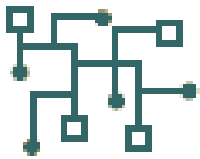
Computation Experience with a Benders' Algorithm for Two-Stage Stochastic Discrete Optimization Problems

Ted Ralphs[1]
Joint work with Menal Güzelsoy[2] and Anahita Hassanzadeh[3]

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

INFORMS Annual Meeting, Philadelphia, PA, November 1, 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

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

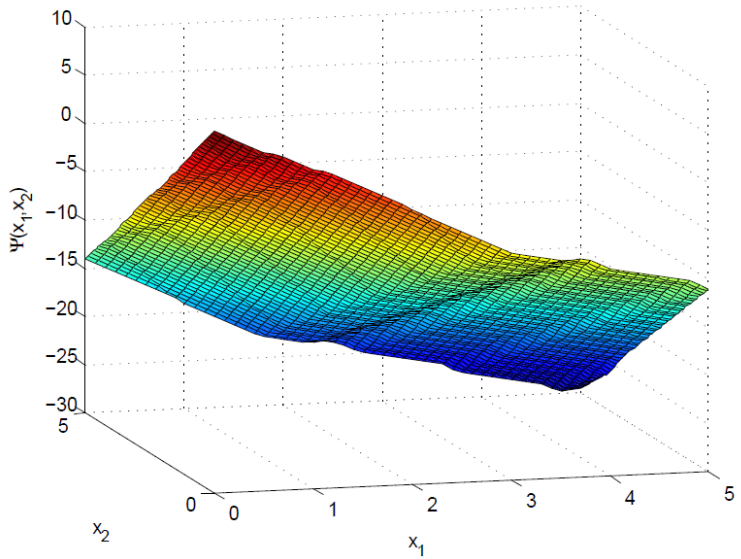
  is the *second-stage feasible region* with respect to a given right-hand side β and $Y = \mathbb{Z}_+^{r_2} \times \mathbb{R}_+^{n_2 - r_2}$.
- The second-stage problem is parameterized on the unknown value β of the right-hand side.
- This value is determined jointly by the realized value of ω and the 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 $x \in X$.

What it Looks Like

Example 1

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

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



Quick Lit Review

	First Stage			Second Stage			Stochasticity			
	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	$\mathbb{R}$	$\mathbb{Z}$	$\mathbb{B}$	**W**	**T**	**h**	**q**
[Laporte and Louveaux, 1993]			*	*	*	*	*	*	*	
[Carøe and Tind, 1997]	*		*	*		*	*	*	*	*
[Carøe and Tind, 1998]	*	*	*		*	*		*	*	
[Carøe and Schultz, 1998]	*	*	*	*	*	*		*	*	*
[Schultz et al., 1998]	*				*	*			*	
[Sherali and Fraticelli, 2002]			*	*		*	*	*	*	*
[Ahmed et al., 2004]	*	*	*		*	*	*		*	*
[Sen and Higle, 2005]			*	*		*		*	*	
[Sen and Sherali, 2006]			*	*	*	*		*	*	
[Sherali and Zhu, 2006]	*		*	*		*	*	*	*	
[Kong et al., 2006]		*	*		*	*	*	*	*	*
[Sherali and Smith, 2009]			*	*		*	*	*	*	*
[Yuan and Sen, 2009]			*	*		*		*	*	*
[Ntaimo, 2010]			*	*		*	*			*
[Gade et al., 2012]			*		*	*	*	*	*	*
[Trapp et al., 2013]		*	*		*	*			*	
Current work	*	*	*	*	*	*		*	*	

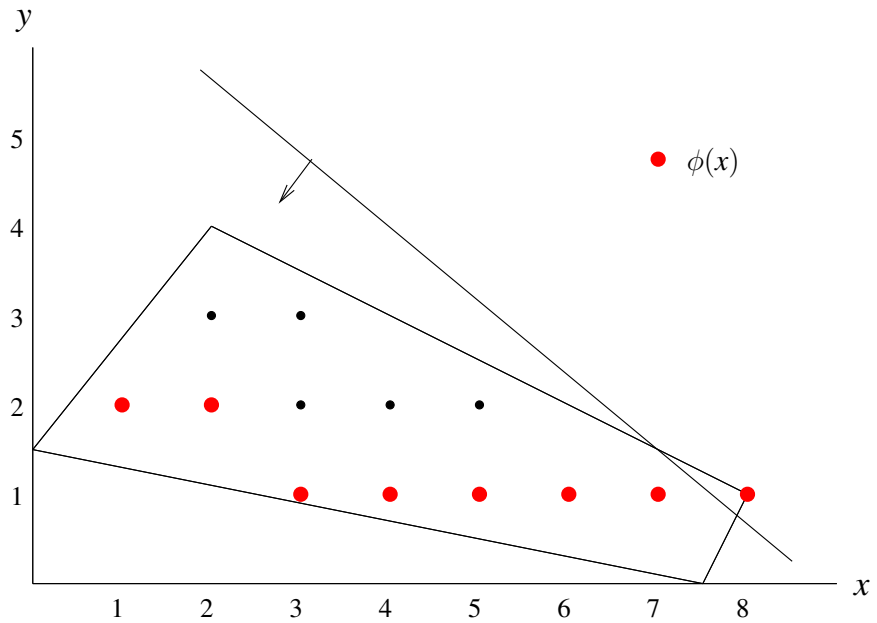
Outline

Generalized Benders' in a Nutshell

$$z_{IP} = \min_{(x,y)\in X\times Y} \left\{ c'x + c''y \;\middle|\; A'x + A''y \geq b \right\}$$
$$= \min_{x\in X} \left\{ c'x + \phi(b - A'x) \right\},$$

where

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



Basic Strategy:

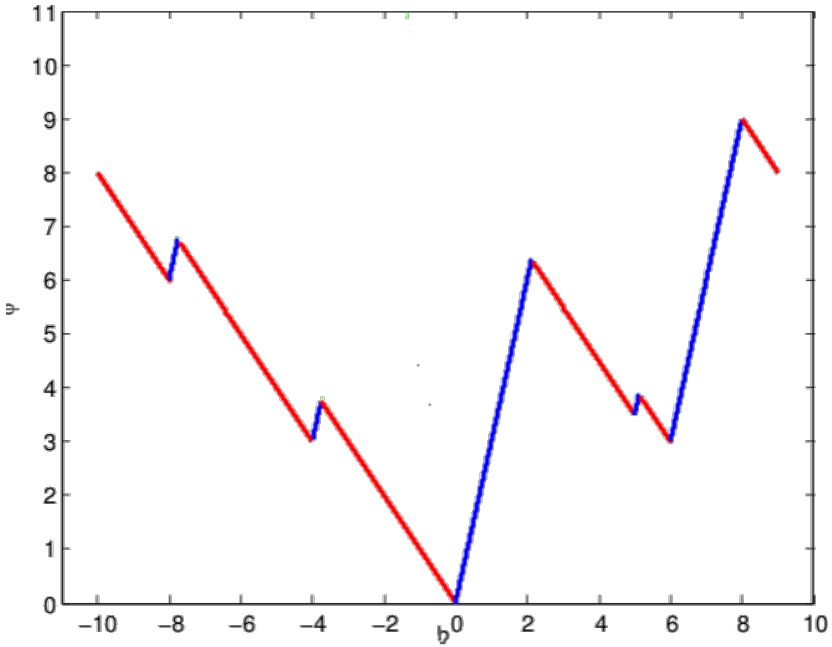
- We have $X = \mathbb{Z}^{r'} \times \mathbb{R}^{n'-r'}$ and $X = \mathbb{Z}^{r''} \times \mathbb{R}^{n''-r''}$.
- Here, ϕ is the value function of a *mixed integer linear program*.
- Benders' algorithm is to solve a relaxation obtained by replacing ϕ with a *lower approximation* that is then iteratively improved.

MILP Value Function

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

Example 2

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



Benders' Master Problem

Benders' master problem is the aforementioned relaxation.

$$\begin{aligned} \min \quad & c'x + w \\ \text{subject to} \quad & A'x \leq b' \\ & w \geq \underline{\phi}(b' - A'x) \\ & x \in X \end{aligned}$$

- $\underline{\phi}$ is a lower approximation of the value function of the second stage problem.
- We have projected out the second-stage variables.
- if $Y = \mathbb{R}^{n''}$, ϕ is a piece-wise linear convex function.
- The classical Benders' algorithm approximates ϕ as the maximum of linear dual functions, which are the dual solutions to a linear program.
- Our method follows the same outline, but our "dual functions" will need to be non-linear.
- The remainder of the talk is about our computational framework for constructing and representing these functions.

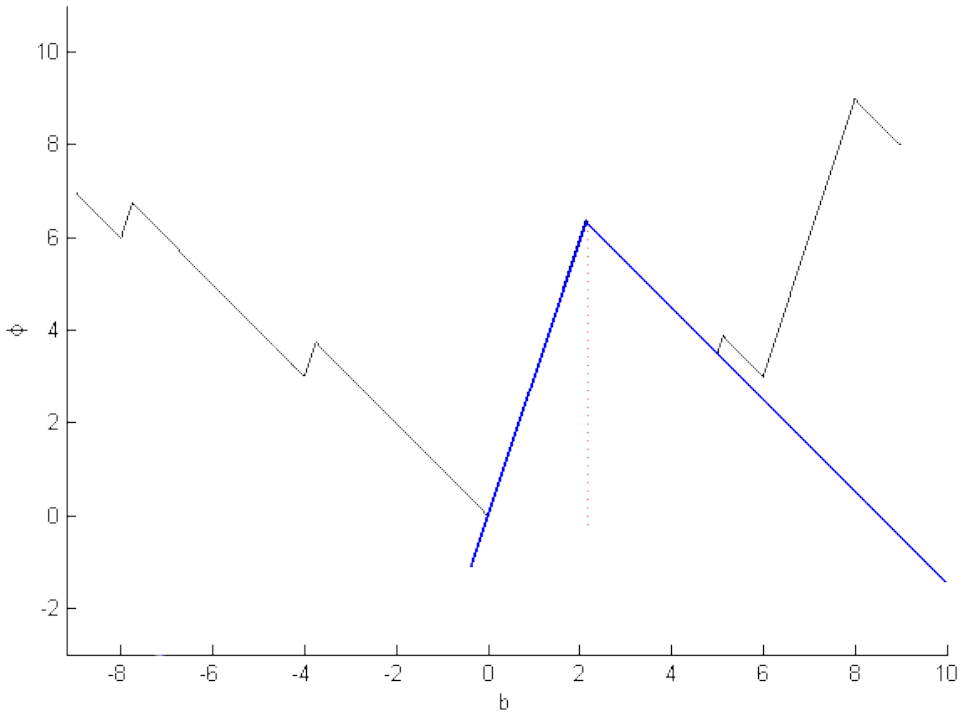
Dual Functions for Integer Programs

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 (strong) problem is

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



These functions are the discrete counterpart to the solutions to the LP dual.

Outline

Dual Functions from Branch-and-Bound [Wolsey, 1981]

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

$$\begin{aligned} \phi^t(b) = \min\ & c^\top x \\ \text{s.t. } & Ax = b, \\ & l^t \leq x \leq u^t, x \geq 0 \end{aligned} \tag{5}$$

The dual at node t:

$$\begin{aligned} \phi^t(b) = \max\ & \{\pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t\} \\ \text{s.t. } & \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ & \underline{\pi} \geq 0, \bar{\pi} \leq 0 \end{aligned} \tag{6}$$

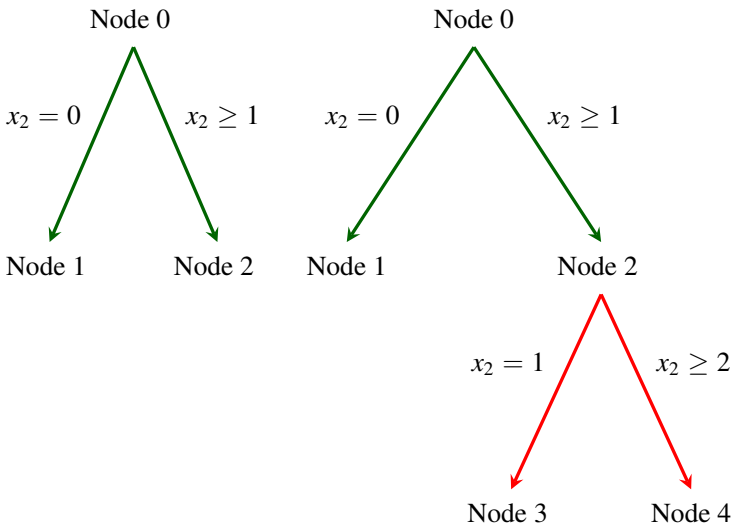
We obtain the following strong dual function:

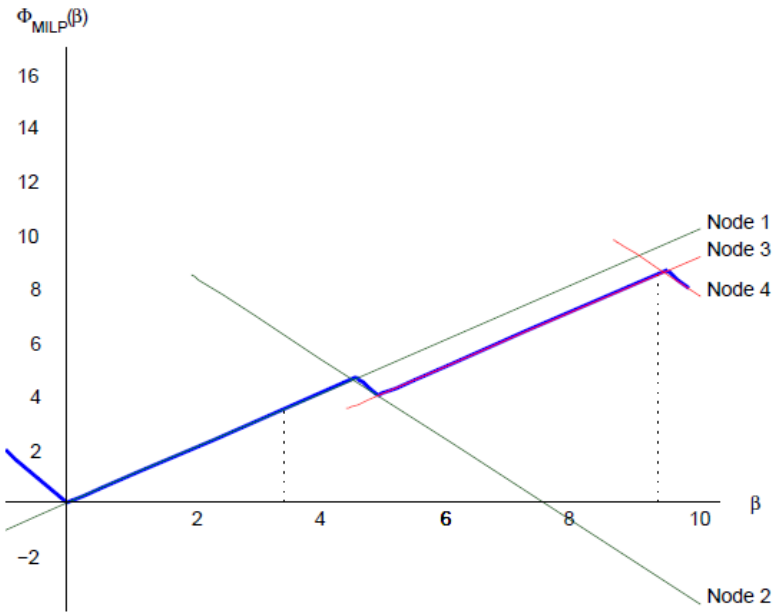
$$\min_{t \in T} \{\hat{\pi}^t b + \underline{\hat{\pi}}^t l^t + \hat{\bar{\pi}}^t u^t\}, \tag{7}$$

where $(\hat{\pi}^t, \underline{\hat{\pi}}^t, \hat{\bar{\pi}}^t)$ is either an optimal solution to the dual (6) (if one exists) or an appropriately chosen dual feasible solution if (6) is unbounded.

Iterative Refinement and Warm Starting

- 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

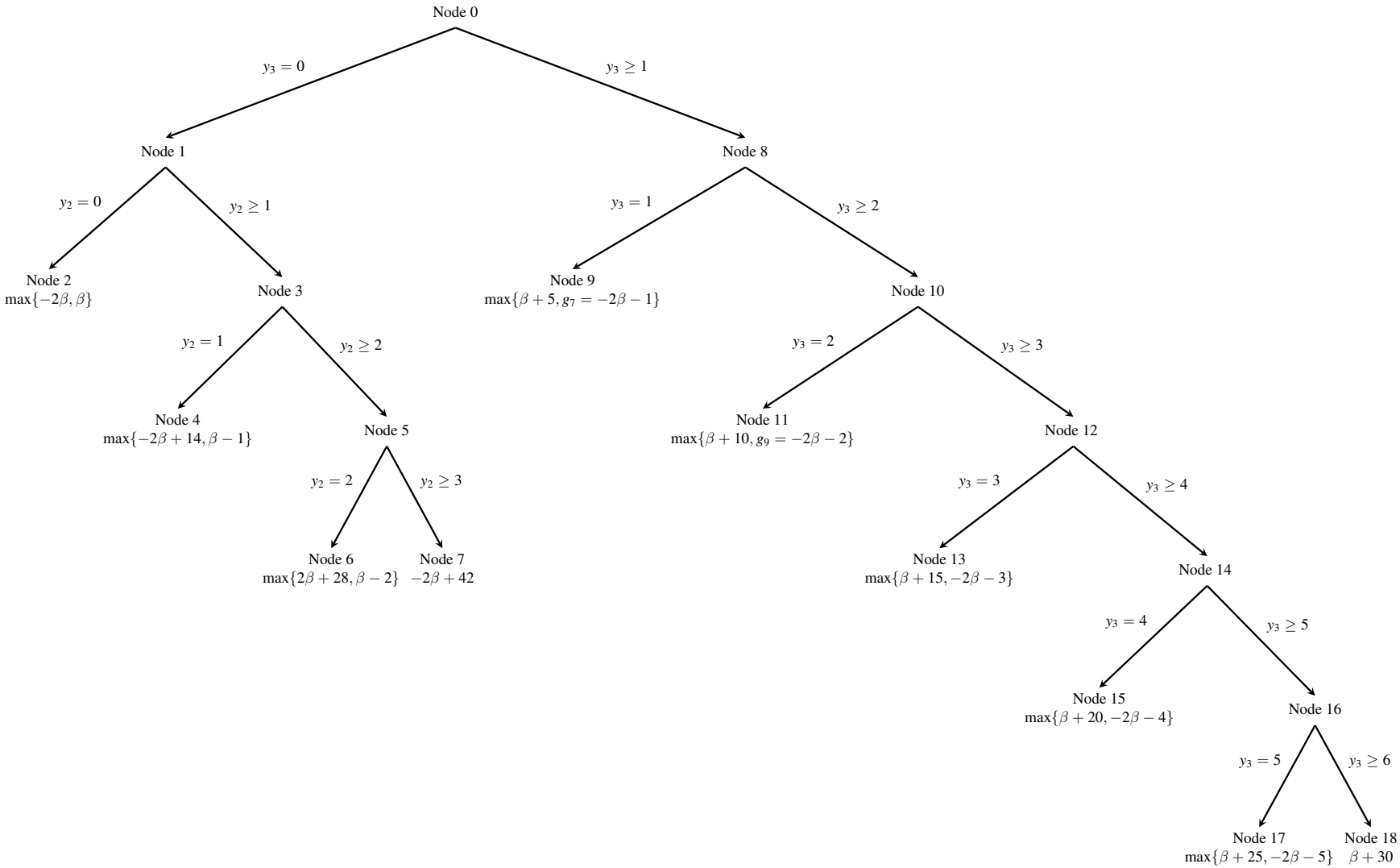
- Continuing the process, we eventually generate the entire value function.
- Consider the strengthened dual

$$\underline{\phi}^*(\beta) = \min_{t \in T} q_{I_t}^\top y_{I_t}^t + \phi_{N \setminus I_t}^t (\beta - W_{I_t} y_{I_t}^t), \tag{8}$$

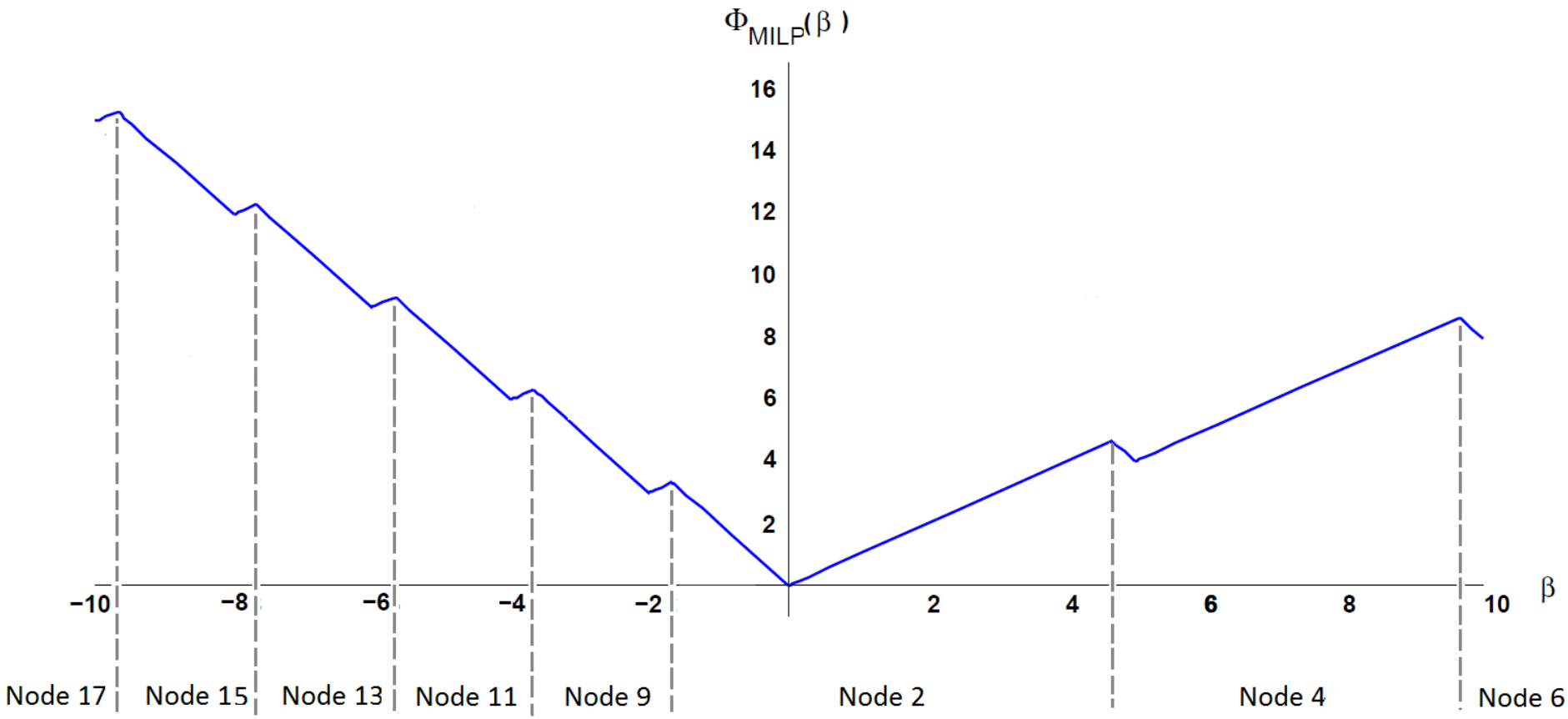
- I_t is the set of indices of fixed variables, $y_{I_t}^t$ are the values of the corresponding variables in node t.
- $\phi_{N \setminus I_t}^t$ is the value function of the linear program at node t including only the unfixed variables.

Theorem 1 *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



Generalized 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.

Generalized Benders' Master: Conceptual Formulation

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

- In this formulation, we have a single approximation from a single branch-and-bound tree.
- In theory, this is enough for a convergent algorithm, but in practice, we may want something more complex.

Implementation Overview

Basic Scheme

1. Solve master problem to obtain new first stage solution and lower bound.
2. Solve scenario subproblems to update value function approximation and obtain new upper bound.
3. Terminate when upper bound equals lower bound.

- As in the classical algorithm, we alternate between solving the master problem and subproblems that update the current approximation.
- The approximation may come from a single tree or a set of trees (more shortly).
- We require a solver capable of exporting the dual function resulting from the solve process.
- Ideally, the solver should also be capable of iterative refinement and warm-starting, though this is not necessary.
- The SYMPHONY MILP solver has this capability.

Computational Challenges: Maintaining Dual Functions

- For the algorithm to be convergent in practice, we need to retain the dual solution to the LP relaxation of any node that was a leaf node in some iteration.
- We have several options for generating dual functions.

Options for Updating Dual Functions

- Single dual function: One continuously refined function across all scenarios (requires warm-starting all computations).
- Scenario dual functions: Separate dual functions for each scenario.

- In the latter case, we still have a choice between solving each scenario subproblem from scratch or warm-starting from the tree of the previous iteration.

Computational Challenges: Solving Subproblems

- If we warm-start the computation, we still need to maintain the dual solutions of nodes that are further refined as part of the dual function.
- This include retaining all nodes that are pruned for any reason.
- This is mostly a matter of bookkeeping, but it means that the LP solver underlying the subproblem solver must be very robust.
- Important! *All* dual information produced by the LP solver must be accurate!
- Because of the rigidity of this scheme, it is not easy to integrate advanced solution techniques
  - Pre-processing,
  - Cut generation, and
  - Reduced cost bound tightening.
- This makes solution of the subproblem less efficient, but this is not really the bottleneck anyway.

Computational Challenges: Master Solver

- The key to the algorithm is solving the master problem effectively.
- To get the strongest possible bound, we would ideally retain every (non-dominated) dual solution generated at any node during any subproblem solve.
- Unfortunately, this may result in severe bloating.
- We have several options for dealing with this.

Options for Controlling Master Problem Size

- Retain a minimal set of dual solutions (discard ones currently unneeded).
- Use lazy cut generation.

- Aside from the size of the formulation, the "big-M" constraints required to model the functions are problematic.
- Indicator constraints are a possible remedy for this.
- So far, our efforts at enhancing solvability of the master problem haven't had as much impact as hoped.

A Glimmer of Hope: Parallelism

- One aspect of this scheme that is advantageous is that it is easy to parallelize.

Parallelization Options

- Scenario subproblems can be solved independently in parallel.
- Master problem may also be solved in parallel.

- In practice, this turns out to be effective.

Quick Example

Consider

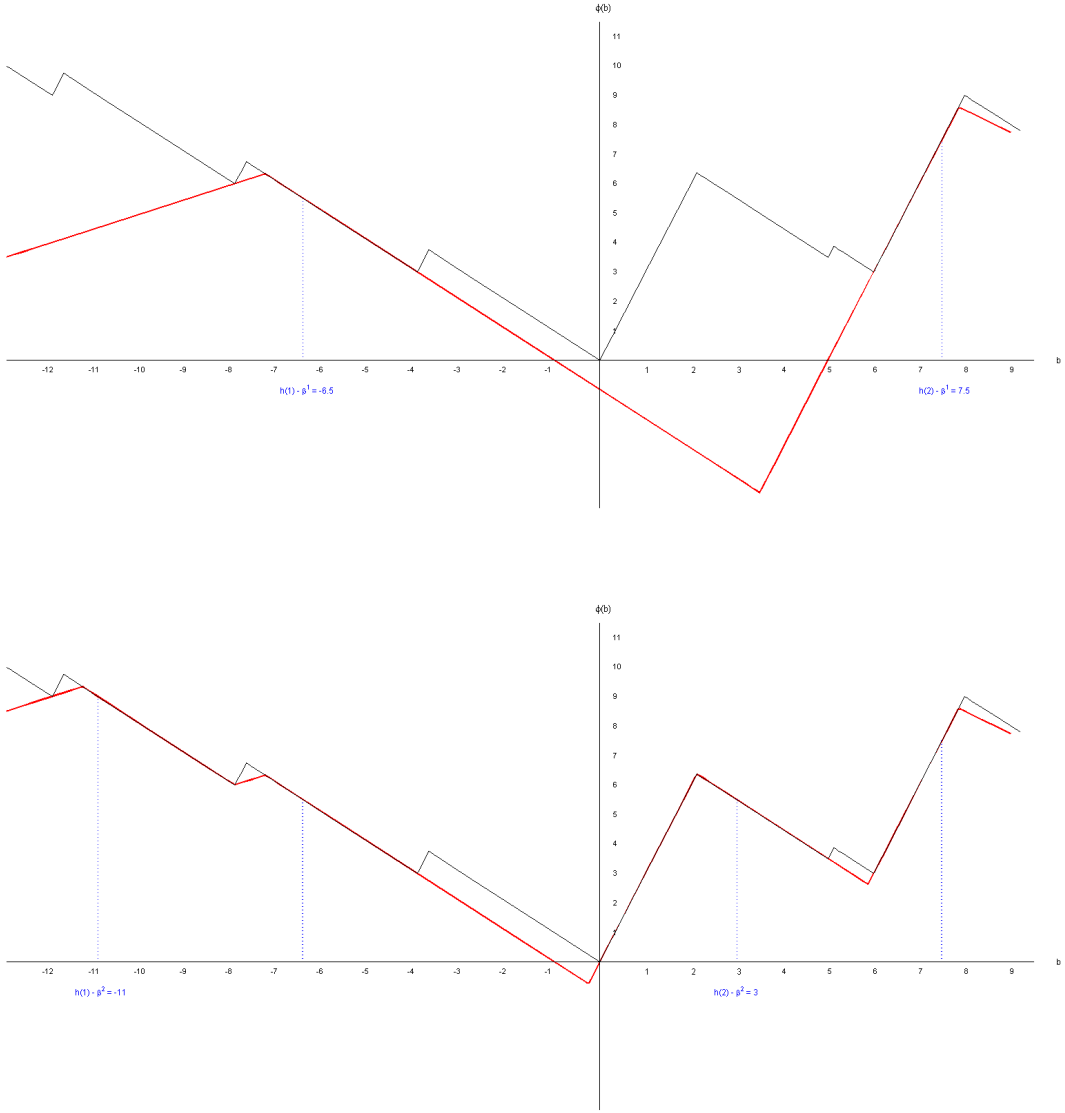
$$\begin{aligned} \min f(x) = \min \; & -3x_1 - 4x_2 + \sum_{s=1}^{2} 0.5Q(x,s) \\ \text{s.t. } & x_1 + x_2 \leq 5 \\ & x \in \mathbb{Z}_+ \end{aligned} \tag{9}$$

where

$$\begin{aligned} Q(x,s) = \min \; & 3y_1 + \frac{7}{2}y_2 + 3y_3 + 6y_4 + 7y_5 \\ \text{s.t. } & 6y_1 + 5y_2 - 4y_3 + 2y_4 - 7y_5 = h(s) - 2x_1 - \frac{1}{2}x_2 \\ & y_1, y_2, y_3 \in \mathbb{Z}_+, \; y_4, y_5 \in \mathbb{R}_+ \end{aligned} \tag{10}$$

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

Example



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
sslp-5-25(50)	1501	6255	250	30	130	5	4.22	0
sslp-5-25(100)	3001	12505	500	30	130	5	14.34	0
sslp-10-50(50)	3001	25010	500	60	510	10	3600+	70
sslp-10-50(100)	6001	50010	1000	60	510	10	3600+	15
sslp-15-45(5)	301	3390	75	60	690	15	3600+	1
sslp-15-45(10)	601	6765	150	60	690	15	1088.69	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	Iteration	Size	Time (s)	%Gap
sslp-5-25(25)	16	(2493, 3639)	182.58	0
sslp-5-25(50)	18	(789, 1821)	12.16	0
sslp-5-25(100)	18	(1322, 3410)	27.91	0
sslp-10-50(50)	8	(40K, 71K)	-	23
sslp-10-50(100)	8	(74K, 125K)	-	9
sslp-15-45(5)	7	(29K, 56K)	-	99
sslp-15-45(10)	26	(17K, 29K)	-	52

Table: Generalized Benders' algorithm applied to SSLP instances

- The results are generated using the MILP solver SYMPHONY version WS revision 2522 and CPLEX 12.5.
- The tests were performed on a 16-core Linux box with 800 MHz AMD processors and 31 GB RAM compiled with g++.
- SSLP 10 and 15 runs were in parallel using 16 cores, other runs sequential.
- The "Gap%" column refers to the relative gap between the upper bound and lower bound of the Generalized Benders' algorithm.

Getting this all to work well is a PITA!

Analysis

- So far, we have been unable to beat straightforward solution of the deterministic equivalent.
- We've tried some obvious things that haven't really worked (yet):

Algorithm Fails

- Lazy constraint generation in solving master
- Indicator constraints in solving master
- Warm-starting of subproblem solves

- On the other hand, parallelization does seem to be a win.
- The master has a lot of structure and we still conjecture that a custom-tuned algorithm should be able to compete with the DEP.

Where Do We Go From Here?

- The Benders approach has not been very successful at solving large SMILPs in its pure form.
- By merging ideas from other algorithm frameworks (convexification), we think we can improve the algorithm substantially.
- However, the real goal of this work has been to solve even more general problems that have no deterministic equivalent.
- With very little tweaking, this general approach can work for bilevel programs.
- We are expecting better results (comparatively) in that arena.

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.
- Solving the master problem is the bottleneck.
- There are still many questions to be answered about how to make all of this as efficient as possible.
- We will soon apply this framework the more general setting of bilevel programming.

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. A cutting-plane approach to mixed 0-1 stochastic integer programs. *European Journal of Operational Research*, 101(2):306–316, 1997.

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

Dinakar Gade, Simge Küçükyavuz, and Suvrajeet Sen. Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, pages 1–26, 2012.

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.

References II

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.

Lewis Ntaimo. Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations research*, 58(1):229–243, 2010.

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.

S. Sen and H.D. Sherali. Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223, 2006. ISSN 0025-5610.

References III

Hanif D Sherali and J Cole Smith. Two-stage stochastic hierarchical multiple risk problems: models and algorithms. *Mathematical programming*, 120(2):403–427, 2009.

H.D. Sherali and B.M.P. Fraticelli. A modification of Benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22(1):319–342, 2002.

H.D. Sherali and X. Zhu. On solving discrete two-stage stochastic programs having mixed-integer first-and second-stage variables. *Mathematical Programming*, 108 (2):597–616, 2006.

Andrew C Trapp, Oleg A Prokopyev, and Andrew J Schaefer. On a level-set characterization of the value function of an integer program and its application to stochastic programming. *Operations Research*, 61(2):498–511, 2013.

L.A. Wolsey. Integer programming duality: Price functions and sensitivity analysis. *Mathematical Programming*, 20(1):173–195, 1981. ISSN 0025-5610.

Yang Yuan and Suvrajeet Sen. Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, 21(3):480–487, 2009.