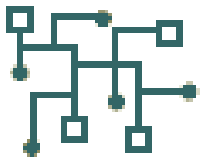
A Benders' Algorithm for Two-Stage Stochastic Optimization Problems With Mixed Integer Recourse

Ted Ralphs[1]
Joint work with Menal Güzelsoy[2] and Anahita Hassanzadeh[1]

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

INFORMS Computing Society Conference, Richmond, VA, January 11, 2015



ISE

Industrial and Systems Engineering

COR@L

COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

Outline

Outline

Two-Stage Mixed Integer Optimization

- We have the following general formulation:

$$z_{\text{2SMILP}} = \min_{x \in \mathcal{P}_1} \Psi(x) = \min_{x \in \mathcal{P}_1} \left\{ c^\top x + \Xi(x) \right\}, \tag{1}$$

  where

$$\mathcal{P}_1 = \{ x \in X \mid Ax = b, x \geq 0 \} \tag{2}$$

  is the *first-stage feasible region* with $X = \mathbb{Z}_+^{r_1} \times \mathbb{R}_+^{n_1 - r_1}$.
- Ξ represents the impact of future uncertainty.
- The canonical form employed in stochastic programming with recourse is

$$\Xi(x) = \mathbb{E}_{\omega \in \Omega} \left[\phi(h_\omega - T_\omega x) \right], \tag{3}$$

- ϕ is the second-stage *value function* to be defined shortly.
- $T_\omega \in \mathbb{Q}^{m_2 \times n_1}$ and $h_\omega \in \mathbb{Q}^{m_2}$ represent the input to the second-stage problem for scenario $\omega \in \Omega$.

The Second-Stage Value Function

- The structure of the objective function Ψ depends primarily on the structure of the *value function*

$$\phi(\beta) = \min_{y \in \mathcal{P}_2(\beta)} q^\top y \tag{RP}$$

  where

$$\mathcal{P}_2(\beta) = \{y \in Y \mid Wy = \beta\} \tag{4}$$

  is the *second-stage feasible region* with respect to a given right-hand side β and $Y = \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}$.
- The second-stage problem is parameterized on the unknown value β of the right-hand side.
- This value is determined jointly by the realized value of ω and the values of the first-stage decision variables.
- We assume
  - ω follows a uniform distribution with a finite support,
  - $\mathcal{P}_1$ is compact, and
  - $\mathbb{E}_{w \in \Omega}[\phi(h_\omega - T_\omega x)]$ is finite for all $x \in X$.

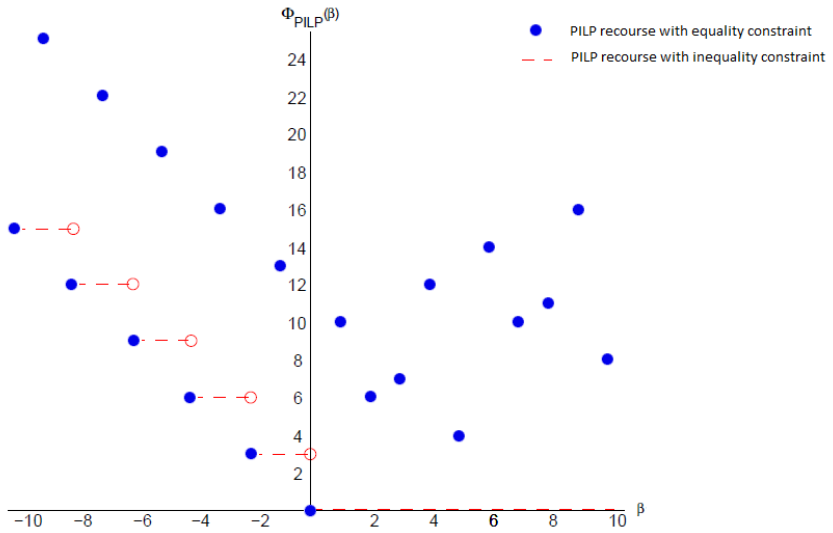
Outline

MILP Value Function (Pure)

The MILP value function is non-convex, discontinuous,and piecewise polyhedral in general.

Example 1

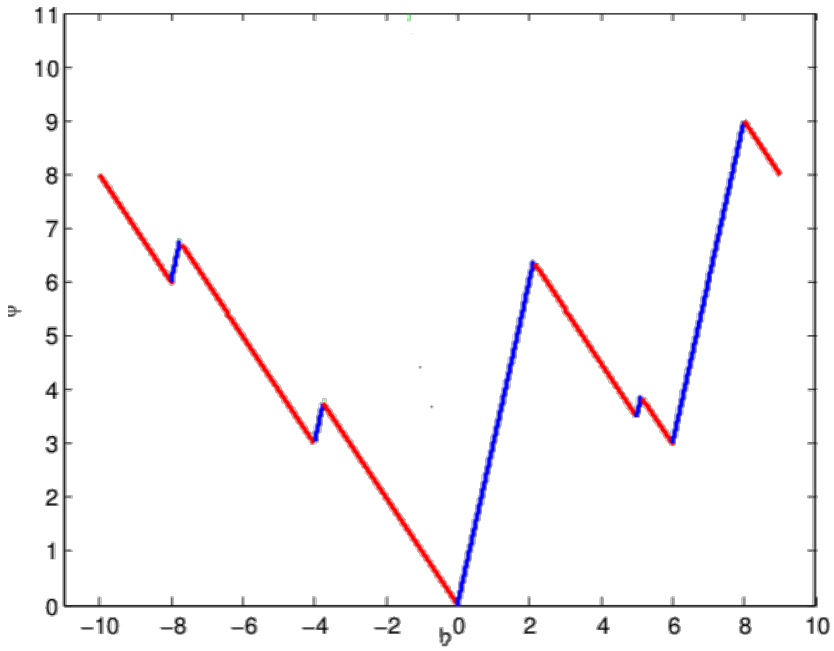
$$\phi(b) = \min\ 3x_1 + \frac{7}{2}x_2 + 3x_3 + 6x_4 + 7x_5 + 5x_6$$
$$s.t.\ 6x_1 + 5x_2 - 4x_3 + 2x_4 - 7x_5 + x_6 = b$$
$$x_1, x_2, x_3, x_4, x_5, x_6 \in \mathbb{Z}_+$$



MILP Value Function (Mixed)

Example 2

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

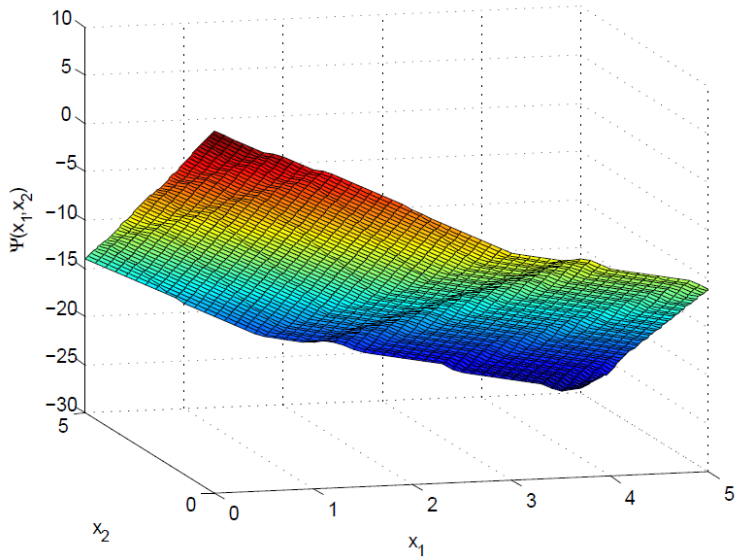


Mixed Integer Two-Stage Problem

Example 3

$$\begin{aligned} \min f(x_1, x_2) = \min \; & -3x_1 - 4x_2 + \mathbb{E}[\Phi_{MILP}(\omega - 2x_1 - 0.5x_2)] \\ & s.t.\; x_1 \leq 5, x_2 \leq 5 \\ & x_1, x_2 \in \mathbb{R}_+, \end{aligned} \tag{Ex.SMP}$$

and $\omega \in \{6, 12\}$ *with a uniform probability distribution.*



Continuous and Integer Restriction of an MILP

Consider the general form of the second-stage value function

$$\begin{aligned} \phi(\beta) = \min\ & q_I^\top y_I + q_C^\top y_C \\ \text{s.t. } & W_I y_I + W_C y_C = b, \\ & y \in \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2} \end{aligned} \tag{MILP}$$

The structure is inherited from that of the *continuous restriction*:

$$\begin{aligned} \phi_C(\beta) = \min\ & q_C^\top y_C \\ \text{s.t. } & W_C y_C = \beta, \\ & y_C \in \mathbb{R}_+^{n_2 - r_2} \end{aligned} \tag{CR}$$

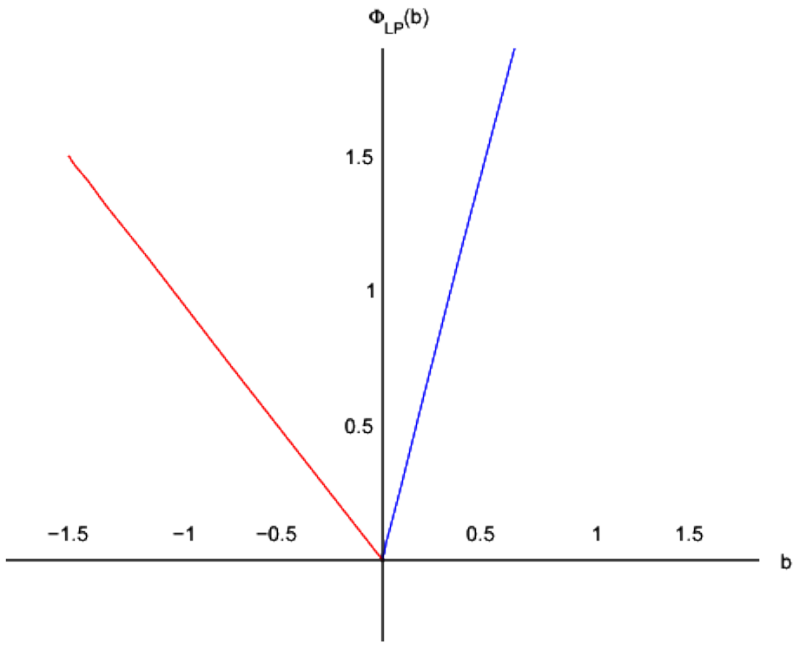
and the similarly defined *integer restriction*:

$$\begin{aligned} \phi_I(\beta) = \min\ & q_I^\top y_I \\ \text{s.t. } & W_I y_I = \beta \\ & y_I \in \mathbb{Z}_+^{r_2} \end{aligned} \tag{IR}$$

Value Function of the Continuous Restriction

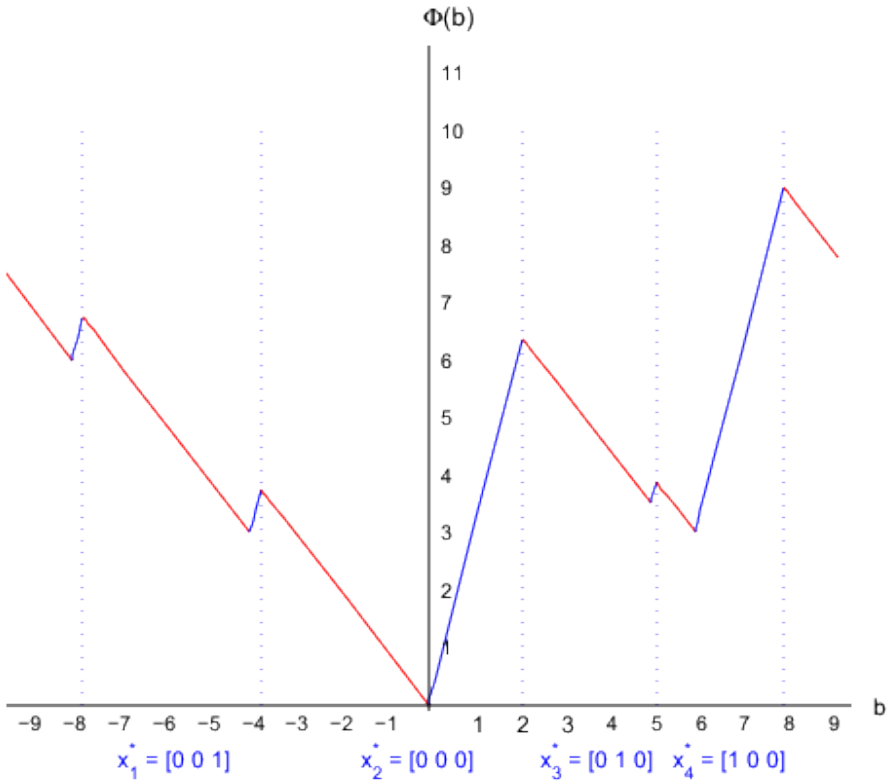
Example 4

$$\begin{aligned}\phi_C(\beta) = \min\ & 6y_1 + 7y_2 + 5y_3 \\ s.t.\ & 2y_1 - 7y_2 + y_3 = \beta \\ & y_1, y_2, y_3 \in \mathbb{R}_+\end{aligned}$$



Points of Strict Local Convexity

Example 5



Theorem 1 *(?)*

Under the assumption that $\{\beta \in \mathbb{R}^{m_2} \mid \phi_I(\beta) < \infty\}$ *is finite, there exists a finite set* $\mathcal{S} \subseteq Y$ *such that*

$$\phi(\beta) = \min_{y_I \in \mathcal{S}} \{q_I^\top y_I + \phi_C(\beta - W_I y_I)\}. \tag{5}$$

Outline

Representing the Value Function

In practice, the value function can be represented in primarily two different ways.

1. On the one hand, the value function is uniquely determined by its points of *strict local convexity*.
   - Embedding the value function using this representation involves explicitly listing these points and choosing one (binary variables).
   - The corresponding continuous part of the solution can be generated dynamically or can also be represented explicitly by dual extreme points.
2. The value function can also be represented explicitly in terms of its *polyhedral pieces*.
   - In this case, the points of strict local convexity are implicit and the selection is of the relevant piece or pieces.
   - This yields a much larger representation.

Embedding the Value Function

Algorithmically, the value function can be embedded in a number of ways.

1. If the objective functions of the inner and outer optimizations "agree," we can embed the inner constraints and evaluate the value function implicitly.
   - This amounts to solving the extensive form in the case of stochastic integer programming.
   - For stochastic programs with many scenarios, this form can be too large to solve explicitly.
2. We can generate the value function a priori and then embed the full representation.
   - The full representation might be very large
   - It is likely that most of the representation is irrelevant to the computation.
3. We can dynamically generate "relevant" parts of the representation ala cut generation.

It is also possible to take a hybrid approach in which we, e.g., generate the full representation a priori, but add parts dynamically.

Conceptual Algorithm for Generating the Value Function

Algorithm

Initialize: Let $\bar{z}(b) = \infty$ for all $b \in B$, $\Gamma^0 = \infty$, $x_I^0 = 0$, $S^0 = \{x_I^0\}$, and $k = 0$.
while $\Gamma^k > 0$ **do**:

- Let $\bar{z}(b) = \min\{\bar{z}, \bar{z}(b; x_I^k)\}$ for all $b \in B$.
- $k \leftarrow k + 1$.
- Solve

$$\begin{aligned} \Gamma^k = \max\ & \bar{z}(b) - c_I^\top x_I \\ \text{s.t. } & A_I x_I = b \\ & x_I \in \mathbb{Z}_+^r. \end{aligned} \tag{SP}$$

  to obtain x_I^k.
- Set $S^k \leftarrow S^{k-1} \cup \{x^k\}$

end while
return $z(b) = \bar{z}(b)$ for all $b \in B$.

Conceptual Algorithm for Generating the Value Function

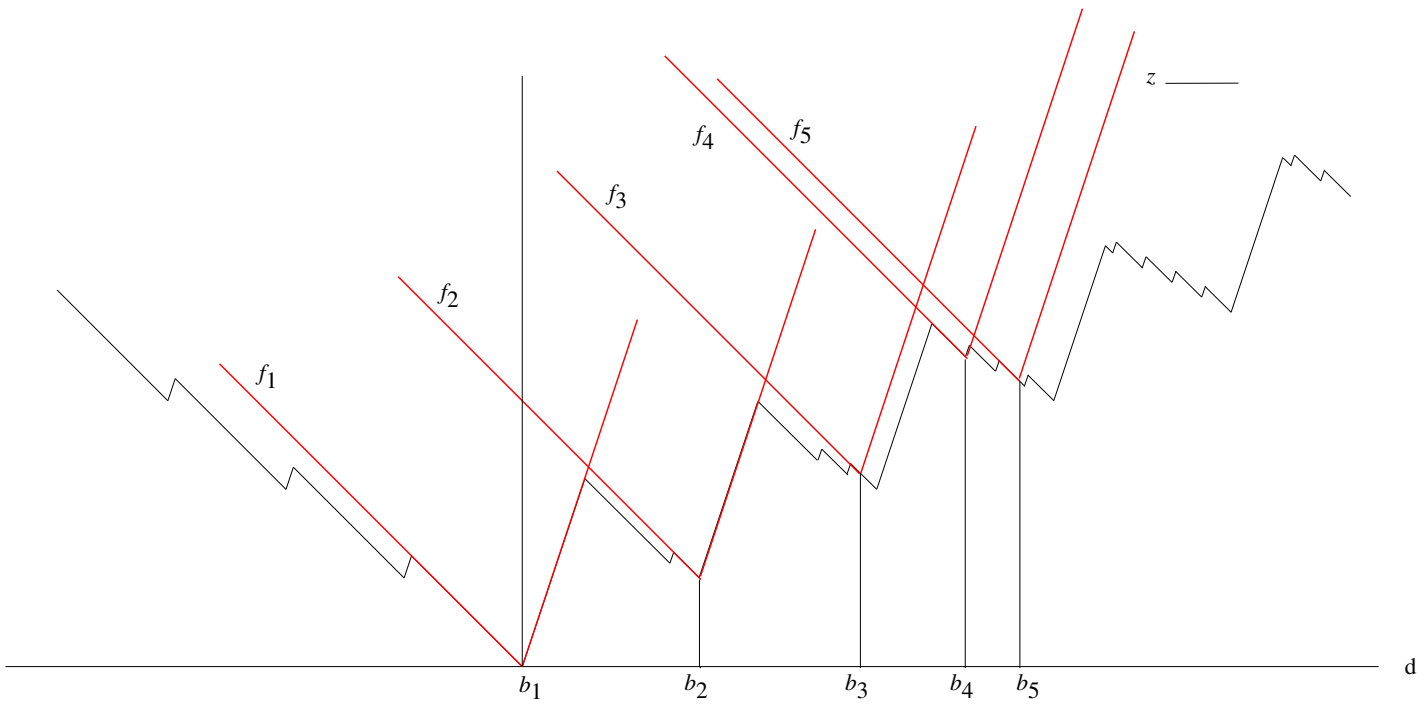


Figure: Upper bounding functions obtained at right-hand sides $b_i, i = 1, \ldots, 5$.

Formulating (SP)

Surprisingly, the "cut generation" problem (SP) can be formulated easily as an MINLP.

$$
\begin{aligned}
\Gamma^k = \; & \max \theta \\
& \text{s.t. } \theta + c_I^\top x_I \leq c_I^\top x_I^i + (A_I x_I - A_I x_I^i)^\top \nu^i \quad i = 1, \ldots, k-1 \\
& \quad A_C^\top \nu^i \leq c_C \quad i = 1, \ldots, k-1 \\
& \quad \nu^i \in \mathbb{R}^m \quad i = 1, \ldots, k-1 \\
& \quad x_I \in \mathbb{Z}_+^r .
\end{aligned} \tag{6}
$$

Computational Results

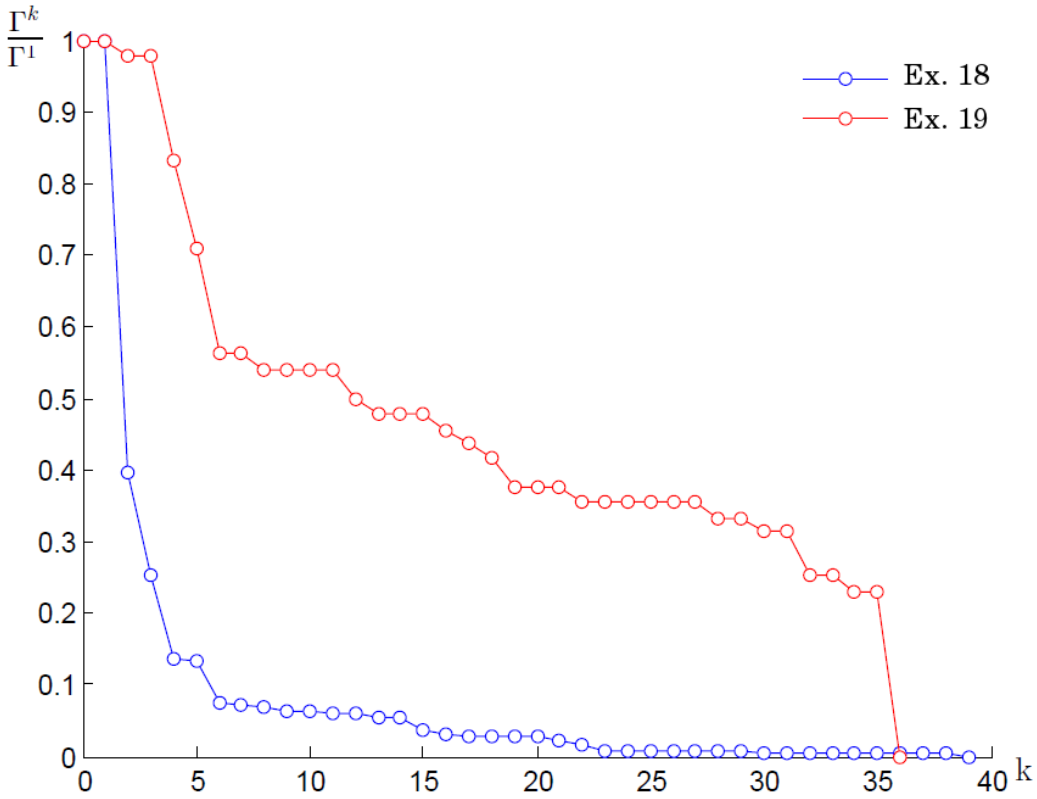
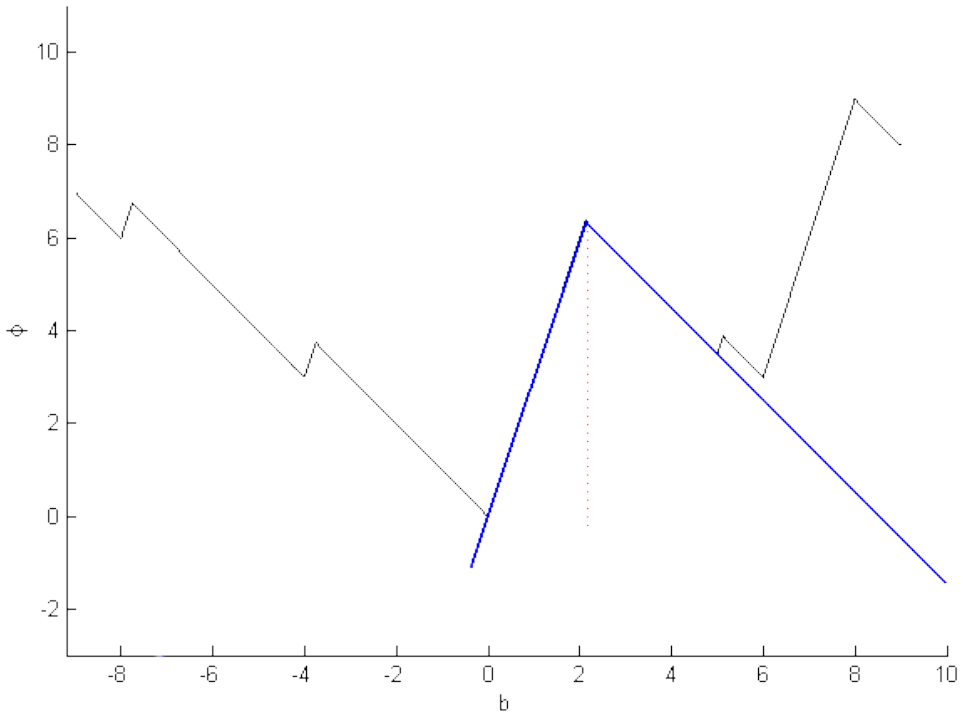


Figure: Normalized approximation gap vs. iteration number.

Generating the Value Function by Branch and Bound

- Our first algorithm was based on refining a single upper approximation (essentially a restriction of the full value function).
- It is also possible to construct the value function by lower approximation using branch and bound.
- Any branch-and-bound tree yields a lower approximation of the value function.



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:

$$\begin{aligned} \phi^t(b) = \min\ & c^\top x \\ \text{s.t. } & Ax = b, \\ & l^t \leq x \leq u^t, x \geq 0 \end{aligned} \tag{7}$$

The dual at node t:

$$\begin{aligned} \phi^t(b) = \max\ & \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \\ \text{s.t. } & \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ & \underline{\pi} \geq 0, \bar{\pi} \leq 0 \end{aligned} \tag{8}$$

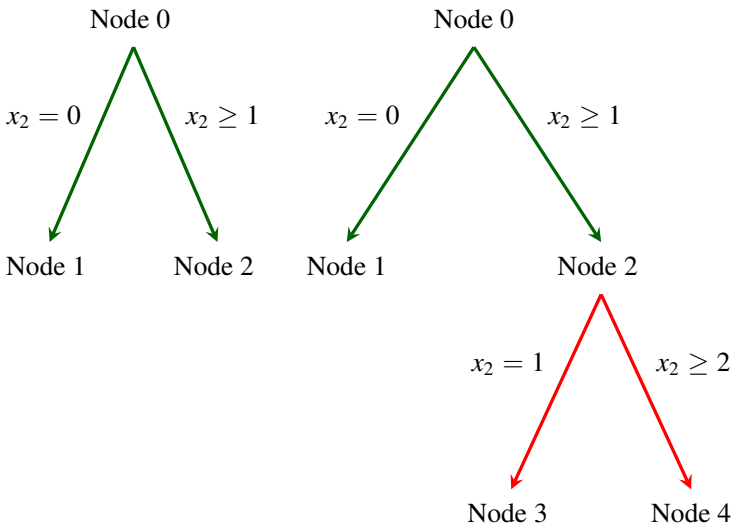
We obtain the following strong dual function:

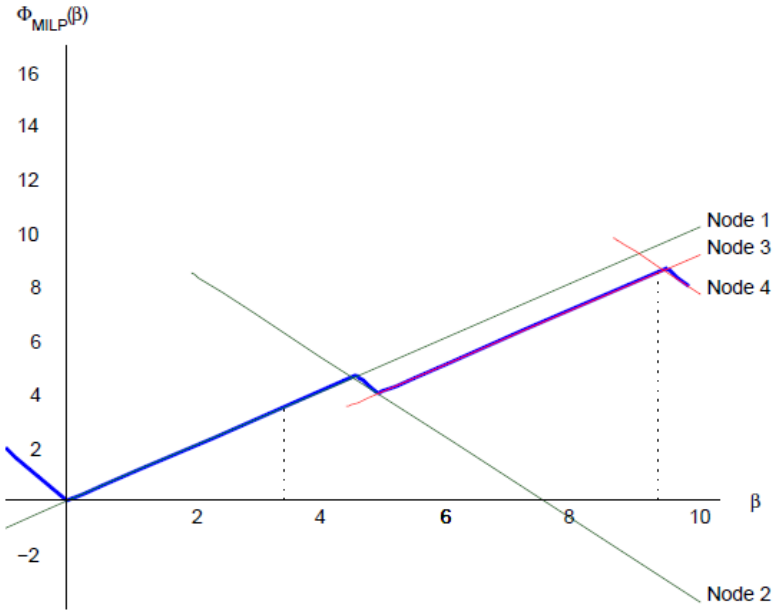
$$\min_{t \in T} \{\hat{\pi}^t b + \hat{\underline{\pi}}^t l^t + \hat{\bar{\pi}}^t u^t\}, \tag{9}$$

where $(\hat{\pi}^t, \hat{\underline{\pi}}^t, \hat{\bar{\pi}}^t)$ is an optimal solution to the dual (8).

Iterative Refinement

- The tree obtained from evaluating $\phi(b)$ yields a dual function strong at b.
- By solving for other right-hand sides, we obtain additional dual functions that can be aggregated.
- These additional solves can be done within the same tree, eventually yielding a single tree representing the entire function.





Tree Representation of the Value Function

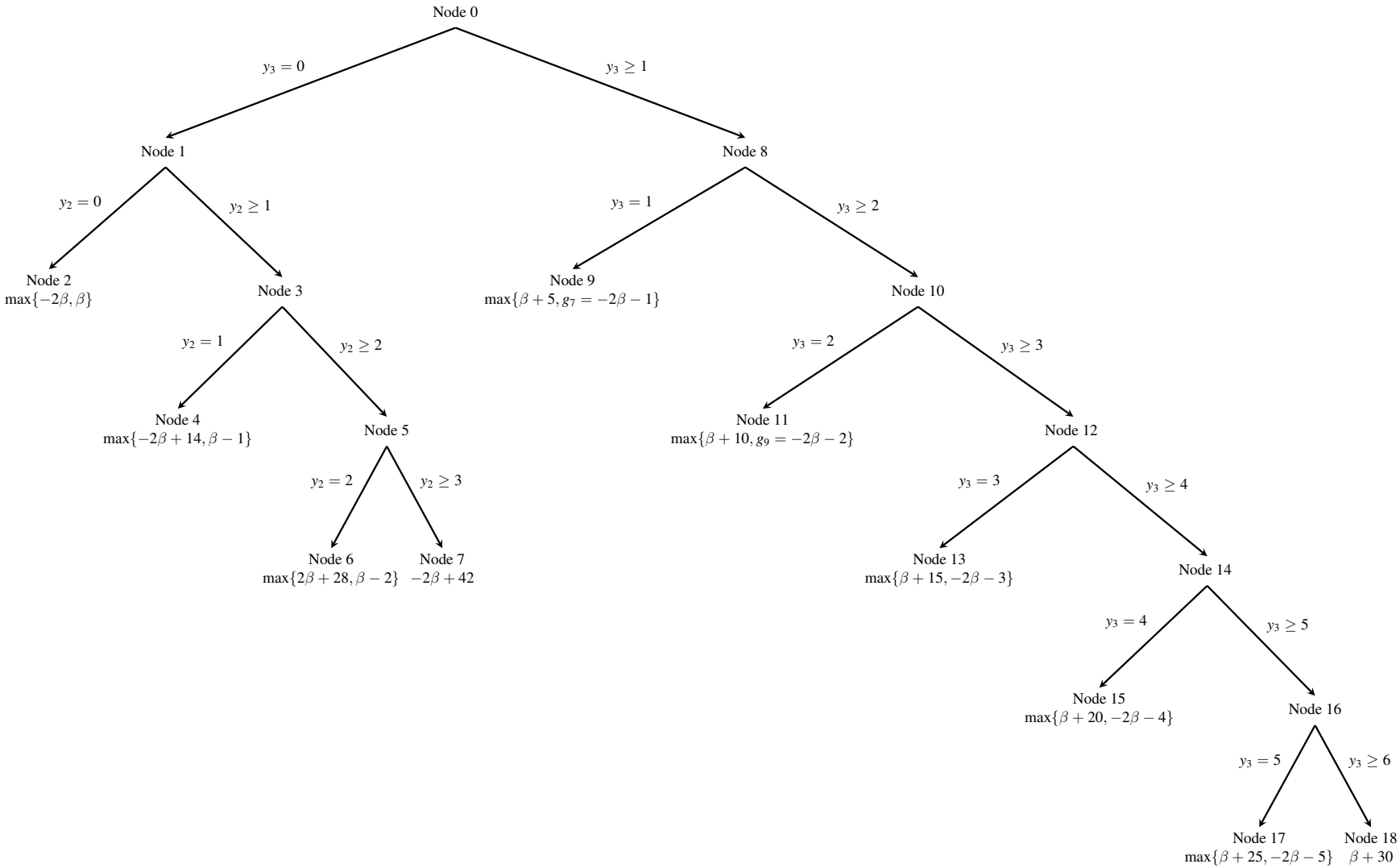
- Continuing the process, we eventually generate the entire value function.
- Consider the strengthened dual

$$\underline{\phi}^*(\beta) = \min_{t \in T} q_{I_t}^\top y_{I_t}^t + \phi_{N \setminus I_t}^t (\beta - W_{I_t} y_{I_t}^t), \tag{10}$$

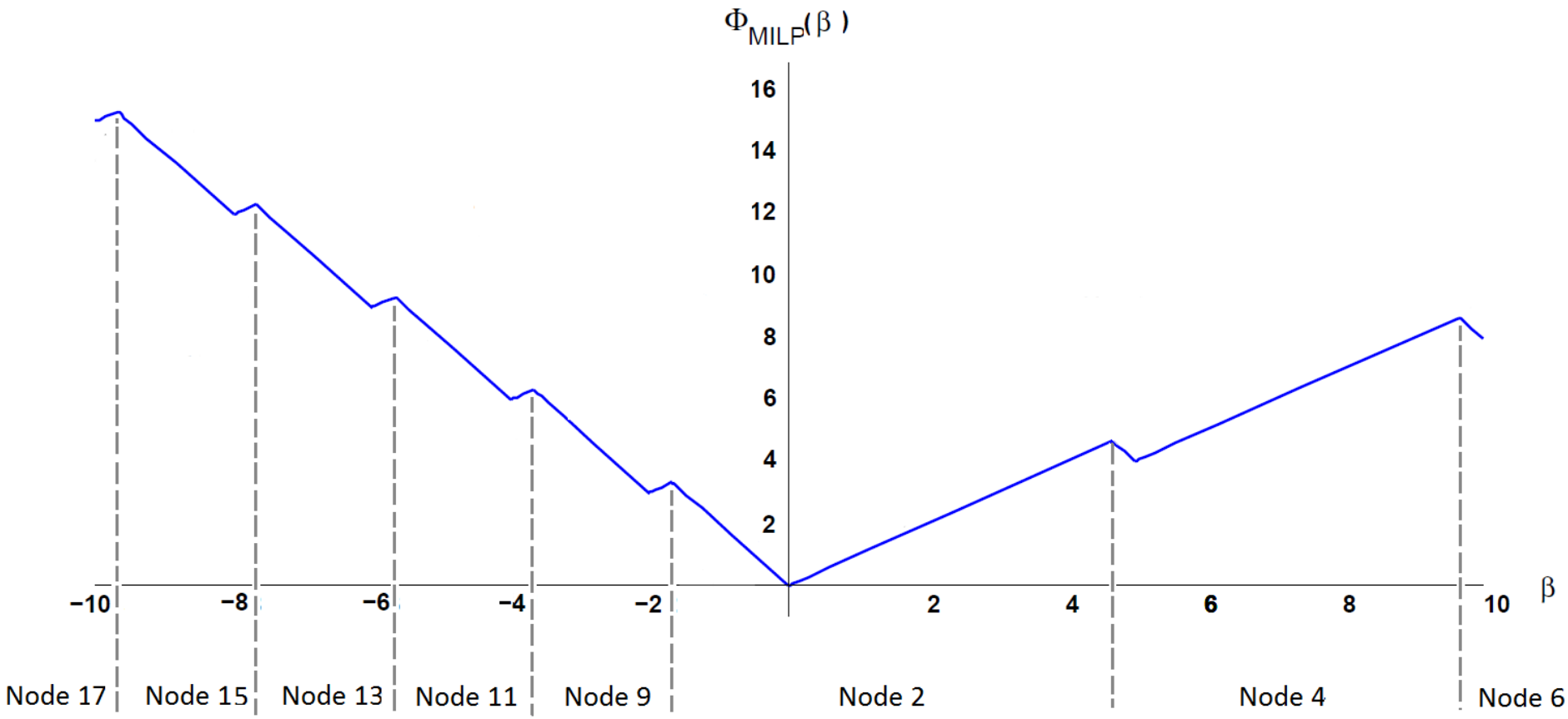
- I_t is the set of indices of fixed variables, $y_{I_t}^t$ are the values of the corresponding variables in node t.
- $\phi_{N \setminus I_t}^t$ is the value function of the linear program at node t including only the unfixed variables.

Theorem 2 *Under the assumption that* $\{\beta \in \mathbb{R}^{m_2} \mid \phi_I(\beta) < \infty\}$ *is finite, there exists a branch-and-bound tree with respect to which* $\underline{\phi}^* = \phi$.

Example of Value Function Tree



Correspondence of Nodes and Local Stability Regions



Back to Stochastic Programming: Lit Review

	First Stage			Second Stage			Stochasticity			
	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	**W**	**T**	**h**	**q**
(?)			*	*	*	*	*	*	*	
(?)	*		*	*		*	*	*	*	*
(?)	*	*	*		*	*		*	*	
(?)	*	*	*	*	*	*		*	*	*
(?)	*				*	*			*	
(?)			*	*		*	*	*	*	*
(?)	*	*	*		*	*	*		*	*
(?)			*	*		*		*	*	
(?)			*	*	*	*		*	*	
(?)	*		*	*		*	*	*	*	
(?)		*	*		*	*	*	*	*	*
(?)			*	*		*	*	*	*	*
(?)			*	*		*		*	*	*
(?)			*	*		*	*			*
(?)			*		*	*	*	*	*	*
(?)		*	*		*	*			*	
Current work	*	*	*	*	*	*		*	*	

Benders Master Formulation

$$\Gamma^k = \min\ c^\top x + \sum_{\omega\in\Omega} p_\omega \underline{z}_\omega$$

$$\begin{aligned}
\text{s.t. } & \underline{z}_\omega \le q_{t,\omega} && \forall t \in T, \omega \in \Omega \\
& \underline{z}_\omega \ge q_{t,\omega} - M_\omega(1-u_{t,\omega}) && \forall t \in T, \omega \in \Omega \\
& q_{t,\omega} \ge (h_\omega - T_\omega x)^\top \eta^i + \alpha^i && \forall i \in \mathcal{I}(t), t \in T, \omega \in \Omega \qquad \text{(MP)} \\
& \sum_{t\in T} u_{t,\omega} = 1 && \forall \omega \in \Omega \\
& u_{t,\omega} \in \mathbb{B}, && \forall t \in T, \omega \in \Omega \\
& x \in X.
\end{aligned}$$

Benders Master Variables

Notation:

- T indexes the leaf nodes of the branch-and-bound tree.
- $\mathcal{I}(t)$ are the indices of the affine functions describing the LP value function associated with node t.
- η^i, α^i represents the affine function indexed by $i \in \mathcal{I}(t)$ for node $t \in T$
- $q_{t,\omega} = \max_{\mathcal{I}(t)} b^\top \eta^i + \alpha^i$ represents the evaluation of the value function of node t in scenario ω.
- $u_{t,\omega}$ represents which of the nodes of the branch-and-bound tree is chosen as the maximum in scenario ω.
- $\underline{z}_\omega = \min_{t \in T} q_{t,\omega}$ represents the evaluation of the current value function approximation for scenario ω.
- x are the first-stage variables.

Overall Approach

- As in the classical algorithm, we alterante between solving the master problem and solving subproblems that update the current approximation (refines T).
- We maintain a single branch and bound tree and warm-start solution of the subproblems.
- This requires a solver capable of warm-starting the branch-and-bound process, as well as exporting the dual function.
- For this purpose, we use the SYMPHONY MILP solver, which has this capability.
- There are many possible variations, such as separate approximations for each scenario.

Example

Consider

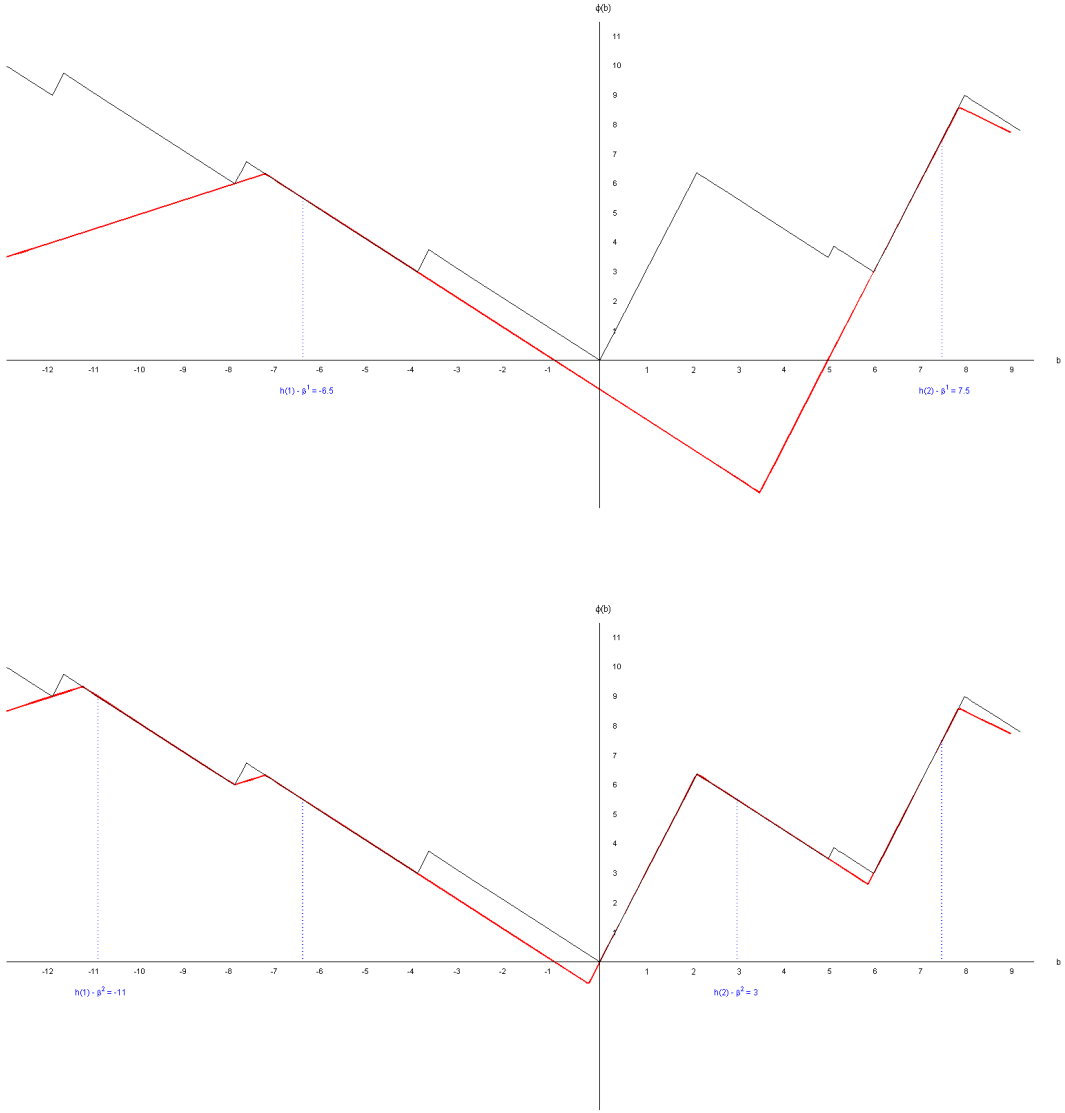
$$\begin{aligned} \min f(x) = \min \; & -3x_1 - 4x_2 + \sum_{s=1}^{2} 0.5Q(x,s) \\ \text{s.t. } & x_1 + x_2 \leq 5 \\ & x \in \mathbb{Z}_+ \end{aligned} \tag{11}$$

where

$$\begin{aligned} Q(x,s) = \min \; & 3y_1 + \frac{7}{2}y_2 + 3y_3 + 6y_4 + 7y_5 \\ \text{s.t. } & 6y_1 + 5y_2 - 4y_3 + 2y_4 - 7y_5 = h(s) - 2x_1 - \frac{1}{2}x_2 \\ & y_1, y_2, y_3 \in \mathbb{Z}_+, \; y_4, y_5 \in \mathbb{R}_+ \end{aligned} \tag{12}$$

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

Example



Solving the Master Problem

- A fundamental question is how to solve the master problem efficiently.
- It is a large integer program with many constraints, most of which will be redundant.
- It seems clear that we need to dynamically manage the LP relaxations.
- There are several options
  1. Put constraints in to a cut pool and add them only as they are violated.
  2. Add all constraints in the beginning, but aggressively remove slack ones.
- Our attempts have so far failed.
- Not having the full formulation makes branching less effective.
- We are currently investigating.

Example from the Literature

Consider

$$\begin{aligned} f(b) = \inf \; & -1.5x_1 - 4x_2 + \mathbb{E}[z(x,\omega)] \\ & \text{s.t. } x_1, x_2 \in \mathbb{B}^1, \end{aligned} \tag{13}$$

where

$$\begin{aligned} z(b) = \inf \; & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \\ & \text{s.t. } 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \leq h(x) - x_1 \\ & \quad 6y_1 + y_2 + 3y_3 + 2y_4 - R \leq h(x) - x_2 \\ & \quad y_i \in \mathbb{B}, i = 1, \ldots, 4, R \in \mathbb{Z}_+. \end{aligned} \tag{14}$$

- Here, the scenarios belong to the set $\{\omega_1, \omega_2\}$, each having a probability of 0.5.
- The distribution of the right-hand side is $(h_1(\omega_1), h_2(\omega_1)) = (5, 2)$ and $(h_1(\omega_2), h_2(\omega_2)) = (10, 3)$.

Example cont.

- To construct larger instances, we allow the right hand sides of the two constraints to be in the set $\{5, 5+\Delta, 5+2\Delta, \ldots, 15\} \times \{5, 5+\Delta, 5+2\Delta, \ldots, 15\}$ to generate $4, 9, 36, 121$ scenarios with $\Delta \in \{1, 2, 5, 10\}$.
- We further extend the set by changing 15 to 20 to generate 225 scenarios with $\Delta = 1$.
- We have the size of the deterministic equivalent of the resulting problems as follows

No. Scen.	4	9	36	121	225
No. Vars	24	47	182	607	1127
No. Const	8	18	72	242	450

Table: Size of the deterministic equivalent of the test instances

Example cont.

Static and Dynamic Cut Generation

Instance	Obj	No. Iter.	Stats. Size of Master	Stat. Time	Dyn. Size of Master	Dyn. Time
4-Scen	-63.50	3	1 (27, 47)	0.16	1 (27, 47)	0.16
9-Scen	-65.67	3	1 (71, 127)	0.32	1 (71, 127)	0.3
36-Scen	-66.83	3	17 (301, 552)	3.21	3 (277, 505)	2.43
121-Scen	-67.17	4	55 (1022,1891)	17.25	3 (986, 1807)	12.81
225-Scen	-79.66	4	5 (1883, 3465)	28.29	7 (1812, 3314)	33.00

Table: Performance of the algorithm applied to test problems

- The third and fifth columns show the size of the master problem in the final iteration in the form of number of tree nodes (number of variables, number of constraints) respectively with static and dynamic cut generation.
- The fourth and sixth columns show the solution time in seconds respectively for static and dynamic cut generation.

SSLP Instances

We apply the algorithm to the SSLP test instances from SIPLIB. The instances have the following properties. The last column shows the deterministic equivalent solution time in seconds.

Instance	DEP			2nd Stage			Time (s)	% Gap
	cons	bin	int	cons	bin	int		
sslp-5-25(25)	751	3130	125	30	130	5	3.17	0.0
sslp-5-25(50)	1501	6255	250	30	130	5	4.22	0.0
sslp-5-25(100)	3001	12505	500	30	130	5	14.34	0.0

Table: The deterministic equivalent of SSLP instances

where “DEP” and “2nd Stage” correspond to the deterministic equivalent and the second stage problems and “cons”, “bins” and “int” respectively represent the number of constraints, binary variables and general integer variables in the corresponding problem.

SSLP Instances

Instance	Static Cut				Dynamic Cut			
	Iteration	Size	Time (s)	% Gap	Iteration	Size	Time (s)	%Gap
sslp-5-25(25)	18	431 (1040, 2016)	89.92	0.0	47	31 (2493, 3639)	39.57	0.0
sslp-5-25(50)	18	26 (1484, 2948)	123.22	0.0	22	9 (2443, 3489)	242.46	0.0
sslp-5-25(100)	18	899 (3703, 7192)	835.76	0.0	12	1533 (2608, 3604)	398.59	0.0

Table: Generalized Benders' algorithm applied to SSLP instances

- The results are generated using the MILP solver SYMPHONY version 5.5.
- A time limit of one hour was enforced on the solution time.
- The "Gap%" column refers to the relative gap between the upper bound and lower bound of the Generalized Benders' algorithm.
- The tests are performed on a a Linux (Debian 6.0.9) operating system running 16 AMD processors on a 800 MHz speed and 31 GB RAM, compiled with g++.

Outline

Conclusions and Future Work

- We have developed an algorithm for the two-stage problem with general mixed integer variables in both stages.
- The algorithm uses the Benders' framework with B&B dual functions as the optimality cuts.
- We need to keep the size of approximations small. Currently, we have implemented a static cut generation as well as a dynamic cut generation by allowing the constraints to be added to the master problem dynamically as needed.
- There are still many questions to be answered about how to make all of this as efficient as possible.
- We are working on making the dynamic cut generation scheme more efficient by incorporating cuts during strong branching. This is work under progress.