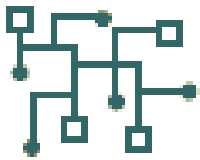
Multilevel Integer Programming, The Polynomial Time Hierarchy, and Parallel Computing

Ted Ralphs[1]

Joint work with Aykut Bulut[1], Scott DeNegre[3], Menal Güzelsoy[2], Anahita Hassanzadeh[1]

[1]COR@L Lab, Department of Industrial and Systems Engineering, Lehigh University
[2]SAS Institute, Advanced Analytics, Operations Research R & D
[3]The Chartis Group

Pennsylvania State University, 27 March, 2013



ISE

Industrial and Systems Engineering

COR@L

COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Outline

A Bit of Game Theory

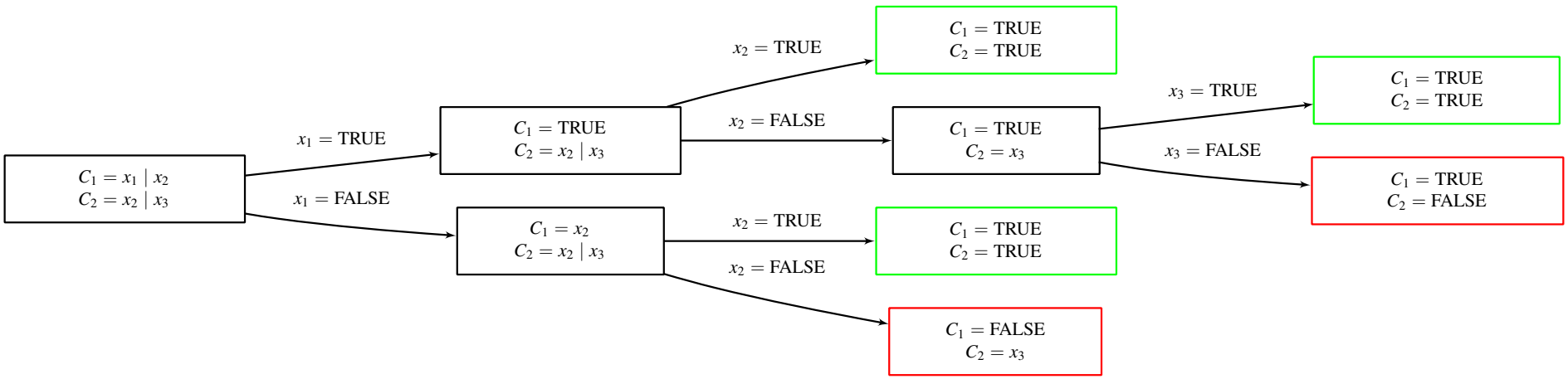
- Our goal is to analyze certain *finite extensive-form games*, which are sequential games involving n players.

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.

Brief Overview of Practical Applications

- Hierarchical decision systems
  - Government agencies
  - Large corporations with multiple subsidiaries
  - Markets with a single "market-maker."
  - Decision problems with recourse
- Parties in direct conflict
  - Zero sum games
  - Interdiction problems
- Modeling "robustness": Chance player is external phenomena that cannot be controlled.
  - Weather
  - External market conditions
- Controlling optimized systems: One of the players is a system that is optimized by its nature.
  - Electrical networks
  - Biological systems

Brief Overview of Technical Applications

- Multilevel structure is inherent in many decision problems that occur within branch and cut and other iterative and recursive algorithms.
- We would like to make the "most effective" algorithmic choice at each step, taking into account the effect of the choice on future iterations.
- The choice problem is an optimization problem that itself may have a multilevel structure similar to that of a multi-round game.
- Multilevel choice problems arise when the effectiveness or validity of the choice is evaluated by solving another optimization problem.
- The number of levels one chooses to "look ahead" determines the complexity of the exact version of the problem.
- Examples
  - Constructing a valid inequality for a given class that maximizes degree of violation.
  - Choosing a branching disjunction that achieves maximal bound improvement.

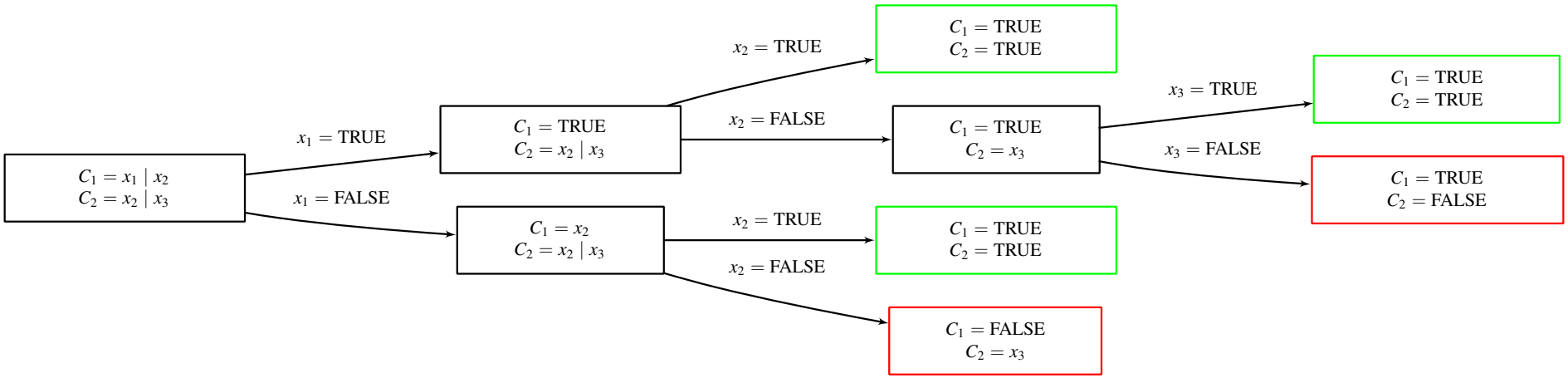
Outline

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 SAT 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 re-formulated as the optimization problem

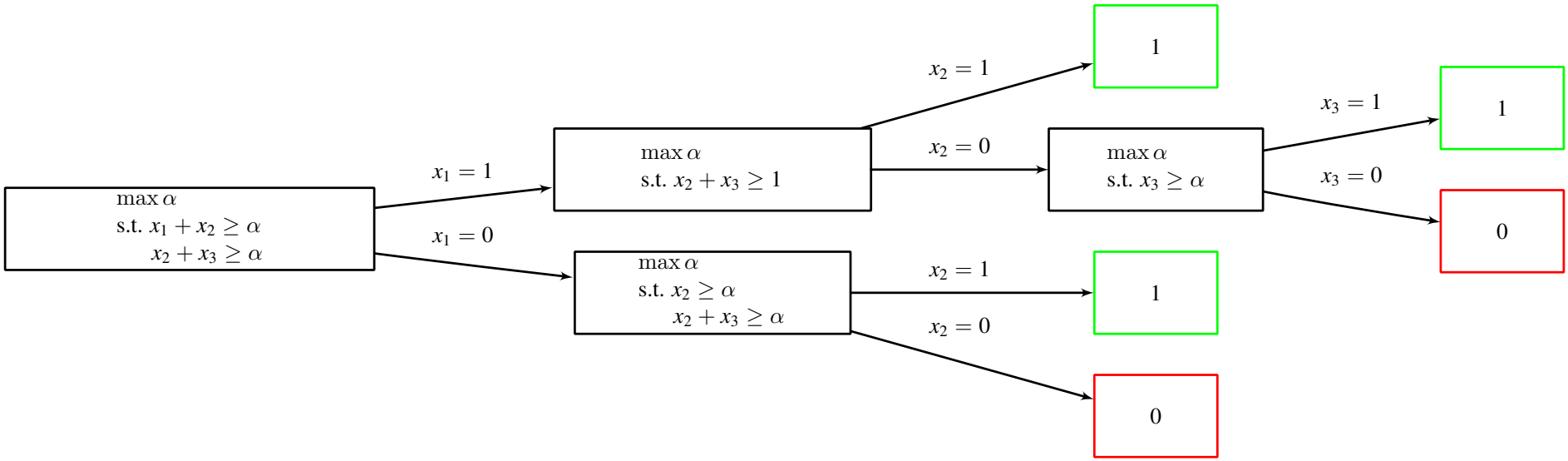
$$\max_{x \in \{0,1\}^n} \alpha$$
$$\text{s.t.} \sum_{i \in C_j^0} x_i + \sum_{i \in C_j^1} (1 - x_i) \geq \alpha \ \forall j \in J$$

- For $k = 2$, we then have

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

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.



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*.

Outline

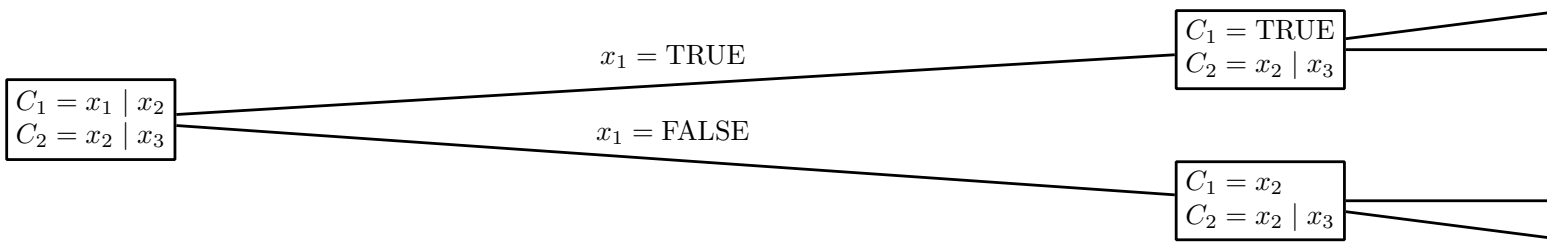
Complexity: Basic Notions

- 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).
  - A DTM specifies an *algorithm* computing the value of a Boolean function.
  - The DTM executes a program, reading the input from a *tape*.
  - We equate a given DTM with the program it executes.
  - The output is YES or NO.
  - A YES answer is returned if the machine reaches an *accepting state*.
- A problem is specified in the form of a *language*, defined to be the subset of the possible inputs over a given *alphabet* (Γ) that are expected to output YES.
- A DTM that produces the correct output for inputs w.r.t. a given language is said to *recognize the language*.
- Informally, we can then say that the DTM represents an "algorithm that solves the given problem correctly."

Non-deterministic Turing Machines

- 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


$C_1 = x_1 \mid x_2$
$C_2 = x_2 \mid x_3$
$x_1 = \text{TRUE}$
$x_1 = \text{FALSE}$
$C_1 = \text{TRUE}$
$C_2 = x_2 \mid x_3$
$C_1 = x_2$
$C_2 = x_2 \mid x_3$

Primitive Complexity Classes

- Languages can be grouped into *classes* based on the *best worst-case running time* of any TM that recognizes the language.
  - The class P is the set of all languages for which there exists a DTM that recognizes the language in time polynomial in the length of the input.
  - The class NP is the set of all languages for which there exists an NDTM that recognizes the language in time polynomial in the length of the input.
  - The class $coNP$ is the set of languages whose complements are in NP.
  - Additional classes can be formed hierarchically by the use of *oracles*.
- A language L_1 can be *reduced* to a language L_2 if there is an output-preserving polynomial transformation of members of L_1 to members of L_2.
- A language L is said to be *complete* for a class if all languages in the class can be reduced to L.
- We are primarily talking here about time complexity, though space complexity must ultimately also be considered.

Outline

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.

Collapsing the Hierarchy

In general, we have

$$\Sigma_0^p \subseteq \Sigma_1^p \subseteq \ldots \Sigma_k^p \subseteq \ldots$$
$$\Pi_0^p \subseteq \Pi_1^p \subseteq \ldots \Pi_k^p \subseteq \ldots$$
$$\Delta_0^p \subseteq \Delta_1^p \subseteq \ldots \Delta_k^p \subseteq \ldots$$

It is not known whether any of the inclusions are strict. We do have that

$$(\Sigma_k^p = \Sigma_{k+1}^p) \Rightarrow \Sigma_k^p = \Sigma_j^p \ \forall j \geq k \tag{4}$$

In particular, if $P = NP$, then every problem in the PH is solvable in polynomial time. Similar results hold for the Π and Δ hierarchies.

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

(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 $Y = \mathbb{Z}^{p_2} \times \mathbb{R}^{n_2 - p_2}$.

Continuous Second Stage

- In general, if $Y = \mathbb{R}^{n_1}$, then the lower-level problem can be replaced with its optimality conditions.
- The optimality conditions for the lower-level optimization problem are

$$\begin{aligned} G^2 y &\geq b^2 - A^2 x \\ uG^2 &\leq d^2 \\ u(b^2 - G^2 - A^2 x) &= 0 \\ (d^2 - uG^2)y &= 0 \\ u, y &\in \mathbb{R}_+ \end{aligned}$$

- When $X = \mathbb{R}^{n_1}$, this is a special case of a class of non-linear mathematical programs known as *mathematical programs with equilibrium constraints* (MPECs).
- An MPEC can be solved in a number of ways, including converting it to a standard integer program.
- Note that in this case, the value function of the lower-level problem is piecewise linear, but not necessarily convex.

Recourse Problems

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

$$\min\{-c^1 x + Q(x) \mid x \in \mathcal{P}_U \cap X\}, \tag{5}$$

where

$$Q(x) = \min\{d^1 y \mid y \in \mathcal{P}_L(x) \cap Y\}. \tag{6}$$

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

Other 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.

Outline

Two-Stage Stochastic Programs with Recourse

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

$$\min\{c^1 x + \mathbb{E}_\xi Q_\xi(x) \mid x \in \mathcal{P}_U \cap X\}, \tag{7}$$

where

$$Q_\xi(x) = \min\{d^2 y \mid y \in Y, G^2 y \geq \omega(\xi) - A^2 x\}, \tag{8}$$

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 (7) is a bilevel program.

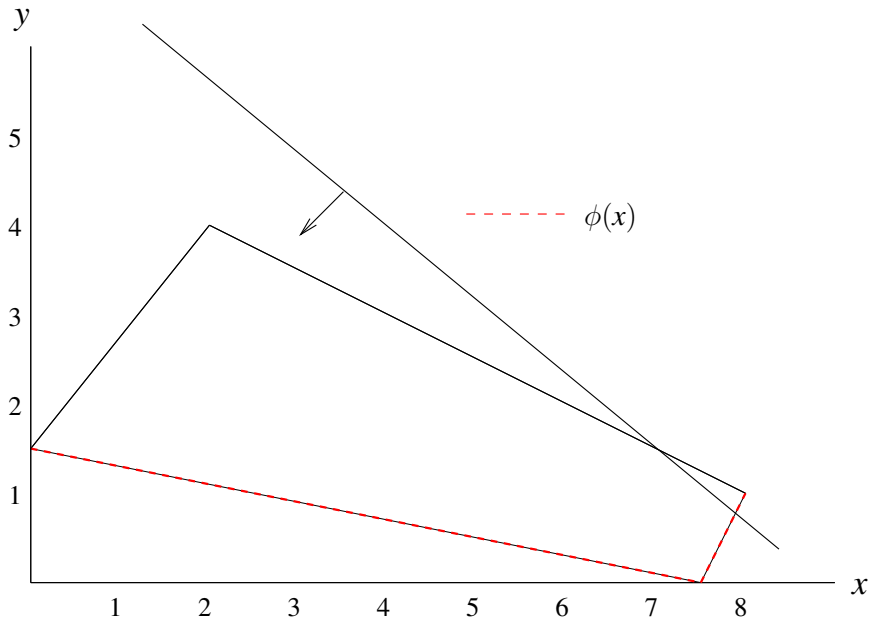
Benders' Principle (Linear Programming)

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

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

where

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

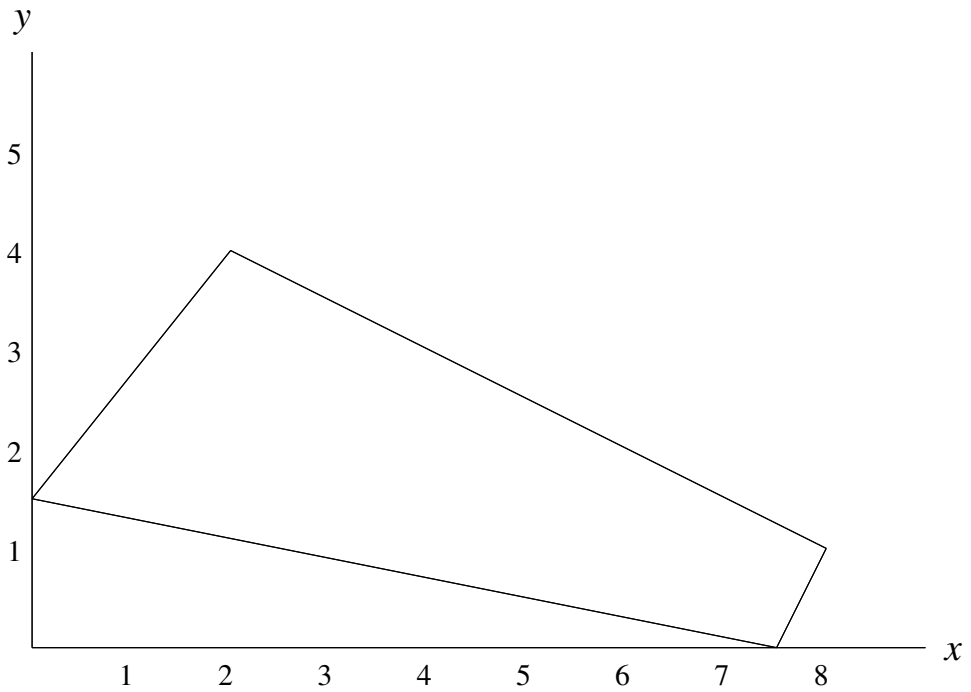


Basic Strategy:

- 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.

Example

$$
\begin{array}{lcll}
z_{LP} & = & \min & x + y \\
 & & \text{s.t.} & 25x - 20y \geq -30 \\
 & & & -x - 2y \geq -10 \\
 & & & -2x + y \geq -15 \\
 & & & 2x + 10y \geq 15 \\
 & & & x, y \in \mathbb{R}
\end{array}
$$

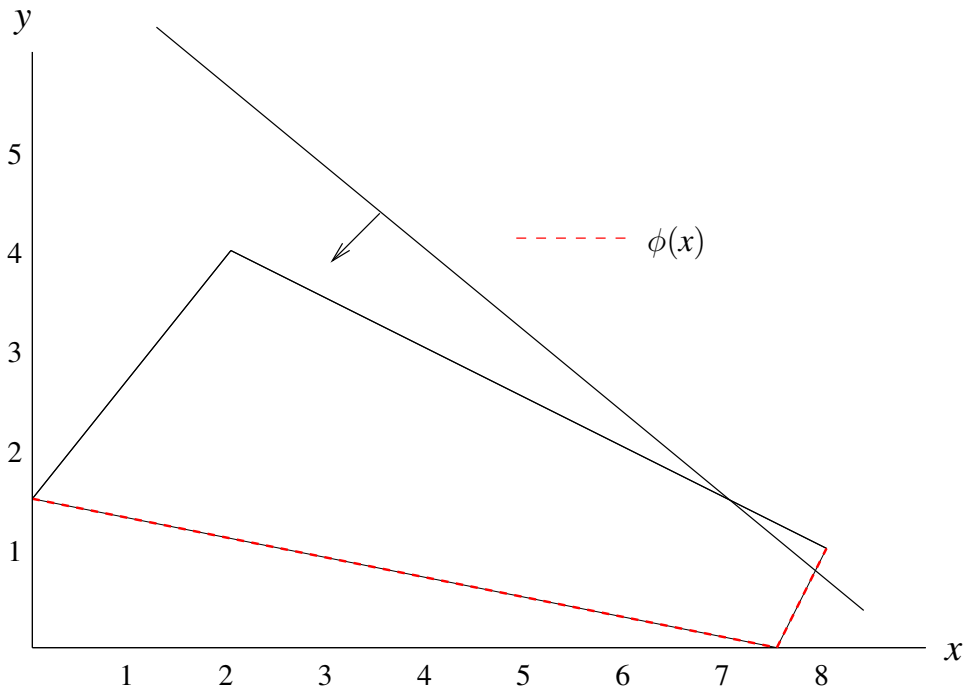


Value Function Reformulation

$$z_{LP} = \min_{x \in \mathbb{R}} x + \phi(x),$$

where

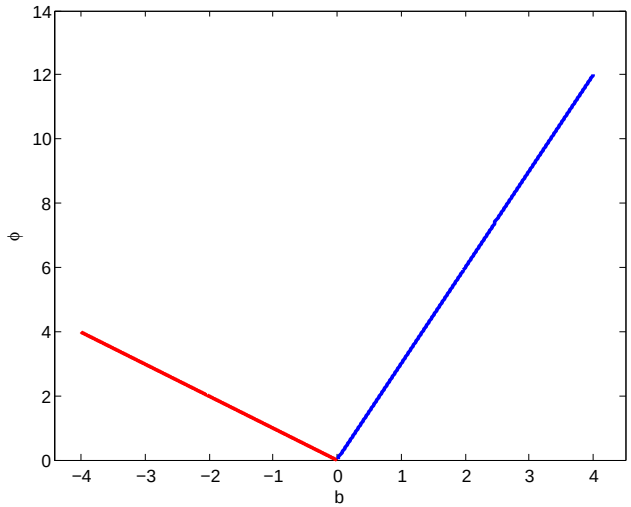
$$\begin{aligned} \phi(x) = \min \quad & y \\ \text{s.t.} \quad & -20y \geq -30 - 25x \\ & -2y \geq -10 + x \\ & y \geq -15 + 2x \\ & 10y \geq 15 - 2x \\ & y \in \mathbb{R} \end{aligned}$$



LP Value Function

Example

$$\phi_{LP}(b) = \min\ 6x_1 + 7x_2 + 5x_3$$
$$\text{s.t. } 2x_1 - 7x_2 + x_3 = b \qquad \text{(Ex.LP)}$$
$$x_1, x_2, x_3 \in \mathbb{R}_+$$



LP Value Function Structure

$$\begin{aligned} \phi_{LP}(b) = \min\ & c^\top x \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{R}^n_+ \end{aligned} \tag{LP}$$

- Assume the dual of (LP) is feasible.
- The epigraph of ϕ_{LP} is a convex cone, call it $\mathcal{L}$:

$$\mathcal{L} := cone\{(A_1, c_1), (A_2, c_2), \ldots, (A_n, c_n), (0, 1)\}$$

- Let $u_1, \ldots, u_k$ be extreme points of the feasible region of the dual of (LP) and $d_1, \ldots, d_p$ be its extreme directions. Then

$$\mathcal{L} := \{(b, z) : z \geq u_i^\top b, i = 1, \ldots, k, d_j^\top b \leq 0, j = 1, \ldots, p\}.$$

- Note that the value function has an underlying discrete structure.

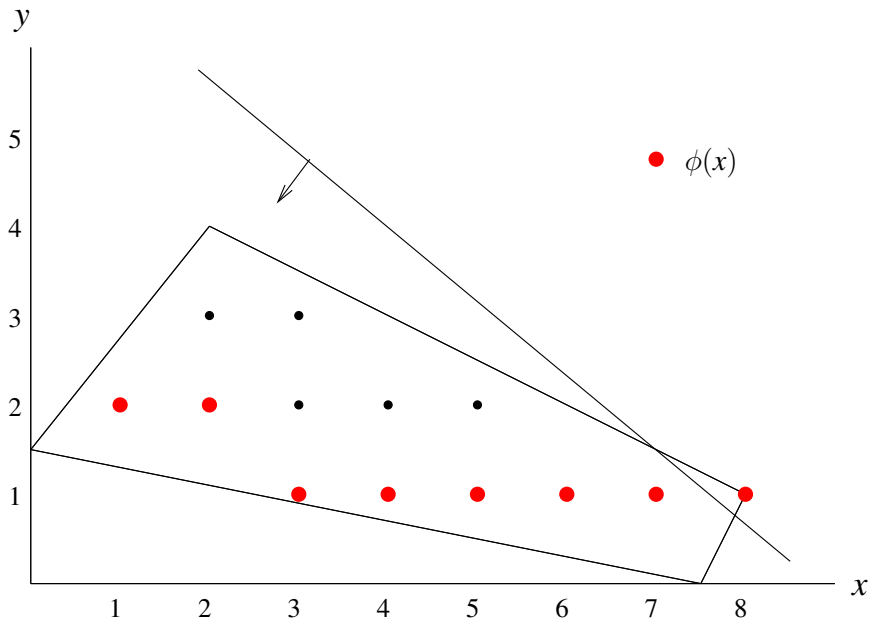
Benders' Principle (Integer Programming)

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

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

where

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

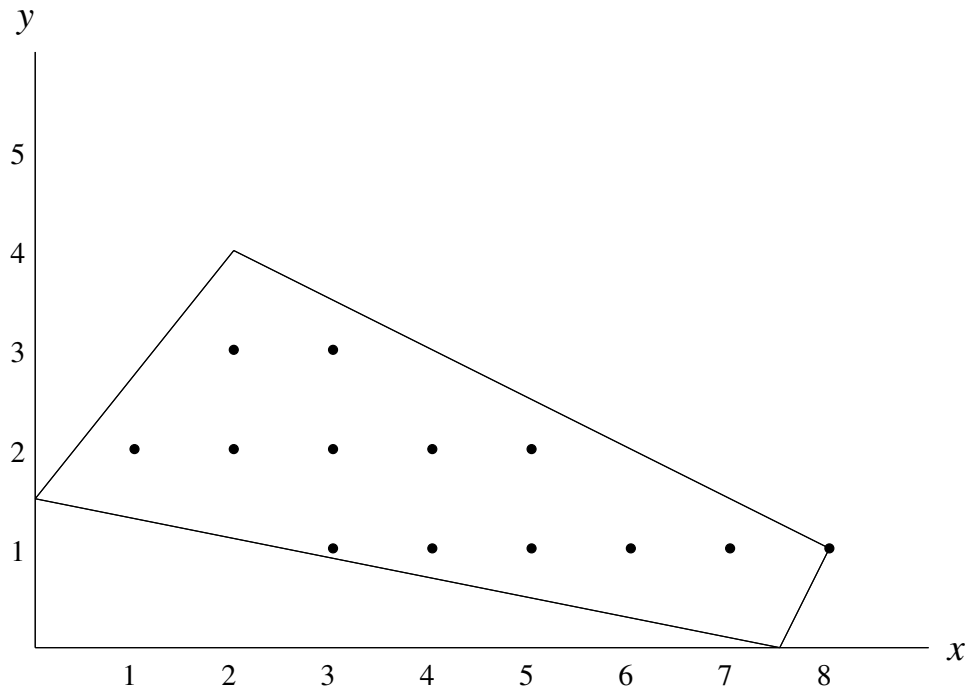


Basic Strategy:

- 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

$$
\begin{aligned}
z_{IP} \quad = \quad & \min \quad && x + y \\
& \text{s.t.} && 25x - 20y \geq -30 \\
& && -x - 2y \geq -10 \\
& && -2x + y \geq -15 \\
& && 2x + 10y \geq 15 \\
& && x, y \in \mathbb{Z}
\end{aligned}
$$

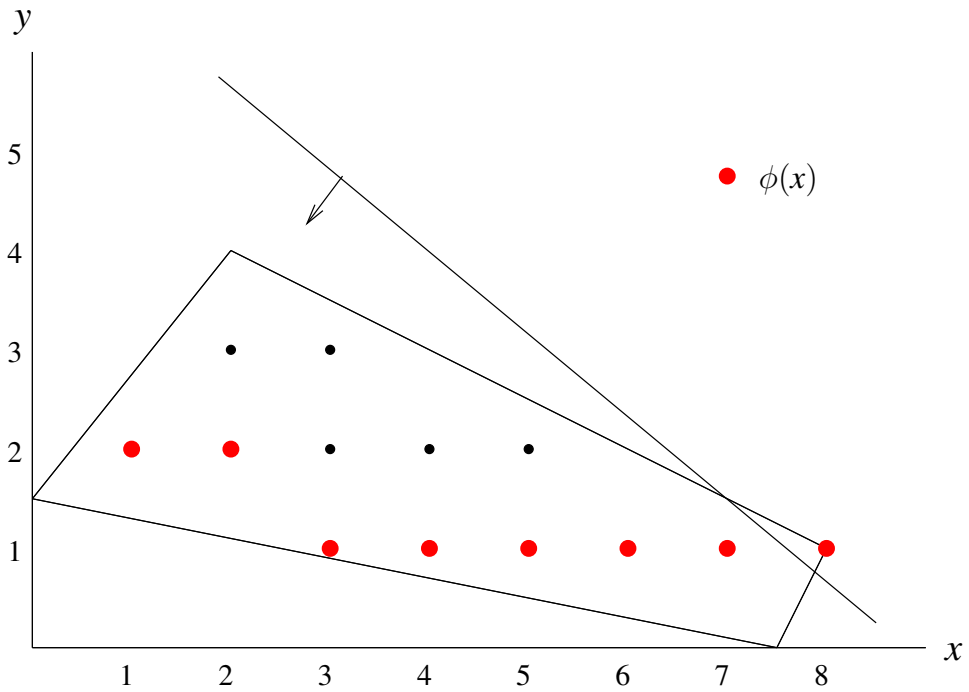


Value Function Reformulation

$$z_{IP} = \min_{x \in \mathbb{Z}} x + \phi(x),$$

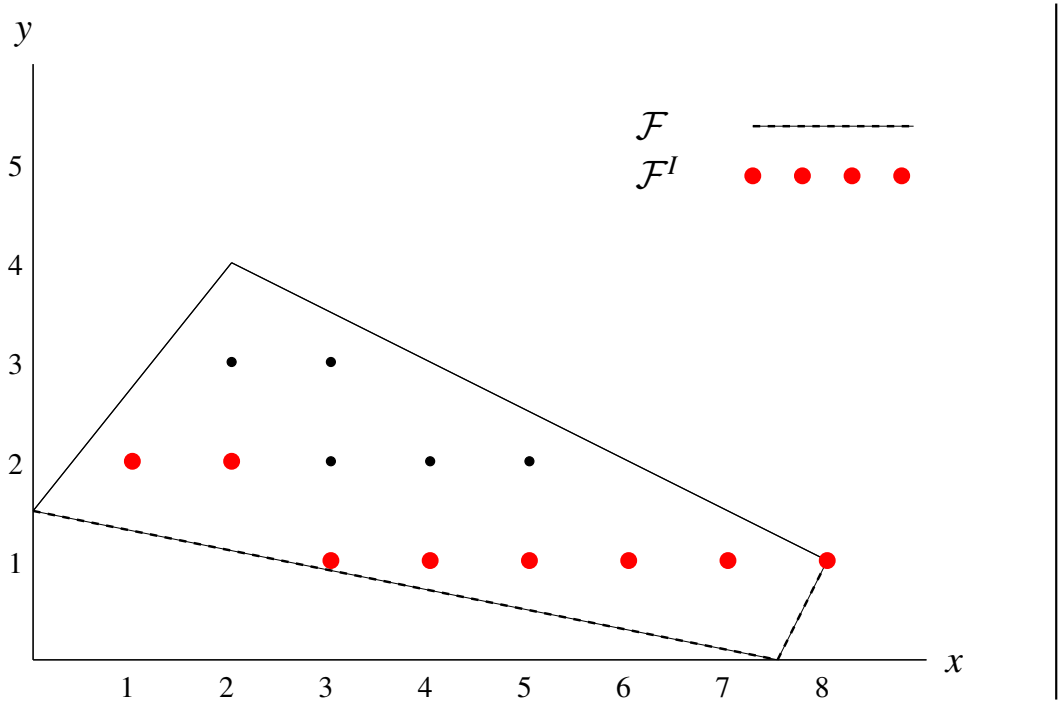
where

$$\begin{aligned} \phi(x) = \min \quad & y \\ \text{s.t.} \quad & -20y \geq -30 - 25x \\ & -2y \geq -10 + x \\ & y \geq -15 + 2x \\ & 10y \geq 15 - 2x \\ & y \in \mathbb{Z} \end{aligned}$$



Value Function Reformulation (General Case)

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.

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.

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

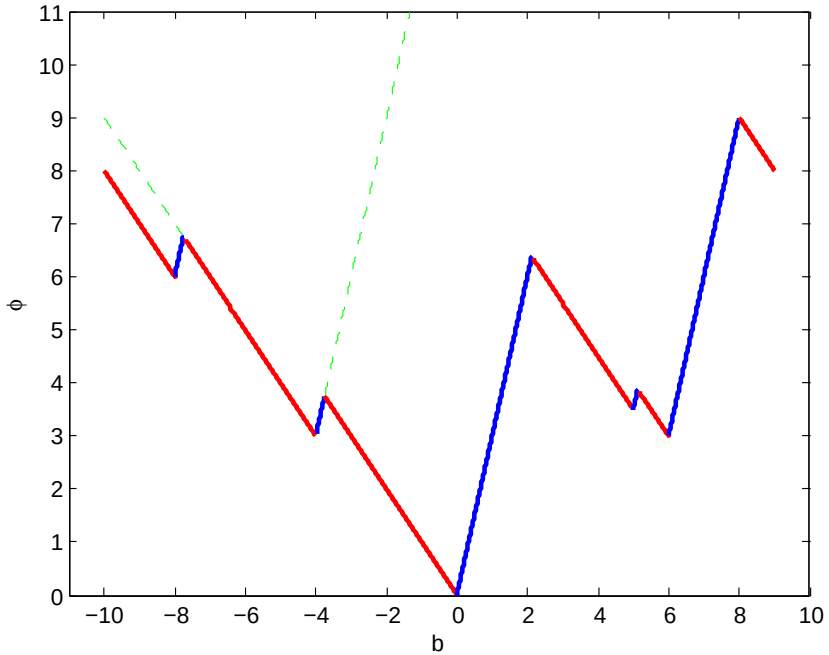
- $S(b) = \{x \in \mathbb{Z}^r_+ \times \mathbb{R}^{n-r}_+ \mid Ax = b\}$.
- $B = \{b \in \mathbb{R}^m \mid S(b) \neq \emptyset\}$.

Example: MILP Value Function

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

Example

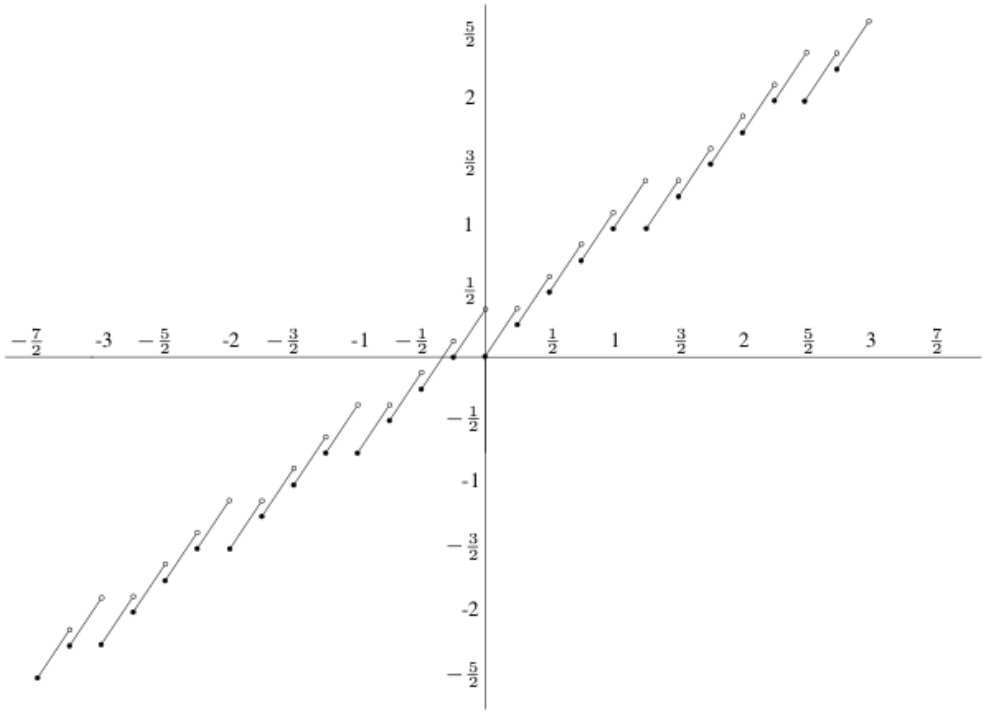
$$\phi(d) = \min\ 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6$$
$$\text{s.t. } 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = d$$
$$x_1, x_2, x_3 \in \mathbb{Z}_+,\ x_4, x_5, x_6 \in \mathbb{R}_+$$



Example: MILP Value Function

Example

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



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

$$\begin{aligned} \phi(b) = \min\ & c_I x_I + \phi_C(b - A_I x_I) \\ \text{s.t. } & x_I \in \mathbb{Z}^r_+ \end{aligned} \tag{9}$$

- 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

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

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

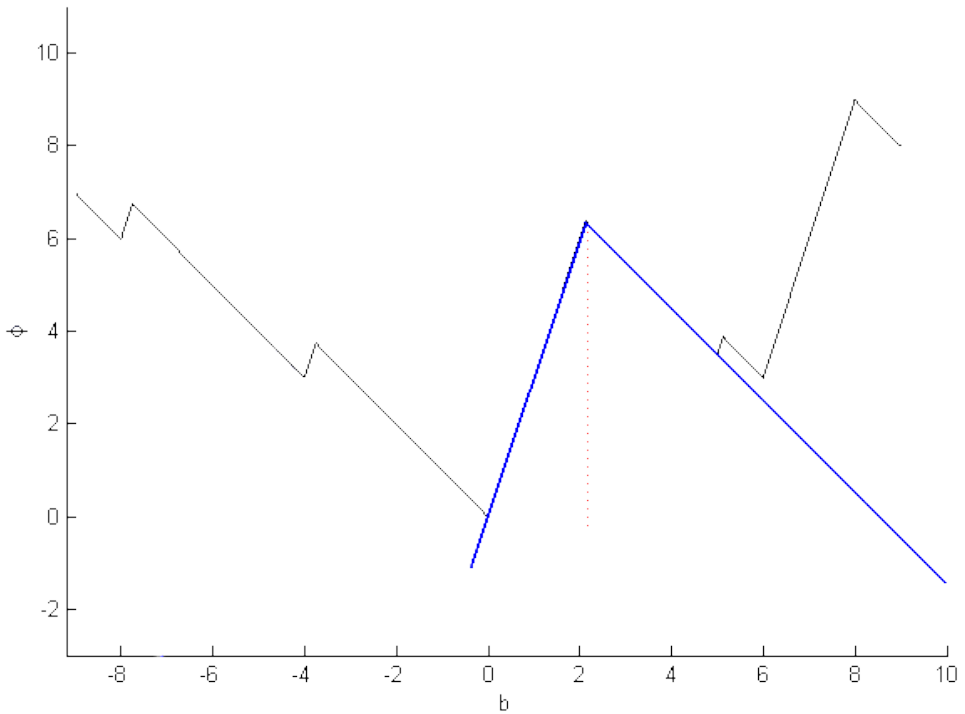
Bounding the Value Function From Below

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

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



Value Function Reformulation of the Two-Stage Problem

Let

- $\mathcal{B} = \{\beta : \beta = Tx, x \in X\}$
- $S_1(\beta) = \{x \in X \mid Tx = \beta\}$
- $S_2(\beta) = \{Wy = \beta,\ y \in Y\}$
- $\psi(\beta) = \min\{c^\top x \mid x \in S_1(\beta)\}$
- $\phi(\beta) = \{q^\top y : y \in S_2(\beta)\}$
- $f(\beta) = \{\psi(\beta) + \min \mathbb{E}_s[\phi(h_s - \beta)] \mid \beta \in \mathcal{B}\}$

Then our problem is to determine $\min_{\beta \in \mathcal{B}} f(\beta)$.

Assumptions:

- q, T, and W are fixed.
- The dual of the LP relaxation of the recourse problem is feasible, i.e.,

$$\{\nu \in \mathbb{R}^{m_2} : W_I^\top \nu \leq q_I, W_C^\top \nu \leq q_C\} \neq \emptyset$$

- X is non-empty and bounded.

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]

Generic Integer Benders' Algorithm

The Algorithm

Step 0. Initialize

a) Set $\beta^1 = Tx^1$ where $x^1 \in \operatorname{argmin}\{c^\top x : x \in X\}$

b) Initialize the dual function lists $\mathcal{F}_1 = \emptyset, \mathcal{F}_s = \emptyset$.

c) Set $k = 1$.

Step 1. Lower bound the problem and check for termination

a) Find optimal dual functions F_1^k and F_s^k for each $s \in 1 \ldots S$ to $\psi(\beta^k)$ and $\phi(h_s - \beta^k)$ respectively.

b) If

$$\max_{f_1 \in \mathcal{F}_1, f_s \in \mathcal{F}_s} \{f_1(\beta^k) + \mathbb{E}_s[f_s(h_s - \beta^k)]\} = F_1^k(\beta^k) + \mathbb{E}_s[F_s^k(h_s - \beta^k)]$$

then stop, $x^* \in \operatorname{argmin}\{c^\top x : x \in X, Tx = \beta^k\}$ is an optimal solution.

Generic Integer Benders' Algorithm

Step 2. Update the lower bound

a) Update the dual functions lists: $\mathcal{F}_1 = \mathcal{F}_1 \cup F_1^k$ and let $\mathcal{F}_s = \mathcal{F}_s \cup_{s\in\Omega} F_s^k$.

b) Solve the problem

$$z^k = \min_{\beta\in\mathcal{B}} \max_{f_1\in\mathcal{F}_1, f_s\in\mathcal{F}_s} \{f_1(\beta) + \mathbb{E}_s[f_s(h_s - \beta)]\}$$

and set its optimal solution to β^{k+1}.

c) Go to Step 1.

Outline

MILP Duals 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:

$$\begin{aligned} \min\ & c^\top x \\ \text{s.t. } & Ax = b, \\ & l^t \leq x \leq u^t, x \geq 0 \end{aligned} \tag{10}$$

The dual at node t:

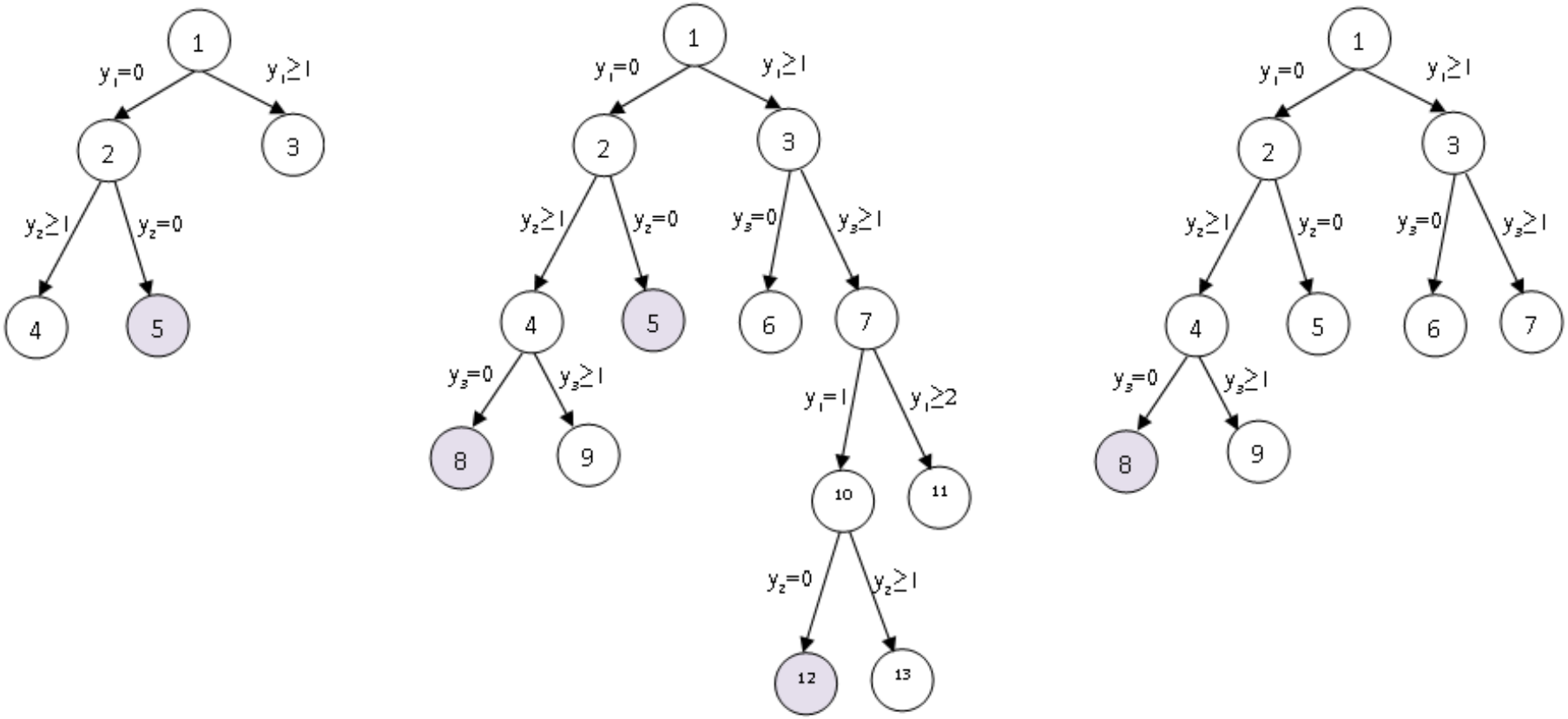
$$\begin{aligned} \max\ & \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \\ \text{s.t. } & \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ & \underline{\pi} \geq 0, \bar{\pi} \leq 0 \end{aligned} \tag{11}$$

We obtain the following strong dual function:

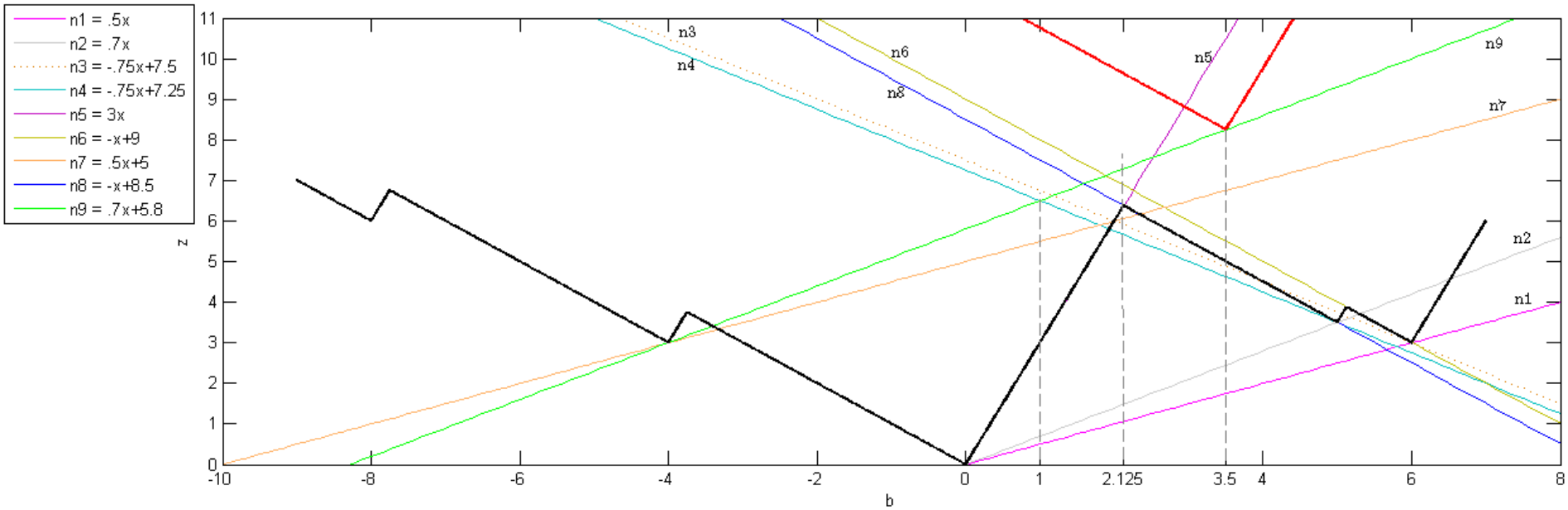
$$\min_{t \in T} \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \tag{12}$$

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

Consider

$$\begin{aligned} \min f(x) = \min \; & -3x_1 - 4x_2 + \sum_{s=1}^{2} 0.5Q(x,s) \\ & \text{s.t. } x_1 + x_2 \leq 5 \\ & x \in \mathbb{Z}_+ \end{aligned} \tag{13}$$

where

$$\begin{aligned} Q(x,s) = \min \; & 3y_1 + \frac{7}{2}y_2 + 3y_3 + 6y_4 + 7y_5 \\ & \text{s.t. } 6y_1 + 5y_2 - 4y_3 + 2y_4 - 7y_5 = h(s) - 2x_1 - \frac{1}{2}x_2 \\ & y_1, y_2, y_3 \in \mathbb{Z}_+, \; y_4, y_5 \in \mathbb{R}_+ \end{aligned} \tag{14}$$

with $h(s) \in \{-4, 10\}$.

Example

Iteration 1

Step 0

- $\mathcal{F} = \emptyset$
- $k = 1$.
- Solve

$$\begin{aligned} \min f(x) = \min \; & -3x_1 - 4x_2 \\ \text{s.t. } & x_1 + x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

$f^0 = 20, x_1^* = 0, x_2^* = 5, \beta^1 = \frac{5}{2}$

Example

Step 1

- Solve the second stage problem for each scenario. That is, with $h(1) - \beta^1 = -6.5$ and $h(2) - \beta^1 = 7.5$.
- The respective dual functions are

  $F^1_{s=1}(\beta) = \min\{-\beta - 1, 0.5\beta + 10\}$ and
  $F^1_{s=2}(\beta) = \min\{3\beta - 15, -0.75\beta + 14.5\}$.

  Then, $\mathcal{F}(\beta) = \max\{F^1_{s=1}, F^1_{s=2}\}$.

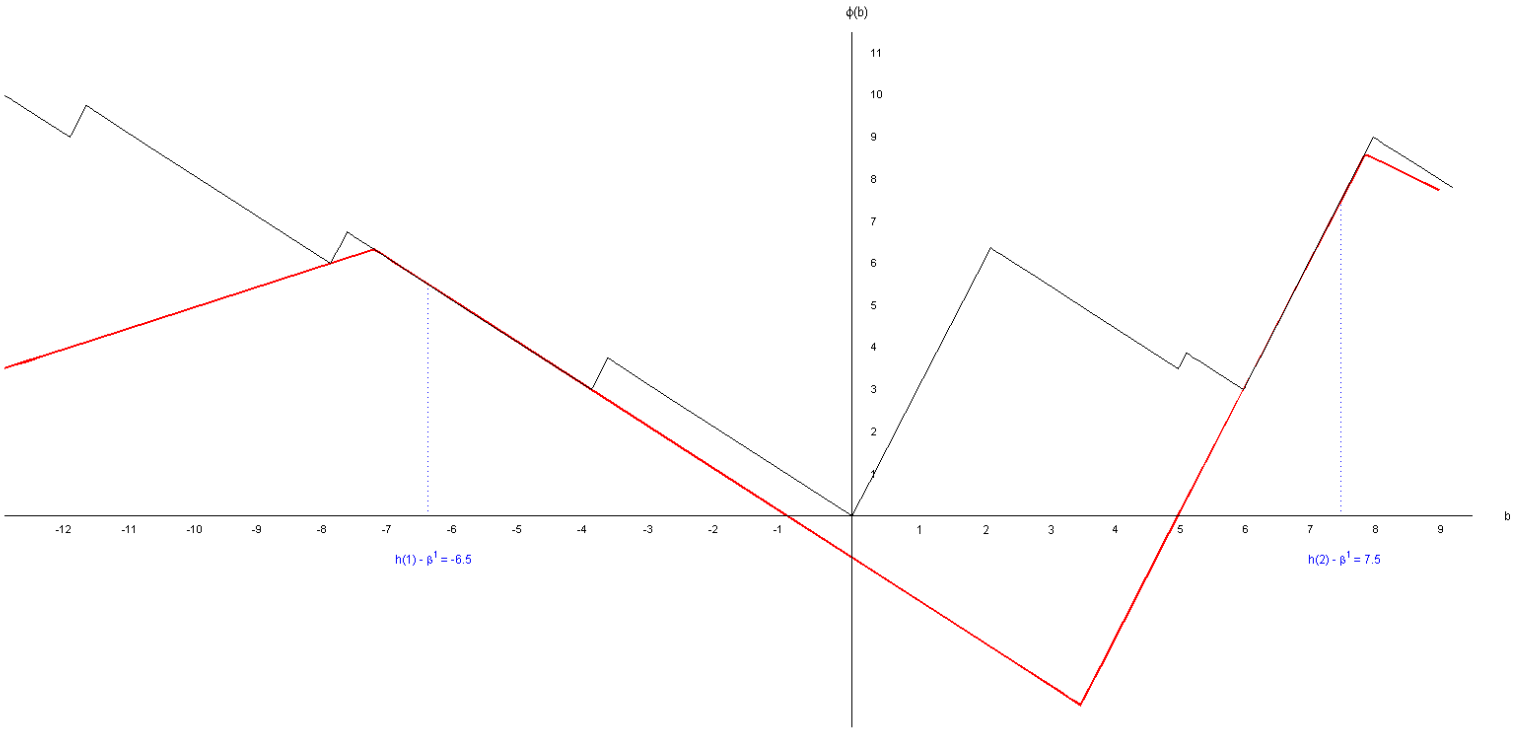
Step 2

- Solve the master problem

$$
\begin{aligned}
f^1 = \min \; & -3x_1 - 4x_2 + 0.5(\mathcal{F}_s(-4-\beta) + \mathcal{F}_s(10-\beta)) \\
\text{s.t. } & x_1 + x_2 \leq 5 \\
& 2x_1 + \frac{1}{2}x_2 = \beta \\
& x_1, x_2 \in \mathbb{Z}_+
\end{aligned}
$$

- The solution to the master problem is $f^1 = -16.75$ with $\beta^1 = 7$.

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

Iteration 2

Step 1

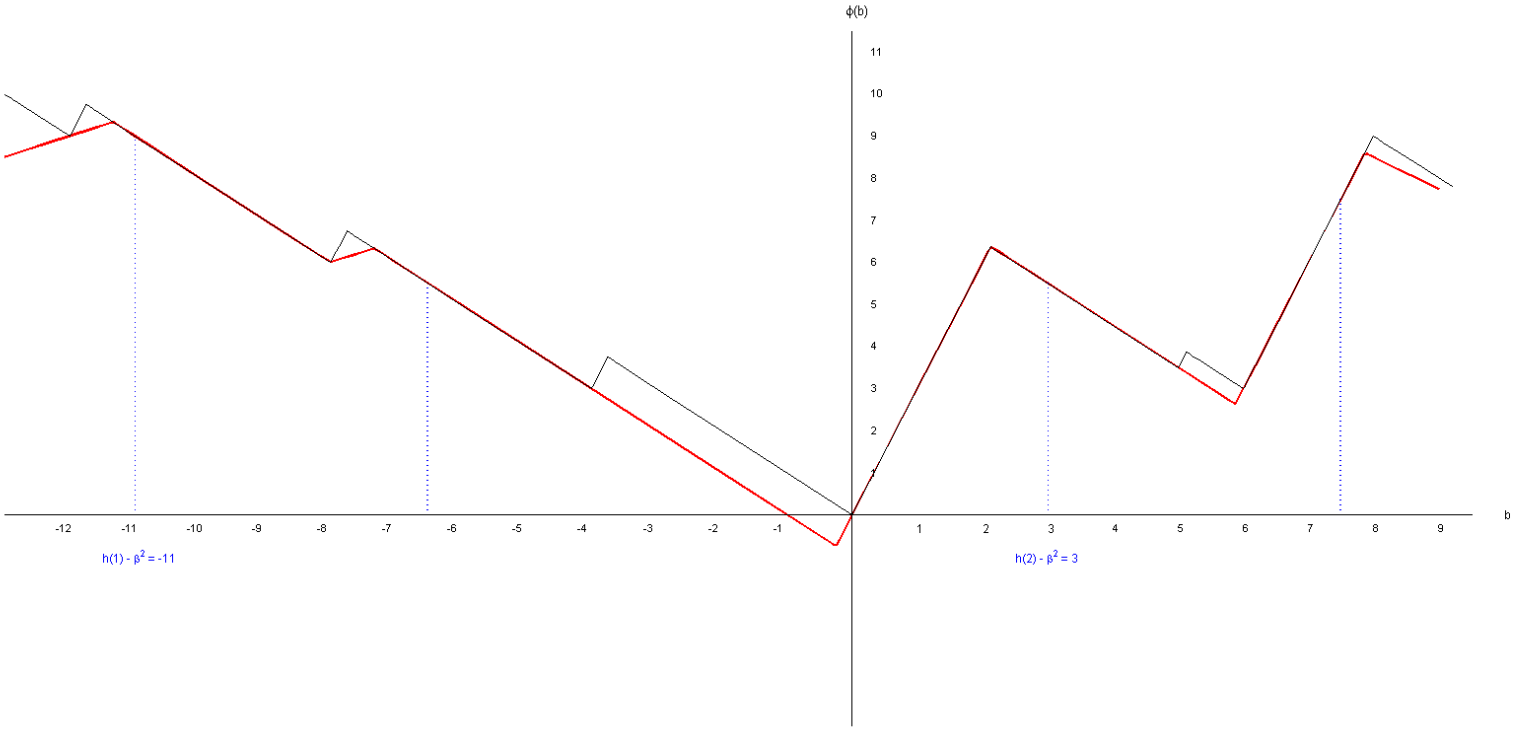
- Solve the second stage problem with right hand sides: -11 and 3.
- The respective dual functions are: $F^2_{s=1}(\beta) = \min\{-\beta - 2, 0.5\beta + 15\}$ and $F^2_{s=2}(\beta) = \min\{3\beta, -\beta + 8.5, 0.7\beta + 5.8\}$.

- Since $\mathcal{F}(-11) + \mathcal{F}(3) < F^2_{s=1}(-11) + F^2_{s=2}(3)$, we continue:
- Update $\mathcal{F}(\beta) = \max\{F^1_{s=1}, F^1_{s=2}, F^2_{s=1}, F^2_{s=2}\}$.

Step 2

- Solve the updated master problem. We obtain $f^2 = -14.5$ with $\beta^2 = 4$.

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

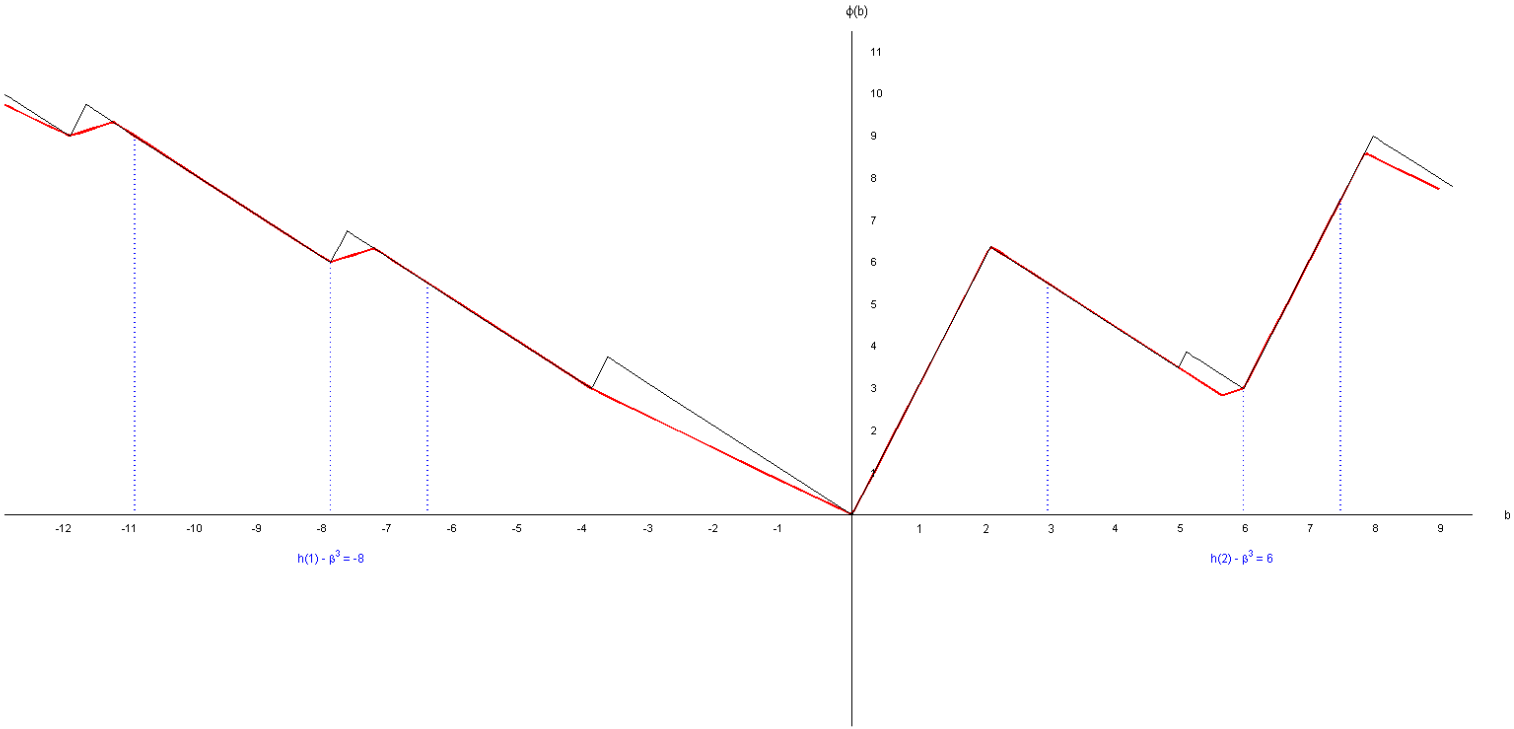
Iteration 3

Step 1

- Solve the second stage problem with right hand sides: -8 and 6.
- The respective dual functions are:
  $F^3_{s=1}(\beta) = -0.75\beta$ and $F^3_{s=2}(\beta) = 0.5\beta$.

- $\mathcal{F}(-8) + \mathcal{F}(6) = F^3_{s=1}(-8) + F^3_{s=2}(6) = 9$, the approximation is exact and the optimal solution to the problem is $f^3 = -14.5$ and $\beta^3 = 4$.

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

Implementation Challenges

- To make the algorithm practical, several issues need to be solved.
- The master problem includes a piecewise linear function which grows in dimensions.
- In each iteration, for a scenario s, $S \times N(s)$ binary variables are added, where $N(s)$ is the number of new pieces of the function.
- Therefore, some "cut pool management" techniques need to be used to keep the size of the master problem manageable.
- This requires using an appropriate database.
- The examined right hand sides and their corresponding dual functions also need to be stored in an efficient manner.

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.

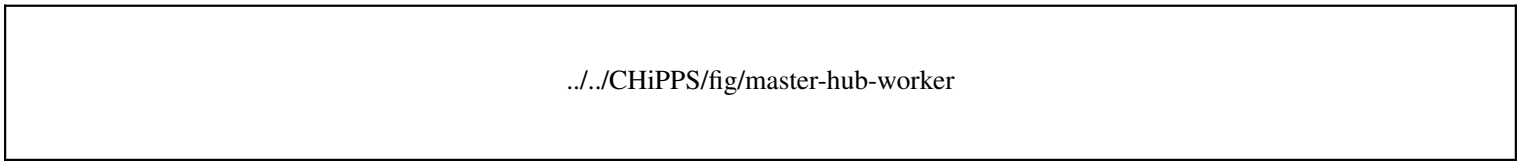
Outline

Why do Parallel Algorithms Arise Naturally?

Parallel algorithms are very natural in this setting for a number of reasons.

- The possible execution paths of a DTM can be thought of as specifying a tree (execution involves searching this tree).
- Problems in NP are those in which exploration of an exponential number of paths is unavoidable (in the worst case).
- Another way of thinking of problems in NP is as problems that can be solved in polynomial time given an exponential number of processors.
- **Problems higher in the hierarchy require even more enumeration and thus present even more potential for parallelization**.

Task Partitioning in Search Algorithms



Why Isn't Parallel Computing a Panacea?

- Practical algorithms use heuristics to avoid enumeration as much as possible.
- We do not know ahead of time what execution paths will be necessary to the computation.
- This makes it very difficult to distribute the computation.
- In essence, practical algorithms are *designed not to be parallelizable*.

The Case of Branch and Bound

- The execution of branch and bound can be thought of as exploring a particular search tree.
- This tree is essentially the one arising from execution of the corresponding DTM.
- Solvers typically endeavor to make this tree as small as possible.
- The decision problem at each node is to determine which disjunction to branch on in order to minimize the resulting subtree.
- Thus, the solution process can be viewed as a kind of multilevel game in itself.
- As mentioned previously, minimizing the size of the tree actually reduces the potential for parallelization.

Seeing the Forest for the Trees

- A theme that has run through all of the topics covered in this talk is the need to explore enormous search trees.
- In two-stage stochastic integer programs with recourse, there is an embarrassing wealth of source for parallelism.
- We are just beginning to understand how to exploit this

Trees

- Game trees
- Branch-and-bound trees
- Scenario trees

Conclusions

- 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 opportunities.
- Our aim is not just to develop the theory, but also to put it into practice.
- Please join us!

http://www.coin-or.org

Questions?

References I

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

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

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

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

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

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

References II

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

S. Sen and J.L. Higle. The C3 theorem and a D2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104 (1):1–20, 2005. ISSN 0025-5610.