Computational Experience Solving Bilevel Integer Programs

The Mixed Integer Bilevel Solver (MibS) Framework

Ted Ralphs[1]

Joint work with:
Scott DeNegre[1], Menal Guzelsoy[2]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
[2]ISyE, Georgia Institute of Technology

INFORMS 2011, Charlotte
16 November 2011

Outline

- Problem Statement
- Branch & Cut for Pure Integer Problems
- Specialized Methods for Interdiction Problems
- Heuristic Methods for General Problems
- The Mixed Integer Bilevel Solver (MibS)
- Computational Results

Bilevel Programming

- Bilevel programming is a generalization of traditional mathematical programming that applies to *hierarchical decision systems*.

- In a *bilevel program*:
  - The *variables* are divided into two groups controlled by separate DMs.
  - The *constraints* of the *lower-level DM* involve the variables of *upper-level DM*.
  - The DMs have independent, possibly conflicting objectives.

- Conceptually, the variables are fixed sequentially in accordance with the inherent system hierarchy.

- We assume *perfect information* and *individual rationality* of the DMs.
  - The upper-level DM will be able to predict the reaction of the lower-level DM to decisions made at the upper-level.
  - We can collapse the hierarchy into a single optimization model in which the decisions made at the highest level determine the system outcome.

Canonical (Mixed Integer) Bilevel Linear Program

The general *bilevel linear program* is described as follows.

- $x \in X \subseteq \mathbb{R}^{n_1}_+$ are the *upper-level variables*
- $y \in Y \subseteq \mathbb{R}^{n_2}_+$ are the *lower-level variables*

(Mixed Integer) Bilevel Linear Program

$$\min \left\{ c^1 x + d^1 y \mid x \in \mathcal{P} \cap X, y \in \operatorname{argmin}\{ d^2 y \mid y \in \mathcal{S}(x) \cap Y \} \right\} \qquad \text{(MIBLP)}$$

The *upper-* and *lower-level feasible regions* are

$$\mathcal{P}_U = \left\{ x \in \mathbb{R}^{n_1}_+ \mid A^1 x \geq b^1 \right\}$$

and

$$\mathcal{S}_L(x) = \left\{ y \in \mathbb{R}^{n_2}_+ \mid G^2 y \geq b^2 - A^2 x \right\}.$$

We consider the general case in which $X = \mathbb{Z}^{p_1} \times \mathbb{R}^{n_1 - p_1}$ and $Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}$.

Notation

We utilize the following notation:

Notation

$$
\begin{aligned}
\Omega^I &= \{(x,y) \in X \times Y \mid x \in \mathcal{P}_U, y \in \mathcal{S}_L(x)\} \\
\Omega &= \{(x,y) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n_2} \mid x \in \mathcal{P}_U, y \in \mathcal{S}_L(x)\} \\
M^I(x) &= \operatorname{argmin}\{d^2 y \mid y \in \mathcal{S}_L(x) \cap Y\} \\
\mathcal{F}^I &= \left\{(x,y) \mid x \in \mathcal{P}_U \cap X, y \in M^I(x)\right\} \\
\mathcal{F} &= \left\{(x,y) \mid x \in \mathcal{P}_U, y \in \operatorname{argmin}\{d^2 y \mid y \in \mathcal{P}_L(x)\}\right\}
\end{aligned}
$$

We make the following additional technical assumptions:

Assumptions

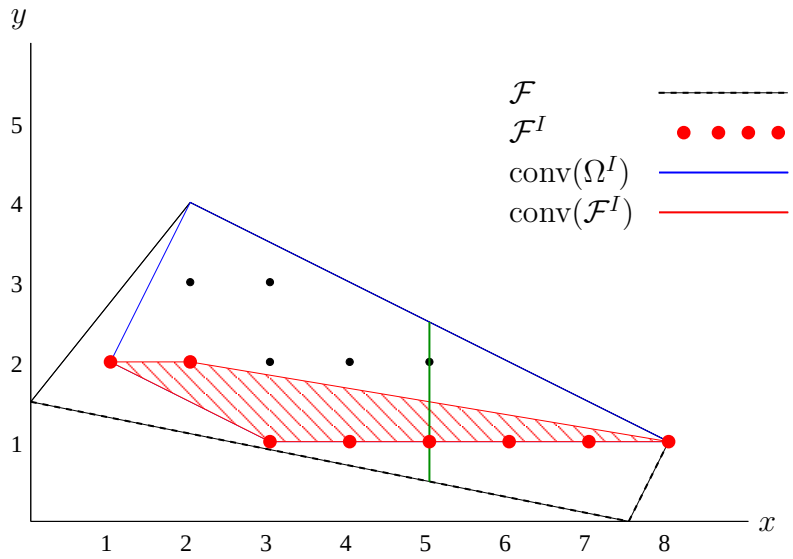
1. Ω^I is nonempty and compact.
2. For every action by the upper-level DM, the problem faced by the lower-level DM is feasible ($\mathcal{S}_L(x) \cap Y \neq \emptyset$).
3. The lower-level LP is bounded ($\min_{y \in \mathcal{S}_L(x)} d^2 y > -\infty$).
4. The lower-level DM is semi-cooperative.

Bilevel Programming Feasible Regions

The following *pure integer* instance is from Moore and Bard (1990).

$$\min_{x \in \mathbb{Z}} \quad -x - 10y$$

$$\begin{aligned} \text{subject to} \quad y \in \operatorname{argmin} \{ y : & \; 25x - 20y \geq -30 \\ & -x - 2y \geq -10 \\ & -2x + y \geq -15 \\ & 2x + 10y \geq 15 \\ & y \in \mathbb{Z} \; \} \end{aligned}$$



From the figure, we can make several observations:

1. $\mathcal{F} \subseteq \Omega$, $\mathcal{F}^I \subseteq \Omega^I$, and $\Omega^I \subseteq \Omega$
2. $F^I \not\subseteq \mathcal{F}$
3. Solutions to bilevel programs do not necessarily occur at extreme points of conv(Ω^I)

Bounding Method

Relaxing integrality conditions and the requirement $y \in M^I(x)$ yields the *relaxation*

$$\min_{(x,y)\in\Omega} c^1 x + d^1 y. \tag{LR}$$

- The resulting bound can be used in combination with a standard variable branching scheme to yield an algorithm that solves (MIBLP).
- The bound is too weak to be effective on interesting problems.
- We can strengthen the linear relaxation with inequalities valid for $\mathcal{F}^I$ to improve the bound.
- To improve the bound, we consider violations of the feasibility conditions.

Bilevel Feasibility Conditions

1. $(x, y) \in \Omega$,
2. $(x, y) \in X \times Y$, and
3. $y \in M^I(x)$.

Bilevel Feasibility Check

- Let $(\hat{x}, \hat{y})$ be a solution to the lower bounding problem (LR).
- We fix $x = \hat{x}$ and solve the lower-level problem

$$\min_{y \in \mathcal{P}_L(\hat{x})} d^2 y \qquad (\mathrm{LL}(x))$$

  with the fixed upper-level solution $\hat{x}$.
- Let y^* be the solution to $(\mathrm{LL}(x))$.
  - $(\hat{x}, y^*)$ is bilevel feasible $\Rightarrow c^1\hat{x} + d^1 y^*$ is a valid upper bound on the optimal value of the original MIBLP
  - Either
    1. $d^2\hat{y} = d^2 y^* \Rightarrow (\hat{x}, \hat{y})$ is bilevel feasible.
    2. $d^2\hat{y} > d^2 y^* \Rightarrow (\hat{x}, \hat{y})$ is *bilevel infeasible*.
- What do we do in the case of bilevel infeasibility?
  - Generate a valid inequality violated by $(\hat{x}, \hat{y})$.
  - Branch on a disjunction violated by $(\hat{x}, \hat{y})$.

Bilevel Feasibility Cuts

In the case where $X = \mathbb{Z}^{n_1}$ and $Y = ^{n_2}$, we can derive simple valid inequalities to strengthen the relaxation. Let

$$A := \begin{bmatrix} A^1 \\ A^2 \end{bmatrix}, \quad G := \begin{bmatrix} 0 \\ G^2 \end{bmatrix}, \quad \text{and} \quad b := \begin{bmatrix} b^1 \\ b^2 \end{bmatrix}.$$

A basic feasible solution $(\hat{x}, \hat{y}) \in \Omega^I$ to (LR) is the *unique* solution to

$$a_i' x + g_i' y = b_i, \quad i \in I$$

where I is the set of active constraints at $(\hat{x}, \hat{y})$.

This implies that

$$\left\{ (x, y) \in \Omega^I \mid \sum_{i \in I} a_i' x + g_i' y = \sum_{i \in I} b_i \right\} = \left\{ (\hat{x}, \hat{y}) \right\}$$

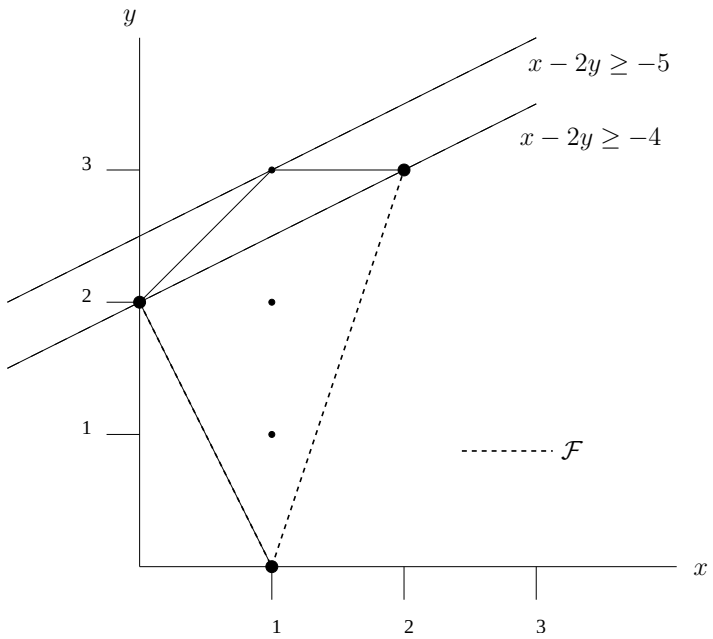
and $\sum_{i \in I} a_i' x + g_i' y \geq \sum_{i \in I} b_i$ is valid for $\Omega^I \setminus \{(\hat{x}, \hat{y})\}$.

Illustrating the Bilevel Feasibility Cut

A Valid Inequality

$\sum_{i \in I} a_i' x + g_i' y \geq \sum_{i \in I} b_i + 1$ is valid for $\mathcal{F}^I$.

$$\begin{aligned} \min_{x \in \mathbb{Z}} \quad & -y \\ \text{subject to} \quad & y \in \operatorname{argmin} \{ y : \quad x - y \geq -2 \\ & \qquad\qquad 2x + y \geq 2 \\ & \qquad\qquad -3x + y \geq -3 \\ & \qquad\qquad -y \geq -3 \\ & \qquad\qquad y \in \mathbb{Z} \ \} \end{aligned}$$



This yields a finite algorithm for pure integer problems.

Primal Heuristics

Improving Objective Cut

- Let $(\hat{x}, \hat{y}) \in \Omega^I$ and $y^* \in M^I(\hat{x})$
- We add the cut $d^2y \leq d^2y^*$ to the current relaxation and reoptimize over

$$\Omega^I \cap \{(x, y) \in (X \times Y) \mid d^2y \leq d^2y^*\}.$$

- Drive solutions towards bilevel feasibility while maintaining quality w.r.t. the upper-level objective.

Lower-level Priority

- Intuitively, this method gives priority to the lower-level DM.
- We optimize d^2y over the joint feasible region Ω^I, to obtain a *bilevel feasible* solution.
- Then, add a cut of the form $c^1x + d^1y \leq L$, where L the value of the best-known solution, and reoptimize over

$$\Omega^I \cap \{(x, y) \in (X \times Y) \mid c^1x + d^1y \leq L\}.$$

- Drive local bilevel feasible solutions towards optimality.

Interdiction Specialization

MibS is designed to allow for easy implementation of specialized methods using specific problem structure.

We have implemented customized methods for *Interdiction Problems*

- Specialized Cuts that exploit problem structure
  - *No-good cuts*

$$\sum_{i \in I_0} x_i + \sum_{i \in I_1} x_i \geq 1 - |I_1|,$$

    impose $x \neq \hat{x}$, where $I_0 := \{i \mid \hat{x}_i = 0\}$ and $I_1 := \{i \mid \hat{x}_i = 1\}$.
  - *Increasing Objective Cuts*, based on the disjunction:

$$\left\{ \begin{array}{rcl} \sum_{i \in I_0} x_i & \leq & 0 \\ d^2 y & \leq & d^2 y^* \end{array} \right\} \bigvee \left\{ \begin{array}{rcl} \sum_{i \in I_0} x_i & \geq & 1, \end{array} \right\}$$

- Greedy Interdiction Heuristic
  - Any feasible interdiction plan combined with the resulting lower-level optimal solution yields a bilevel feasible solution.
  - One way to obtain a feasible interdiction plan is to inderdict lower-level variables with the smallest objective coefficients until our budget is expended.

Weighted Sums Heuristic

We can attempt to balance *optimality* and *feasibility* using a *weighted objective*.

- Previous heuristic methods favored one objective or the other.
- This method aims to find solutions that balance the objectives of the DMs.

We form the related *multiobjective program*:

$$\text{vmin}_{(x,y)\in\Omega^I} [c^1x + d^1y, d^2y]. \tag{1}$$

We can find *efficient solutions* to (1) using the weighted-sum subproblem:

$$\min_{(x,y)\in\Omega^I} \delta(c^1x + d^1y) + (1-\delta)d^2y,$$

for $0 \leq \delta \leq 1$ (Geoffrion, 1968).

- This method does not generate all efficient solutions.
- For the purposes of a heuristic method, generating a portion of the efficient set is likely sufficient.
- Note, efficient solutions are not bilevel feasible in general, but a feasible solution is obtained via the bilevel feasibility check.

Stationary Point Heuristic

Alternatively, we can construct solutions composed of upper- and lower-level solutions of high quality w.r.t. their individual objective functions.

- Solutions to the underlying MILP are typically infeasible, due to suboptimality w.r.t. the lower-level objective.
- Solutions found by optimizing w.r.t. d^2y over Ω^I are unlikely optimal for the original (upper-level) objective.
- The heuristic attempts to find an *equilibrium* between the objectives.

Recall the lower-level problem, for fixed $x \in X$:

$$z_{LL}(x) = \min_{y \in \mathcal{S}(x)} d^2 y. \tag{2}$$

We define the *constrained upper-level problem*, for fixed $y \in Y$:

$$\begin{aligned} \min_{x \in X} \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \geq b^1 \\ & A^2 x \geq b^2 - G^2 y \end{aligned} \tag{3}$$

We alternate between solutions to (2) and (3) until we arrive at a stationary point $(\hat{x}, \hat{y})$.

Implementation

The Mixed Integer Bilevel Solver (MibS) implements the branch and bound framework and heuristic methods described here using software available from the Computational Infrastructure for Operations Research (COIN-OR) repository.

COIN-OR Components Used

- The COIN High Performance Parallel Search (CHiPPS) framework to perform the branch and bound.
- The COIN Branch and Cut (CBC) framework for solving the MILPs.
- The COIN LP Solver (CLP) framework for solving the LPs arising in the branch and cut.
- The Cut Generation Library (CGL) for generating cutting planes within CBC.
- The Open Solver Interface (OSI) for interfacing with CBC and CLP.

Please visit `www.coin-or.org` for information on obtaining these codes.

Primary MibS Classes

The primary classes the comprise MibS are as follows:

MibS Classes

1. `MibSModel`: Derived from the virtual BLIS class `BlisModel` and stores information about the original problem.
2. `MibSCutGenerator`: Derived from the virtual BLIS class `BlisConGenerator`, and used to generate cuts when CHiPPs finds integer, bilevel infeasible solutions.
3. `MibSSolution`: Derived from the virtual BLIS class `BlisSolution` and stores and prints integer bilevel feasible solutions.
4. `MibSBilevel`: Specific to MibS and used to test bilevel feasibility of integer solutions.
5. `MibSHeuristic`: Specific to MibS and used to generate heuristic solutions.

What Is Implemented

The current MibS release includes:

- Branch and Cut for IBLPs
  - *Bounding problems*
  - *Bilevel feasibility cuts*
  - Several *primal heuristics*
  - Simple *preprocessing*
- Specialized methods (primarily cuts) for specific problem classes
  - *Pure binary at the upper level*
  - *Interdiction problems*
- Standalone heuristics
  - *Greedy* method for interdiction problems
  - *Weighted sums* method for general problems
  - *Stationary point* method for general problems

MibS will be in COIN "soon" and is available now at

`http://coral.ie.lehigh.edu/projects/MibS`

Computational Results: Branch & Cut

Random IBLP Instances

- We created biobjective ILP instances with two objectives.
- Then, randomly chose a set of lower-level columns.
- All coefficients were chosen in the range $[-50, 50]$.
- All ILP rows were controlled by the lower-level DM (as in the examples from Moore and Bard (1990)).

Problem Class	Num Rows	Num Cols	Num Lower
1	20	20	5
2	20	20	10
3	20	20	15
4	30	20	10
5	40	20	10

Results from MibS on the random instances:

Class	No. Optimal	Avg. Gap (%)	Avg. No. Nodes	Avg. No. Cuts	Avg. CPU (s)
1	9	25.04	104156.40	43879.70	198.23
2	5	34.49	294798.30	135170.30	3787.24
3	2	49.45	231665.80	132728.10	600.82
4	5	61.55	286957.90	153211.00	3101.58
5	6	25.28	190189.10	79750.50	414.09

- Limit of 30000 CPU seconds for each instance.
- Only bilevel feasibility cuts were employed (no generic MILP cuts).
- AMD Opteron Processor 6128 with 32GB of memory.

Computational Results: Heuristic Methods

We tested our heuristic methods on the same instance classes.

- The optimal value of the underlying MILP was used as lower bound.
- Simple heuristics were used to determine an upper bound for each instance.

Weighted Sums Heuristic Results:

- Average gap between heuristic solution value and the lower bound is 45%.
- Average improvement over simple heuristics is 41%.

Stationary Point Heuristic Results:

- Average gap between the Weighted Sums objective value and that of the lower bound is 54%
- Average improvement over simple heuristics is 10%.

Comparison to MibS:

- The MibS solution is always at least as good as those obtained by the heuristic methods.
- In 11/50 instances tested, at least one of the heuristics performs as well as MibS.
- Average increase in objective value over the best MibS solution is 34%.
- In 24/50 instances this gap is less than 10%.

Computational Results: Heuristic Comparison

- The Weighted Sums Heuristic obtained a lower objective value than the Stationary Point Heuristic in 25 instances
- The reverse was true only 6 times.
- When the Weighted Sums Heuristic performed better, it tended to do so by a lot (average of 74%).
- When the Stationary Point Heuristic yielded the better solution, it was only about 2% better.
- The Stationary Point Heuristic is twice as fast, on average.

Computational Results: ILP Interdiction

We tested our solver customization on a set random ILP interdiction problems.

- The goal of the upper-level DM is to minimize the maximum cost solution achievable by the lower-level DM, who is solving a random ILP.
- A cost is associated with the interdiction of each lower-level variable.
- Instances were generated from a set of ILPs with randomly-chosen coefficients in the range $[-50, 50]$ and an interdiction cost vector.
  - *Unit Costs*: Unit interdiction cost assigned to all lower-level variables, budget of 3 interdictions.
  - *Random Costs*: Randomly-selected cost assigned to all lower-level variables, budget chosen to allow roughly 3 interdictions.

Problem Class	Num Rows	Num Cols
1	10	10
2	15	10
3	20	20

Results from the exact solver:

	Avg. No. Nodes		Avg. Depth		Avg. Gap (%)		Avg. No. Cuts		Avg. CPU (s)	
Class	Unit	Random	Unit	Random	Unit	Random	Unit	Random	Unit	Random
1	13255.20	13199.40	23.00	23.60	—	—	13327.40	10828.80	187.12	162.42
2	58639.40	54055.20	24.30	24.60	—	—	49633.20	35861.00	583.62	452.66
3	364098.70	559028.00	46.20	47.40	161.23	160.78	294524.40	249707.80	—	—

Computational Results: Greedy Interdiction

We tested the greedy heuristic method on the same instance classes.

- The optimal value of the underlying MILP was used as lower bound.
- Simple heuristics were used to determine an upper bound for each instance.
- The *interdiction effect* is defined as the decrease in the optimal lower-level solution value and is reported as a % of the lower-level objective value.

	Unit		Random	
Class	Avg. Best Effect (%)	Avg. Heuristic Effect (%)	Avg. Best Effect (%)	Avg. Heuristic Effect (%)
1	79.15	72.23	74.16	56.02
2	83.87	70.59	80.57	65.51
3	49.20	36.82	51.12	30.63

- The exact solver produces more effective interdiction strategies (as expected).
- The heuristic solutions do, however, have a significant effect on the lower-level objective value and can suffice if a quick solution is needed.
- The difference between the two methods is less dramatic using symmetric interdiction costs.

Computational Results: Preliminary Conclusions

- Even pure integer problems are hard to solve.
- IBLP problem difficulty appears to be dependent on the % of lower-level columns in an instance.
- The primal heuristics yield a significant reduction in the average number of tree nodes and cuts required to solve the IBLPs.
- The difficulty of random instances is volatile and may be linked to the relationship between the objective functions.
- The standalone heuristic methods find reasonably good solutions to IBLPs with very little computational effort.
- Unit interdiction costs tend to increase the difficulty of determing an optimal interdiction strategy.

Future Work

- We have developed an exact algorithm for the general case that relies on approximations of the lower-level *value function* (DeNegre, 2011)
  - The algorithm iteratively improves this approximation using a bounding function
  - The idea is similar in spirit to Benders' Reformulation for MILP
- Implementing this algorithm is challenging
  - Each iteration requires solution of an MILP with a piecewise linear constraint
  - Reformulation of this subproblem can be accomplished if we are able to enumerate the vertices of a particular LP
- In addition to improving the current implementation, we intend to augment MibS with the algorithm for the general case

References I

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

Geoffrion, A. 1968. Proper efficiency and the theory of vector maximization. *Journal of Mathematical Analysis and Applications* **22**, 618–630.

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