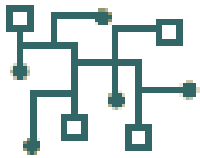
Multilevel and Multistage Integer Programming: Complexity and Algorithms

Ted Ralphs[1]

Joint work with Aykut Bulut[1], Andrea Lodi[2], Gerhard Woeginger[3]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
[2]DEI, University of Bologna
[3]Eindhoven University of Technology

INFORMS Annual Meeting, Minneapolis, MN, 7 October 2013



ISE Industrial and Systems Engineering

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Outline

Setting

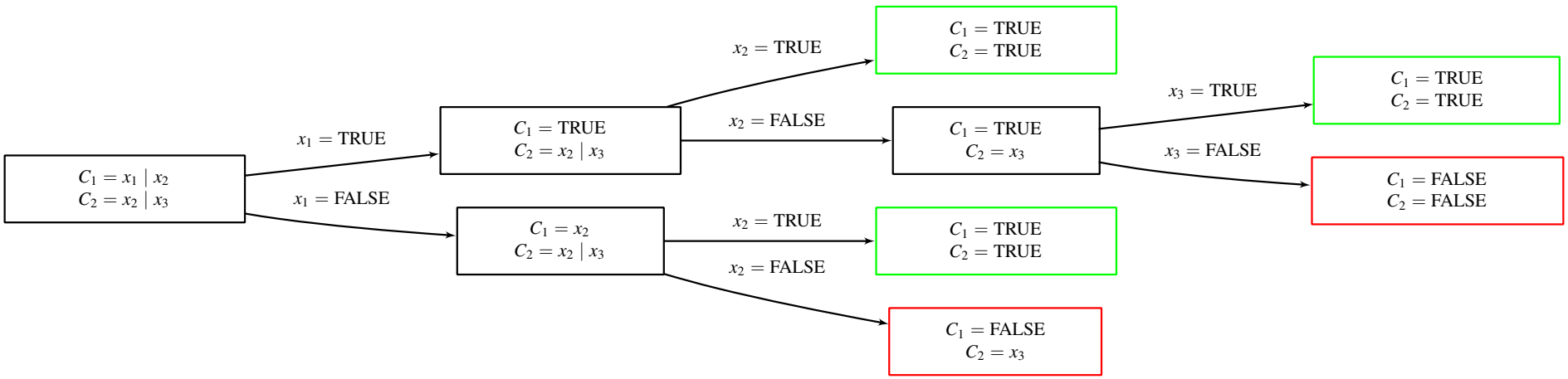
- Our goal is to analyze certain *finite extensive-form games*, which are sequential games involving n players.

Loose Definition

- The game is specified on a tree with each node corresponding to a move and the outgoing arcs specifying possible choices.
- The leaves of the tree have associated payoffs.
- Each player's goal is to maximize payoff.
- There may be *chance* players who play randomly according to a probability distribution and do not have payoffs (*stochastic games*).

- All players are rational and have perfect information.
- The problem faced by a player in determining the next move is a *multilevel/multistage* optimization problem.
- The move must be determined by taking into account the *responses of the other players*.

Example Game Tree



Multilevel and Multistage Games

- We use the term *multilevel* for competitive games in which there is no chance player.
- We use the term *multistage* for cooperative games in which all players receive the same payoff, but there are 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.

Multilevel and Multistage Optimization

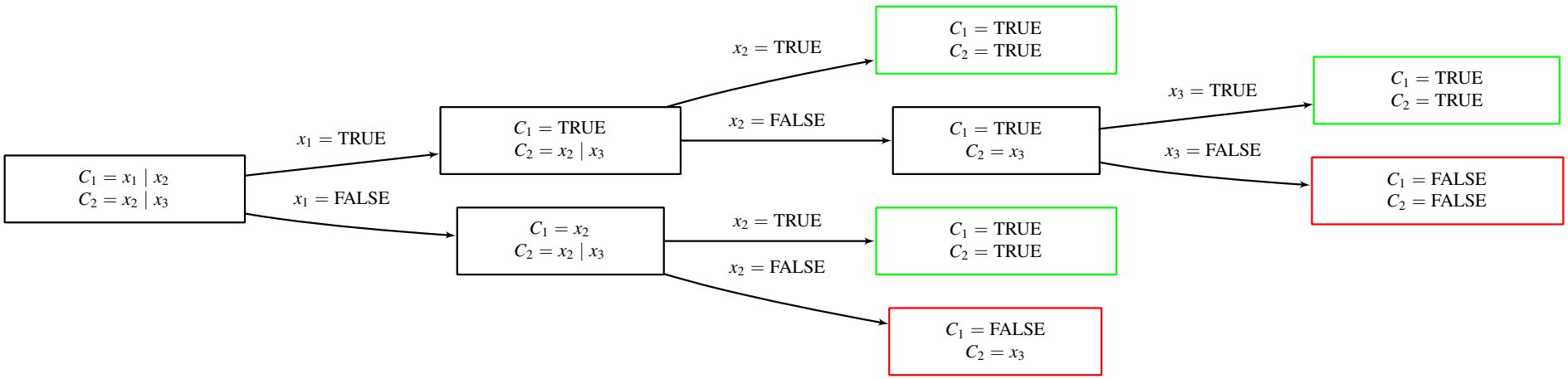
- A standard mathematical program models a (set of) decision(s) to be made *simultaneously* by a *single* decision-maker (i.e., with a *single* objective).
- Decision problems arising in sequential games and other real-world applications involve
  - multiple, independent decision-makers (DMs),
  - sequential/multi-stage decision processes, and/or
  - multiple, possibly conflicting objectives.
- Modeling frameworks
  - Multiobjective Programming $\Leftarrow$ multiple objectives, single DM
  - Mathematical Programming with Recourse $\Leftarrow$ multiple stages, single DM
  - Multilevel Programming $\Leftarrow$ multiple stages, multiple objectives, multiple DMs
- *Multilevel programming* generalizes standard mathematical programming by modeling hierarchical decision problems, such as finite extensive-form games.
- Such models arises in a remarkably wide array of applications.

A Canonical Example: Satisfiability Game

- A canonical extensive-form game that illustrates many of the basic principles is the *k-player satisfiability game*.
  - k players determine the value of a set of Boolean variables with each in control of a specific subset.
  - In round i, player i determines the values of her variables.
  - Each player tries to choose values that force a certain end result, given that subsequent players may be trying to achieve the opposite result.
- Examples
  - k = 1: SAT
  - k = 2: The first player tries to choose values such that any choice by the second player will result in satisfaction.
  - k = 3: The first player tries to choose values such that the second player cannot choose values that will leave the third player without the ability to find satisfying values.
- Note that the odd players and the even players are essentially "working together" and the same game can be described with only two players.

A Simple Example

- This diagram illustrates the search for solutions to the problem as a tree.
- The nodes in green represent settings of the truth values that satisfy all the given clauses; red represents non-satisfying truth values.
  - With one player, the solution is any path to one of the green nodes.
  - With two players, the solution is a subtree in which there are no red nodes.
- The latter requires knowledge of *all* leaf nodes (important!).



More Formally

- More formally, we are given a Boolean formula with variables partitioned into k sets $X_1, \ldots, X_k$.
- For k odd, the SAT game can be formulated as

$$\exists X_1 \forall X_2 \exists X_3 \ldots ? X_k \tag{1}$$

- for even k, we have

$$\forall X_1 \exists X_2 \forall X_3 \ldots ? X_k \tag{2}$$

- A more general form of this problem, known as the *quantified Boolean formula problem* (QBF) allows an arbitrary sequence of quantifiers.

From SAT Game to Multilevel Optimization

- For $k = 1$, SAT can be formulated as the (feasibility) integer program

$$\exists x \in \{0,1\}^n : \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \; \forall j \in J. \qquad \text{(SAT)}$$

- (SAT) can be formulated as the optimization problem

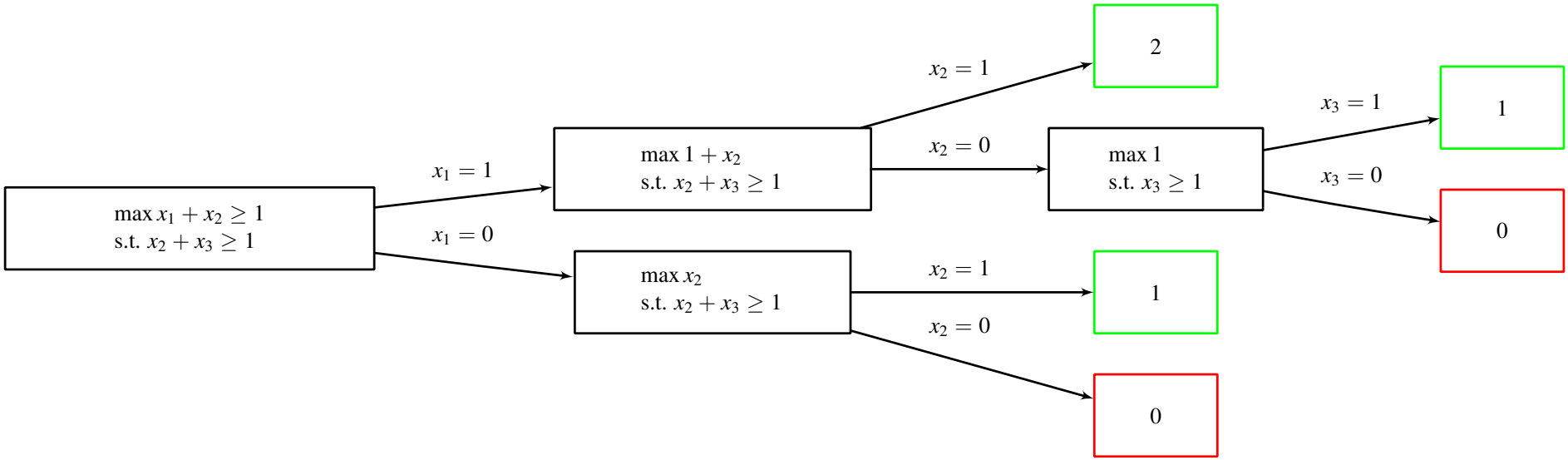
$$\max_{x \in \{0,1\}^n} \sum_{i \in C_0^0} x_i + \sum_{i \in C_0^1} (1 - x_i)$$
$$\text{s.t.} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \; \forall j \in J \setminus \{0\}$$

- For $k = 2$, we then have

$$\max_{x_{I_1} \in \{0,1\}^{I_1}} \min_{x_{I_2} \in \{0,1\}^{I_2}} \sum_{i \in C_0^0} x_i + \sum_{i \in C_0^1} (1 - x_i)$$
$$\text{s.t.} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \; \forall j \in J \setminus \{0\}$$

Branch and Bound for Optimization Version of SAT

- Consider the earlier example of the SAT game, now as an optimization problem.
- In the one player version, the goal is simply to maximize payoff.
- The two player game is zero-sum with the first player attempting to maximize while the second player attempts to minimize.
- The complexity of the two-player game comes from the requirement to account for the payoff at *all* leaf nodes.



Outline

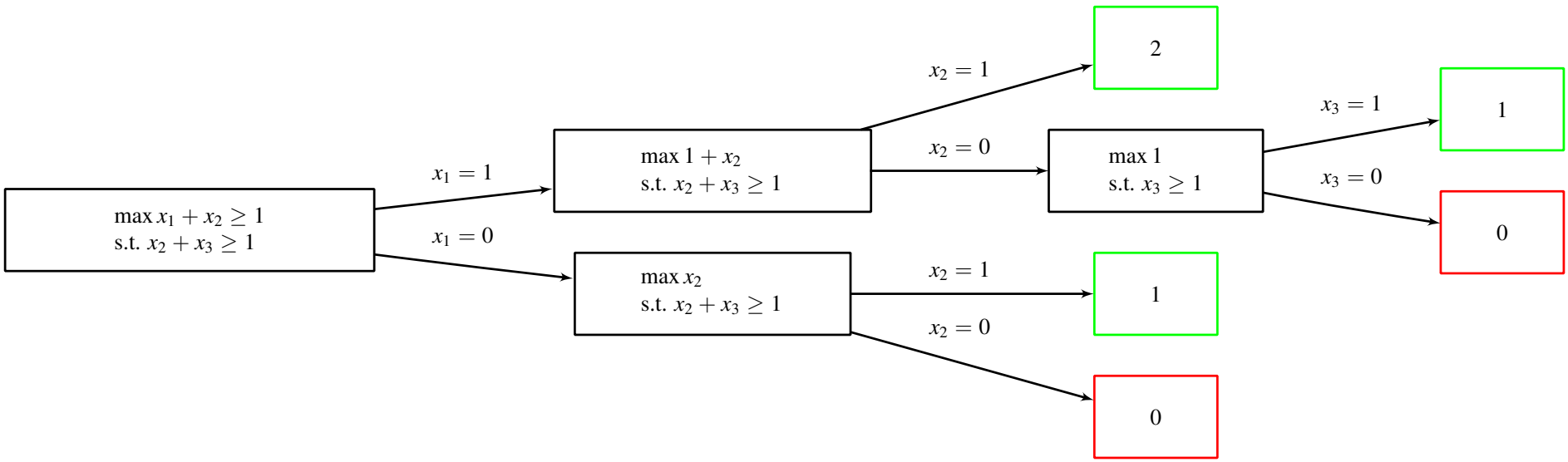
How Difficult is the SAT Game?

- Fundamentally, we would like to know how difficult it is to solve player one's decision problem.
- It is well-known that the (single player) satisfiability problem is is in the complexity class *NP*-complete.
- It is perhaps to be expected that the k-player satisfiability game is in a different class.
  - The k^{th} player to move is faced with a satisfiability problem.
  - The $(k-1)^{th}$ player is faced with a 2-player subgame in which she must take into account the move of the k^{th} player.
  - And so on . . .
- Each player's decision problem appears to be exponentially more difficult than the succeeding player's problem.
- This complexity is captured formally in the hierarchy of complexity classes known as the *polynomial time hierarchy*.

Turing Machines

- The formal complexity framework traditionally employed in discrete optimization applies to *decision problems* (Garey and Johnson, 1979).
- The formal model of computation is a *deterministic Turing machine* (DTM).
- The possible execution paths of a DTM can be thought of as forming a tree.
- For problems that are efficiently solvable, we know how to construct an execution path that is guaranteed to end in an accepting state.
- For more difficult problems, some enumeration is needed.
- A *non-deterministic Turing machine* (NDTM) can be thought of as a Turing machine with an infinite number of parallel processors.
- An NDTM follows all possible execution paths simultaneously.
- It returns YES if an accepting state is reached on *any* path.
- The running time of an NDTM is the *minimum* running time (length) of any execution paths that end in an accepting state.
- The "running time" is the minimum time required to verify that some path (given as input) leads to an accepting state.

Back to SAT


max $x_1 + x_2 \geq 1$
s.t. $x_2 + x_3 \geq 1$
$x_1 = 1$
$x_1 = 0$
max $1 + x_2$
s.t. $x_2 + x_3 \geq 1$
max x_2
s.t. $x_2 + x_3 \geq 1$
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
max 1
s.t. $x_3 \geq 1$
$x_3 = 1$
$x_3 = 0$
2
1
0
1
0

The Polynomial Hierarchy

The polynomial hierarchy is a scheme for classifying multi-level and multi-stage decision problems. We have

$$\Delta_0^p := \Sigma_0^p := \Pi_0^p := P, \tag{3}$$

where P is the set of decision problems that can be solved in polynomial time. Higher levels are defined recursively as:

$$\begin{aligned} \Delta_{k+1}^p &:= P^{\Sigma_k^p}, \\ \Sigma_{k+1}^p &:= NP^{\Sigma_k^p}, \text{and} \\ \Pi_{k+1}^p &:= coNP^{\Sigma_k^p}. \end{aligned}$$

PH is the union of all levels of the hierarchy.

Complexity of Multilevel Games and Optimization

- The satisfiability games with k players is complete for Σ_k^p.
- For the corresponding k-level optimization problem, the optimal value is one if and only if the first player has a winning strategy.
- This means the satisfiability game can be reduced to the (decision) problem of whether the optimal value ≥ 1?
- Thus, the (the decision version of) k-level mixed integer programming is also complete for Σ_k^p.
- By swapping the "min" and the "max," we can get a similar decision problem that is complete for Π_k^p.

$$\min_{x_{N_1} \in \{0,1\}^{N_1}} \max_{x_{N_2} \in \{0,1\}^{N_2}} \sum_{i \in C_0^0} x_i + \sum_{i \in C_0^1} (1 - x_i)$$
$$\text{s.t.} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq 1 \; \forall j \in J \setminus \{0\}$$

- The question remains whether the optimal value is ≥ 1, but now we are asking it with respect to a minimization problem.

Outline

Multilevel (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 $Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}$. This basic model can be extended to multiple levels.

Multistage (iInteger) Linear Programming

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

$$\min\{-c^1 x + Q(x) \mid x \in \mathcal{P}_U \cap X\}, \tag{4}$$

where

$$Q(x) = \min\{d^1 y \mid y \in \mathcal{P}_L(x) \cap Y\}. \tag{5}$$

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

Two-Stage Stochastic Programs with Recourse

- For most of the remainder of the talk, we consider the two-stage stochastic mixed integer program

$$\min\{c^1 x + \mathbb{E}_\xi Q_\xi(x) \mid x \in \mathcal{P}_U \cap X\}, \tag{6}$$

where

$$Q_\xi(x) = \min\{d^2 y \mid y \in Y, G^2 y \geq \omega(\xi) - A^2 x\}, \tag{7}$$

ξ is a random variable from a probability space $(\Xi, \mathscr{F}, \mathcal{P})$, and for each $\xi \in \Xi$, $\omega(\xi) \in \mathbb{R}^{m_2}$.

- If the distribution of ξ is discrete and has finite support, then (6) is a bilevel program.

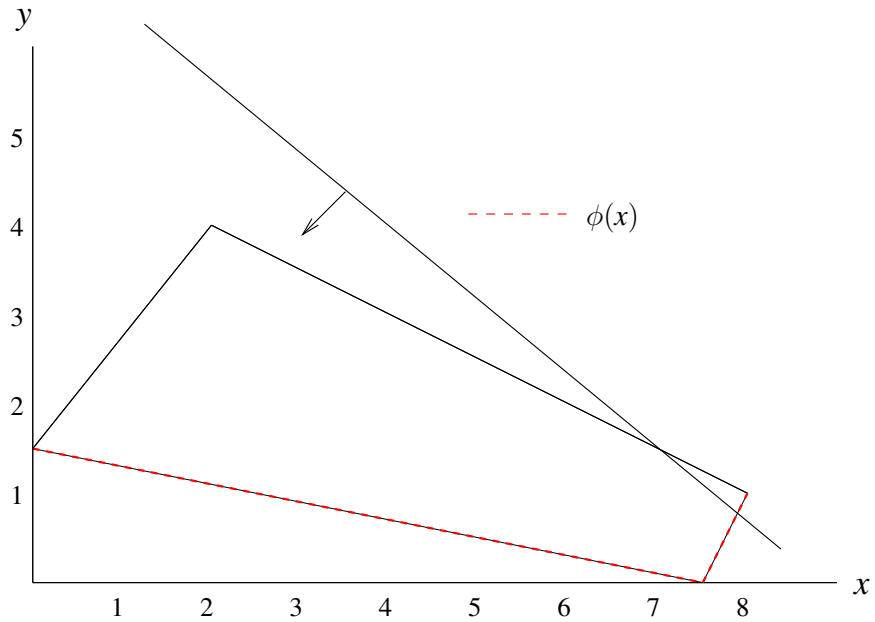
Benders' Principle (Linear Programming)

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

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

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 generate a lower approximation by sampling the domain.

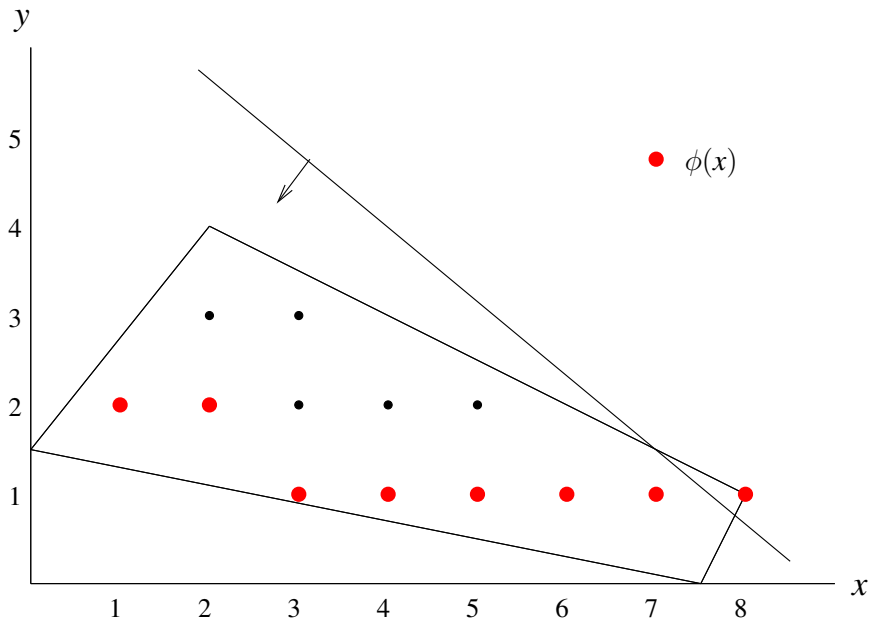
Benders' Principle (Integer Programming)

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

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

where

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



Basic Strategy:

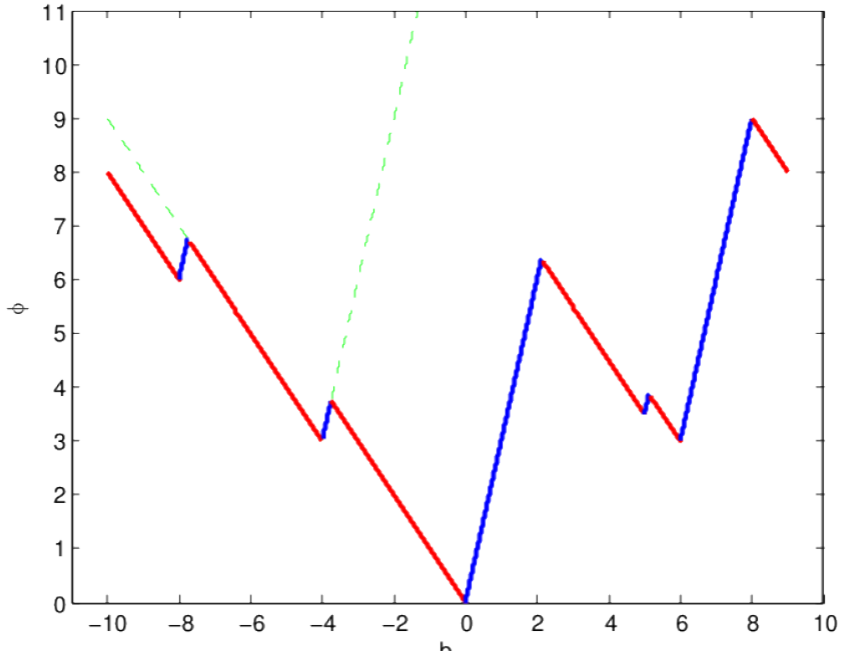
- Here, ϕ is the value function of an *integer program*.
- In the general case, the function ϕ is piecewise linear but not convex.
- Here, we also iteratively generate a lower approximation by evaluating ϕ.

Example: MILP Value Function

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

Example

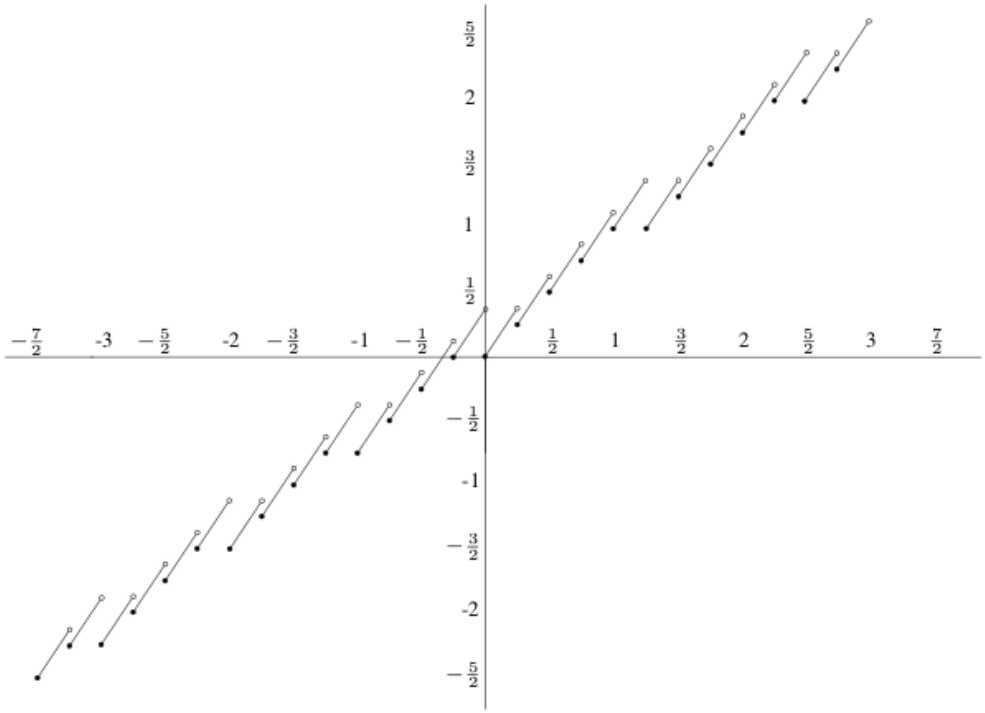
$$\phi(d) = \min\ 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6$$
$$\text{s.t. } 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = d$$
$$x_1, x_2, x_3 \in \mathbb{Z}_+,\ x_4, x_5, x_6 \in \mathbb{R}_+$$



Example: MILP Value Function

Example

$$\phi(b) = \min\ x_1 - \frac{3}{4}x_2 + \frac{3}{4}x_3$$
$$\text{s.t. } \frac{5}{4}x_1 - x_2 + \frac{1}{2}x_3 = b \qquad \text{(Ex2.MILP)}$$
$$x_1, x_2 \in \mathbb{Z}_+,\ x_3 \in \mathbb{R}_+$$



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 upper or lower bounding functions

Lower bounds

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

Upper bounds

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

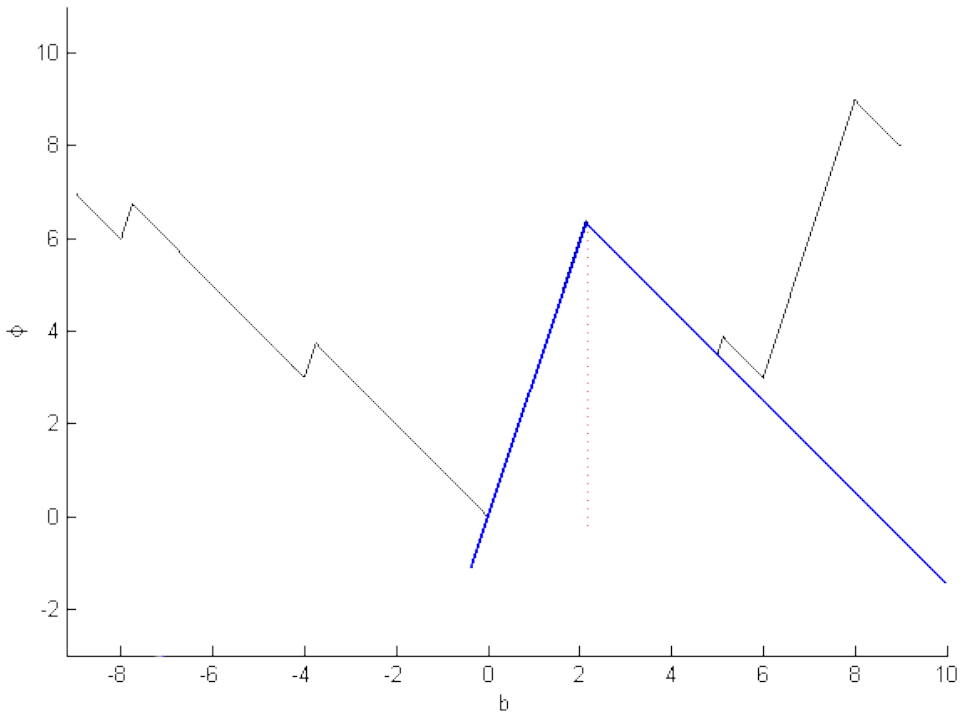
Bounding the Value Function From Below

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

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



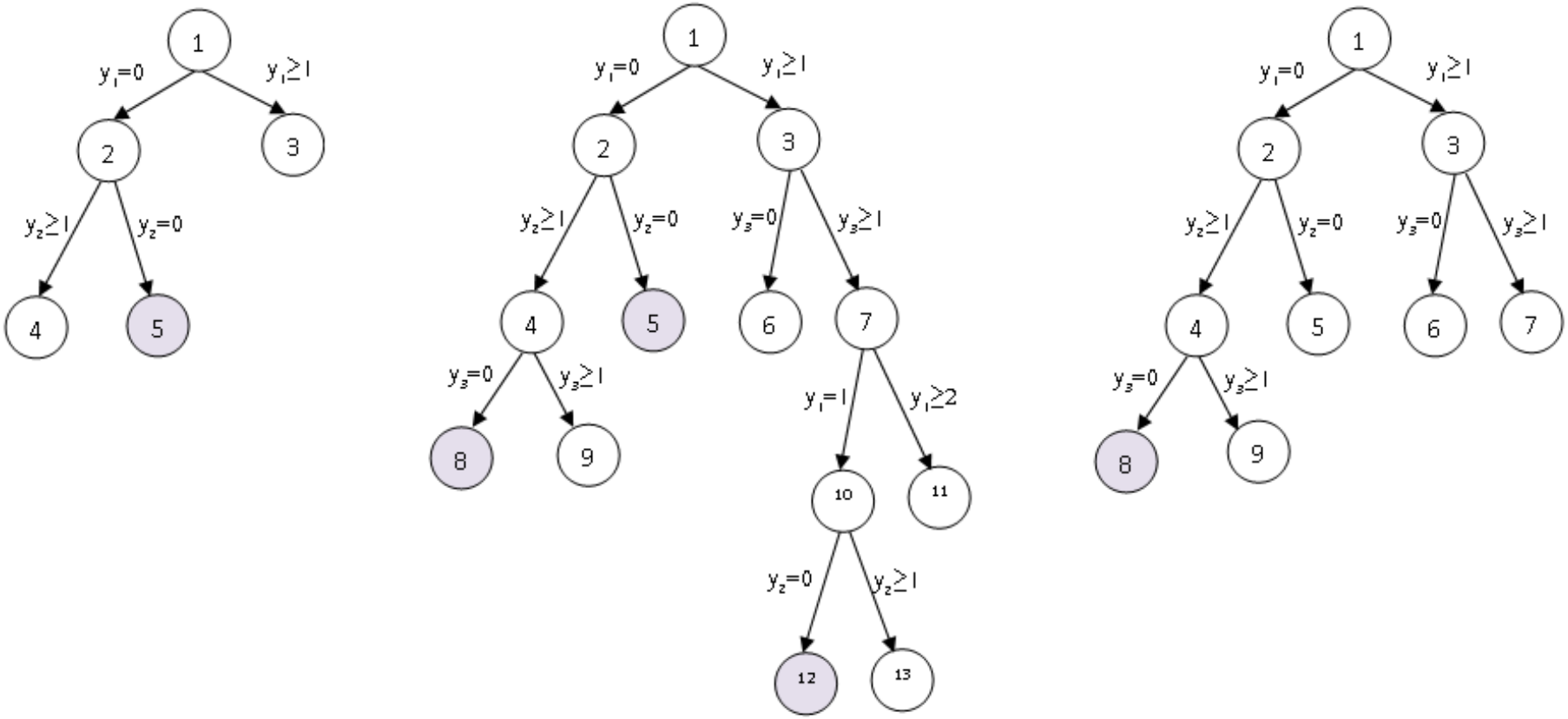
Related Algorithms

The algorithmic framework we utilize builds on a number of previous works.

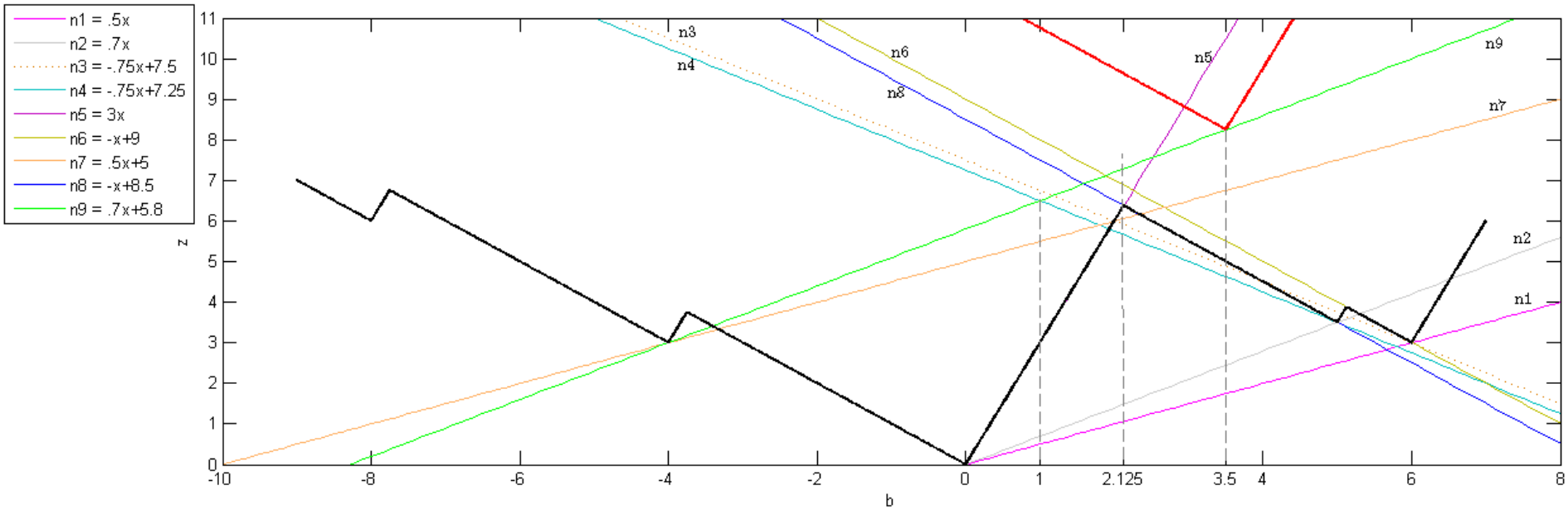
- Modification to the L-shaped framework (Laporte and Louveaux, 1993; Carøe and Tind, 1998; Sen and Higle, 2005)
  - Linear cuts in first stage for binary first stage
  - Optimality cuts from B&B and cutting plane, applied to pure integer second stage
  - Disjunctive programming approaches and cuts in the second stage
- Value function approaches: Pure integer case (Ahmed et al., 2004; Kong et al., 2006)
- Scenario decomposition (Carøe and Schultz, 1998)
- Enumeration/Gröbner basis reduction (Schultz et al., 1998)

MILP Duals from Branch-and-Bound

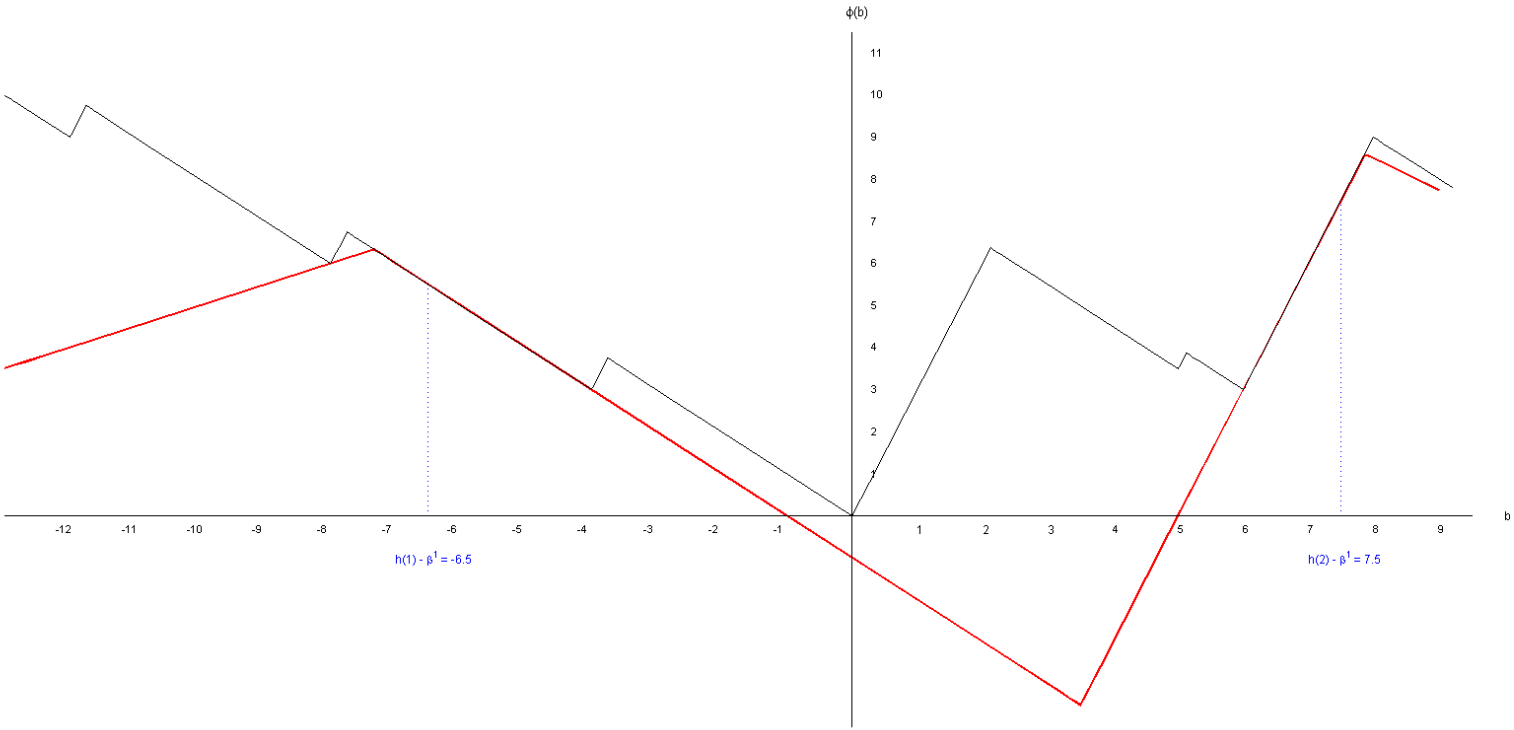
Figure : Dual Functions from B&B for right hand sides 1, 2.125, 3.5



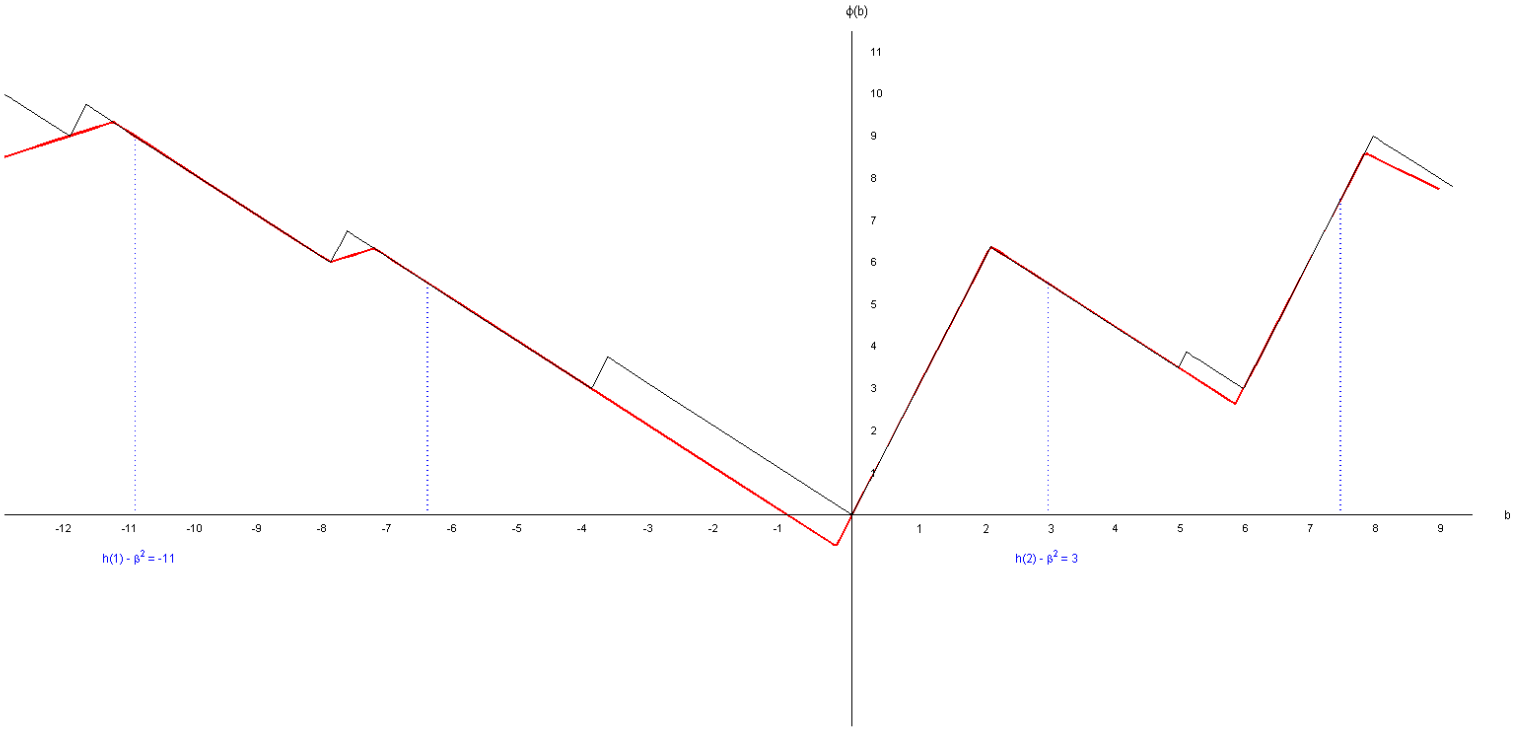
MILP Duals from Branch-and-Bound



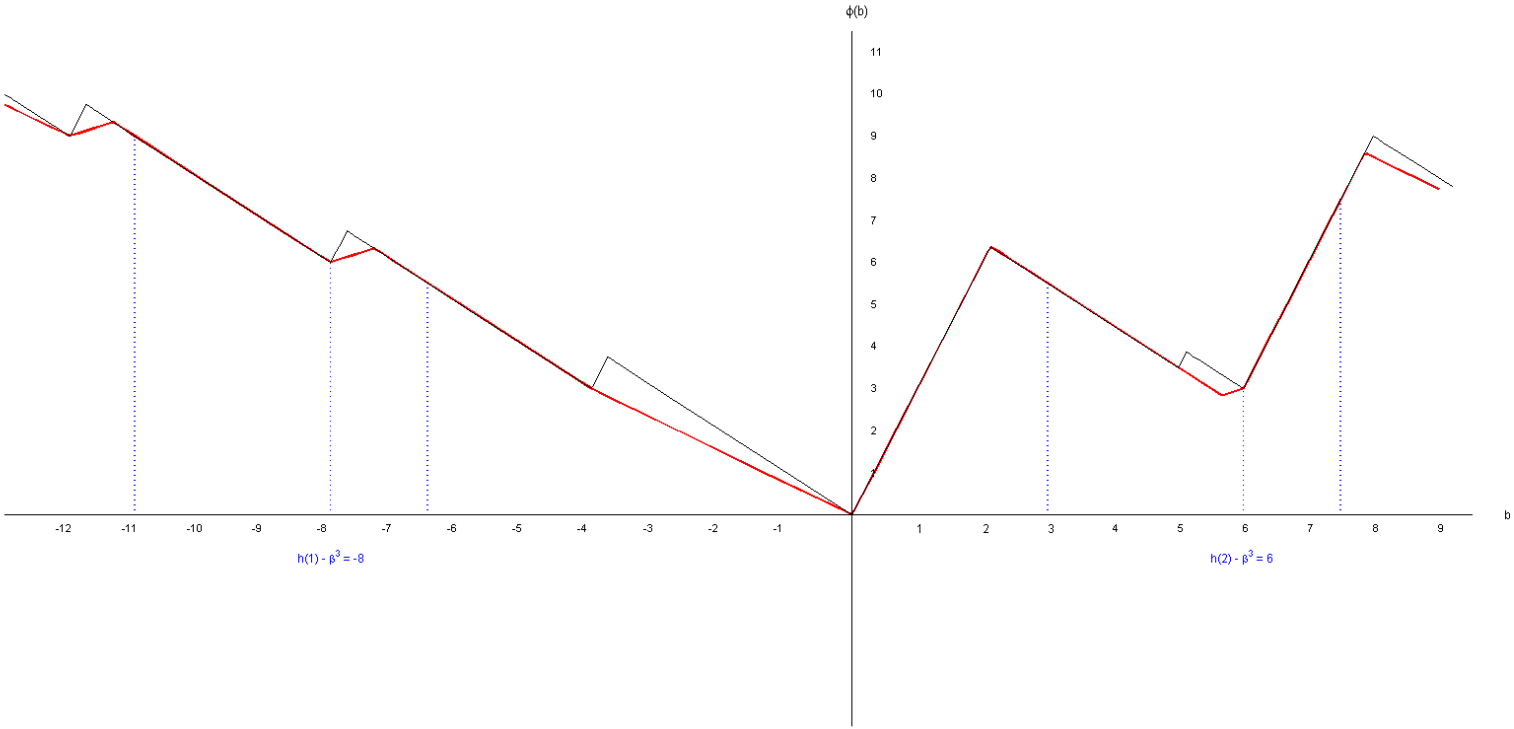
Example


ϕ(b)
11
10
9
8
7
6
5
4
3
2
1
-12
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
b
h(1) - β¹ = -6.5
h(2) - β¹ = 7.5

Example


ϕ(b)
11
10
9
8
7
6
5
4
3
2
1
-12
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
b
h(1) - β² = -11
h(2) - β² = 3

Example


ϕ(b)
11
10
9
8
7
6
5
4
3
2
1
-12
-11
-10
-9
-8
-7
-6
-5
-4
-3
-2
-1
1
2
3
4
5
6
7
8
9
b
h(1) - β3 = -8
h(2) - β3 = 6

Algorithms for General Bilevel Programs

- The general case is much more difficult because we need the *solution* to the lower-level problem, not just the *value*.
- Algorithms must involve some kind of relaxation of the problem.
- Relaxations are inherently weak.
- Some progress has been made, but incorporating knowledge of the value function into the relaxation has proven exceptionally challenging.

Outline

Conclusions

- This has been a high level overview of the complexity of this broad class of problems.
- The complexity of these problems is closely related to the complexity of more traditional types of problems.
- Understanding these issues can provide important insight and lead to practical algorithms.
- This a wide open area and there is much more to be done.
- See talks by Anahita Hassanzadeh and Aykut for more details.

References I

Ahmed, S., M. Tawarmalani, and N. Sahinidis 2004. A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming* **100**(2), 355–377.

Carøe, C. and R. Schultz 1998. Dual decomposition in stochastic integer programming. *Operations Research Letters* **24**(1), 37–46.

Carøe, C. and J. Tind 1998. L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming* **83**(1), 451–464.

Garey, M. and D. Johnson 1979. *Computers and Intractability: A Guide to the Thoery of NP-Completeness*. W.H. Freeman and Company.

Kong, N., A. Schaefer, and B. Hunsaker 2006. Two-stage integer programs with stochastic right-hand sides: a superadditive dual approach. *Mathematical Programming* **108**(2), 275–296.

Laporte, G. and F. Louveaux 1993. The integer l-shaped method for stochastic integer programs with complete recourse. *Operations research letters* **13**(3), 133–142.

References II

Schultz, R., L. Stougie, and M. Van Der Vlerk 1998. Solving stochastic programs with integer recourse by enumeration: A framework using Gröbner basis. *Mathematical Programming* **83**(1), 229–252.

Sen, S. and J. Higle 2005. The C 3 theorem and a D 2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming* **104**(1), 1–20. ISSN 0025-5610.