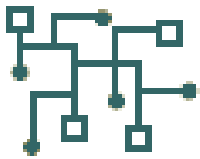
Bilevel Integer Optimization: Theory and Algorithms

Ted Ralphs[1]

Joint work with Sahar Tahernajad[1], Scott DeNegre[3], Menal Güzelsoy[2], Anahita Hassanzadeh[4]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
[2]SAS Institute, Advanced Analytics, Operations Research R & D
[3]The Hospital for Special Surgery
[4]Climate Corp

ISMP, Pittsburgh, PA, July 15, 2015



ISE Industrial and Systems Engineering

COR@L COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Multilevel and Multistage Games

- *Multilevel* games are those in which two players compete to optimize their separate utility functions.
- *Multistage* games are cooperative games in which all players receive the same payoff, but there are so-called *chance players*.
- A *subgame* is the part of a game that remains after some moves have been made.

Stackelberg Game

- A Stackelberg game is a game with two players who make one move each.
- The goal is to find a *subgame perfect Nash equilibrium*, i.e., the move by each player that ensures that player's best outcome.

Recourse Game

- A cooperative game in which play alternates between cooperating players and chance players.
- The goal is to find a *subgame perfect Markov equilibrium*, i.e., the move that ensures the best outcome in a probabilistic sense.

(Standard) Mixed Integer Linear Programs

- In parts of the talk, we will need to consider a (standard) *mixed integer linear program* (MILP).
- To simplify matters, when we discuss a standard MILP, it will be of the form

MILP

$$\min\{c^\top x \mid x \in \mathcal{P} \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p})\}, \qquad \text{(MILP)}$$

where $\mathcal{P} = \{x \in \mathbb{R}^n_+ \mid Ax = b\}$, $A \in \mathbb{Q}^{m\times n}$, $b \in \mathbb{Q}^m$, $c \in \mathbb{Q}^n$.

Bilevel (Integer) Linear Programming

Formally, a *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*

Bilevel (Integer) Linear Program

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

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

$$\mathcal{P}_U = \left\{ x \in \mathbb{R}_+ \mid A^1 x \leq b^1 \right\} \text{ and}$$
$$\mathcal{P}_L(x) = \left\{ y \in \mathbb{R}_+ \mid G^2 y \geq b^2 - A^2 x \right\}.$$

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

Special Cases

- Pure integer.
- Positive constraint matrix at lower level.
- Binary variables at the upper and/or lower level.
- *Interdiction problems*.

Mixed Integer Interdiction

$$\max_{x \in \mathcal{P}_U^I} \min_{y \in \mathcal{P}_L^I(x)} dy \qquad \text{(MIPINT)}$$

where

$$\mathcal{P}_U^I = \left\{ x \in \mathbb{B}^n \mid A^1 x \leq b^1 \right\}$$
$$\mathcal{P}_L^I(x) = \left\{ y \in \mathbb{Z}^p \times \mathbb{R}^{n-p} \mid G^2 y \geq b^2, y \leq u(e - x) \right\}.$$

- The case where follower's problem has network structure is called the *network interdiction problem* and has been well-studied.
- The model above allows for lower-level systems described by general MILPs.

Recourse Problems

- If $d^1 = -d^2$, we call this as a *mathematical program with recourse*.
- We can reformulate the bilevel program as follows.

$$\min\{-c^1 x + Q(x) \mid x \in \mathcal{P}_U \cap X\}, \tag{1}$$

where

$$Q(x) = \min\{d^1 y \mid y \in \mathcal{P}_L(x) \cap Y\}. \tag{2}$$

where Q is known as the *value function* of the recourse problem.

- *Two-stage stochastic programs with recourse* are a related class, formulated as

$$\min\{-c^1 x + \mathbb{E}_\xi Q_\xi(x) \mid x \in X\}, \tag{3}$$

where

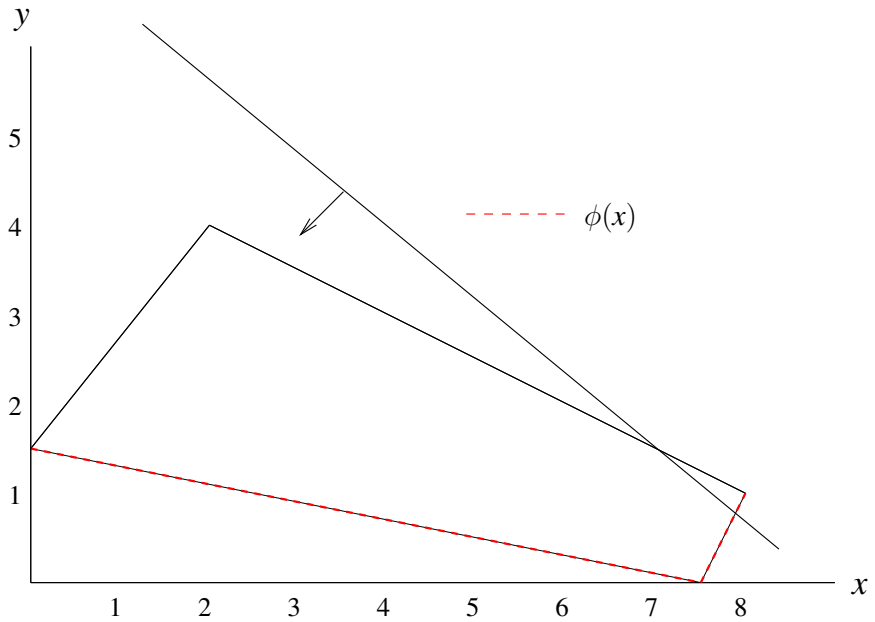
$$Q_\xi(x) = \min\{d^1 y \mid y \in Y, Wy = h_\xi - Tx\}, \tag{4}$$

Benders' Principle (Linear Programming)

$$z_{LP} = \min_{(x,y)\in\mathbb{R}^n} \left\{ c'x + c''y \;\middle|\; A'x + A''y \geq b \right\}$$
$$= \min_{x\in\mathbb{R}^{n'}} \left\{ c'x + \phi(b - A'x) \right\},$$

where

$$\phi(d) = \min\ c''y$$
$$\text{s.t. } A''y \geq d$$
$$y \in \mathbb{R}^{n''}$$



Basic Strategy:

- The function ϕ is the *value function* of a linear program.
- The value function is piecewise linear and convex.
- We iteratively approximate it by generating *dual (lower-bounding) functions*, as we will see.

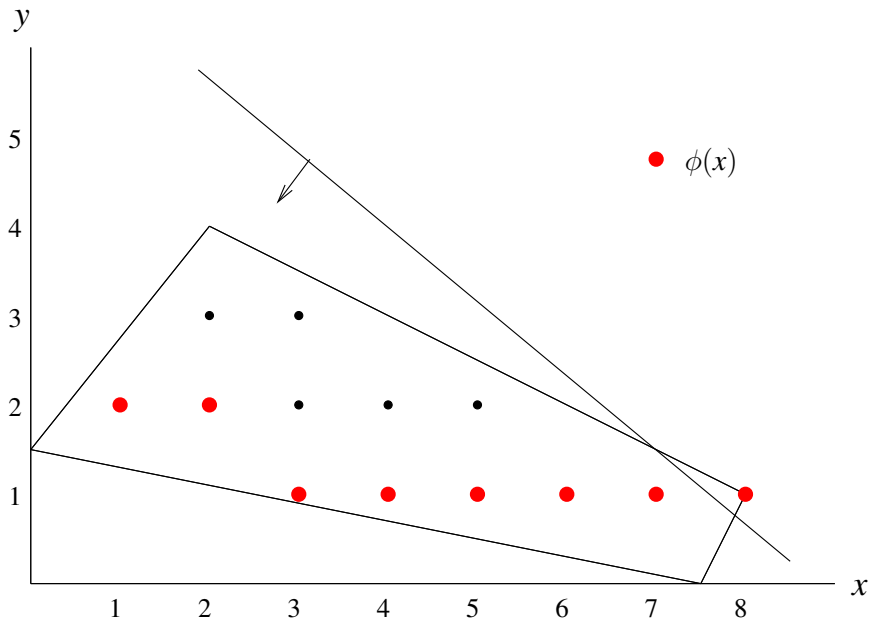
Benders' Principle (Integer Programming)

$$z_{IP} = \min_{(x,y)\in\mathbb{Z}^n} \left\{ c'x + c''y \;\middle|\; A'x + A''y \geq b \right\}$$

$$= \min_{x\in\mathbb{R}^{n'}} \left\{ c'x + \phi(b - A'x) \right\},$$

where

$$\phi(d) = \min\ c''y$$
$$\text{s.t. } A''y \geq d$$
$$y \in \mathbb{Z}^{n''}$$

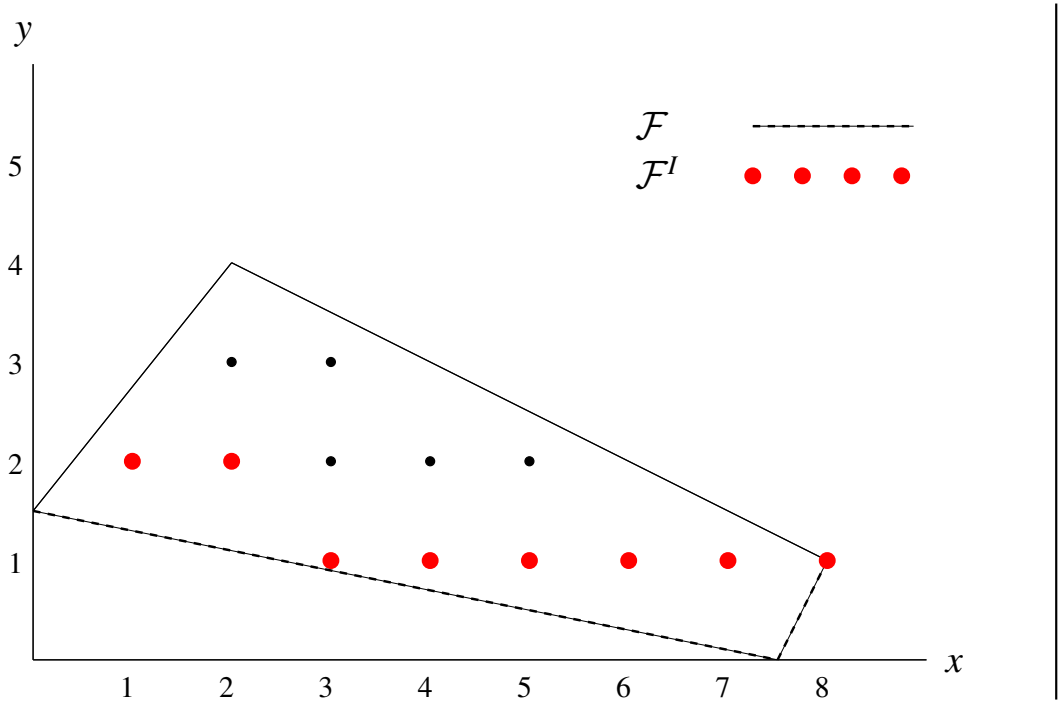


Basic Strategy:

- Here, ϕ is the value function of an *integer program*.
- In the general case, the function ϕ is piecewise linear but not convex.
- Here, we also iteratively generate an approximation by constructing dual functions.

Value Function Reformulation (General MIBLP)

More generally, we can reformulate (MIBLP) as



$$\begin{aligned} \max \quad & c^1 x + d^1 y \\ \text{subject to} \quad & A^1 x \leq b^1 \\ & G^2 y \geq b^2 - A^2 x \\ & d^2 y = z_{LL}(b^2 - A^2 x) \\ & x \in X, y \in Y, \end{aligned}$$

where z_{LL} is the value function of the lower-level problem.

- This is, in principle, a standard mathematical program.
- Note that relaxing integrality does not yield a valid bound on the optimum, as in the single-level case.
- Relaxing integrality effectively replaces z_{LL} with the value function of the LP relaxation.

Value Function Reformulation (Recourse and Zero Sum)

An important special case is when either $d^1 = -d^2$ (recourse) or $d^1 = d^2$ (zero sum), we can reformulate (MIBLP) as

$$\begin{aligned} \max \quad & c^1 x + w \\ \text{subject to} \quad & A^1 x \leq b^1 \\ & w \leq z_{LL}(b^2 - A^2 x) \\ & x \in X, y \in Y, \end{aligned}$$

where z_{LL} is again the value function of the lower-level problem.

- In these special cases, we can project out the lower-level variables, as in a traditional Benders algorithm.
- This leads to a generalized Benders algorithm obtained by constructing approximations of z_{LL} dynamically.
- Non-linear "cuts" arising from upper-bounding funtions are added in each step to improve the approximations.
- The upper-bounding functions are primal functions in the zero-sum case and dual functions in the recourse case.

Approximating the Value Function

- In general, it is difficult to construct the value function explicitly.
- We therefore propose to approximate the value function by either primal (upper) or dual (lower) bounding functions.

Dual bounds

Derived by considering the value function of *relaxations* of the original problem or by constructing *dual functions* $\Rightarrow$ Relax constraints.

Primal bounds

Derived by considering the value function of *restrictions* of the original problem $\Rightarrow$ Fix variables.

MILP Value Function

Now we consider the MILP value function $\phi : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$

$$\begin{aligned} \phi(b) = \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \end{aligned} \tag{MILP}$$

We define

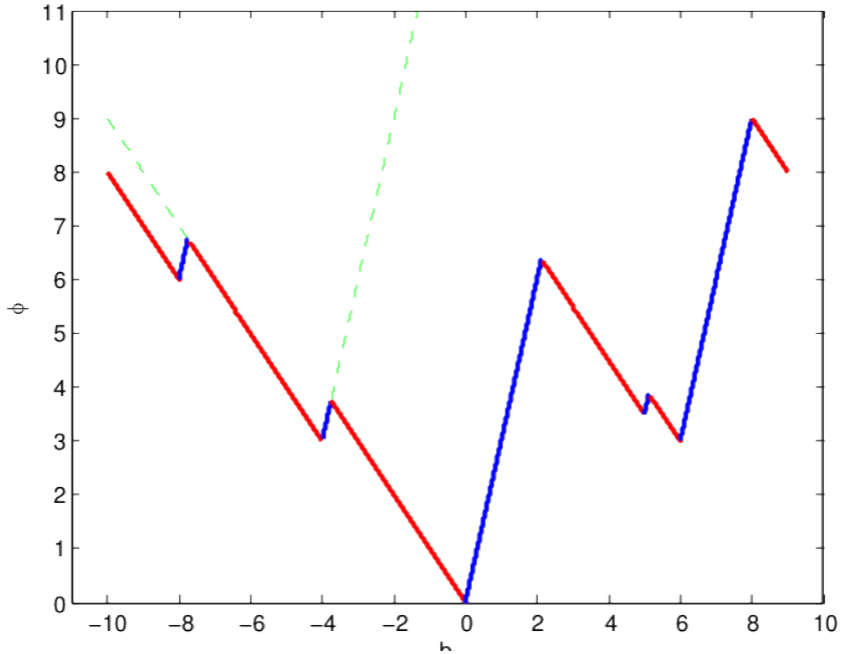
- $\mathcal{S}(b) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = b\}$.
- $B = \{b \in \mathbb{R}^m \mid S(b) \neq \emptyset\}$.

Example: MILP Value Function

The value function of a MILP is non-convex and discontinuous piecewise polyhedral.

Example

$$\phi(d) = \min\ 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6$$
$$\text{s.t. } 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = d$$
$$x_1, x_2, x_3 \in \mathbb{Z}_+,\ x_4, x_5, x_6 \in \mathbb{R}_+$$



Continuous and Integer Restriction of an MILP

Consider

$$\begin{aligned} \phi(b) = \min\ & c_I^\top x_I + c_C^\top x_C \\ \text{s.t. } & A_I x_I + A_C x_C = b, \\ & x \in \mathbb{Z}_+^r \times \mathbb{R}_+^{n-r} \end{aligned} \tag{MILP}$$

Define the *continuous restriction* of (MILP) as

$$\begin{aligned} \phi_C(b) = \min\ & c_C^\top x_C \\ \text{s.t. } & A_C x_C = b, \\ & x \in \mathbb{R}_+^{n-r} \end{aligned} \tag{CR}$$

and its *integer restriction* as

$$\begin{aligned} \phi_I(b) = \min\ & c_I^\top x_I \\ \text{s.t. } & A_I x_I = b \\ & x_I \in \mathbb{Z}_+^r \end{aligned} \tag{IR}$$

Discrete Representation of the Value Function

For $b \in \mathbb{R}^m$, we have that

$$\begin{aligned} \phi(b) = \min\ & c_I^\top x_I + \phi_C(b - A_I x_I) \\ \text{s.t. } & x_I \in \mathbb{Z}_+^r \end{aligned} \tag{5}$$

- From this we see that the value function is comprised of the minimum of a set of translations of ϕ_C.
- The set of translations, along with ϕ_C describe the value function exactly.
- For $\hat{x}_I \in \mathbb{Z}_+^r$, let

$$\phi_C(b, \hat{x}_I) = \phi_C(b - A_I \hat{x}_I) + c_I^\top x_I \ \ \forall b \in \mathbb{R}^m.$$

- Then we have that $\phi(b) = \min_{x_I \in \mathbb{Z}_+^r} \phi_C(b, \hat{x}_I)$.

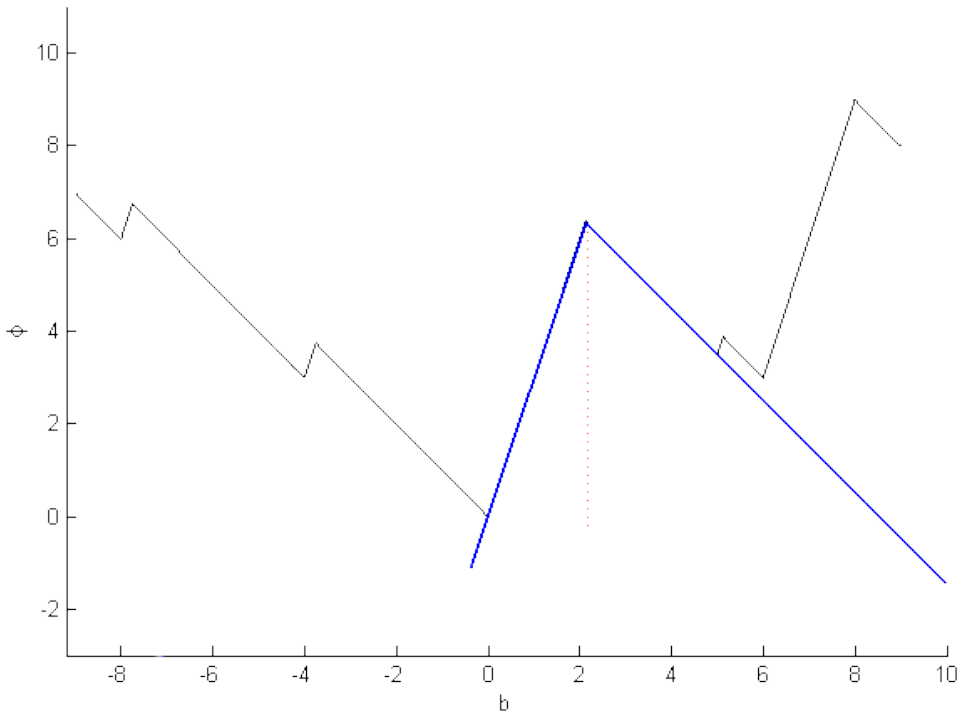
Dual Bounding Functions

A *dual function* $\varphi : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}$ is and function such that

$$\varphi(b) \leq \phi(b) \quad \forall b \in \Lambda$$

For a particular value of $\hat{b}$, the dual problem is

$$\phi_D = \max\{\varphi(\hat{b}) : \varphi(b) \leq \phi(b) \quad \forall b \in \mathbb{R}^m,\ \varphi : \mathbb{R}^m \to \mathbb{R} \cup \{\pm\infty\}\}$$



Dual Functions from Branch-and-Bound

Let T be set of the terminating nodes of the tree. Then in a terminating node $t \in T$ we solve:

$$\begin{aligned} &\min c^\top x \\ &\text{s.t. } Ax = b, \\ &\quad l^t \leq x \leq u^t, x \geq 0 \end{aligned} \tag{6}$$

The dual at node t:

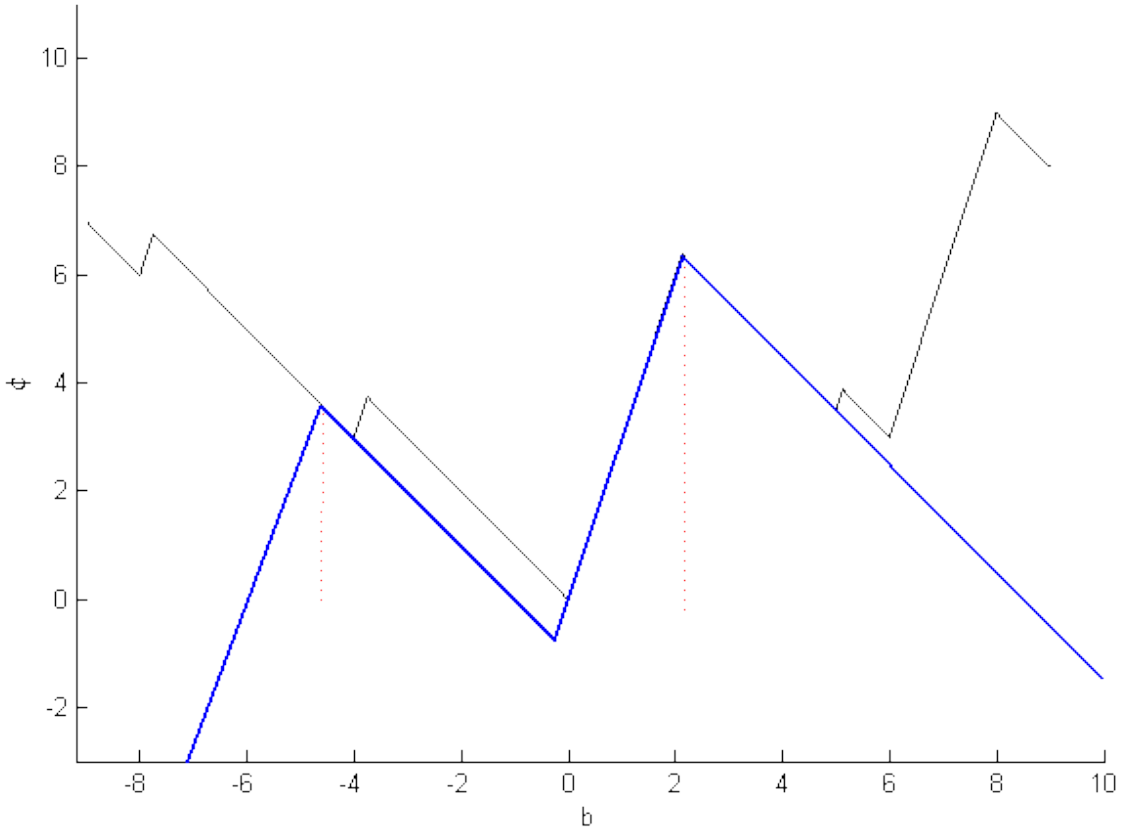
$$\begin{aligned} &\max \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \\ &\text{s.t. } \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ &\quad \underline{\pi} \geq 0, \bar{\pi} \leq 0 \end{aligned} \tag{7}$$

We obtain the following strong dual function:

$$\min_{t \in T} \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \tag{8}$$

This can be further strengthened by including internal nodes.

Dual Functions from Branch-and-Bound



Primal Bounding Functions

- Just like the dual (lower) bounding futions, we are interested in a function that is a valid primal (upper) bound for the value function.
- Primal bounds are inherently more difficult to obtain than dual bounding functions.
- One way is to consider the continuous relaxation of the primal instance.
  - Assume that $\{u \in \mathbb{R}^m \mid uA_C \leq c_C\}$ is not empty and bounded.
  - Let $z_C(d) = \max\{vd \mid v \in V\}$ where V is the set of extreme points of dual polytope.
  - Then, $z_C(d) \geq z(d) \ \ \forall d \in \mathbb{R}^m$.
- Furthermore, we can move z_C to a given right hand side b to obtain strong primal bounding functions.

Theorem 1 *Let x^* be an optimal solution to the primal problem with right-hand side b. Define the function f as*

$$f(d) = c_I x_I^* + z_C(d - A_I x_I^*) \quad \forall d \in \mathbb{R}^m.$$

Then, $f(d) \geq z(d) \ \ \forall d \in \mathbb{R}^m$ with $f(b) = z(b)$, and hence, is a strong primal bounding function at b.

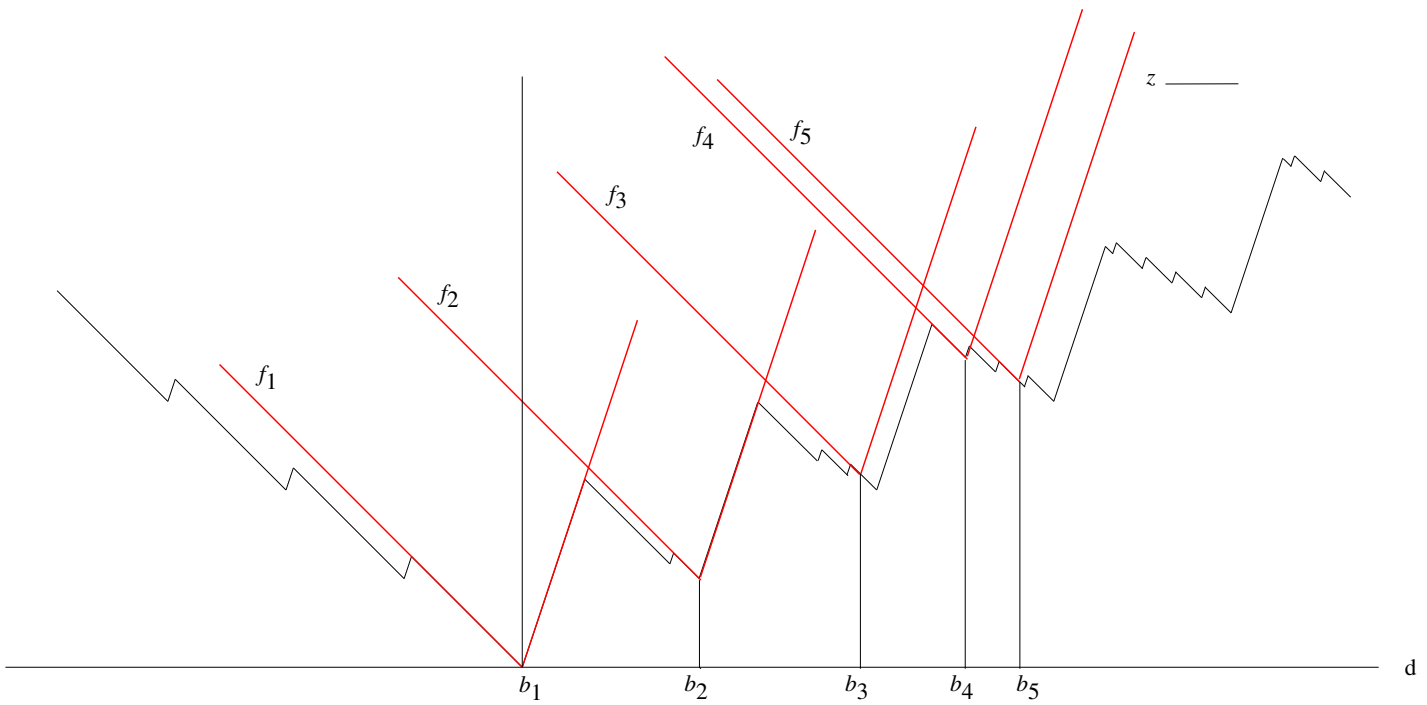


Figure: Primal bounding functions obtained at right-hand sides $b_i, i = 1, \ldots, 5$.

Warm Starting and Sensitivity Analysis

We can exploit the approximations to solve from a warm start or do sensitivity analysis.

Sensitivity Analysis

Sensitivity analysis can be performed in the obvious way.

- For right-hand side changes, evaluate the approximation for the new right-hand side (LP relaxation dual solutions remain feasible).
- For objective function changes, we re-evaluate all stored feasible solutions.
- It is easy to extend the basic theory to allow sensitivity on variable bounds, too (important later!)

Warm Starting

- Warm starting of branch and bound is also theoretically straightforward.
- We adjust the relaxations in the leaf nodes (after possible tree-trimming), re-evaluate stored solutions, and continue the computation.

Improving Variable Bounds

- Prior to any warm solve, we can improve variable bounds by computing generalized reduced costs.
- For each variable, we increase its lower bound temporarily and perform a sensitivity analysis.
- If such a bound change leads to a dual (lower) bound (on the modified instance) exceeding the known primal (upper) bound (on the instance), we can improve the variable's upper bound.
- The same can be done to improve lower bounds.

A Branch and Cut Algorithm

- Putting all of this together, we propose a branch-and-bound approach.

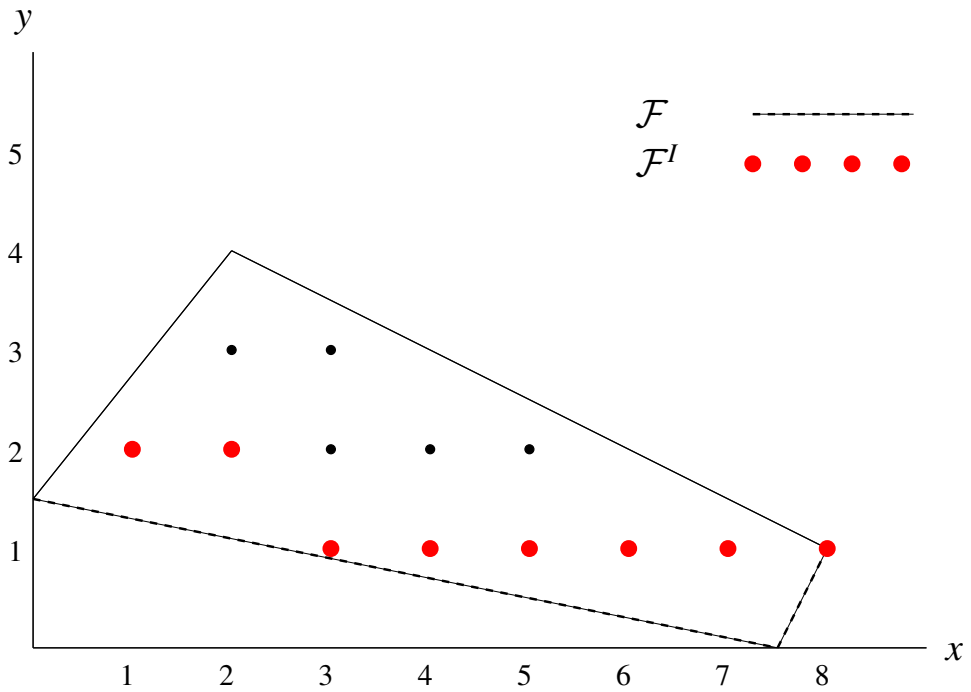
Components

- Bounding methods
- Branching methods
- Search strategies
- Preprocessing methods
- Primal heuristics

- In the remainder of the talk, we address development of these components, focusing on bounding.

Dual Bounds

Dual bounds for the MIBLP can be obtained by relaxing the value function constraint.



$$
\begin{aligned}
\max \quad & c^1 x + d^1 y \\
\text{subject to} \quad & A^1 x \leq b^1 \\
& G^2 y \geq b^2 - a^2 x \\
& d^2 y \leq \bar{z}_{LL}(b^2 - A^2 x) \\
& x \in X, y \in Y,
\end{aligned}
$$

where $\bar{z}_{LL}$ is an *upper (primal) approximation* of the value function of the lower-level problem.

- The upper approximation $\bar{z}_{LL}$ is piecewise linear in general and generated dynamically.
- We can then reformulate the above as a single-level program.

Bilevel Feasibility Check

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

$$\min_{y \in \mathcal{P}_L^I(\hat{x})} d^2 y \tag{9}$$

  with the fixed upper-level solution $\hat{x}$.
- Let y^* be the solution to (9).
  - $(\hat{x}, y^*)$ is bilevel feasible $\Rightarrow c^1\hat{x} + d^1 y^*$ is a valid primal 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 $(\hat{x}, \hat{y})$.
  - Improve our approximation of the value function so that $(\hat{x}, \hat{y})$ is no longer feasible.
  - Branch on a disjunction violated by $(\hat{x}, \hat{y})$.

Improving Efficiency of the Feasibility Check

- Unlike the MILP case, the main difficulty of the algorithm is in the feasibility check.
- In solving the lower-level problem repeatedly, we obtain information about the value function that can be exploited to improve efficiency.
- By maintaining primal and dual approximations, we can
  - Improve variable bounds.
  - Warm start the solves.
- Solves can also be parallelized.

Bilevel Feasibility Cut (Pure Integer Case)

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 the dual bounding problem 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

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

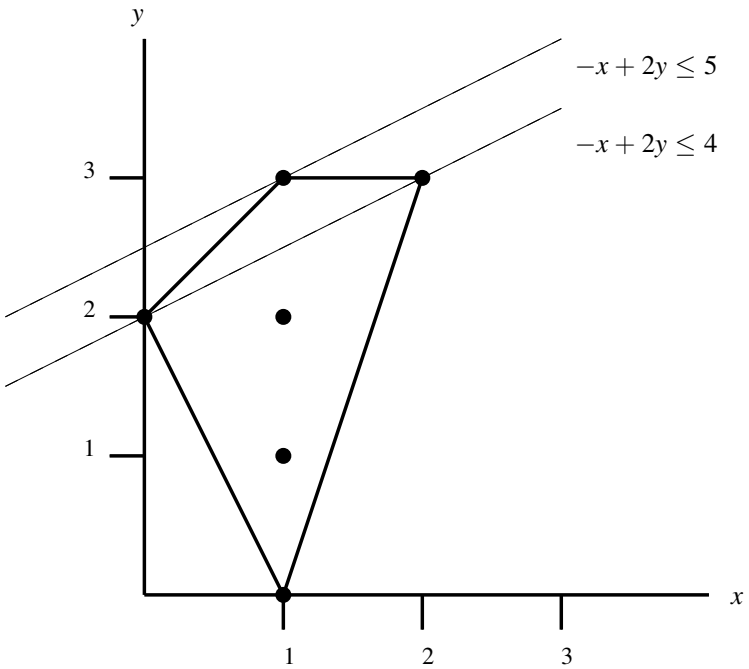
and $\sum_{i \in I} a_i' x + g_i' y \leq \sum_{i \in I} b_i$ is valid for Ω.

Bilevel Feasibility Cut (cont.)

A Valid Inequality

$\sum_{i \in I} a_i' x + g_i' y \leq \sum_{i \in I} b_i - 1$ for all $(x, y) \in \Omega^I \setminus \{(\hat{x}, \hat{y})\}$.

$$\max_x \min_y \{y \mid -x + y \leq 2, -2x - y \leq -2, 3x - y \leq 3, y \leq 3, x, y \in \mathbb{Z}_+\}.$$



This yields a finite algorithm in the pure integer case.

No Goods

- The previous cut is weak, as it only eliminates a single combination of upper and lower level.
- It doesn't incorporate bound information obtained during the feasibility check.
- Ideally, what we would really like to impose is a disjunction something like

$$d^\top y \leq d^\top \hat{y} \quad \text{OR} \quad x \neq \hat{x}$$

- In fact, if it were possible, we could simply impose $x \neq \hat{x}$!
- Once we have seen $\hat{x}$ once, we don't need to see it again.
- In the binary case, it is easy to enforce $x \neq \hat{x}$ by imposing the cut

$$\sum_{i:\hat{x}_i=0} x_i + \sum_{i:\hat{x}_i=1} (1 - x_i) \geq 1.$$

Special Case: $G^2 \in \mathbb{R}^{m_2 \times n_2}_+$

- When $G^2 \in \mathbb{R}^{m_2 \times n_2}_+$, we have the property that

$$x \geq \hat{x} \Rightarrow x \in \mathcal{P}_L(\hat{x})$$

- This means that

$$x \geq \hat{x} \quad \Rightarrow \quad z_{LL}(G^2 - A^2 x) \geq z_{LL}(G^2 - A^2 \hat{x})$$

- We then have the following valid disjunction (Xu, 2012).

$$\begin{array}{rcl} (A^2 x \leq b^2 - G^2 \hat{y} & \text{AND} & d^\top y \leq d^\top \hat{y}) \text{ OR} \\ (A^2 x)_1 & > & b^2_1 - (G^2 \hat{y})_1 \text{ OR} \\ (A^2 x)_2 & > & b^2_2 - (G^2 \hat{y})_2 \text{ OR} \\ \vdots & & \\ (A^2 x)_{m_2} & > & b^2_{m_2} - (G^2 \hat{y})_{m_2} \text{ OR} \end{array}$$

- This can be imposed in practice by branching with appropriate choice of ϵ's to enforce strict inequality.

Special Case: Upper Level Binary and $G^2 \in \mathbb{R}_+^{m_2 \times n_2}$

- If we additionally have binary upper level variables, we then the disjunction can be simplified to

$$\sum_{i:\hat{x}_i=0} x_i = 0 \qquad \text{AND} \qquad (d^2)^\top y \leq (d^2)^\top \hat{y}$$

$$\text{OR}$$

$$\sum_{i:\hat{x}_i=0} x_i \qquad \geq \qquad 1$$

Special Case: Interdiction

- In the case of interdiction problems, we have

$$x_i = 1 \Rightarrow y_i = 0$$

and we impose the previous disjunction with the cut

$$(d^2)^\top y \leq \sum_{1 \leq i \leq n_2} d_i^2 \hat{y}_i (1 - x_i)$$

- We call these *Benders cuts* because if we add one such cut for each lower-level solution (with no interdiction), we get the exact value function.
- Essentially, we are constructing a primal bounding function.

Implementation: MibS

The *Mixed Integer Bilevel Solver* (MibS) implements the branch and bound framework 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 manage the global branch and bound.
- The SYMPHONY framework for checking bilevel feasibility..
- 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 both SYMPHONY and MibS itself.
- The Open Solver Interface (OSI) for interfacing with SYMPHONY and CLP.

What Is Implemented: SYMPHONY

- We have implemented all of the advanced required of the MILP solver to support what is described.
- Warm-start of MILPs from an existing tree after problem modification.
  - Change of bounds.
  - Change in right-hand side.
  - Change in objective function.
  - Addition of constraints and variables.
- Inexpensive sensitivity analysis for same types of problem modifications.

What Is Implemented: MibS

MibS is an open source solver for bilevel integer programs built on top of the BLIS layer of CHiPPS. Features include

- Branch and Cut framework for general IBLPs
  - *Warm-starting* during feasibility check.
  - *Dual improvement* of variable bounds.
  - *All classes of 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.

Computational Results

- Computations were done on a set of 164 interdiction problems with various lower-level problems.
  - Knapsack problems.
  - Assignment problems.
  - Random MILP.
- Computations were done on the COR@L compute cluster consisting of 16-core AMD Opteron 2.0 GHz nodes with 32 Gb of memory.

Computational Results: All instances

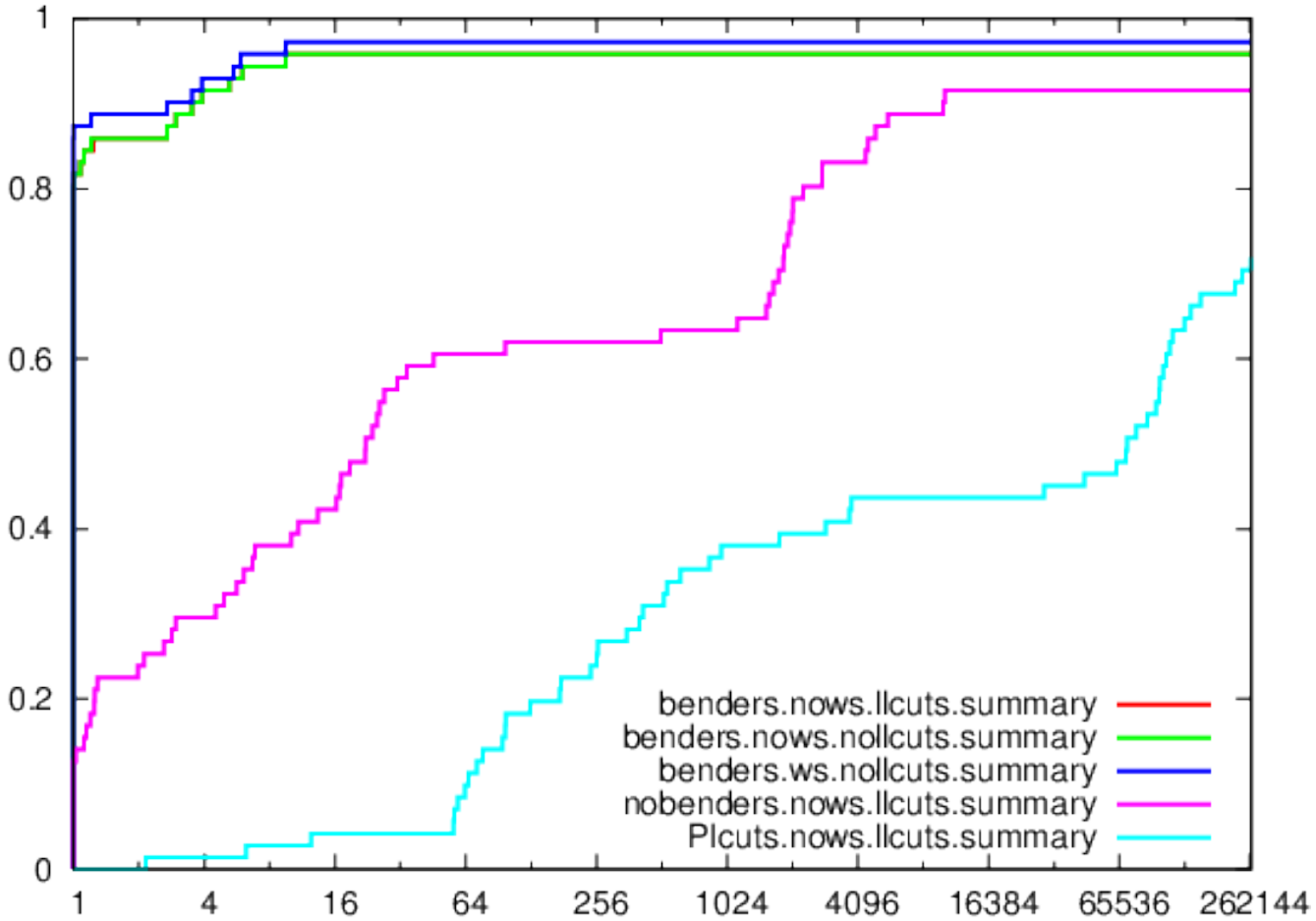


Figure: Performance profile of all instances that can be solved by at least one method with number of nodes as a performance measure

Computational Results: All instances

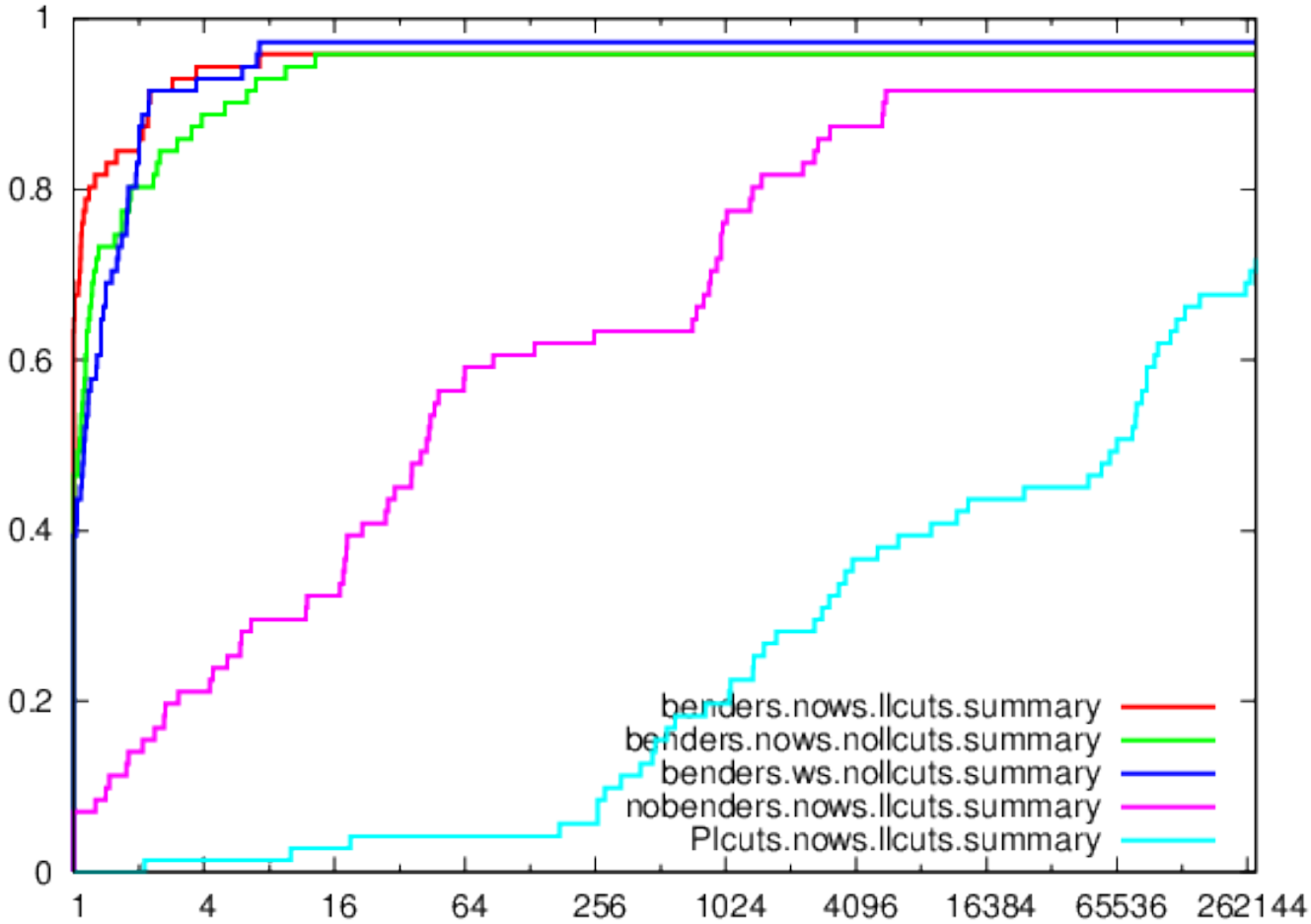


Figure: Performance profile of all instances that can be solved by at least one method with time as a performance measure

Computational Results: Binary Lower Level

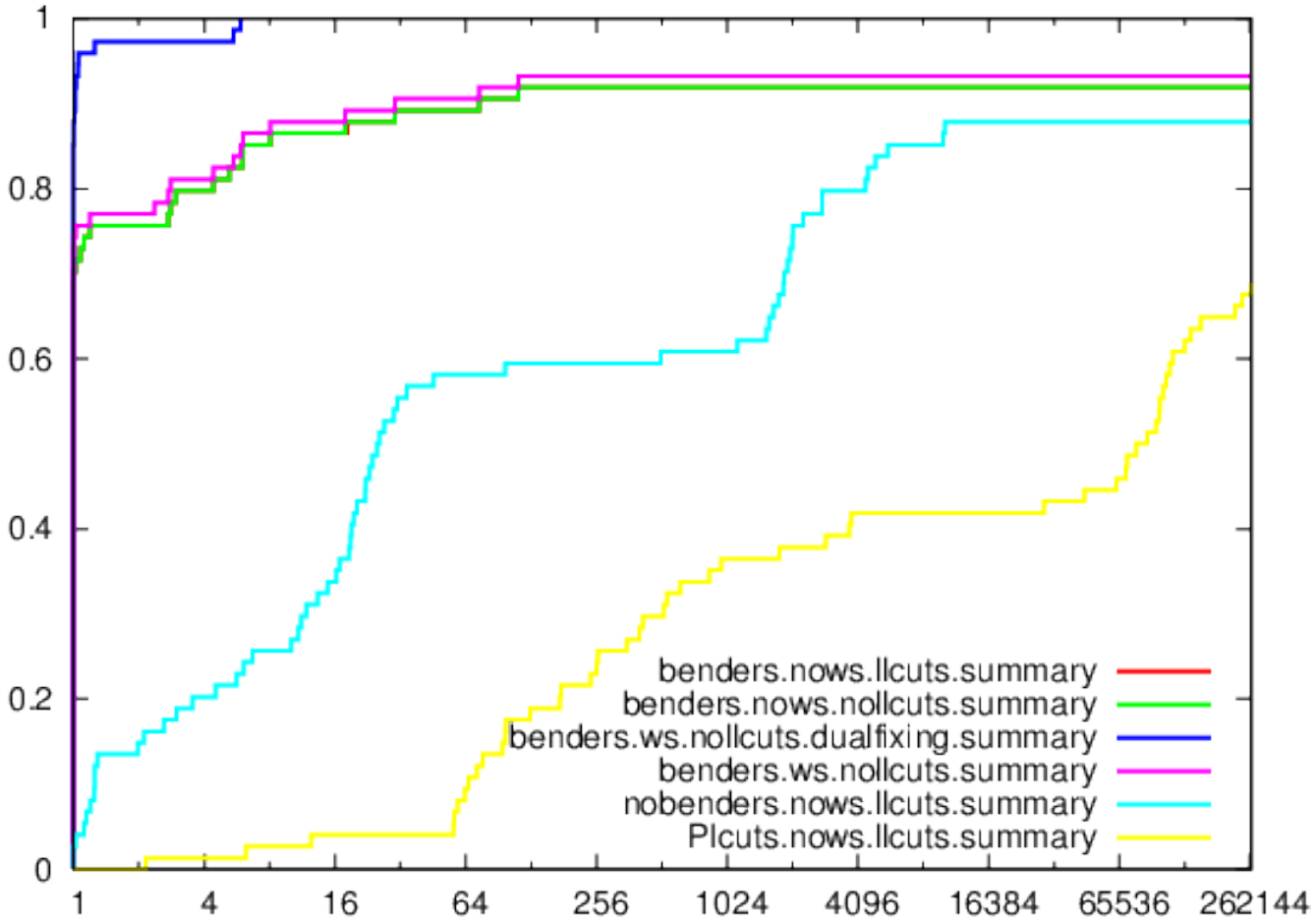


Figure: Performance profile of instances with binary lower-level variables that can be solved by at least one method with number of nodes as a performance measure

Computational Results: Binary Lower Level

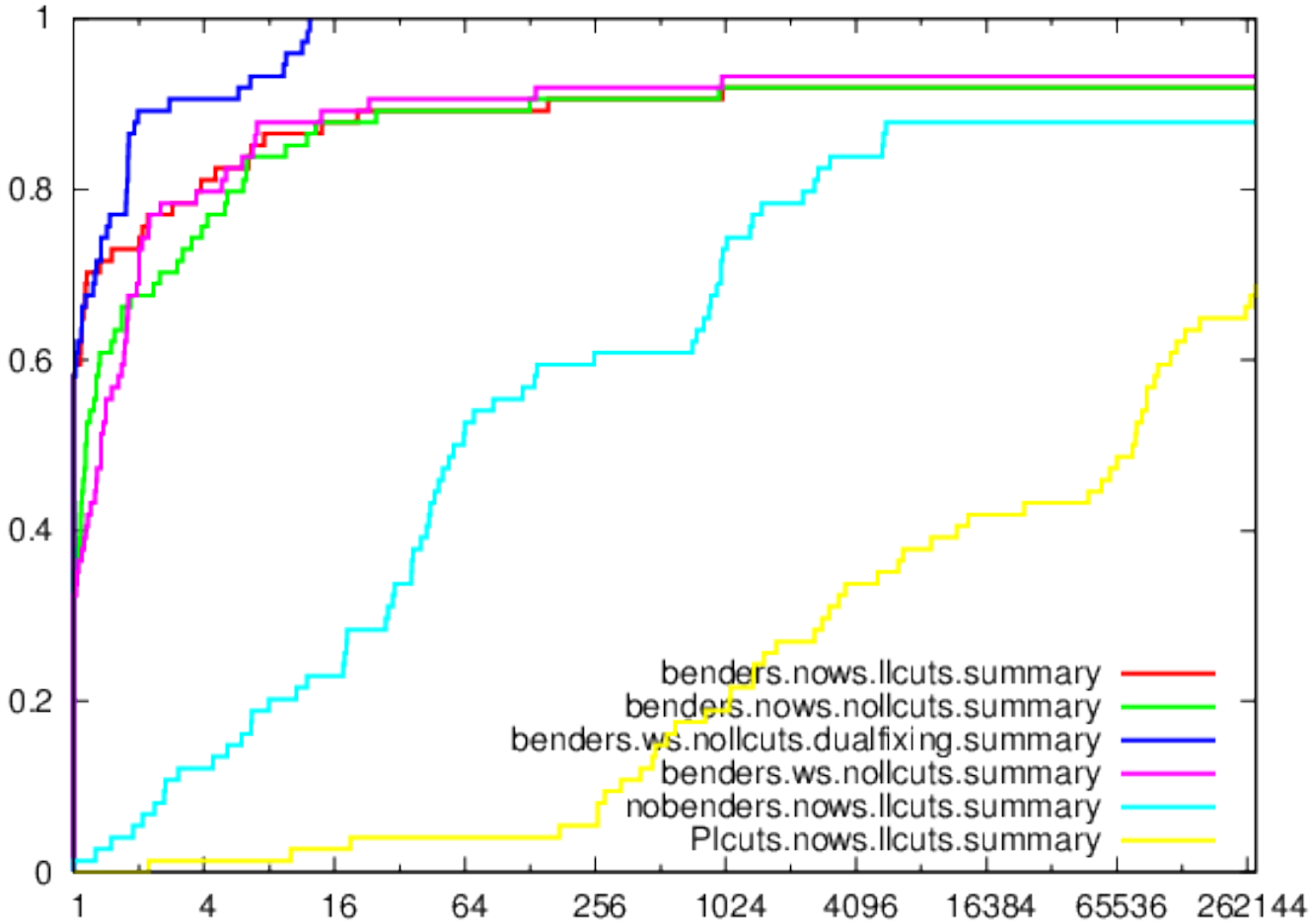


Figure: Performance profile of all instances with binary lower-level that can be solved by at least one method with time as a performance measure

Conclusions

- Upper bounds are found quickly for problems of this size—it is the lower bound that is difficult to improve.
- Not surprisingly, pure integer cuts are relatively ineffective.
- No good cuts result in an improvement of several orders of magnitude.
- Benders cuts result in several more orders of magnitude improvement.
- For now, warm-start seems to have almost no impact, but it's not clear exactly why this is the case.
- Parallelizing the feasibility checks also seems to have little impact.
- Dual fixing seems to work well for more difficult instances, though the impact is much smaller than adding cuts.
- All in all, we can now effectively solve small to medium size interdiction problems using generic methods out-of-the-box.
- It is unclear whether any of this can help in solving general MIBLPS.

Final Remarks

- This has been a high-level overview of a very wide swath of problems that present immense computational challenges.
- There is much work to be done and many obvious extensions of the general approach. .
- The code is available open source so feel free to play!

http://github.com/tkralphs/MibS

Questions?

References I

Xu, P. 2012. *Three Essays on Bilevel Optimization and Applications*. Ph.D. thesis, Iowa State University.