the above standard parameters, there are parameters specific to `MibS` that further control cut generation. The determination of whether or not to attempt to generate inequalities of a particular class in a given iteration at a given node is also a function of

1. the integrality status of the current solution (whether the linking variables and/or second-level variables are integer-valued); and
2. whether the second-level problem has been solved with respect to the first-level part of the solution to the LP relaxation, which is in turn controlled by another set of parameters.

We discuss Item 2 in more detail below in Section 5.2; Item 1 relates to the fact that some inequalities can only be used to separate a solution to the relaxation from the feasible region when the solution has certain structure, as described in Section 4. The structure required for the inequalities described in this paper is as follows.

- $(\hat{x}, \hat{y}) \in \mathcal{S}$: Integer no-good, Generalized Chvátal.

- $\hat{x}_L \in \mathbb{Z}^L$: Benders binary, Benders interdiction, Generalized no-good, Hypercube IC.
- $(\hat{x}, \hat{y}) \in \mathcal{P}$: ISIC of types I and II, IDIC (though separation is not guaranteed when $(\hat{x}, \hat{y}) \notin \mathcal{S}$, as discussed earlier).

For all classes except for ISICs and IDICs, we have no choice but to separate only solutions with the specified structure and this means waiting to generate cuts until a solution to the LP relaxation with such structure arises. It is possible this will not happen with high frequency, which limits the impact of generating inequalities of these classes.

The situation is different with ISICs and IDICs. Because these cuts are relatively expensive to generate (requiring an MILP oracle call in general), the strategy for generating them should take into account not only the difficulty of the oracle problem, but the probability of success at separating the solution. As we have previously mentioned, cut generation is only guaranteed to succeed when the solution is in $\mathcal{S}$ (fully integral) and this conservative policy of only attempting to separate fully integer solutions was the one taken in `MibS 1.1`. While this seemed like a logical approach at the time and we had not questioned it, we realized during the writing of this paper and the development of the new version of `MibS` that restricting generation of cuts *only* to iterations where the solution is in $\mathcal{S}$ severely limits the frequency of cut generation and therefore the effectiveness.

`MibS 1.2` has additional parameters controlling precisely what types of points the solver should (attempt to) separate. Solutions in $\mathcal{S}$ are always separated. Separation of other types of solutions are controlled by which of the following strategies is chosen. For the present, we offer six strategies for generating intersection cuts.

- `Always`: Try to separate every iteration.
- `AlwaysRoot`: Try to separate every iteration, but only in the root node.
- `XYInt`: Try to separate when $(\hat{x}, \hat{y}) \in \mathcal{S}$.
- `LInt`: Try to separate when $\hat{x}_L \in \mathbb{Z}^L$.
- `YInt`: Try to separate when $\hat{y} \in Y$.
- `YLInt`: Try to separate when either $\hat{x}_L \in \mathbb{Z}^L$ or $\hat{y} \in Y$.

The `Always` strategy results in the maximum number of cuts being generated (and the smallest search trees) but can also result in wasted effort when cut generation failure rates are high. On the other end of the spectrum, the `XYInt` strategy ensures a failure rate near zero, but also means many fewer cuts are generated. We analyze the empirical performance of these strategies in Section 6.

Finally, it should be mentioned that there are situations in which cut generation is mandatory and parameters settings must be over-ridden. Most prominently, when branching only on variables with fractional values, we are required to generate a cut whenever $(\hat{x}, \hat{y}) \in \mathcal{S} \setminus \mathcal{F}$. In such a case, all parameters that would prevent a cut from being generated or added are ignored.

5.2 Controlling Oracle Calls

The implementation of the overall bounding loop was described in Algorithm 1 of Tahernejad et al. [2020]. In this scheme, parameters control when the feasibility of a given solution is checked. Unlike in the MILP case, checking feasibility is an expensive operation and it can be advantageous to delay it, even if that may results in additional branching.

Similarly, the generation of most classes of inequalities described here requires a call to an MILP oracle. In the case of Benders binary cuts, Benders interdiction cuts, Generalized no-good cuts, ISICs of type I, and Hypercube ICs, the oracle is the very same one that is used to check feasibility, which ties the feasibility check to cut generation in an important way.

ISICs of type II and IDICs have their own associated oracle calls, which currently serve no other purpose and is called only when cuts are to be generated. The oracle that solves the second-level problem must be called either when we want to check feasibility of a solution *or* when a cut is to be generated. This has the apparently strange consequence that we may sometimes solve the second-level problem even when $\hat{x} \notin X$. When $\hat{x} \in X$, the effect of solving the second-level problem is to produce a point $y^* \in \mathcal{R}(\hat{x})$ so that $(\hat{x}, y^*) \in \mathcal{F}$ and can hence be treated as a new heuristic solution for the purpose of improving the global upper bound as a side benefit. When $\hat{x} \notin X$, then $(\hat{x}, y^*) \notin \mathcal{F}$, but y^* can still be used as an input to generate cuts.

`MibS` attempts to generate cuts requiring a second-level solution only when that solution is already available. This means that if parameter settings for cut generation are such that points for which $\hat{x} \notin X$ are to be separated, we must also ensure that the second-level problem is solved, though it would not otherwise be.

In Tahernejad et al. [2020], we determined that doing the feasibility check only when the $(\hat{x}, \hat{y}) \in \mathcal{S}$ or when the values of all linking variables have been fixed (and we can thus prune the node by solving (UB)) was the most advantageous default settings. However, we noted in this study that this results in certain classes of inequalities, e.g., Hypercube ICs and ISICs of type I, being rarely or never generated in many of the instances in our test set. We thus experimented with solving the second-level problem whenever linking variables were integer-valued and we report on those experiments in Section 6.

5.3 Branching Strategy

Although this paper is ostensibly about valid inequalities, the branching strategy employed has an intricate effect both on the generation of valid inequalities and the overall effectiveness of the algorithm. The underlying solver provides a number of the standard strategies and control parameters for branching that are typically used in solving MILPs. Version `1.2` of `MibS` uses a basic pseudo-cost strategy [Bénichou et al. (1971)] to select the final branching candidate by default and implements three strategies for determining the set of candidates for branching.

1. Select the branching candidate from among all variables with fractional value in the solution to the current LP relaxation (we refer to this as `fractional` branching).
2. Prioritize branching on linking variables, only branching on non-linking variables if the values of all linking variables are fixed by branching constraints (we refer to this as `linking` branch-

ing). This means we may need branch on linking variables whose values are not fixed but are integer in the solution to the LP relaxation. The mechanisms for this and the reasons why branching on integer variables may be necessary in solving MIBLPs is discussed in Tahernejad et al. [2020].

3. Prioritize branching on lower-level variables, as long as some lower-level variable has a fractional value (the intuition behind the possible efficacy of this strategy is explained below).

There are several goals that are implicitly being traded off in the choice of branching strategy and we reserve detailed discussion of how these strategies behave empirically for Section 6.4.1.

6 Empirical Analysis

In this section, we present detailed empirical analyses. The goals of the experiments are several-fold. First and foremost, we aim to provide an objective evaluation of the effectiveness of generating inequalities from the classes we've discussed. Separation routines for all of the described classes of valid inequalities have been implemented in the open-source solver `MibS` version `1.2.2`. Extensive experiments were conducted in order to gain insight into the performance of the different classes of valid inequalities for MIBLPs. In addition, we've tried to gain insight into how the strategy for cut generation interacts with the other elements of the branch-and-cut algorithm, particularly the branching strategy. Although the theory presented parallels that of the MILP case in many respects, the practical aspects of cut generation for MIBLPs are not as closely aligned as one might expect in a number of ways. As we've described and further emphasize below, however, teasing out the overall effect of generating valid inequalities is not easy.

We emphasize that this work should only be considered as the first step towards a long-term goal of understanding how to manage the cut generation procedure within a general branch-and-cut algorithm for MIBLPs. For the MILP case, this maturation process involved many research works and the accumulation of experience by many solver developers. Though this vast trove of knowledge about the MILP case gave us a head start on the MIBLP case, there is good reason to believe that the MIBLP case also requires further deep investigation and the accumulation of a similar degree of empirical experience. What we present here is a further step along this path, but likely raises as many questions as it answers.

The remainder of this section is organized as follows. In Section 6.1, we discuss the goals of the experiments and how we measured performance. This is a bit different and more difficult than in the MILP case and so requires discussion. Following that, in Section 6.2, we discuss the test set and its properties, after which we describe the experimental setup in Section 6.3. Finally, in Section 6.4, we discuss the empirical results.

6.1 Measures of Performance

The effectiveness of generating a particular class of inequalities is more difficult to measure in the MIBLP setting than in the MILP one, but we first review a number of common measures of performance for classes of inequalities that are typical in the MILP setting. Perhaps the most

objective measure of performance, because it is independent of any other details of the algorithm, is the bound obtained by optimizing over the so-called closure associated with the given class of valid inequalities. Informally, this closure is the region described by the initial relaxation strengthened by all inequalities in the class (see Lodi et al. [2014] for a discussion). It is thus a measure of the maximum amount of improvement in the relaxation bound that can be achieved in the root node. The closure bound has some drawbacks. First, it is sometimes not easily computed and can require a separately designed algorithm (see, e.g., [Balas and Saxena (2008)]). Second, it may or may not be indicative of performance in practice, since the closure bound may only be achieved as a result of the interaction of particular sets of inequalities in the class and may or may not actually be achievable in practice. And of course, as with all empirical measure, the closure bound is computed for each instance individually, so results depend on the test set and general conclusions may be difficult to draw.

The appeal of the closure bound is its objectivity, but in practice, more subjective measures may be a better indicator of practical effectiveness. Common empirical measures include (1) the improvement in the bound achieved in the root node using the standard root processing of the branch-and-cut solver within which the testing is taking place; (2) the reduction in the size of the branch-and-bound tree when generating valid inequalities from a given class over a baseline without; and (3) the reduction in empirical running time with and without cut generation with respect to a similar baseline. The difficulty with these measures is that it is much more difficult to separate out the effects of ostensibly unrelated algorithmic parameter settings, particularly the branching strategy.

In version `1.1` of `MibS`, cut generation could only be done once integrality of either the upper-lever or of both upper- and lower-level solution had been achieved, which typically required branching. It also resulted in relatively few cuts being generated and made root bound improvement a largely meaningless measure. As a result of the possibility of separating arbitrary fractional solutions, the processing of the root node in `MibS 1.2` plays a role more similar to its role in MILP, where the strategy is generally to expend substantial effort in improving both the primal and dual bounds before switching to a phase in which there is more focus on enumeration. Essentially, root processing can be seen as a pre-processing step in which the initial formulation is tightened as much as possible.

Nevertheless, some classes of inequality can still only be used (or at least are most effectively used) to separate points in $\mathcal{S}$ or with specific structure, while it is typically the case that solutions generated in the root node are not members of $\mathcal{S}$ (i.e., some integer variables have fractional values). For all these reasons, we rely on a "holistic" collection of measures: root bound improvement (when appropriate), the time to optimality (for solved instances), the size of the branch-and-bound tree (for solved instances), and the final optimality gap (for unsolved instances). These measures are summarized using several different kinds of profiles, each of which reveals a different aspect of the results.

- *Performance profiles* are empirical cumulative distribution functions (CDFs) of ratios of a given performance measure against the virtual best [Dolan and Moré (2002)]. We use performance profiles primarily for comparing CPU time across instances that could be solved to optimality.
- *Baseline profiles* are similar to performance profiles, but present empirical CDFs of ratios of a given performance measure against a fixed baseline rather than against the virtual best. We

use baseline profiles for comparing both the reduction in tree size and the reduction in root gap, since there is a natural baseline (the tree size/root gap when solving with no MIBLP cuts).

- *Cumulative profiles* are a combination of two profiles. On the left side of the plot is the empirical CDF of the fraction of instances solved within a given time interval. On the right side is the empirical CDF of the fraction of instances that achieved a given final gap within the time limit (the lines on the two halves of the graph connect because the fraction of instances solved within the time limit is the same as the fraction of instances with zero gap at the time limit). We use cumulative profiles to compare performance across all instances, including those that could not be solved within the time limit.

We filtered the instances as follows.

- For all profiles, we excluded instances that were solved in less than 1 second by all methods and instances that were solved in less than .01 seconds by any one method,
- For performance and baseline profiles, we additionally dropped instances that could not be solved by any of the methods.
- For cumulative profiles, we dropped instances for which no solution was found by any method.

6.2 Test Set

Perhaps as important a factor in performance as the details of the implementation is the specific classes of problems under consideration. As with MILPs, specific classes of inequalities can behave very differently on different data sets. The seven data sets employed in our testing have differing properties summarized in Table 1. (Note that there is now a standard library of test instances called BOBILib [Thürauf et al. (2024)] that includes these instances and more, but this instance library came out after most of these experiments had been completed). The columns of Table 1 have the following interpretations. The first column is the name used to denote the data set in the rest of this paper. The second is the number of instances in the data set. The third is the type of variables in the first and second levels (B indicates binary, I indicates general integer and C indicates real-valued/continuous). The fourth and fifth columns indicate the number of variables and constraints (upper and lower limits of the range over all instances in the data set). The sixth column indicates the degree of alignment of the objectives (a normalized inner product between -1 and 1, with -1 indicating the objectives are exactly opposed and 1 indicating they are exactly aligned).

There are some important properties that affect problem difficulty, especially properties that determine whether particular classes of inequalities apply. Of course, one very important property is whether the instance has continuous variables, especially at the lower level. Pure integer instances with all integer coefficients in the constraint matrices have a much wider range of cuts available. The signs of the coefficients is also an important factor. Instances in which the lower-level constraints are "$\leq$" and have all non-negative coefficients, for example, tend to be easier, as one might expect, since this is the case even for classical single-level MILP. Instances that have coefficients with mixed signs are generally more difficult to solve.

Data Set	# of instances	Variable Type	# of variables	# of constraints	Degree of alignment	Original source
INT-DEN	300	B B	10-40 10-40	1 11-41	-1	Interdiction [DeNegre (2011)]
DEN	50	I I	5-15 5-15	0 20	Varies	[DeNegre (2011)]
DEN2	110	I I	5-10 5-20	0 5-15	Varies	[DeNegre (2011)]
ZHANG	30	B I	50-80 70-110	0 6-7	0.6-0.8	[Zhang and Ozaltın (2017)]
ZHANG2	30	I I	50-80 70-110	0 6-7	0.6-0.8	[Zhang and Ozaltın (2017)]
FIS	57	B B	Varies	Varies	-1	MIPLIB [Fischetti et al. (2018)]
XU	100	I IC	10-460 4-184	10-460 4-184	≈ 0	Mixed [Xu and Wang (2014)]

Table 1: The summary of data sets

The degree of alignment of the objectives is one particularly important factor in determining the difficulty of instances. Instances with alignments near 1 (such as those in ZHANG and ZHANG2) tend to be relatively easy. On other hand, zero-sum instances (alignment of -1) have specialized cuts and are also easier than general instances (interdiction problems in particular are relatively easier to solve than general instances). An alignment of -1 also ensures that any fractional solution to the LP relaxation must violate the optimality constraint and that case C1, as described in the beginning of Section 4 cannot arise. As we discuss later, fractional solutions that satisfy the optimality constraint (those in case C1) may arise quite frequently when the alignment is not -1 and this may be an important issue in managing cut generation in particular cases.

More detailed descriptions of each of the test sets are included in Appendix A.

6.3 Experimental Setup

Hardware Platform. All computational results we report were generated on compute nodes running the Linux (Debian 8.7) operating system with dual AMD Opteron 6128 processors and 32 GB RAM. All experiments were run on a single core with a time limit of 3600 seconds and a memory limit of 8 GB.

Software Versions. `MibS 1.2.2` was employed for conducting all experiments and `SYMPHONY 5.7.2` was employed as the MILP solver for all required subproblems except where noted. Although running times can be improved to some extent by using either Cbc [Forrest (2024)] or a commercial solver (CPLEX [*IBM ILOG CPLEX Optimization Studio* (2024)] is the only currently supported alternative), results presented herein indicated the achieved speedups would not have substantively changed any of the conclusions and since we are actively developing methodologies for solving the

various subproblems more efficiently using SYMPHONY's unique capabilities, we prefer to report results here using SYMPHONY.

Parameters. The only settings that differed between experiments were those for turning specific classes of valid inequalities off/on and those for specifying branching priorities. All parameters of `MibS` and `Blis` were left at their defaults. However, we note here that in a number of cases, default parameter settings in `MibS 1.2` were changed from those in `1.1` and some of these now over-ride the default parameters in `BLIS` as a result of the extensive testing done here. We report those separately here.

- The pseudocost branching strategy was used to choose the best variable among the branching candidates. We have not put much effort so far into tuning the branching strategy (see empirical analysis of branching strategy in Section 6.4.1).
- The generation of generic MILP inequalities by the Cut Generation Library, which was previously disabled is now enabled (see the empirical analysis backing this choice in Section 6.4.5).
- We now do not filter cuts based on dynamism or density (`scaleConFactor` = `denseConFactor` = ∞), since this sometimes results in cuts necessary for correctness being rejected. In practice, we did not encounter any numerical issues as a result of this, at least on these instances.
- We generate cuts in every iteration in which they are eligible to be generated except when tailoff detection forces branching and do not limit the number of cut passes (`cutPass` = ∞) or total number of cuts `cutFactor` = ∞
- The tailoff detection mechanism was improved in `MibS 1.2` and is now based on relative improvement in the bound after adding cuts. After testing, the default cutoff was set to `.05`, which seems to provide the best overall performance across all instances in this set.

Scripts for precisely replicating these experiments, raw output files from `MibS`, and spreadsheets with summary statistics are all publicly available in the Github repository of `MibS` [DeNegre, Ralphs, and Tahernejad (2024)].

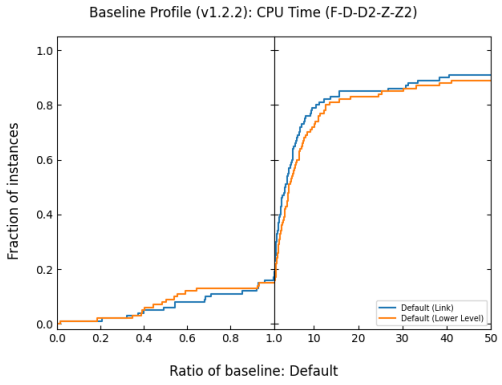
6.4 Results

We now present the summarized results of the experiments.

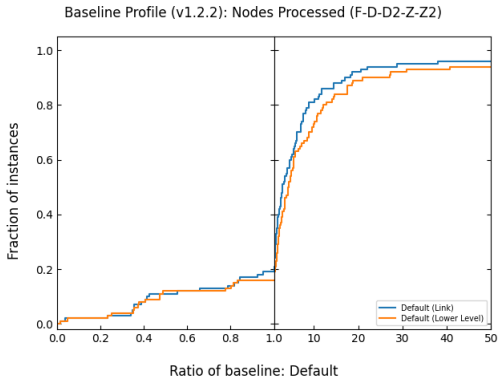
6.4.1 Branching Strategy

As we mentioned in Section 5.3, selection of branching strategy in the case of MIBLP is more intricate than in MILP. In Tahernejad et al. [2020], we concluded that the effectiveness of the branching strategy was affected mainly by the ratio of the number of integer variables at the first and second level. In particular, with that earlier version of `MibS` (version `1.1`), when the number of first level integer variables was smaller than the number of second level integer variables, performance was better when using `linking` branching.

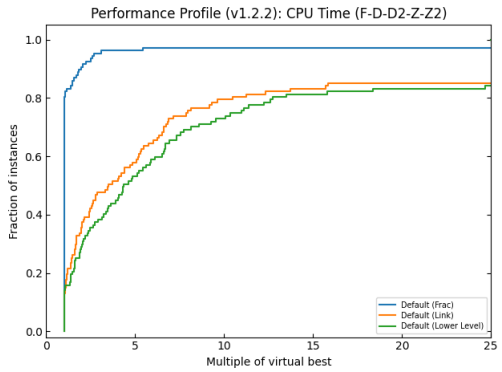
Here, we have revisited that result in light of the completely revamped cut generation strategy. We have tried the three general strategies mentioned already: branching on all fractional variables, prioritizing branching on linking variables, and prioritizing branching on lower-level variables. Both cumulative profiles and baseline profiles, with the baseline being the `fractional` branching strategy, across all instances in our test set (except for the interdiction ones), are shown in Figure 8. It can



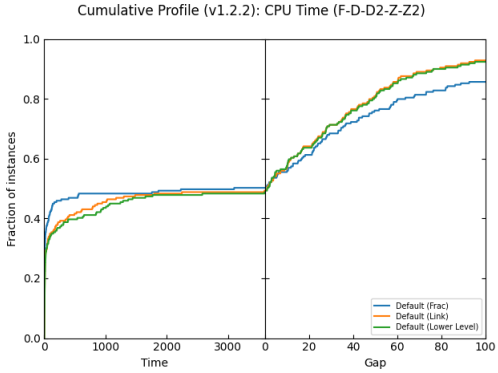
(a) Baseline profile of CPU time



(b) Baseline profile of number of processed nodes



(c) Performance profile of CPU time



(d) Cumulative profile of CPU time

Figure 8: Impact of branching strategy

be clearly seen that branching on all fractional variables is advantageous in the vast majority cases.

These results seem to be in conflict with what was reported in our earlier paper. In particular, branching on all fractional variables is more effective now than it was previously and as a result, we have changed the default branching strategy for all non-interdiction instances across the board.

We have investigated the reasons for this change and they are quite intricate. There are a number of effects that result in complex tradeoffs that must be navigated.

1. As in MILP, the branching variable that is chosen from among the allowed candidates is the one that is predicted to have the biggest impact in terms of bound improvement. Restricting the set of branching candidates may reduce the effectiveness of this strategy by removing what would have been the most effective candidate for branching from consideration.

2. On the other hand, the logic of branching exclusively on linking variables is to reduce the

problem to a (lexicographic) MILP as quickly as possible by fixing the values of all linking variables first. Once all linking variables are fixed, the problem can be handed off to a subsolver. This effectively moves the branching on second-level variables out of `MibS` altogether and down into the subsolver used to solve the second-level problem, which is better suited for the job once the problem is reduced to an MILP. This means an increased emphasis and dependence on oracle calls and perhaps less emphasis on cut generation. In theory, the fact that we are enumerating over just the set of linking variables instead of all variables could result in smaller search trees.

3. A third effect of the choice of branching strategy is that it influences the structure of the solutions produced by solving the LP relaxation in ways that may affect our ability to generate strong valid inequalities. Unlike in MILP, generation of MIBLP cuts is not guaranteed to succeed when the the point being separated is not integral, as we discussed in Section 4.3. It can be argued that the intersection cuts are less likely to fail when the lower-level variables have integer values because this increases the chance that there exists an improving direction/solution. One thing is for certain—both the branching strategy and the separation strategy have a potentially important effect on the failure rate for cut generation, which in turn affects the overall efficiency of the algorithm.

Because of these factors, the three branching strategies lead to substantively different algorithmic behavior and different algorithm emphases. Despite the dominance of `fractional` branching across almost all of our experiments, it is difficult to extract a simple explanation that provides a universal explanation for this result.

Table 2 shows statistics averaged across all non-interdiction, pure integer instances with only IDIC inequalities being generated using the `Always` strategy. The columns have the following meanings:

- The first set of two columns are the averages for total tree size and total number of oracle calls across all instances.
- The second set of three columns shows how the CPU time breaks down between oracle calls and cut generation, with the first column showing total CPU time and the second and third columns showing time spent on oracle calls and generation of MIBLP cuts, respectively (the difference between the sum of the second and third columns and the first column is the time spent in all other parts of the code—solving LP relaxations, branching, generating MILP cuts, etc.—and is typically negligible).
- The final set of three columns shows averages per node for (1) CPU time, (2) number of calls to the MIBLP cut generator; and (3) total number of cuts successfully generated.

Here, the dominance of the `fractional` branching can be clearly seen. Most strikingly, the tree size is smaller by an order of magnitude, despite the much larger search space. This is probably attributable to a combination of the increased effectiveness of choosing from a larger set of branching candidates and the increased effectiveness of cut generation. The number of oracle class is negligible in the case of `fractional` branching, while it is substantial in the case of `linking` branching. This can be attributed to dramatic differences in the number of nodes in which all linking variables have fixed values. This condition forces an oracle call according to current default parameter settings

	Total Number		Total Time			Average Per Node		
Branching Strategy	Nodes	Oracle Calls	CPU	Oracle Calls	CG	CPU	CG Calls	Cuts
Fractional	2226	8	52	0.4	51	.023	1.13	0.23
Linking	24498	1400	175	23	138	.007	0.53	0.07

Table 2: Statistics for solving with different branching rules

in `MibS`. This is also reflected in the average number of rounds of MIBLP cut generation, which is more than one with `fractional` branching, but approximately a half with `linking` branching. And finally, the number of cuts successfully generated and added per node is significantly higher for `fractional` branching. We attribute this to a combination of there simply being more frequent cut generation and also to a higher rate of success. It is unclear whether the cuts generated when branching on linking variables are actually more effective.

Before closing the section, we note that interdiction problems (and, we conjecture, other zero-sum problem as well) are an exception to the general rule because of the very close linking of the first- and second-level variables. For these instances, `linking` branching is a clearly dominant strategy.

6.4.2 Cut Generation Strategy

To test the effectiveness of our cut generation, we first tested the effectiveness of generating each type of valid inequality individually under both of our main branching strategies and, in the case of interdiction cuts, with the different cut generation strategies described in Section 5.1. Based on these results, we also tried combinations of multiple classes of inequalities in order to determine the best default values for cut generating parameters.

In each figure that follows, the classes of inequalities are indicated by the following shorthand.

- `Benders Interdict`: The Benders interdiction cut.
- `Gen No Good`: The generalized no-good cut.
- `Benders Binary`: The Benders binary cut.
- `Int No Good`: The integer no-good cut.
- `ISIC Type1`: Improving solution ICs of type I.
- `ISIC Type2`: Improving solution ICs of type II.
- `IDIC`: Improving direction ICs.
- `Hypercube IC`: The hypercube IC.
- `No Cuts`: No MIBLP cuts generated (integrality cuts are still generated).

For intersection cuts, the specific strategy utilized from those described in Section 5.1 is also indicated as part of the shorthand description. When `link` is in parentheses, the `linking` branching strategy was used and similarly, when `frac` is in parentheses, `fractional` branching was used (we primarily show results with `fractional` branching since this strategy was dominant for almost all classes). We also experimented with prioritizing branching on second-level variables, but those results were not promising. Note that when no cuts are generated, we are forced to branch only on linking variables (e.g., to branch on non-fixed linking variables, even if they have integer values in the relaxation solution).

In most of the experiments, a single class of inequality was turned on and tested with each of the two branching strategies and possibly also with different parameter settings, subject to applicability of each class (based on instance structure). In the case of intersection cuts, all classes were tested with all settings described in Section 5.1, with only the best settings for each class used in the global comparison between classes. Some testing with combinations of multiple classes generated simultaneously was also done in order to determine if there any classes that were "complementary" and have an additive effect. We did find some additive effects, described below, but we did not do extensive testing in this regard. Because of the (very) large number of different parameters settings that were tested, as well as the number of different test sets, we present only summary figures in the main text. Detailed figures for each test set individually are shown in Appendix B.

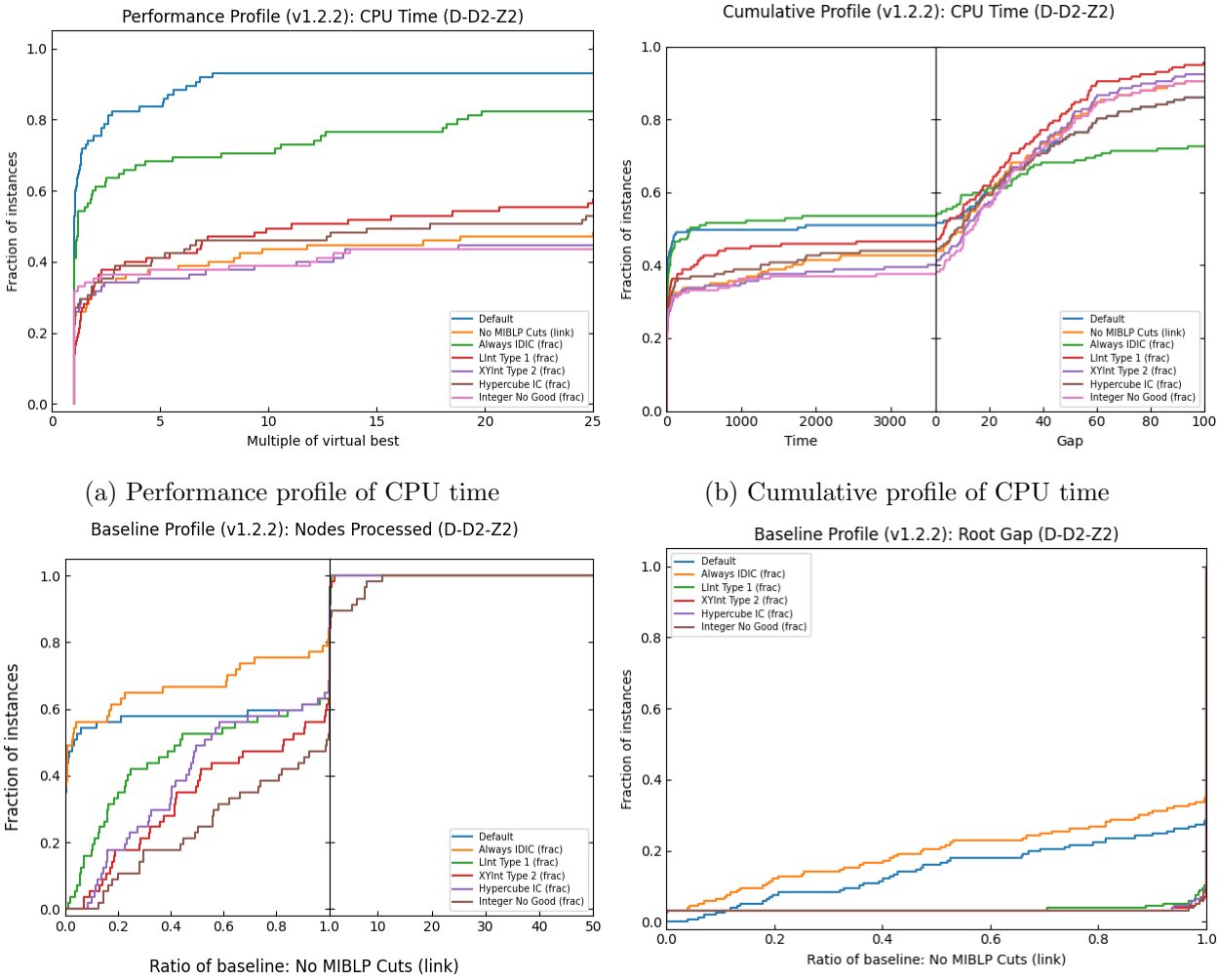
Pure Integer Instances. Figure 9 shows the results on the non-binary, pure integer instances (DEN, DEN2, and ZHANG2) for all applicable classes of inequalities, with IDICs generated using the `Always` strategy; type I ISICs generated with the `LInt` strategy; and type II ISICs generated with the `XYInt` strategy. We also tested with what is now the new default strategy, which uses multiple classes and is described later. The performance profile in Figure 9a shows that among the instances which could be solved by one of the methods, generating IDICs with the `Always` strategy is the clear winner in terms of CPU time and Figure 9c shows that the resulting trees are almost universally smaller. Somewhat surprisingly, no other class on its own does much better than no MIBLP cuts at all, with Hypercube ICs and Integer No Goods faring even worse then the baseline with no MIBLP cuts. On the other hand, Figure 9c shows that all classes reduce the tree size substantially over the case of no MIBLP cuts, so it seems that the issue is that the expense of the cut generation does not pay off.

The fact that IDICs are the most effective inequalities in almost all cases is expected in general, based on the discussion in Section 4.6. What was less clear before running the experiments was whether aggressive generation strategies for ICs would pay dividends. The answer to that question seems to be a resounding yes, despite the high failure rate for generating these cuts (see further discussion below). Note, however, that Figure 15 shows that for the single instance set `IBLP-DEN2`, the IDICs vastly under-perform. The reasons for this are not known, but it can be observed from Figure 15d that root gap closed by IDICs is roughly the same as that closed by Integer No Goods, which are much cheaper to generate. This seems to indicate that IDICs are not effective enough to be worth the computational cost in this case. These instances differ from the others in one obvious way—the constraint matrices have both positive and negative coefficients—and it's possible that this makes the cut generation more expensive for this class.

An interesting observation from Figure 9b is that while IDICs dominate for instances that can be

solved within an hour using IDICs, they are not as effective in general at closing the gap for more difficult instances. ISICs are better for this. This suggests that a dynamic scheme in which IDICs are switched off if they're not effective in the early stages of the algorithm could have an impact.

Based on these results, our default strategy for pure integer instances in `MibS` is to use a combination of IDICs with the `Always` strategy along with ISICs of Type I with the `LInt` strategy. Although ISICs alone do not seems to be effective, the combination of these two classes is superior to any individual class. There is one exception to this default strategy and that is for instances with constraint matrices that have both positive and negative coefficients (like `IBLP-DEN2`). For now, we have chosen to use Hypercube ICs for these instances by default, though this requires more investigation.



(a) Performance profile of CPU time

(b) Cumulative profile of CPU time

(c) Baseline profile of tree size

(d) Baseline profile of root gap

Figure 9: Comparing the performance of different inequalities on the pure integer instances

Binary First-Level Instances. Figure 10 shows the results on the instances with binary first-level variables (FIS and ZHANG) on all applicable classes of inequalities. Here, we have tested

all classes applicable to only binary instances in addition to the previous inequalities applicable to pure integer problems, with the only difference being that type I inequalities are generated using the `YLInt` strategy instead of the `Lint`. For these instances, the performance profile in Figure 10a again shows that among the instances that could be solved by one of the methods, generating IDICs with the `Always` strategy is the clear winner in terms of CPU time and Figure 10c shows that the resulting trees are again almost universally smaller.

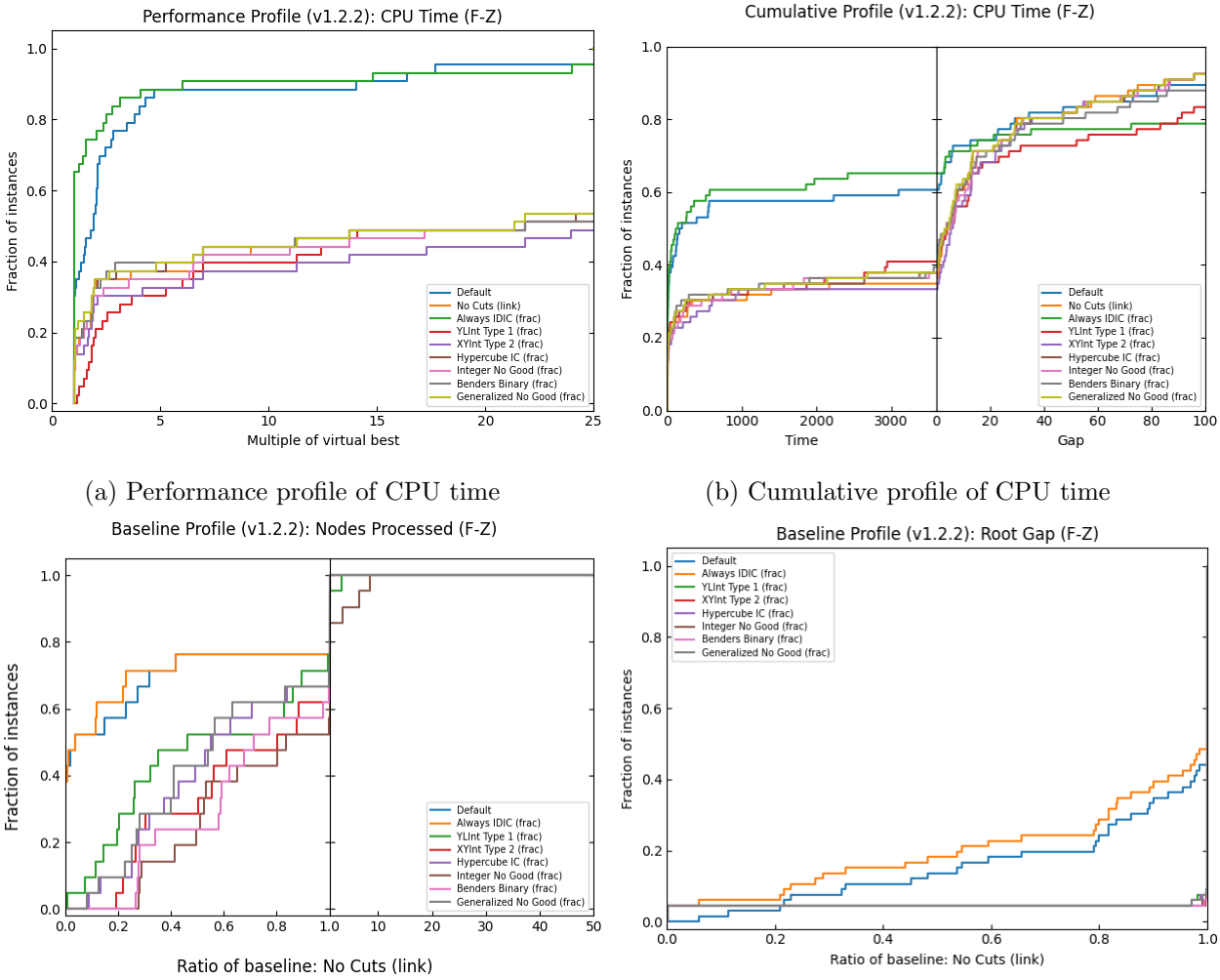
As before, almost no other class outperforms the setting in which we generate no MIBLP cuts in terms of CPU time, but tree size is consistently reduced. The conclusion once again is that the expense of cut generation is not worthwhile in many cases. It may seem surprising that the cuts specialized for the binary case are not as effective as general MIBLP cuts, but this is in line with our earlier analysis. This is a phenomenon worth further investigation, given the effectiveness of methods specialized to the binary case in solving MILPs. We observe once again from Figure 10b that also for these instances, while IDICs dominate for instances that can be solved within an hour, they are not as effective in general at closing the gap for other instances.

Based on these results, our default strategy for pure integer instances in `MibS` is to use a combination of Benders binary cuts, generalized no-good cuts, ISICs of Type I with the `XYInt` strategy, and IDICs with the `Always` strategy. Although the specialized cuts are not as effective as ICs on their own, they are cheap to generate and do improve performance when generated in combination with ICs.

Interdiction Instances. Figure 11 shows the results on the interdiction instances from the `INT-DEN` set with the Benders Interdiction cuts, ISICs, and IDICs (other classes of cuts were clearly dominated and not included to keep the profile from being overly crowded). For these instances, we are also showing the difference between the `fractional` and `linking` branching strategies to highlight that this class of problems indeed has very different properties when it comes to selecting branching variables, owing to the obvious connection between the linking and second-level variables.

For these instances, the performance profile in Figure 11a shows that among the instances that could be solved by one of the methods, generating Benders Interdiction cuts is the very clear winner in all respects. The strength of these cuts has been known for some time and this is not a surprise. `Linking` branching is also clearly dominant. Figure 11c shows that the trees resulting from the generation of cuts of all classes are again universally smaller and the root gap closed is much larger with Benders Interdiction cuts than others, as shown in Figure 11d. All classes close some amount of gap, with the exception of type I ISICs for which the lower bound remains at its initial value of zero. Unlike in other cases, the Benders Interdiction cuts are also able to close the gap effectively on unsolved instances, as shown in Figure 11b.

Based on these results, our default strategy for interdiction instances in `MibS` is to use a combination of BendersbBinary cuts, Benders interdiction cuts, and ISICs of type I with a generation strategy of `Lint`. We do not generate feasibility cuts (MILP cuts) by default, as discussed below.



(a) Performance profile of CPU time

(b) Cumulative profile of CPU time

(c) Baseline profile of tree size

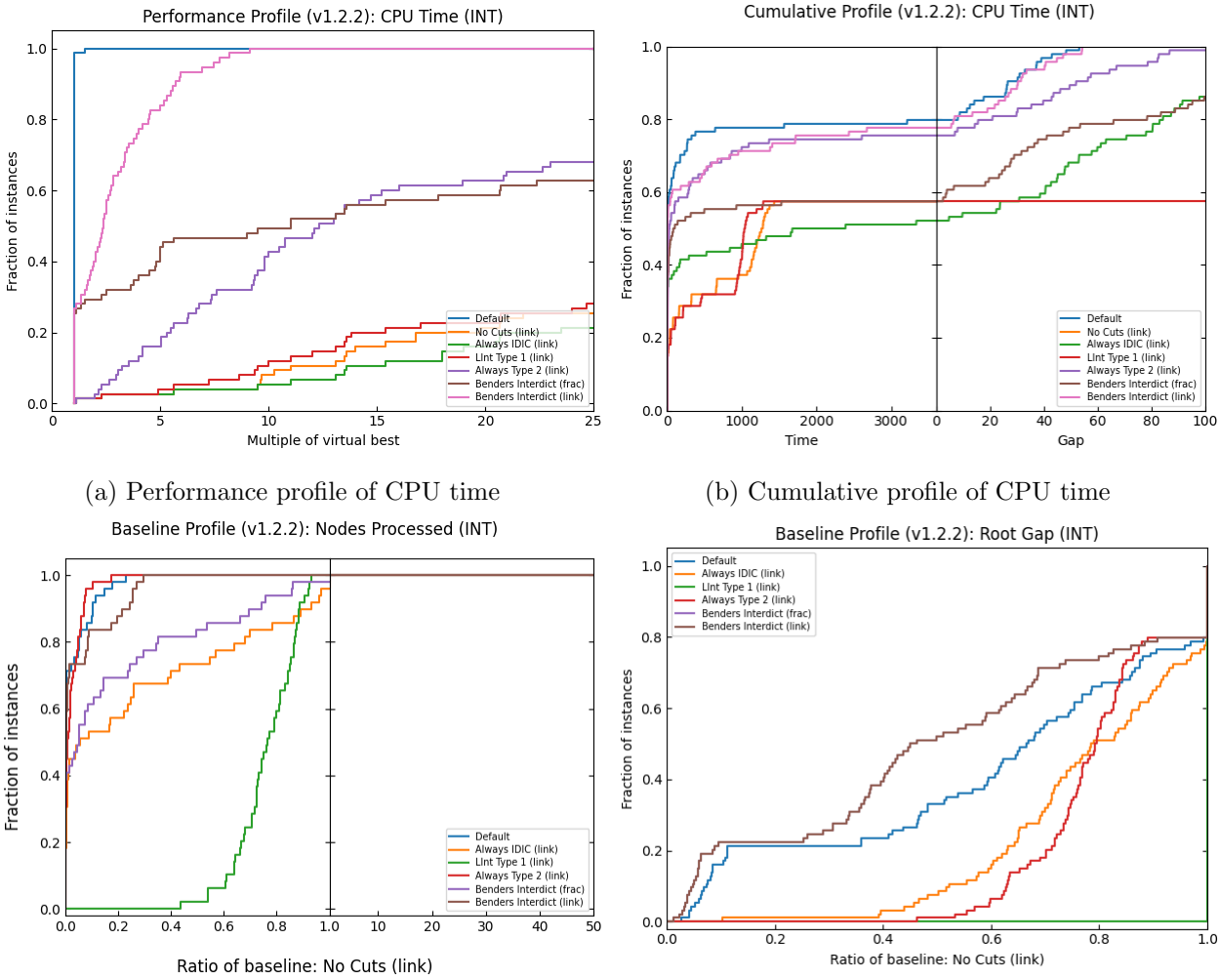
(d) Baseline profile of root gap

Figure 10: Comparing the performance of different inequalities on the binary instances

6.4.3 Strength of Inequalities

We selected randomly 10 instances of the `INT-DEN` and `IBLP-DEN` sets and solved each instance with different inequalities as shown in Tables 3 and 4. For each instance, we considered the first generated cut and compared its right-hand side with the "best possible" right-hand side, obtained by optimizing over $\text{conv}(\mathcal{F})$ with the coefficient vector of the cut as the objective function. These values are shown in the `Orig RHS` and `Best RHS` columns, respectively. The `Obj before cut`, `Obj after orig cut` and `Obj after best cut` columns show, respectively, the objective values of the relaxation problem before adding the first cut, after adding this cut and after adding this cut with the best right-hand side.

As one can observe in Table 3, the right-hand sides of the generated Benders cuts are equal to the best possible right-hand sides for all five instances and it is one of the reasons for the strength of



(a) Performance profile of CPU time

(b) Cumulative profile of CPU time

(c) Baseline profile of tree size

(d) Baseline profile of root gap

Figure 11: Comparing the performance of different inequalities on the `INT-DEN` set

this cut. Furthermore, the right-hand sides of the generated IDICs are closer to their best possible values comparing with the integer no-good cuts and it verifies the results shown in Section 6.4.2.

Note that in Table 4, for the instances 1, 2 and 3, we observed that the bilevel feasible region of the node in which the first cut is generated is empty and the result is that the best value of the right-hand side is ∞ (because the best right-hand side is obtained by minimizing the left-hand side over the bilevel feasible region) and the node should be pruned. This table also shows that the right-hand sides of the IDICs and hypercube ICs are closer to their best values in comparison with the integer no-good cuts and it verifies the superiority of these inequalities over the integer no-good cut. Moreover, since the hypercube IC and generalized no-good cut may remove a part of the non-improving bilevel feasible solutions, the right-hand sides of these inequalities may be greater than their best possible values and it can be observed for the instances 3 and 4.

Table 3: Analysis of the right-hand sides of different inequalities for the `INT-DEN` set

Instance	Obj before cut	Benders binary				IDIC				Integer no-good cut			
		Orig RHS	Best RHS	Obj after orig cut	Obj after best cut	Orig RHS	Best RHS	Obj after orig cut	Obj after best cut	Orig RHS	Best RHS	Obj after orig cut	Obj after best cut
1	0	4520	4520	424.64	424.64	1	1.64	0	514.43	1	7	0	0
2	0	6252	6252	1270.08	1270.08	1	1.11	0	407.73	1	12	0	659.91
3	0	5662	5662	1285.7	1285.7	1	1.07	0	61.21	1	14	0	481.27
4	0	7113	7113	1274.73	1274.73	1	1.15	0	390.19	1	16	0	701.53
5	0	9685	9685	1720.64	1720.64	1	1.11	0	406.65	1	18	0	1070.59

Table 4: Analysis of the right-hand sides of different cuts for the `IBLP-DEN` set

Instance	Obj before cut	IDIC				Hypercube IC				Integer no-good cut			
		Orig RHS	Best RHS	Obj after orig cut	Obj after best cut	Orig RHS	Best RHS	Obj after orig cut	Obj after best cut	Orig RHS	Best RHS	Obj after orig cut	Obj after best cut
1	-669	0	∞	-623.47	prune	5	∞	-645	prune	-789	∞	-668.5	prune
2	-688	0	∞	-544	prune	1	∞	-632.46	prune	-29	∞	-676.36	prune
3	-766	0	0	-286	-286	3	2	-761.08	-766	-17	-4	-761.08	-508.05
4	-724	0	0	-578.63	-578.63	19	18	-709.75	-724	-407	-402	-721	-706
5	-659	0	∞	-656.67	prune	15	∞	prune	prune	-603	∞	-656.83	prune

6.4.4 Cut Generation Failures

As mentioned in Section 4.3, it is possible for cut generation to fail in the case of both ISICs and IDICs for several reasons. Here, we provide a brief analysis of how prevalent this is. Table 5 shows overall statistics for each of the test sets in a variety of scenarios.

- Lines 1 to 5 in the table are for the cut generation strategy `Always`. The first three of these five lines are the results when only generating IDICs with both `fractional` branching and `linking` branching. The second line shows the effect of disabling the generation of MILP cuts. Lines 4 and 5 show the results when generating ISICs of types I and II with the `fractional` branching strategy.
- Lines 6 to 8 show the effect of generating cuts only when the linking variables are integer (the strategy `LInt`) and with the `fractional` branching strategy, with different types of intersection cuts.
- Lines 9 to 11 are similar to lines 6 to 8, but with the cut generation strategy `XYInt`.
- Finally, lines 12 and 13 are both for the default strategy, but one details failure rates for IDICs and the other for ISIC of type I.

The statistics provide some evidence for several conjectured behaviors, though more study is needed. First, we conjectured high rates of failure for ICs for problems in which the objective alignment is near 1 and this seems to be the case, as per the results on lines 1 to 3 for ZHANG and ZHANG2. We also conjectured that generation of MILP cuts should help and this also seems to be the case.

On the other hand, we conjectured that there would be a low failure rate when the objective alignment is -1. As per lines 1 to 3 for the interdiction problems, this is not the case. In fact, this seems to be because it can very easily be the case that the problem of producing an improving direction is infeasible. For binary problems, anytime all of the free variables have fractional values

Scenario	DEN	DEN2	FIS	ZHANG	ZHANG2	INT
Always IDIC (frac)	.56	.38	.53	.65	.63	.46
Always IDIC w/o MILP cuts (frac)	.57	.37	.53	.78	.85	.82
Always IDIC (link)	.60	.28	.75	.92	.87	.50
Always Type 1 (frac)	.18	.15	.20	.04	.18	.01
Always Type 2 (frac)	.18	.27	.45	.03	.14	.06
LInt IDIC (frac)	.18	.16	.13	.31	.38	.01
LInt Type 1 (frac)	.07	.06	0	.03	.11	0
LInt Type 2 (frac)	.05	.14	.14	.02	.11	$< .01$
XYInt IDIC (frac)	$< .01$	$< .01$	.03	$< .01$	0	$< .01$
XYInt Type 1 (frac)	0	0	0	0	0	0
XYInt Type 2 (frac)	$< .01$	.01	.03	$< .01$	$< .01$	$< .01$
Default (IDIC)	.61	N/A	.57	.74	.67	N/A
Default (ISIC)	$< .01$	N/A	0	.05	.03	N/A

Table 5: Cut generation failure rates

in the solution to the relaxation, it is clear there can be no improving direction (recall that the direction needs to be both feasible and have integral coefficients). This can very easily happen once some branching has occurred. The high failure rate indicates that is indeed an issue. Fortunately, such infeasibility should be easily detected in pre-processing and should not greatly affect running times.

As expected, failure rates are reduced substantially with the generation strategy `LInt`, but as we have seen, although the lower rates of generation do reduce time spent generating inequalities in most cases, this does not pay off in general, as many fewer cuts are successfully generated in the end. As expected, failure rates are close to zero with the strategy `XYInt`, which was the only strategy available in `MibS 1.1` (there can still be some failures when the intersection with the radial cone and the BFS is empty).

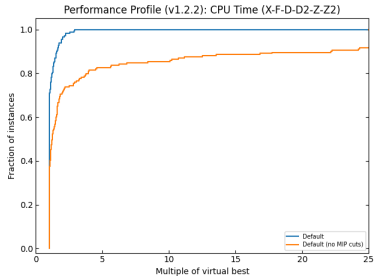
This raises the tricky issue, which remains to be addressed, of the fact that it is difficult to identify whether a given fractional point is in case C1 or not without trying to separate it. This separation is expensive and will fail if the point is in case C1. Hence, we are not only unable to apply strong cuts, but we are also wasting CPU time with failed separation attempts.

6.4.5 Effect of Integrality Cuts

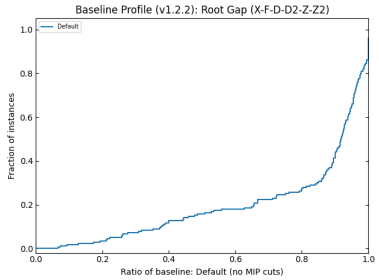
In `MibS 1.1`, we decided not to experiment with MILP cuts because we were focused on MIBLP-specific cuts that we felt would be much stronger. In developing `MibS 1.2` and with a better understanding of the behavior of the MIBLP cuts, we decided to bring MILP cuts back into the picture. Our empirical testing has shown that generating MILP cuts is indeed impactful. Our best understanding of this is that the use of MILP cuts encourages more integer solutions to be produced and this in turn reduces failure rates arising because of being in case C1 (which can only happen with fractional solutions), resulting in more frequent generation of strong ICs. This is born

out in the performance profile shown in Figure 12a. The baseline profile in Figure 12b shows that integrality (MILP) cuts close a significant amount of gap above what MIBLP cuts already close.

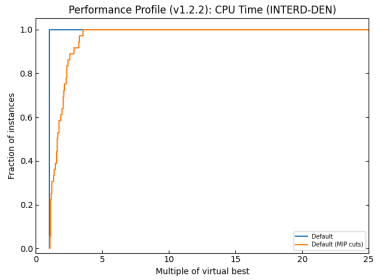
We have also conjectured that MILP cuts should help to a greater extent for instances with objective alignment near 1 and may not help as much for instances with objective alignment -1. Figures 12c and 12d demonstrate that for the test sets ZHANG2 and INT-DEN, this is indeed the case.



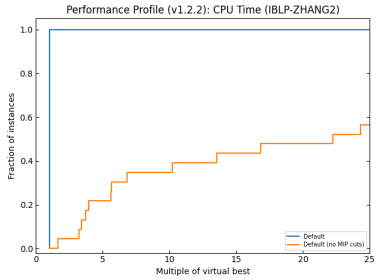
(a) Performance profile (all instances)



(b) Baseline profile (all instances)



(c) Performance profile (interdiction)



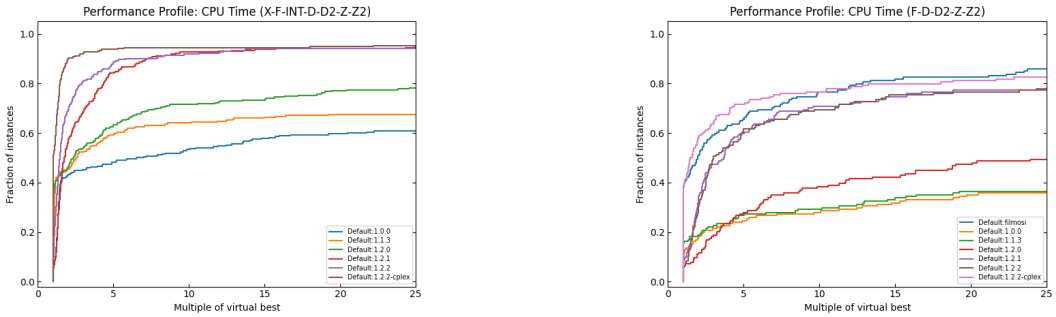
(d) Performance profile (ZHANG2)

Figure 12: Demonstrating the impact of generating MILP cuts in combination with MIBLP cuts

6.4.6 Comparisons to Other Solvers

Finally, we present some overall comparisons that show both the evolution of `MibS` over time and a comparison to `filmosi`, the most competitive alternative solver available Fischetti et al. [2017b]. Figure 13a show the comparison of different versions of `MibS` over time, including a version built with CPLEX 12.7 as the subsolver. The impact of the improvements made between `MibS 1.1` and `1.2` can be clearly seen. The use of CPLEX instead of SYMPHONY provides only a modest boost.

Figure 13b adds the `filmosi` solver to the mix. The `filmosi` solver was also built on top CPLEX 12.7 and uses CPLEX not only as a subsolver but also uses CPLEX's own branch-and-cut as the base for the entire algorithm. As such, it is using the powerful engine of CPLEX's MILP solver to do everything from pre-processing to cut generation to making branching decisions. Nevertheless, through a combination of all of the methodologies described in this paper, we are able to achieve performance more or less on par with `filmosi`. It is an interesting question whether `filmosi` could itself be improved by some of the tweaks we have made in `MibS` or whether a version of `MibS` built on top of the CPLEX branch and cut rather than that of `BLIS` would be possible and could be effective.



(a) Performance profile

(b) Cumulative profile

Figure 13: Overall comparison of different version of `MibS` to `filmosi`

7 Conclusions

Although the generation of valid inequalities has proven to be an invaluable tool in solving MIBLPs, there remains a large scope for developing the fundamental theory underlying the branch-and-cut algorithms that have become a standard solution method. We have taken an initial step towards filling in some of the gaps, as well as providing a framework within which to view the known methods, clarifying the relationship to and distinction from similar methods already studied in the MILP case. We also introduced new general methodology for generating valid inequalities that applies not only to MIBLPs, but also potentially to other classes of optimization problem (such as MILPs themselves). Finally, we provided a comprehensive comparison of known methods for the generation of valid inequalities. There remains a wide range of possibilities for innovation in leveraging the decades of research solving MILPs to derive improved methodology for MIBLPs. It is our hope that this work motivates future work along these lines, leading to further discoveries in this important field of research.

There also remain many additional open questions regarding how to efficiently incorporate the generation of valid inequalities into a branch-and-cut algorithm. Although one can simply add cut generation to an existing MILP solver and obtain a solver for MIBLPs in a relatively straightforward way, many of the control mechanisms that have been well-honed for solving MILPs do not seem to work as well when naively utilized for solving MIBLPs. The work of re-designing many of these control mechanisms in a way that works well for MIBLPs is a rich source of future challenges in this space.

Acknowledgements

This research was made possible with support from National Science Foundation Grants CMMI-1435453, CMMI-0728011, and ACI-0102687, as well as Office of Naval Research Grant N000141912330.

References

Balas, E. (1979). "Disjunctive Programming". In: *Annals of Discrete Mathematics 5: Discrete Optimization*. Ed. by P. L. Hammer, E. L. Johnson, and B. H. Korte. North Holland, pp. 3–51.

Balas, E., S. Ceria, and G. Cornuéjols (1993). "A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs". In: *Mathematical Programming* 58, pp. 295–324.

Balas, E. and A. Saxena (2008). "Optimizing Over the Split Closure". In: *Mathematical Programming* 113.2, pp. 219–240.

Balas, E. (1971). "Intersection Cuts-a New Type of Cutting Planes for Integer Programming". In: *Operations Research* 19.1, pp. 19–39.

– (1985). "Disjunctive Programming and a Hierarchy of Relaxations for Discrete Optimization Problems". In: *SIAM Journal on Algebraic Discrete Methods* 6.3, pp. 466–486.

– (1998). "Disjunctive Programming: Properties of the Convex Hull of Feasible Points". In: *Discrete Applied Mathematics* 89, pp. 3–44.

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

Basu, A., C. T. Ryan, and S. Sankaranarayanan (2018). "Mixed-Integer Bilevel Representability". In: *arXiv preprint arXiv:1808.03865*. arXiv: 1808.03865.

Bénichou, M., J. M. Gauthier, P. Girodet, G. Hentges, G. Ribière, and O. Vincent (1971). "Experiments in Mixed-Integer Linear Programming". In: *Mathematical Programming* 1, pp. 76–94.

Bienstock, D., C. Chen, and G. Munoz (2016). "Outer-Product-Free Sets for Polynomial Optimization and Oracle-Based Cuts". In: *arXiv preprint arXiv:1610.04604*. arXiv: 1610.04604.

Bixby, R. E., S. Ceria, C. M. McZeal, and M. W. Savelsbergh (1998). *An Updated Mixed Integer Programming Library: MIPLIB 3.0*. Tech. rep.

Bolusani, S., S. Coniglio, T. Ralphs, and S. Tahernejad (2020). "A Unified Framework for Multistage Mixed Integer Linear Optimization". In: *Bilevel Optimization: Advances and Next Challenges*. Ed. by S. Dempe and A. Zemkoho. Springer, pp. 513–560. ISBN: 978-3-030-52118-9. DOI: 10.1007/978-3-030-52119-6. URL: https://arxiv.org/abs/2104.09003.

Bolusani, S. and T. Ralphs (2022). "A Framework for Generalized Benders' Decomposition and Its Application to Multilevel Optimization". In: *Mathematical Programming Series B* 196, pp. 389–426. DOI: 10.1007/s10107-021-01763-7. URL: https://arxiv.org/abs/2104.06496.

Caprara, A., M. Carvalho, A. Lodi, and G. Woeginger (2016). "Bilevel Knapsack with Interdiction Constraints". In: *INFORMS Journal on Computing* 28.2, pp. 319–333.

Conforti, M., G. Cornuéjols, and G. Zambelli (2014). *Integer Programming*. Vol. 271. Springer.

Cornuéjols, G. (2008a). "Valid Inequalities for Mixed Integer Linear Programs". In: *Mathematical Programming* 112, pp. 3–44. DOI: 10.1007/s10107-006-0086-0.

– (2008b). "Valid Inequalities for Mixed Integer Linear Programs". In: *Mathematical Programming* 112.1, pp. 3–44.

IBM ILOG CPLEX Optimization Studio (2024). URL: https://www.ibm.com/products/ilog-cplex-optimization-studio.

Del Pia, A. and R. Weismantel (2012). "On Convergence in Mixed Integer Programming". In: *Mathematical programming* 135.1-2, pp. 397–412.

Dempe, S. (2002). *Foundations of Bilevel Programming*. Springer Science & Business Media.

DeNegre, S. (2011). "Interdiction and Discrete Bilevel Linear Programming". PhD Dissertation. Lehigh University. URL: http://coral.ie.lehigh.edu/~ted/files/papers/ScottDeNegreDissertation11.pdf.

DeNegre, S. and T. Ralphs (2009). "A Branch-and-Cut Algorithm for Bilevel Integer Programming". In: *Proceedings of the Eleventh INFORMS Computing Society Meeting*, pp. 65–78. DOI: 10.1007/978-0-387-88843-9_4. URL: http://coral.ie.lehigh.edu/~ted/files/papers/BILEVEL08.pdf.

DeNegre, S. and T. Ralphs (2024). "MIBLP Instance Generator". In: URL: https://github.com/tkralphs/MIBLPInstanceGenerator.

DeNegre, S., T. Ralphs, and S. Tahernejad (2015). "MibS Version 1.0". In: URL: https://github.com/coin-or/MibS.

– (2024). "MibS Version 1.2". In: DOI: 10.5281/zenodo.1439384. URL: https://github.com/coin-or/MibS.

Dey, S. and M. Molinaro (2018). "Theoretical Challenges towards Cutting-Plane Selection". In: *Mathematical Programming* 170, pp. 237–266.

Dolan, E. D. and J. J. Moré (2002). "Benchmarking Optimization Software with Performance Profiles". In: *Mathematical Programming* 91.2, pp. 201–213.

Figueira, J. (2000). *MCDM Numerical Instances Library*.

Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017a). "A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs". In: *Operations Research* 65.6, pp. 1615–1637.

– (2017b). *Filmosi*. URL: https://msinnl.github.io/pages/bilevel.html.

– (2018). "On the Use of Intersection Cuts for Bilevel Optimization". In: *Mathematical Programming* 172.1-2, pp. 77–103.

– (2019). "Interdiction Games and Monotonicity, with Application to Knapsack Problems". In: *INFORMS Journal on Computing* 31.2, pp. 390–410. ISSN: 1091-9856. DOI: 10.1287/ijoc.2018.0831.

Forrest, J. (2024). *COIN-OR Branch and Cut*. URL: https://github.com/coin-or/Cbc.

– (2025). *Cut Generation Library*. URL: https://github.com/coin-or/Cgl.

Gade, D. and S. Küçükyavuz (2011). "Pure Cutting-Plane Algorithms and Their Convergence". In: *Wiley Encyclopedia of Operations Research and Management Science*, pp. 1–11.

Gomory, R. E. (1958). "Outline of an Algorithm for Integer Solutions to Linear Programs". In: *Bulletin of the American Mathematical Monthly* 64, pp. 275–278.

– (1960). *A Algorithm for the Mixed Integer Problem*. Tech. rep. The RAND Cooperation.

Grötschel, M., L. Lovász, and A. Schrijver (1993a). *Geometric Algorithms and Combinatorial Optimization*. New York: Springer-Verlag.

– (1993b). *Geometric Algorithms and Combinatorial Optimization*. New York: Springer-Verlag.

Lodi, A., T. Ralphs, and G. Woeginger (2014). "Bilevel Programming and the Separation Problem". In: *Mathematical Programming* 148, pp. 437–458. DOI: 10.1007/s10107-013-0700-x. URL: http://coral.ie.lehigh.edu/~ted/files/papers/BilevelSeparation12.pdf.

Loridan, P. and J. Morgan (1996). "Weak via Strong Stackelberg Problem: New Results". In: *Journal of global Optimization* 8, pp. 263–287.

Marchand, H., A. Martin, R. Weismantel, and L. Wolsey (2002). "Cutting Planes in Integer and Mixed Integer Programming". In: *Discrete Applied Mathematics* 123.1, pp. 397–446.

Meyer, R. (1974). "On the Existence of Optimal Solutions to Integer and Mixed Integer Programming Problems". In: *Mathematical Programming* 7, pp. 223–235.

Moore, J. T. and J. F. Bard (1990). "The Mixed Integer Linear Bilevel Programming Problem". In: *Operations research* 38.5, pp. 911–921.
Munkres, J. (2014). *Topology*. Pearson Education.
Ralphs, T. (July 2015). *Bilevel Integer Optimization: Theory and Algorithms*. Invited Talk. International Symposium on Mathematical Programming, Pittsburgh, PA. URL: https://coral.ise.lehigh.edu/~ted/files/talks/BILEVEL-ISMP15.pdf (visited on 09/16/2025).
Ralphs, T., L. Ladányi, and M. Saltzman (2004). "A Library Hierarchy for Implementing Scalable Parallel Search Algorithms". In: *Journal of Supercomputing* 28, pp. 215–234. DOI: 10.1023/B:SUPE.0000020179.55383.ad. URL: http://coral.ie.lehigh.edu/~ted/files/papers/JSC02.pdf.
Stockmeyer, L. (1976). "The Polynomial-Time Hierarchy". In: *Theoretical Computer Science* 3.1, pp. 1–22. ISSN: 0304-3975. DOI: 10.1016/0304-3975(76)90061-X.
Tahernejad, S. (2019). "Two-stage Mixed Integer Stochastic Bilevel Linear Optimization". PhD Dissertation. Lehigh University.
Tahernejad, S., T. Ralphs, and S. DeNegre (2020). "A Branch-and-Cut Algorithm for Mixed Integer Bilevel Linear Optimization Problems and Its Implementation". In: *Mathematical Programming Computation* 12, pp. 529–568. DOI: 10.1007/s12532-020-00183-6. URL: https://arxiv.org/abs/2104.09010.
Thürauf, J., T. Kleinert, I. Ljubić, T. Ralphs, and M. Schmidt (2024). *BOBILib: Bilevel Optimization (Benchmark) Instance Library*. (Visited on 09/26/2025).
Vicente, L. N., G. Savard, and J. J. Judice (1996). "Discrete Linear Bilevel Programming Problem". In: *Journal of optimization theory and applications* 89, pp. 597–614.
Wolter, K. (2006). "Implementation of Cutting Plane Separators for Mixed Integer Programs". MA thesis. Technische Universität Berlin.
Xu, P. and L. Wang (2014). "An Exact Algorithm for the Bilevel Mixed Integer Linear Programming Problem under Three Simplifying Assumptions". In: *Computers & operations research* 41, pp. 309–318.
Xu, Y. and T. Ralphs (2022). "BLIS Version 0.94". In: DOI: 10.5281/zenodo.5846505. URL: https://github.com/coin-or/CHiPPS-BLIS.
Zhang, J. and O. Y. Ozaltın (2017). "A Branch-and-Cut Algorithm for Discrete Bilevel Linear Programs". In: *Optimization Online*.

A Detailed Descriptions of Test

Here are detailed descriptions of the test instances.

- `INT-DEN`: This set was generated by DeNegre [2011] and contains 300 knapsack interdiction problems. These problems originate from the *Multiple Criteria Decision Making library* [Figueira (2000)] and has the same structure as (MIPINT). The number of first-level variables ($n_1 = n_2$) varies in $n_1 \in \{10, 11, ..., 19, 20, 30, 40, 50\}$ and the number of first- and second-level constraints are 1 and $n_1 + 1$, respectively. There are 20 instances corresponding to each level of n_1, except 40 instances for $n_1 \in \{10, 20\}$. Due to the difficulty of the instances with $n_1 = 50$, we excluded them in the experiments.

- `IBLP-DEN`: This set was generated using the publicly available generator of DeNegre and Ralphs [2024] that were used in support of experiments in DeNegre [2011]. It contains 50 pure integer instances with all variables non-negative and upper bounded by 1500. The number of upper- and lower-level constraints is 0 and 20, respectively. The constraints are expressed in "$\geq$" form with negative coefficients having absolute value at most 50. Thus, all second-level variables likely have implicit upper bounds less than the given bound of 1500. The number of upper- and lower-level variables are as follows.

# of instances	n_1	n_2
10	5	10
10	10	10
10	15	5
20	15	5

- `IBLP-DEN2`: This set was generated using the publicly available generator of DeNegre and Ralphs [2024] that were used in support of experiments in DeNegre [2011]. It contains 110 pure integer instances with all variables non-negative and upper bounded by 500. The number of upper- and lower-level constraints is 0 and the number of lower-level variables is as indicated in the table below. The constraints are expressed in "$\leq$" form with coefficients of mixed sign having absolute value at most 50. Thus, it is once again likely that all second-level variables have implicit upper bounds less than the given bound of 500. The number of lower-level constrains and upper- and lower-level variables is as follows.

# of instances	m	n_1	n_2
10	5	5	5
10	10	10	10
10	10	15	10
10	10	20	10
10	10	15	15
10	10	10	20
10	15	10	10
10	15	15	10
10	15	20	10
10	15	15	15
10	15	10	20

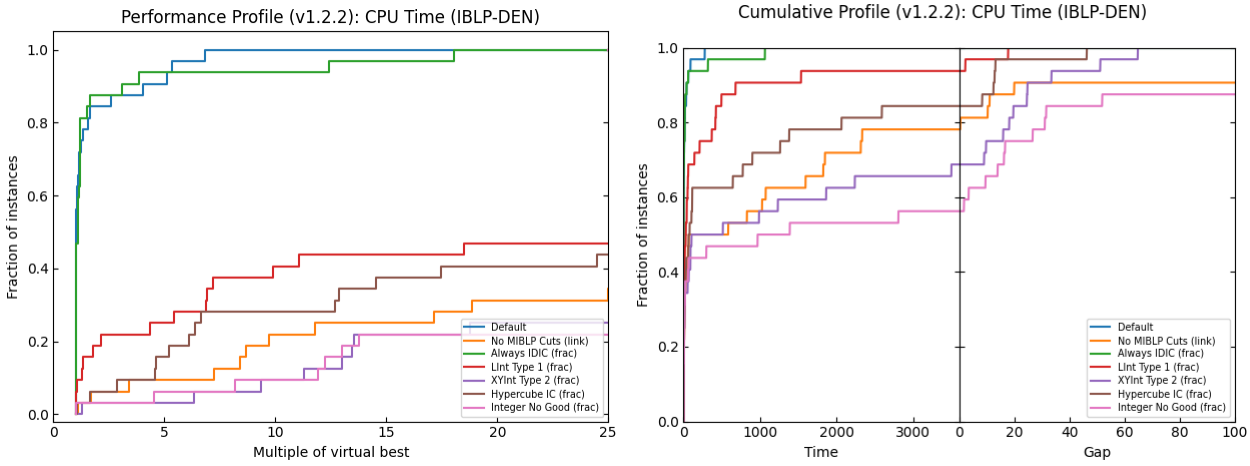
- `IBLP-ZHANG`: This set was generated by Zhang and Ozaltın [2017] and includes 30 instances with binary first-level variables, integer second-level variables and no first-level constraints. The number of first-level variables varies in $n_1 \in \{50, 60, 70, 80, 90\}$ and the number of second-level variables is set to $n_2 = n_1 + 20$. There are 3 instances with $m_2 = 6$ and 3 instances with $m_2 = 7$ corresponding to each level of n_1. The linear constraints are expressed in "$\leq$" form with positive coefficients on both upper and lower level variables ranging between 0 and 10 while the right-hand side is in the range of 20 to 30. Thus, all second-level variables have small implicit upper bounds.

- `IBLP-ZHANG2`: This set is a modification of `IBLP-ZHANG` in which the bounds on first-stage variables are increased from 1 to 10 to make them non-binary.

- `IBLP-FIS`: This set was generated by Fischetti et al. [2018] and are constructed from well-known instances originating in `MILPLIB 3.0` [Bixby et al. (1998)] in which all variables are binary and there are no first-level constraints. The variables are split between upper and lower level in proportions reflected in the file name of each converted instance. Due to memory limitations, we did not consider 3 instances of this set in our experiments.

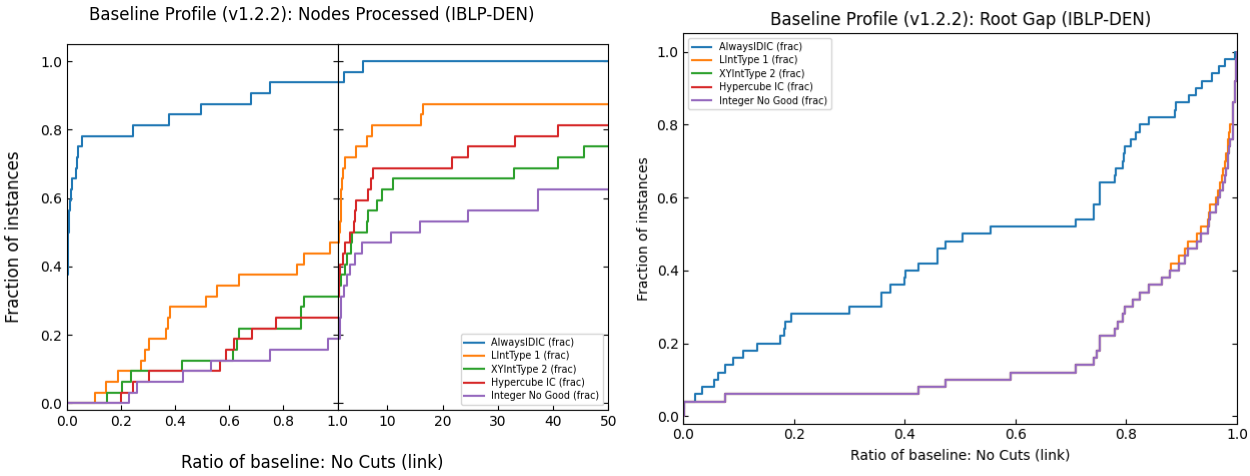
- `MIBLP-XU`: Xu and Wang [2014] generated a set of mixed integer bilevel instances for the testing of their algorithm. The instances have $n = n_1 = n_2 \in \{10, 60, 110, 160, 210, 260, 310, 360, 410, 460\}$. The first-level variables are constrained to be integer but some of the second-level variables are continuous. The number of first-level as well as second-level constraints is $0.4n$. All matrices, vectors, right-hand sides, etc. are uniformly distributed integers. The constraint matrices have coefficients in $[0, 10]$, while the objective vectors have entries in $[-50, 50]$. The first-level right-hand side vector b_1 has entries in $[30, 130]$, and the second-level right-hand side vector b_2 has entries in $[10, 110]$. There are 10 instances for every value of n, which gives a total of 100 instances.

B Results for Individual Test Sets



(a) Performance profile of CPU time

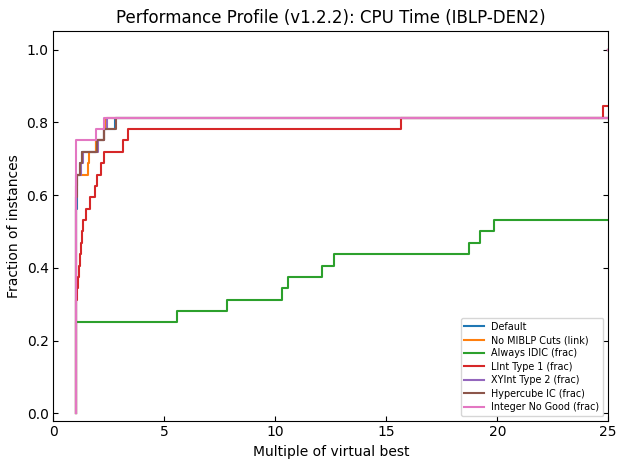
(b) Cumulative profile of CPU time



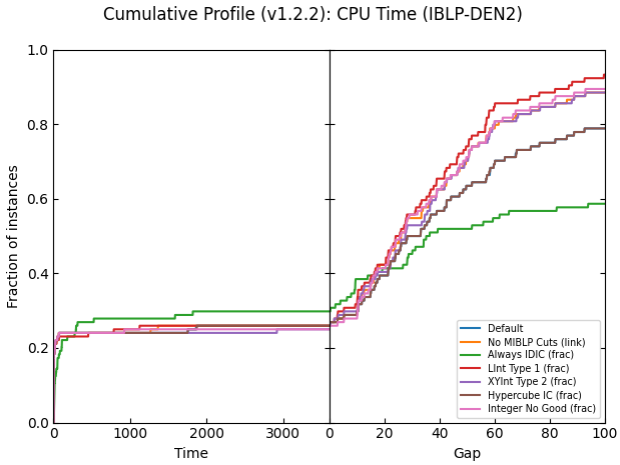
(c) Baseline profile of tree size

(d) Baseline profile of root gap

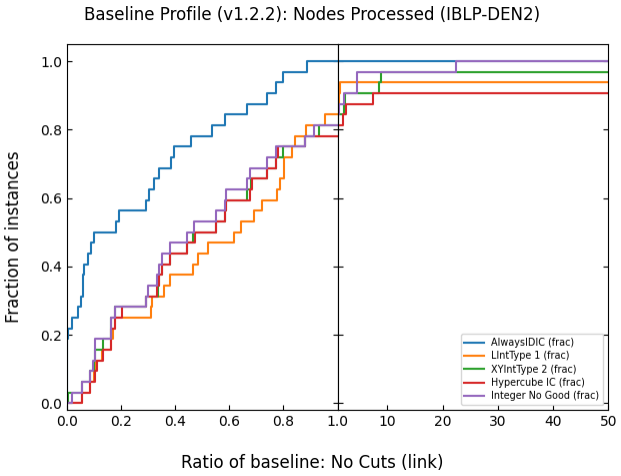
Figure 14: Comparing the performance of different inequalities on the `IBLP-DEN` set



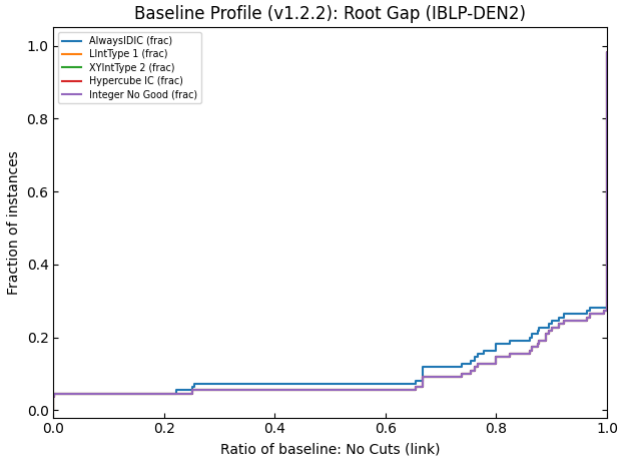
(a) Performance profile of CPU time



(b) Cumulative profile of CPU time

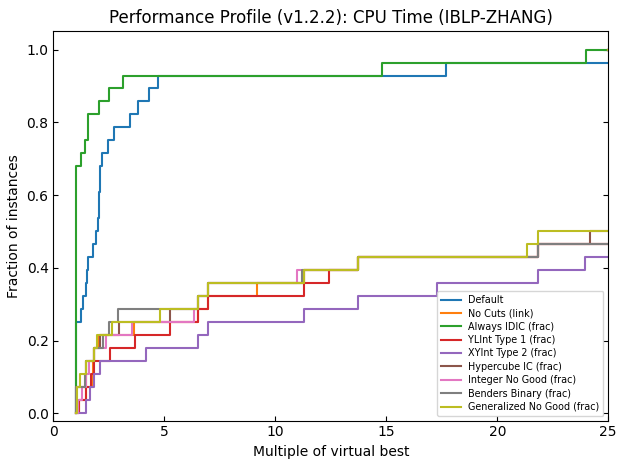


(c) Baseline profile of tree size

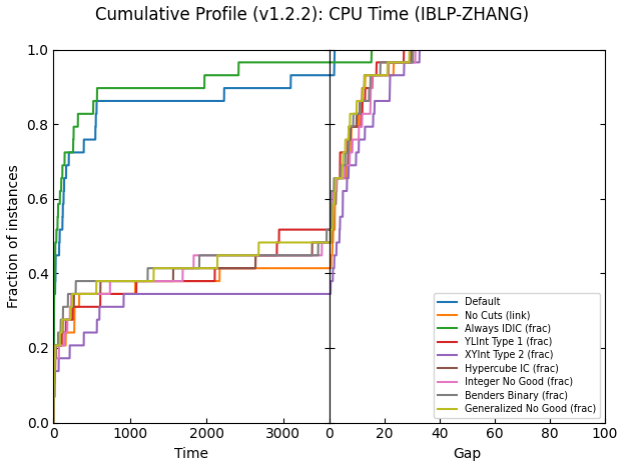


(d) Baseline profile of root gap

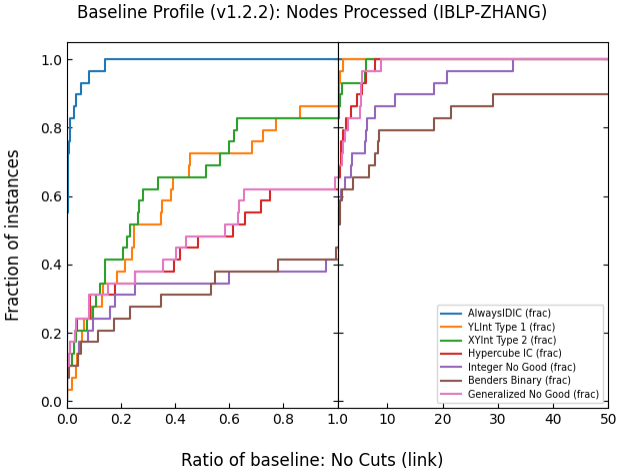
Figure 15: Comparing the performance of different inequalities on the `IBLP-DEN2` set



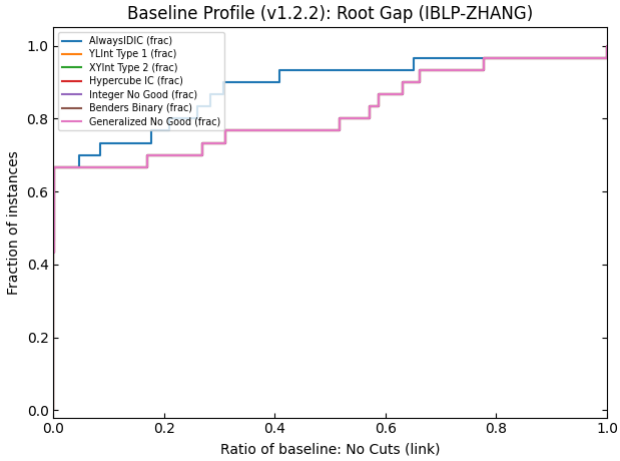
(a) Performance profile of CPU time



(b) Cumulative profile of CPU time

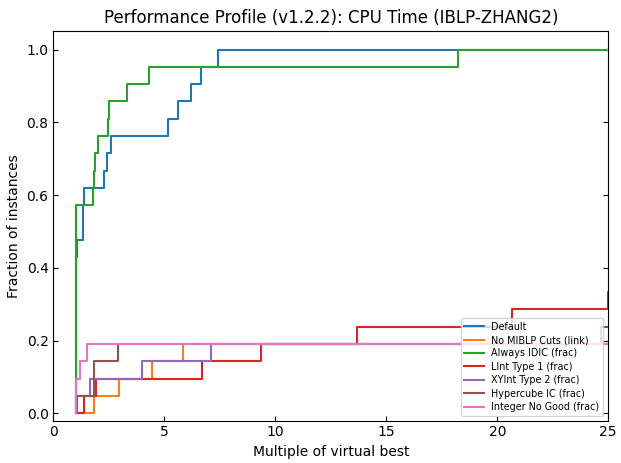


(c) Baseline profile of tree size

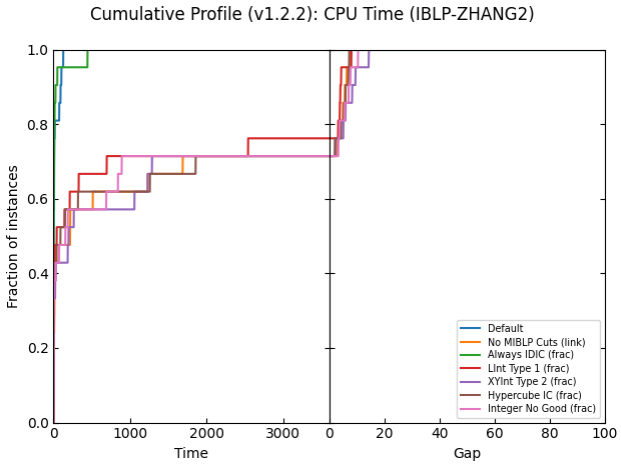


(d) Baseline profile of root gap

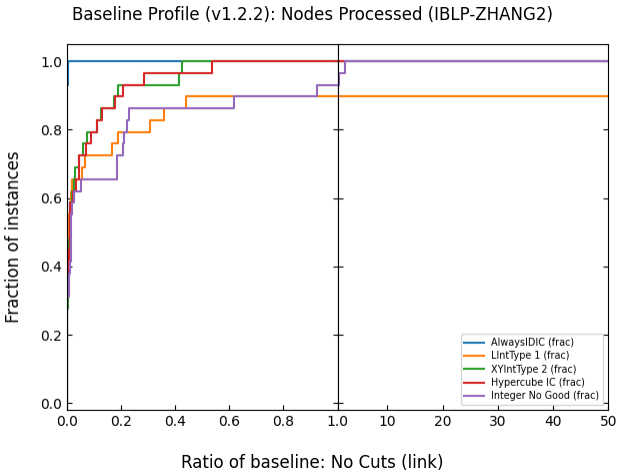
Figure 16: Comparing the performance of different inequalities on the `IBLP-ZHANG` set



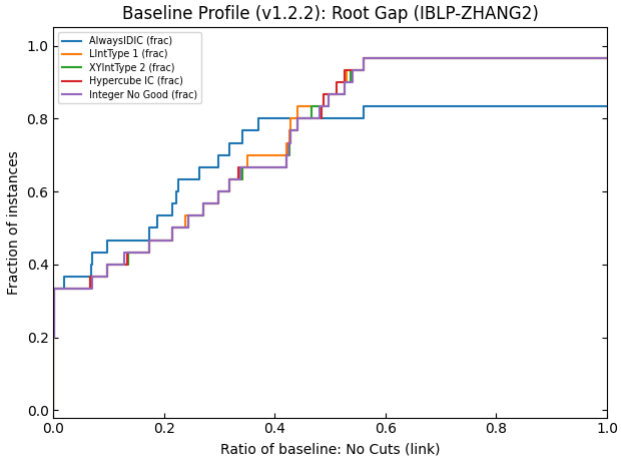
(a) Performance profile of CPU time



(b) Cumulative profile of CPU time

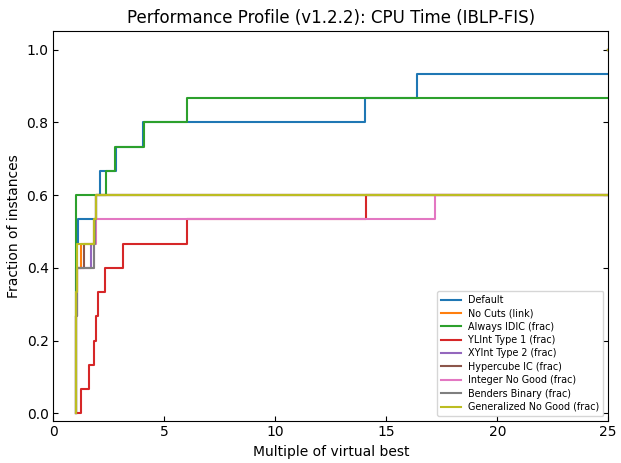


(c) Baseline profile of tree size

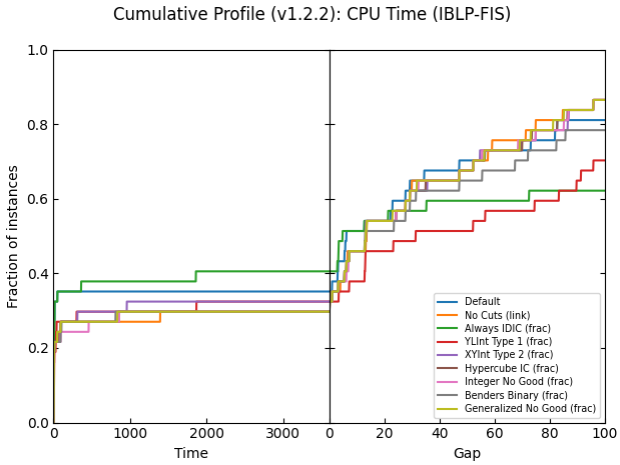


(d) Baseline profile of root gap

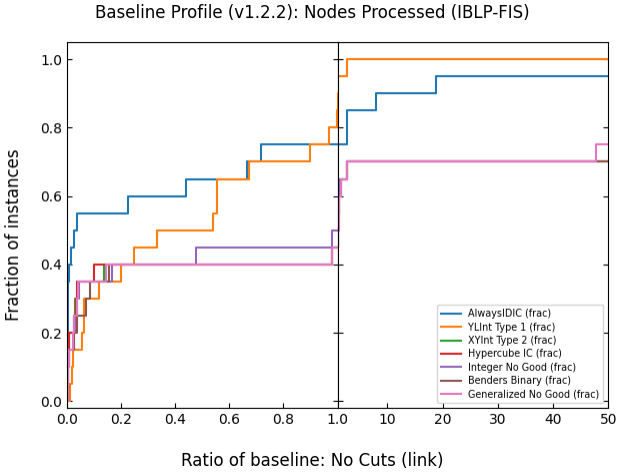
Figure 17: Comparing the performance of different inequalities on the `IBLP-ZHANG2` set



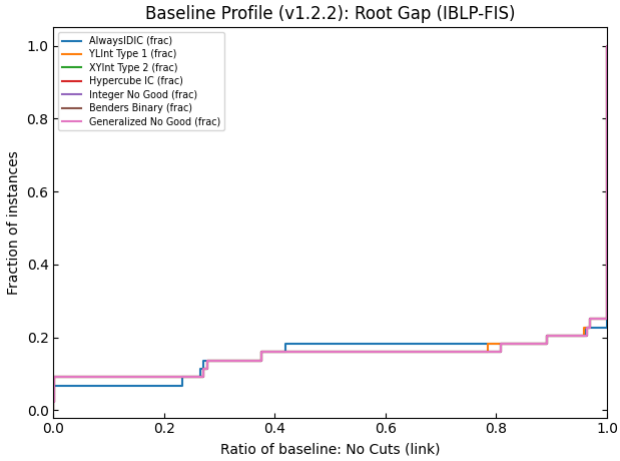
(a) Performance profile of CPU time



(b) Cumulative profile of CPU time



(c) Baseline profile of tree size



(d) Baseline profile of root gap

Figure 18: Comparing the performance of different inequalities on the `IBLP-FIS` set