Branch-and-Cut for Integer Bilevel Linear Optimization

~~Recent Progress and Remaining Challenges~~

~~How to Burn Through 20K+ CPU Hours~~

Some Things I Believed That Turned Out To Be Wrong

Ted Ralphs[1]

Primary Collaborators: Sahar Tahernejad[1], Scott DeNegre [1]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University

IWOBIP, Chile, January, 2024

ISE Industrial and Systems Engineering

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Attributions

Many Ph.D students and postdocs contributed to development of this work over time.

Current/Former Students/Postdocs

- Federico Battista
- Suresh Bolusani
- Scott DeNegre
- Samira Fallah
- Menal Gúzelsoy
- Anahita Hassanzadeh
- Ashutosh Mahajan
- Sahar Tahernajad
- Yu Xie

Thanks!

Thanks to the Organizers!

My Pandemic Projects

- Me at the beginning of the pandemic: "At least now I'll have some time to code!"
- My wife at the beginning of the pandemic: "Guess what? We're having a baby!!"
- Result: `MibS` 1.2 and a healthy baby boy!

Caveats

- This talk chronicles my attempts over the last 2.5 years at understanding the behavior of `MibS` and tuning it to work well "off-the-shelf."
- Digging deeper into `MibS` made me realize how much I still didn't know.
- The MIP solver `MibS` is built on (`Blis`) is not exactly "state-of-the-art," but its performance as an MIBLP solver is now quite competitive.
- I'll try to give a window into what I've learned, but some of it is still guesswork.
- There are lots of complex interactions, take everything with a grain of salt.
- Many, many questions remain.

Outline

Setting

- *First-level variables*: $x \in X$ where $X = \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1}$
- *Second-level variables*: $y \in Y$ where $Y = \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}$

MIBLP

$$\min_{x,y} \left\{ cx + d^1 y \;\middle|\; x \in X, y \in \mathcal{P}_1(x), y \in \operatorname{argmin}\{d^2 z \mid z \in \mathcal{P}_2(x) \cap Y\} \right\} \tag{MIBLP}$$

where

$$\mathcal{P}_1(x) = \left\{ y \in \mathbb{R}_+^{n_2} \;\middle|\; G^1 y \geq b^1 - A^1 x \right\}$$
$$\mathcal{P}_2(x) = \left\{ y \in \mathbb{R}_+^{n_2} \;\middle|\; G^2 y \geq b^2 - A^2 x \right\}$$

Later, we'll need to refer to

$$\mathcal{P} = \{ (x, y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid y \in \mathcal{P}_1(x) \cap \mathcal{P}_2(x) \}$$

Simplifying Assumptions

- We discuss only the **optimistic case** (the pessimistic case is more involved, but the methodology is not very different).
- To ensure solutions exist, we make the standard assumption that all first-level variables that appear in second-level constraints are integer.
- We further assume (without loss of generality in the optimistic case) that ***all* first-level variables are integer** ($r_1 = n_1$).
- We assume $\mathcal{P}$ **is bounded** and that $\left\{r \in \mathbb{R}^{n_2}_{+} \;\middle|\; G^2 r \geq 0, d^2 r < 0\right\} = \emptyset$ (the latter ensures **the lower-level problem is bounded for any feasible upper-level solution**).
- We assume **all input data is integer**.

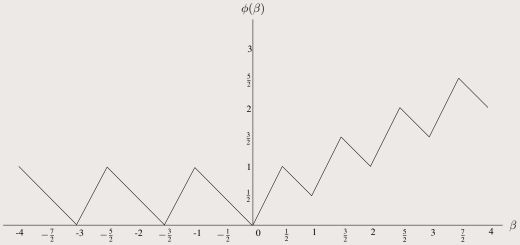
The Second-level Value Function

- The second-level *value function* is

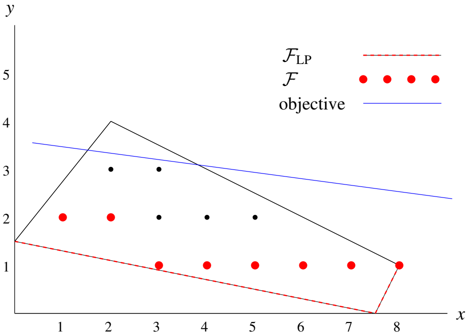
MILP Value Function

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

We let $\phi(\beta) = \infty$ if $\{y \in Y \mid G^2 y \geq \beta\} = \emptyset$.



Example 1 Moore and Bard [1990]



$$
\begin{aligned}
\min_{x \in \mathbb{Z}_+} \quad & -x - 10y \\
\text{s.t.} \quad & y \in \operatorname{argmin} \{ y : \\
& \quad -25x + 20y \leq 30 \\
& \quad x + 2y \leq 10 \\
& \quad 2x - y \leq 15 \\
& \quad 2x + 10y \geq 15 \\
& \quad y \in \mathbb{Z}_+ \ \}
\end{aligned}
$$

□

Value Function Reformulation

- *First-level variables*: $x \in X$ where $X = \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1}$
- *Second-level variables*: $y \in Y$ where $Y = \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}$

MIBLP

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

Bilevel Feasible Region

$$\mathcal{F} = \left\{ (x, y) \in \mathcal{S} \;\middle|\; d^2 y \leq \phi(b^2 - A^2 x) \right\},$$

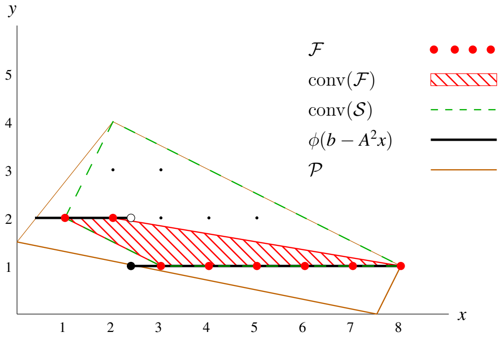
where

$$\mathcal{S} = \{ (x, y) \in X \times Y \mid y \in \mathcal{P}_1(x) \cap \mathcal{P}_2(x) \}$$

- This reformulation seems to suggest a Benders-type algorithm in which we approximate the second-level value function.
- Convexification helps avoid approximating the entire function.

Polyhedral Reformulation

Convexification considers the following conceptual reformulation.



$$\begin{aligned} \min \quad & cx + d^1 y \\ \text{s.t.} \quad & (x, y) \in \mathrm{conv}(\mathcal{F}) \end{aligned}$$

- This reformulation suggests a branch-and-cut algorithm similar to that used for solving MILPs DeNegre and Ralphs [2009].
- To get dual bounds, we optimize over a relaxed feasible region.
- We iteratively approximate $\mathrm{conv}(\mathcal{F})$ with linear inequalities.

Basic Principle: Disjunction

Definition 1 (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

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

Two classes of disjunction

- $(\bar{x}, \bar{y}) \in \mathcal{P} \setminus \mathcal{S} \Leftarrow$ must violate a variable disjunction.
- $(\bar{x}, \bar{y}) \in \mathcal{S} \setminus \mathcal{F} \Leftarrow$ must violate this valid disjunction (points in $\mathcal{P} \setminus \mathcal{S}$ may also).

$$\begin{pmatrix} A^1 x \geq b^1 - G^1 y^* \\ A^2 x \geq b^2 - G^2 y^* \\ d^2 y \leq d^2 y^* \end{pmatrix} \quad \text{OR} \quad \begin{pmatrix} A^1 x \not\geq b^1 - G^1 y^* \\ \text{OR} \\ A^2 x \not\geq b^2 - G^2 y^*, \end{pmatrix} \qquad \text{(OPT-DISJ)}$$

where $y^* \in \mathcal{P}_2(\bar{x}) \cap Y$ and $d^2 \bar{y} > d^2 y^*$.

- Note that such a $y^* \neq \bar{y}$ must exist when $\bar{y} \in \mathcal{S}$.

Exploiting Disjunctions

- In branch-and-cut for MILPs, we have the following nice properties.
  - The only infeasible points that arise are those in $\mathcal{P} \setminus \mathcal{S}$, which are easy to identify.
  - We typically utilize the same disjunctions for deriving disjunctive cuts and branching.
- In MIBLP, points in $\mathcal{S} \setminus \mathcal{F}$ are not easy to identify and violated valid disjunctions are not compact or easy to generate.
- For points in $\mathcal{P} \setminus \mathcal{S}$, it is still easy to identify violated variable disjunctions.
- Nevertheless, points in $\mathcal{P} \setminus \mathcal{S}$ may also violate disjunctions of the form (OPT-DISJ).
- It may be worthwhile to generate such violated valid disjunctions, even though this can be expensive.

Basic Principle: Identifying Infeasible Solutions

- Just as in MILP, an important key to solving MIBLPs is identifying large (convex) subsets of $\mathcal{P}$ that contain no member of $\mathcal{F}$.
- This should be done by carefully exploiting available information and keeping computational overhead low.
- Two methods for proving a solution infeasible underlie much of the methodology for doing this.

Second-level Improving Solutions

Let $(x, y) \in \mathcal{P}$ and $y^* \in \mathcal{P}_2(x) \cap Y$. Then $d^2 y > d^2 y^* \Rightarrow (x, y) \notin \mathcal{F}$.

Second-level Improving Directions

Let $(x, y) \in \mathcal{P}$ and $\Delta y \in \mathbb{Z}^{n_2}$ such that $d^2 \Delta y < 0$. Then $y + \Delta y \in \mathcal{P}_2(x) \Rightarrow (x, y) \notin \mathcal{F}$.

Valid Inequalities

Valid inequality: The triple $(\alpha^x, \alpha^y, \beta) \in \mathbb{R}^{n_1+n_2+1}$ is a *valid inequality* for $\mathcal{F}$ if

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

Valid improving inequality: The triple $(\alpha^x, \alpha^y, \beta) \in \mathbb{R}^{n_1+n_2+1}$ is a *valid improving inequality* for $\mathcal{F}$ with respect to $(\bar{x}, \bar{y}) \in \mathcal{F}$ if

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

Cutting plane: As usual, a *cutting plane* (cut) refers to a valid (improving) inequality violated by a given (infeasible) solution to the current relaxation.

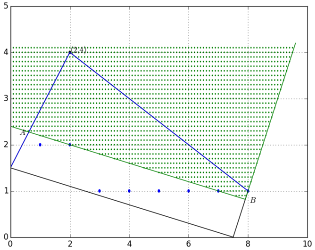
Basic Principle: Bilevel Free Sets [Fischetti et al., 2018]

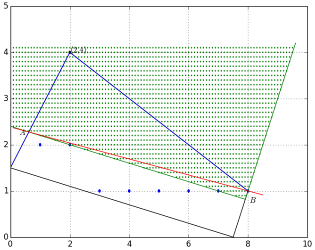
Bilevel Free Set

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

General Recipe for Valid Inequalities

- Identify a BFS $C \subseteq \mathbb{R}^{n_1+n_2}$.
- Then inequalities valid for for $\text{conv}(\overline{\text{int}(C)} \cap \mathcal{P})$ are also valid for $\mathcal{F}$.





Outline

Branch-and-Cut Algorithm

- The basic framework is very similar to that used for solving MILPs, but with many subtle differences.

Components

- **Bounding**
  - **Dual bound** $\Rightarrow$ A "tractable" relaxation strengthened with valid inequalities
  - Primal bound $\Rightarrow$ Feasible solutions
- **Branching** $\Rightarrow$ Valid disjunctions
- **Cut generation** $\Rightarrow$ Inequalities valid for $\text{conv}(\mathcal{F})$.
- Search strategies
- Preprocessing methods
- Primal heuristics
- **Control mechanisms** $\Rightarrow$ Important but tricky!

- This talk will focus on the highlighted areas.

Challenges

- On the surface, branch-and-cut for MIBLPs looks similar to that for MILPs.
- Digging deeper, they are *very* different and there is a lot we still don't know.
- We have to tear down the solver and re-examine every aspect of its performance. Some challenges that remain.
  - In contrast with MILP, it can be difficult to move the bound in the root node.
  - Thus, we don't have a very good approximation of $\mathrm{conv}(\mathcal{F})$ in the early stages.
  - This (probably) makes it difficult to predict the impact of branching.
  - Because the disjunctions used for cutting are much stronger than those used for branching, it seems more important to emphasize cuts.
  - On the other hand, cuts are expensive to generate.
  - We don't really know how to integrate MILP cuts and MIBLP cuts.
  - In general, the interaction of cutting and branching is much more intricate, which makes good control mechanisms vitally important.
  - Specific properties of instances (e.g., degree of alignment of objectives) can affect performance dramatically and this needs to be understood better.

Dual Bound

Possible relaxations

1. Remove the *optimality constraint of the second-level problem* (MIP relaxation)

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

2. Remove the *optimality constraint of the second-level problem* and the *integrality constraints* (LP relaxation)

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

3. Something in between? (Neighborhood relaxation)

$$\mathcal{R}_{\mathcal{N}}(x) = \{y \in \mathrm{Proj}_y(\mathcal{S}) \mid d^2 y \leq d^2 \bar{y} \quad \forall\, \bar{y} \in \mathcal{N}(y) \cap \mathrm{Proj}_y(\mathcal{S})\}$$

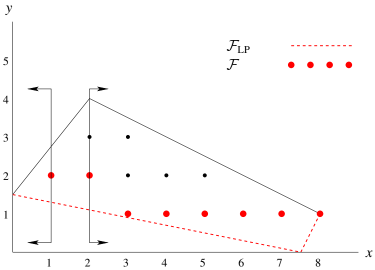
where $\mathcal{N}(y)$ is a neighborhood of y Xueyu et al. [2022].

Which Relaxation?

- What relaxation performs best is ultimately an empirical question, but we can reason about it.
- When solving MIBLPs, solving MILP subproblems seems to be a unavoidable. and these must be "tractable" to have any hope of solving.
- It is tempting to think that a stronger relaxation should be better.
- However, solving an MILP subproblem at each node means undertaking the very same process of branching that outer branch-and-cut will undertake.
- But more importantly, cut generation requires quick reoptimization.
- All in all, it only seems to make sense to use the LP relaxation for bounding..

Branching

- In general, there has been very little study of how to branch in solving MIBLPs.
- What we do today is use roughly the same rules for branching that are used in solving MILPs.
- Does this make sense? Not always...
- We may need to branch on variables that already have an integer value (more on this).
- MILP strategies predict the impact of branching using the dual bound as a proxy.
- In MIBLP, this is probably not a very good proxy.

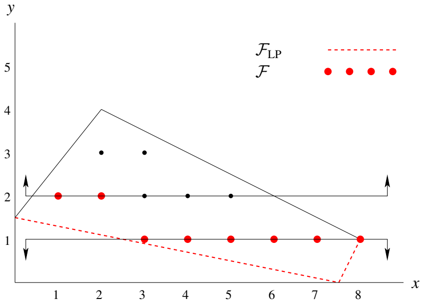


- One of the open challenges is to figure out a better prediction function.
- Currently, `MibS` uses straightforward pseudo-cost branching.

Branching Priorities

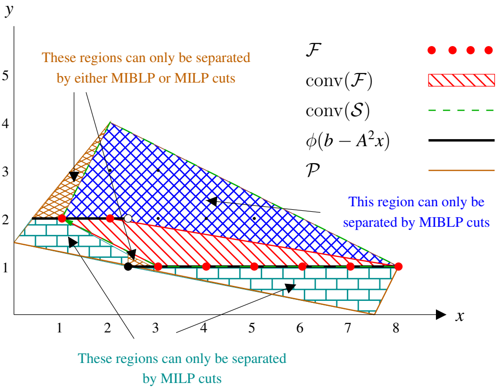
- There are several reason why one might think it is a good idea to prioritize branching on first-level variables.
  - The goal of bilevel optimization is to produce a first-level solution.
  - Once first-level variable values are fixed, the problem is reduced to an MILP.
  - Solving this resulting MILP effectively means that we just switch to branching on second-level variables.
  - But we do it using the heavy machinery of an MILP solver!! This is a win!!

- Unfortunately, this intuition seems to be (completely) wrong in `MibS` 1.2!
- In fact, it may be somewhat the opposite!
- Note, however, that standard MILP branching schemes do not work in pure branch-and-bound.



Cut Generation

- Unlike in MILP, we have several distinct classes of infeasible solution.
- Each requires different handling.
- Which types arise is (somewhat) dictated by the objective alignment.



1. $(\bar{x}, \bar{y}) \in \mathbb{R}^{n_1+n_2}$ for which $d^2\bar{y} \leq \phi(b^2 - A^2\bar{x}) \Leftarrow (\bar{x}, \bar{y}) \notin \mathcal{S}$
   - Need MILP cuts, but it's not easy to recognize this case!
2. $(\bar{x}, \bar{y}) \in \mathbb{R}^{n_1+n_2}$ for which $d^2\bar{y} > \phi(b^2 - A^2\bar{x}) \Leftarrow (\bar{x}, \bar{y})$ may or may not be in $\mathcal{S}$.
   - $\bar{x} \in X \Leftarrow$ Can evaluate $\phi(b^2 - A^2\bar{x})$ or $\Xi(\bar{x})$ to separate.
   - $\bar{y} \in Y \Leftarrow$ Relatively easier to separate with MIBLP cuts
   - $\bar{x} \notin X, \bar{y} \notin Y \Leftarrow$ Important, but tricky case!

Classes of Inequalities Valid for MIBLPs

Generalized Chvátal Cuts

- Let $C = \{(x, y) \in \mathcal{P} \mid \pi^x x + \pi^y y \leq \beta\}$ be a BFS, where $(\pi^x, \pi^y) \in X \times Y$, $\beta \in \mathbb{Z}$.
- Then $(\pi^x, \pi^y, \beta + 1)$ is valid for $\mathcal{F}$.

Intersection Cuts

- Let C be a convex set containing no improving solutions and let (x, y) be an extreme point of $\mathcal{P}$ in the interior of C.
- Then the intersection cut with respect to C and (x, y) is valid for $\mathcal{F}$.

Benders Cuts

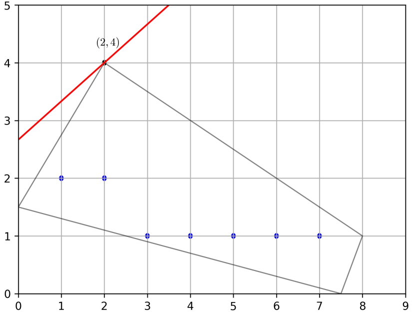
- Let $\bar{\psi} : \mathbb{R}^{n_1} \to \mathbb{R}$ be such that $\bar{\psi}(x) \geq \phi(b^2 - A^2 x)$ (a *primal function*).
- Then $C = \{(x, y) \in \mathcal{P} \mid d^2 y \geq \bar{\psi}(x)\}$ is a BFS and $d^2 y \leq \bar{\psi}(x)$ for all $(x, y) \in \mathcal{F}$.

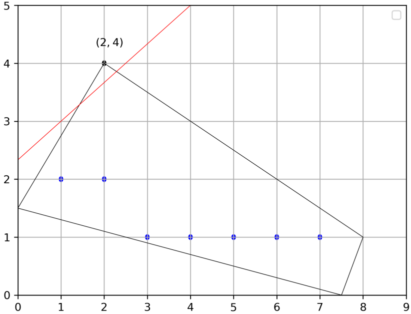
Classes Implemented in `MibS`

- MILP cuts.
- Generalized Chvátal (Integer no-good cut) [DeNegre and Ralphs, 2009]
- Benders Cuts
  - Benders Binary Cut [DeNegre, 2011]
  - Benders Interdiction Cut [**?**Caprara et al., 2014]
  - Benders Bound Cut [Tahernejad, 2019]
- Intersection cuts [Fischetti et al., 2017, 2018]
  - Improving Solution (Types I and II)
  - Improving Direction
  - Hypercube
- Generalized no-good cut [DeNegre, 2011]

Generalized Chvátal Inequalities

- Basic Idea: Just as in MILP, generate an inequality $(\alpha^x, \alpha^y, \beta)$ valid for $\mathcal{P}$ supporting $\mathcal{P}$ at extreme point $(x, y) \in \mathcal{S}$, where
  - $(\alpha^x, \alpha^y) \in X \times Y$,
  - $\beta \in \mathbb{Z}$
- If $(x, y) \notin \mathcal{F}$, then $\{(x, y) \in \mathcal{P} \mid \alpha^x x + \alpha^y y \leq \beta\} \cap \mathcal{F} = \emptyset$.
- Hence, $(\alpha^x, \alpha^y, \beta + 1)$ is valid for $\mathcal{F}$.





Benders Cuts

- Derive a primal bounding function for the second-level value function ϕ.
- Typically, this is done by exploiting known primal solutions.

Benders Binary Cut

Let $(\hat{x}, y^*) \in \mathcal{P}$ be such that $y^* \in \mathcal{P}_1(\hat{x}) \cap \mathcal{P}_2(\hat{x}) \cap Y$.

$$\bar{\psi}(x) = \begin{cases} d^2 y^* & \text{if } \begin{cases} x_i = 1 & \forall \{i \in L \setminus L^- \mid y_i^* = 0\} \\ x_i = 0 & \forall \{i \in L \setminus L^+ \mid y_i^* > 0\} \end{cases} \\ \infty & \text{otherwise} \end{cases}$$

where

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

- Basic Idea: For $(x, y) \in \mathcal{P}$, if $y^* \in \mathcal{P}_2(x)$ and $d^2 y > d^2 y^*$, then $(x, y) \notin \mathcal{F}$.
- This cut can be linearized with an appropriate big-M.

Benders Cuts (cont'd)

- A stronger version of the Benders Binary cut is valid for interdiction problems.

Benders Interdiction Cut

Let $(\hat{x}, y^*) \in \mathcal{P}$ be such that $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$.

$$\bar{\psi}(x) = \begin{cases} d^2 y^* - \delta(x) & \text{if } \begin{cases} x_i = 1 & \forall \{i \in L \setminus L^- \mid \hat{x}_i = 1\} \\ x_i = 0 & \forall \{i \in L \setminus L^+ \mid \hat{x}_i = 0\} \end{cases} \\ \infty & \text{otherwise} \end{cases}$$

where

- $L^- = \left\{ i \in L \;\middle|\; A_i^2 \leq 0 \right\}$,
- $L^+ = \left\{ i \in L \;\middle|\; A_i^2 \geq 0 \right\}$, and
- $\delta(x) = \sum_{\{i \in L^+ : y_i^* = 0, y_i = 1\}} d_i - \sum_{\{i \in L^- : x_i = y_i^* = 1\}} d_i$

- As before, this cut can be linearized with an appropriate big-M.
- As before, the cut eliminates $(x, y) \in \mathcal{P}$ such that $y^* \in \mathcal{P}_2(x)$ and $d^2 y > d^2 y^*$.

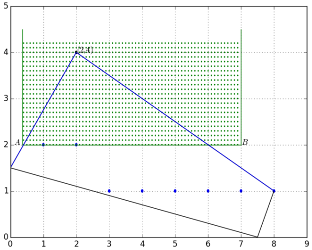
Improving Solution Intersection Cut (ISIC)

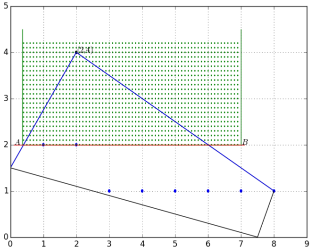
- For simplicity, assume all problem data are integral.
- Let $(\hat{x}, \hat{y})$ be an extreme point of $\mathcal{P}$ such that $d^2\hat{y} > d^2y^*$ for some $y^* \in \mathcal{P}_2(\hat{x}) \cap Y$ ($\Leftarrow$ the improving solution).

Bilevel Free Set

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

- The basic logic is very similar to the Benders cut.
- Crucially, note that we don't need $\hat{x} \in X$ or $\hat{y} \in Y$.





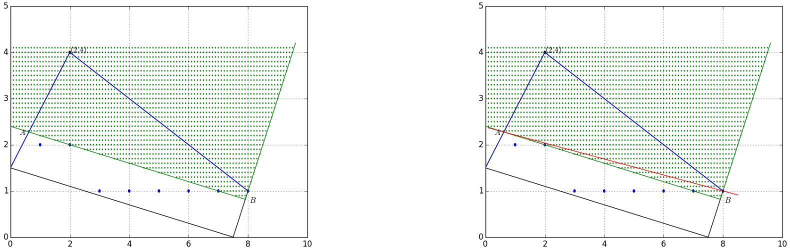
Improving Direction Intersection Cut (IDIC)

- Once again, assume all problem data are integral.
- Let $(\hat{x}, \hat{y})$ be an extreme point of $\mathcal{P}$ and let $\Delta y \in \mathbb{Z}^{n_2}$ ($\Leftarrow$ the improving direction) such that $\hat{y} + \Delta y \in \mathcal{P}_2(\hat{x})$ and $d^2 \Delta y < 0$

Bilevel Free Set

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

- Once again, note that we don't need $\hat{x} \in X$ or $\hat{y} \in Y$.



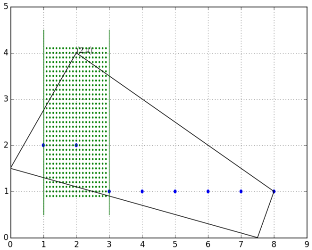
Hypercube Intersection Cut (HIC)

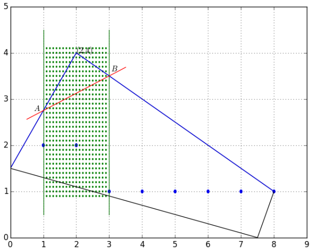
- Let $(\hat{x}, \hat{y}) \in \mathcal{P}$ with $\hat{x} \in X$.

Bilevel Free Set

$$C = \left\{ (x, y) \in \mathbb{R}^{n_1 \times n_2} \;\middle|\; \hat{x}_i - 1 \le x_i \le \hat{x}_i + 1 \quad \forall i < r_1 \right\}.$$

- Note that any solutions $(\hat{x}, y) \in \mathcal{F}$ may violate this cut, so we need to evaluate $\Xi(\hat{x})$ before imposing it.





Generalized No-good Cut

- This cut is valid when first-level variables are binary.
- Let $(\hat{x}, \hat{y}) \in \mathcal{P}$ such that $\hat{x} \in \mathbb{B}^{n_1}$.
- Once again, all solutions $(\hat{x}, y) \in \mathcal{F}$ violate this cut, so we need to evaluate $\Xi(\hat{x})$ before imposing it.

Generalized No Good

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

- The inequality is violated by all $(x, y) \in \mathcal{P}$ with $x = \hat{x}$.

Comparing the Classes Analytically : Size of $\mathrm{int}(C)$

Generalized Chvátal cuts

Only a single point $(x, y) \in \mathcal{S} \setminus \mathcal{F}$

HICs and Generalized no-good cuts

All $(\hat{x}, y) \in \mathcal{S}$ (feasible or not) for some $\hat{x} \in X$ such that $\Xi(\hat{x})$ is known
$\Rightarrow$ All combinations of a **fixed** $\hat{x}$ with any y.

Benders cuts and ISICs

All $(x, y) \in \mathcal{P}$ such that $y^* \in \mathcal{P}_2(x)$ and $d^2 y > d^2 y^*$
$\Rightarrow$ All (x, y^*) for which a **fixed** y^* proves infeasibility.

IDICs

$(x, y) \in \mathcal{P}$ such that Δy is an improving feasible direction for y, given x
$\Rightarrow$ All (x, y) for which **a fixed** Δy proves infeasibility.

ISICs versus IDICs

- For general IBLPs, it seems apparent that ISICs and IDICs provide the most "bang for the buck," but how do they compare to each other?
  - Both classes of inequalities can be used to separate arbitrary fractional solutions, which sets them apart.
  - Both also require solving an MILP subproblem.
  - The feasible regions of these subproblems are even (in a certain sense) equivalent.
  - Let
  $$\mathcal{W}(\hat{x}, \hat{y}) = \left\{ w \in \mathbb{Z}^{r_2} \times \mathbb{R}^{n_2 - r_2} \mid d^2 w < 0,\ \hat{y} + w \in \mathcal{P}_2(\hat{x}) \right\}.$$
  be the set of improving feasible directions with respect to $(\hat{x}, \hat{y}) \in \mathcal{P}$.
  - Then for any $(x, y) \in \mathcal{S}$,

$$(x, y) \in \mathcal{F} \Leftrightarrow \mathcal{W}(\hat{x}, \hat{y}) = \emptyset \Leftrightarrow \exists y^* \in \mathcal{P}_2(x) \cap Y \text{ with } d^2 y^* < d^2 y$$

- The crucial difference is that the construction of large bilevel free sets using the two different recipes requires much different solutions/directions.
  - To construct large bilevel free sets with IDICs, directions should be *short*
  - To construct large bilevel free sets with ISICs, solutions should be *high quality*.

Generating Improving Solutions/Directions

- Currently, the improving solutions and directions are generated a subproblem.
- Exactly which solution/direction is generated can affect performance dramatically.
- Hence, the objective function used is crucial. The goal is generally to get the deepest cut, but making the BFS as large as possible is a proxy.
- The objective function used for the subproblem determines what BFS will be produced.
- Currently, for ISIC, we have two objective functions.
  - Type I: Most improving solution.
  - Type II: Maximize the number of redundant constraints.
- For IDICs, we only have the second objective function.

Outline

Software Framework

`MibS` is an open-source solver for MIBLPs.

- Implements the branch-and-cut algorithm for MIBLPs described here.
- Implemented in C++.
- Built on top of the BLIS MILP solver [Xu et al., 2009].
- Employs software available from the *Computational Infrastructure for Operations Research (COIN-OR)* repository
  - *COIN High Performance Parallel Search (CHiPPS)*: To manage the global branch-and-bound
  - *SYMPHONY*: To solve the required MIPs (can also use Cbc or CPLEX)
  - *COIN LP Solver (CLP)*: To solve the LPs arising in the branch and cut.
  - *Cut Generation Library (CGL)*: To generate cutting planes within both SYMPHONY and `MibS`
  - *Open Solver Interface (OSI)*: To interface with other solvers

Data Sets

Table: The summary of data sets

Data Set	#	VT	V#	C#	Align	Notes
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]

Computational Experiments

- Nearly 20K CPU hours with four different versions of `MibS` with both SYMPHONY and CPLEX as subsolvers (and filmosi for comparison).
- Run on the COR@L cluster: 14 nodes, dual 8-core .8 GHz CPUs, 32 Gb memory
- Instances that took less than 5 seconds to solve for all versions were filtered.
- Which data sets are included are indicated in the title (X = XU, F=FIS, etc.)

Control Mechanism: Cut Generation

- As mentioned, not all cuts can separate all solutions.
  - $(\hat{x}, \hat{y}) \in \mathcal{S}$: all classes *except MILP cuts*.
  - $\hat{x} \in X$: no Generalized Chvátal
  - $\hat{y} \in Y$: ISICs of types I and II, IDIC,
  - $\hat{y} \notin Y$: Depends on whether $d^2\hat{y} > \phi(B^2 - A^2\hat{x})$!
- Whether/how to separate $(\hat{x}, \hat{y}) \notin \mathcal{S}$ involves important tradeoffs.
  - MILP cuts are relatively cheap, but may be redundant.
  - IDICs and ISICs are expensive, but it may be worth it in order to move the dual bound sooner (especially in the root node).
- Control mechanism for ISICs and IDICs.
  - If either $\hat{x} \in X$ or $\hat{y} \in Y$, IDICS are always generated and ISICs are only generated when the second-level problem has already been solved to check feasibility.
  - If $\hat{x} \notin X$ and $\hat{y} \notin Y$, IDICs and/or ISICs are generated if the associated parameter is set (and the second-level problem is already solved in the case of ISICs).
- Note that cut filtering by dynamism and density are disabled.

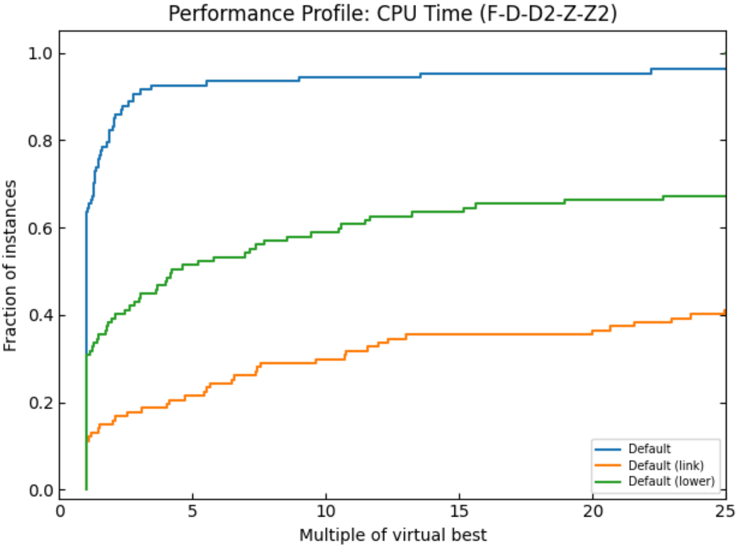
Control Mechanism: Solving Subproblems

- Options for when to solve the second-level problem.
  - Every iteration
  - When $x \in X$
  - When $(x, y) \in \mathcal{S}$ (equivalent to checking feasibility)
  - When first-level variables are fixed by branching (in this case, we evaluate Ξ)
- Evaluating Ξ involves solving one additional MILP.
- Options for when to do this are similar.
- Solving these subproblems is a pre-requisite for generating certain cuts.
  - When generating Benders binary cuts, Benders interdiction cuts, or ISICs of type 1, we must first solve the second-level problem.
  - When generating generalized no-good cuts and Hypercube ICs, we must also evaluate Ξ.
- By default, `MibS` currently only solves the second-level problem when $(x, y) \in \mathcal{S}$ and when x is fixed by branching.

Control Mechanism: Branching

- As with MILP, performance is highly sensitive to the stopping criteria for cut generation.
- In `MibS` 1.2, this stopping criteria is based on a simple tailing off scheme.
- Branching is forced when the relative change in gap is less than a parameter value (.05 is the current default).
- When branching on all variables and not just first-level, we need to allow cut generation to continue whenever the solution is integral, regardless.

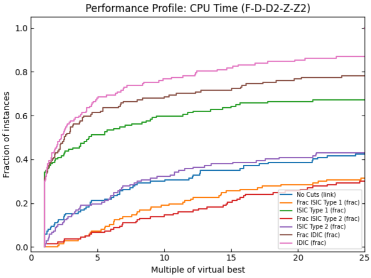
Comparing Branching Schemes

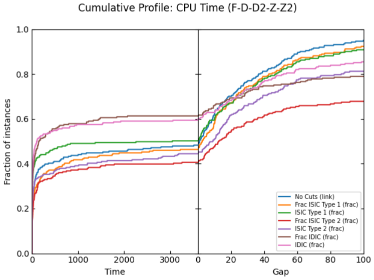


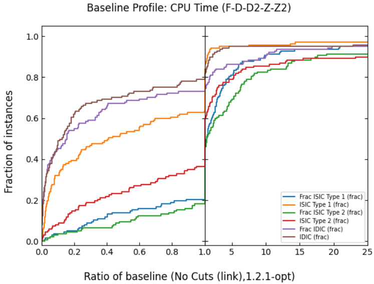
Comparing Cuts Empirically

- In the MILP context, it is typical to compare cuts using a closure bound or root gap to isolate the separate effects of branching and cutting.
- Results are displayed using a combination of
  - Performance profiles (CDF of the ratio
  - Cumulative profiles
  - Baseline profiles
- Performance measure
  - CPU time
  - Nodes evaluated
  - Root bound

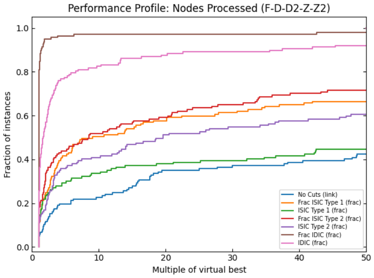
Summary Results (IDICs versus ISICs)

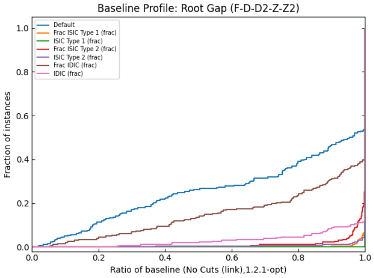






Summary Results (IDICs versus ISICs)



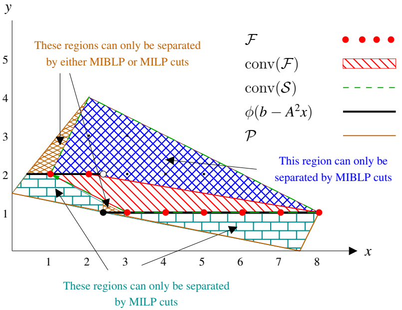


Analysis

- The overall winner by performance profile using CPU time as a performance measure is separating only integer solutions with IDICs.
- The cumulative profile shows that for problems that cannot be solved in one hour, the gap is more effectively closed by separating fractional solutions with IDICs.
- Note, however, that “No cuts” is actually the overall winner in terms of gap closed.
- In terms of tree size, separating fractional solution with IDICs is easily the winner, but the expense of generating doesn’t pay off in most cases.
- Nor surprisingly, fractional IDICs are also effective at closing the gap in the root node for some but not all instances.
- MILP cuts do help on top of MIBLP cuts in general.

Cut Generation Failure

- Recall that cut generation may fail when $d^2\hat{y} \leq \phi(b^2 - A^2\hat{x})$.
- The degree to which this is an issue varies a lot!
- In practice, it may be a big issue, but may be mitigated with better control mechanisms.

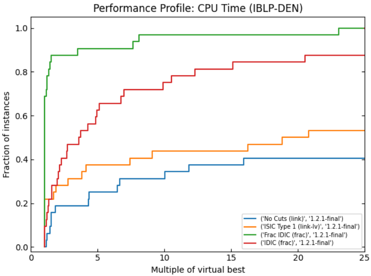


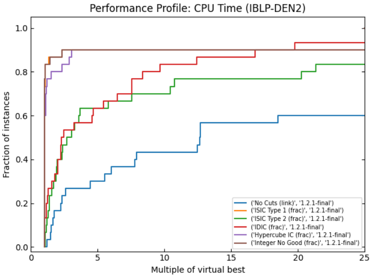
Branching Priority	Fractional Separation	DeNegre	Zhang	Interdiction
All	Yes	.62	.89	.84
Link	Yes	.62	.97	.79
All	No	.28	.15	.07
Link	No	.31	.96	.04

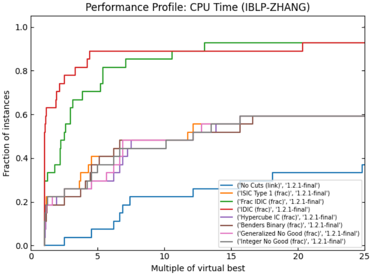
Performance on Individual Datasets

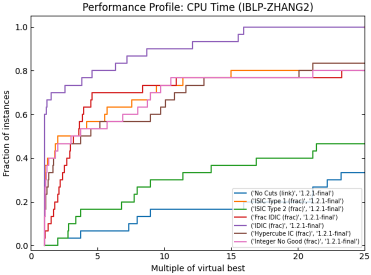
- Although the results look very uniform when aggregated, performance varied greatly between datasets.
- For the IBLP-DEN instances, separating fractional points with IDICs is the magic bullet.
- For the IBLP-DEN2 instances, Integer No Good Cuts perform as well as any other class and outperform IDICs.

Results on Individual Datasets

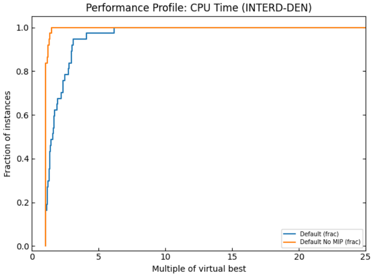


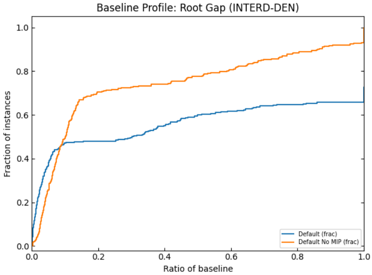


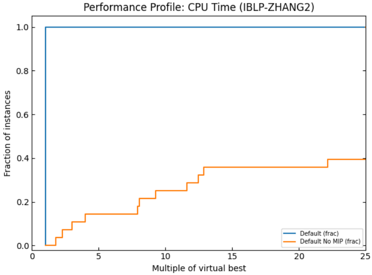


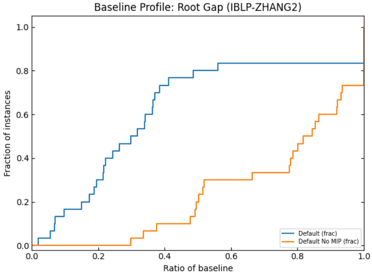


Do MILP Cuts Help?





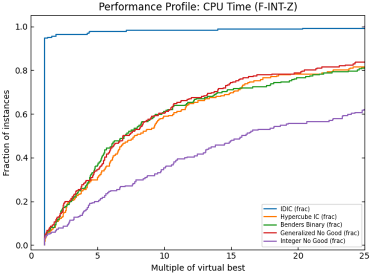


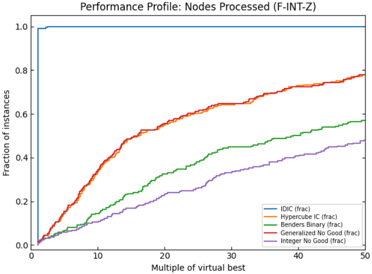


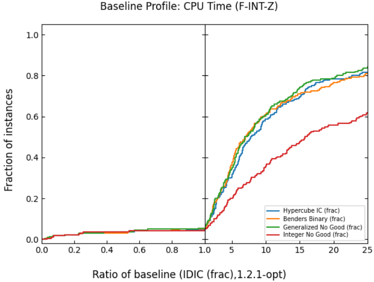
Binary Instances

- As with MILPs, instances with only binary first-level variables are a special case for which there are additional classes of inequalities.
- Surprisingly, however, the cuts specialized to binary instances do not outperform IDICs.

Summary Results (Binary Instances)



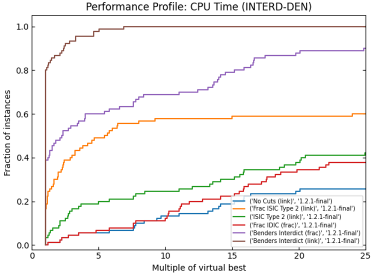


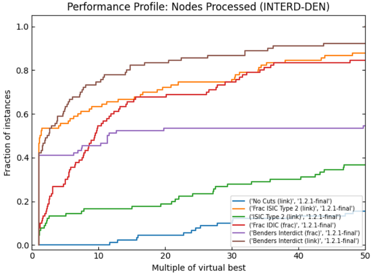


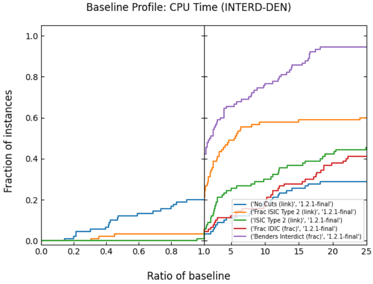
Interdiction Instances

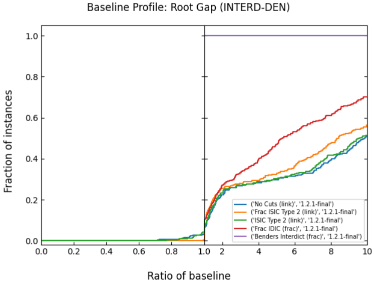
- For interdiction instances, Benders cuts are clearly dominant.
- This is not at all unexpected.
- It is the one class of problems for which there are specialized cuts that help.

Summary Results (Interdiction)

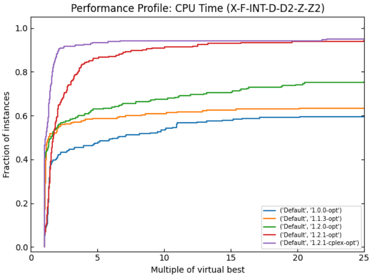


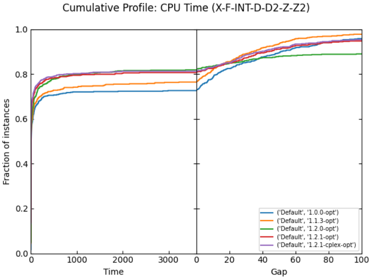


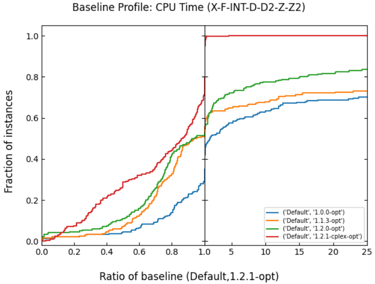




Overall Results: Different Versions of MibS



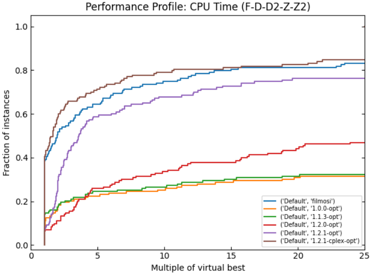


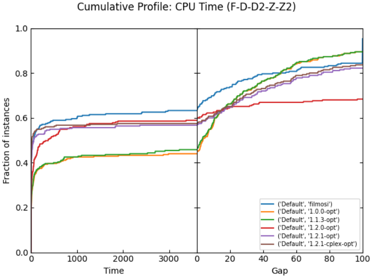


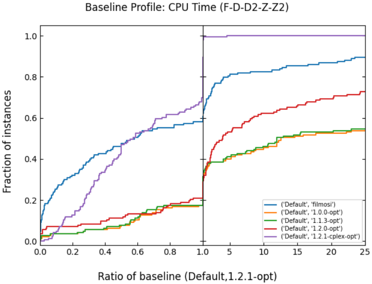
Analyzing the Results

- The improvements have mainly been in how many instances could be solved.
- Each new version has brought a new set of instances to solvability.
- For closing the gap on unsolved instances, older versions were better.
- This makes sense and is consistent with previous results.
- Using CPLEX as a subsolver only results in marginal gains.

Overall Results: Comparing MibS with filmosi







Analyzing the Results

- Using CPLEX as the underlying MILP solver, results are competitive with `filmosi` using default parameters settings for each.
- It is very difficult to tell what is going on inside `filmosi` and this is a bit of an apples-oranges comparison.
- There is still a lot of low-hanging fruit to improve `MibS`, but it is unclear what can be done with `filmosi`.
- Many of the things one might want to do to improve performance are not possible with a closed-source solver.

Outline

Future Directions

- **There are still many avenues for improving performance and much low-hanging fruit.**
  - Improved branching
  - Better dynamic control mechanisms for cut generation (better integration of MIBLP and MILP cuts)
  - Warm-starting of subproblem solvers (SYMPHONY)
  - Pools of solutions/directions/cuts
  - ...
- **Existing capabilities that need further development**.
  - Stochastic bilevel solver
  - Pessimistic solver
  - Bounded rationality

The Dream

How would we design a solver if we could do it from the ground up?

- No explicit subsolvers, just one tightly integrated solver.
- Flexible reaction sets (bounded rationality).
- Flexible base relaxations.
- Solver based completely on improving directions?

Conclusions

- Solutions of MIBLPs is where solution of MILPs was 15 years ago.
- The basic theory is well-developed, but in practice, solvers are well-tuned bags of tricks.
- MILP solvers are still improving, thanks largely to commercial viability and fierce competition.
- It remains to be seen if MIBLP solvers will follow a similar trajectory.

References I

A. Caprara, M. Carvalho, A. Lodi, and G.J. Woeginger. Bilevel knapsack with interdiction constraints. Technical Report OR-14-4, University of Bologna, 2014.

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

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

M. Fischetti, I. Ljubić, M. Monaci, and M. Sinnl. A new general-purpose algorithm for mixed-integer bilevel linear programs. *Operations Research*, 65(6):1615–1637, 2017.

Matteo Fischetti, Ivana Ljubić, Michele Monaci, and Markus Sinnl. On the use of intersection cuts for bilevel optimization. *Mathematical Programming*, 172(1-2): 77–103, 2018.

References II

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

S. Tahernejad. *Two-stage Mixed Integer Stochastic Bilevel Linear Optimization*. PhD, Lehigh University, 2019.

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

Y. Xu, T.K. Ralphs, L. Ladányi, and M.J. Saltzman. Computational Experience with a Software Framework for Parallel Integer Programming. *The INFORMS Journal on Computing*, 21:383–397, 2009. doi: 10.1287/ijoc.1090.0347. URL `http://coral.ie.lehigh.edu/~ted/files/papers/CHiPPS-Rev.pdf`.

Shi Xueyu, Oleg A. Prokopyev, and Ted K. Ralphs. Mixed Integer Bilevel Optimization with a k-optimal Follower: A Hierarchy of Bounds. *Mathematical Programming Computation*, 15:1—-51, 2022. doi: 10.1007/s12532-022-00227-z. URL `http://coral.ie.lehigh.edu/~ted/files/papers/HierarchyOfBounds20.pdf`.

References III

Junlong Zhang and Osman Y Ozaltın. A branch-and-cut algorithm for discrete bilevel linear programs. *Optim. Online*, 2017.