Stochastic Integer Programming

Jim Luedtke

Dept. of Industrial and Systems Engineering
Wisconsin Institute for Discovery
University of Wisconsin-Madison, USA
jim.luedtke@wisc.edu

Centre de Reserches Mathématiques
Workshop on Optimization Under Uncertainty
September 27, 2021

Example: Generation/Transmission Capacity Expansion

Given candidate generators (sizes/locations) and transmission lines

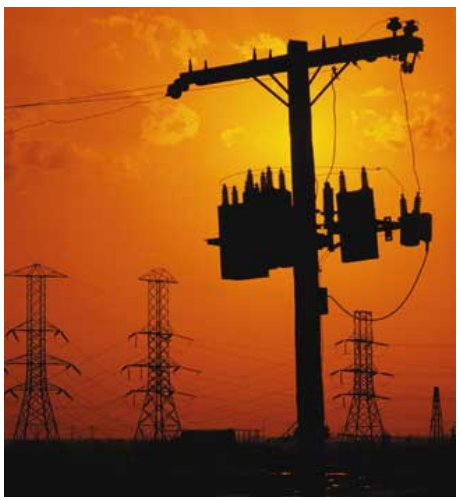
- Which to open to minimize capacity expansion and operating costs to meet demands?

Discrete decisions:

- Select generator/transmission line or not

Uncertainty:

- Future demands by location, weather/renewable yield



Example: Deterministic Model

Simplified generation expansion only model (ignoring transmission):

- Candidate generators: G, Demand locations: J
- Fixed cost f_i, capacity C_i for generator $i \in G$
- Load d_j: for $j \in J$
- x_i: Binary choice for generator opening decisions
- y_{ij}: Amount of demand at location j met from facility i
- z_j: Load shed at location j

$$\min \sum_{i \in G} f_i x_i + \sum_{i \in G} \sum_{j \in J} c_{ij} y_{ij} + \sum_{j \in J} p_j z_j$$

$$\text{s.t.} \quad \sum_{i \in G} y_{ij} + z_j = d_j \quad \forall j \in J$$

$$\sum_{j \in J} y_{ij} \leq C_i x_i \quad \forall i \in G$$

$$x_i \in \{0, 1\}, y_{ij} \geq 0 \qquad \forall i \in G,\ j \in J$$

Two-stage Stochastic Optimization

What if capacity/demands are random?

Classic two-stage framework

1. Choose here-and-now decisions (aka first-stage)

$\Rightarrow$ Observe random variables

2. Make recourse decisions (in response to observed random variables, aka second-stage)

Goal: Choose current decisions to minimize immediate cost plus expected value of cost of "best response" decisions

- Can also consider risk-averse objectives, but we'll ignore that today

Must determine which decisions are here-and-now and which are recourse

Example: Stochastic model

- Load D_j: Random load for $j \in J$
- Capacity C_i: Random capacity for $i \in G$
- x_i: Binary choice for generator open decisions (here-and-now)
- y_{ij}: Amount of customer j demand met from facility i (recourse)
- z_j: Amount of customer j demand met that is not met (recourse)

$$\min_{x \in \{0,1\}^G} \sum_{i \in G} f_i x_i + \mathbb{E}[Q(x, D, C)]$$

$$Q(x, D, C) := \min_{y,z \geq 0} \sum_{i \in G} \sum_{j \in J} c_{ij} y_{ij} + \sum_{j \in J} p_j z_j$$

$$\text{s.t.} \sum_{i \in G} y_{ij} + z_j \geq D_j \quad \forall j \in J$$

$$\sum_{j \in J} y_{ij} \leq C_i x_i \quad \forall i \in G$$

General Model

Two-Stage Stochastic Mixed Integer Program (SMIP)

$$\begin{aligned} \min\ & c^\top x + \mathbb{E}[Q(x,\xi)] \\ \text{s.t.}\ & Ax \geq b \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

where $\xi = (q, h, T, W)$ and

$$\begin{aligned} Q(x,\xi) = \min\ & q^\top y \\ \text{s.t.}\ & Wy = h - Tx \\ & y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+ \end{aligned}$$

- x: first-stage decision variables
- y: second-stage decision variables
- Sometimes assume n_1, p_1, n_2, or p_2 are zero

Other Example Applications

Stochastic unit commitment

[Takriti et al., 1996, Carøe and Schultz, 1998, Cheung et al., 2015]

- Random electricity loads and wind/solar production
- First-stage decisions: units to commit and when (binary)
- Second-stage decisions: production amounts, line switching (binary or continuous)

Wildfire initial response planning [Ntaimo et al., 2012]

- Random fire locations/sizes
- First-stage decisions: where to pre-position equipment (integer)
- Second-stage decisions: equipment deployment in response to fires

Stochastic vehicle routing (a priori routing) [Gendreau et al., 1996]

- Random customer demands
- First-stage decisions: planned vehicle routes (binary)
- Second-stage decisions: recourse actions when capacity violated (binary or continuous)

What makes SMIP hard?

Stochastic integer programming combines challenges from Stochastic programming and Integer Programming

Stochastic programming challenges

- Evaluating expectation
- Huge size even with finite scenario approximation

Integer programming challenges

- Huge number of discrete options
- Weak relaxations can lead to huge enumeration trees

Tutorial Overview

- Sample Average Approximation (SAA) for approximating expected value (brief!)
- How to solve the SAA approximation
  - Deterministic equivalent form
  - Integer programming methdology background
  - Benders cut based methods
  - Lagrangian dual based methods
  - Recent hybrid approaches

Many interesting topics we won't cover, e.g.,

- Multistage
- Risk-averse
- Chance constraints
- Distributionally robust

First Challenge: Evaluating Expected Value

Two-Stage Stochastic Mixed Integer Program (SMIP)

$$\begin{aligned} \min\ & c^\top x + \mathbb{E}[Q(x,\xi)] \\ \text{s.t.}\ & Ax \geq b \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

High-dimensional ξ:

- Evaluating $\mathbb{E}[Q(x,\xi)]$ challenging even for a single fixed x
- Need approximation at many values of x

Simple idea: Sample average approximation

- Let ξ^s, $s = 1, \ldots, S$ be a Monte Carlo sample of ξ
- Use sample average to approximate expected value

$$\mathbb{E}[Q(x,\xi)] \approx \frac{1}{S}\sum_{s=1}^{S} Q(x,\xi^s)$$

Sample average approximation $\Rightarrow$ Deterministic, but very large-scale optimization model

Approximating Expected Value

Key question: How many scenarios required for "good approximation"?

- Significant research into this
  [Mak et al., 1999, Shapiro and Homem-de-Mello, 2000, Ahmed and Shapiro, 2002, Kleywegt et al., 2002]
- Roughly: Achieving ϵ accurate solution requires $O(n_1/\epsilon^2)$ scenarios
- Good news: Surprisingly, required number grows "mildly" with number of decision variables and random variables
- Bad news: Required number grows fast with desired accuracy

Conclusion

SAA enables solving SMIP problems to "modest accuracy"

Opinion: High accuracy is pointless anyway given likely errors in distribution estimation, model approximation

Next challenge: How to solve the very large-scale optimization model defined by sample average approximation?

SAA $\Rightarrow$ Finite Scenario SMIP

Scenarios: $\xi^s = (q_s, h_s, T_s, W_s)$, for $s = 1, \ldots, S$

$$\begin{aligned} \min\ & c^\top x + \frac{1}{S}\sum_{s=1}^{S} Q(x, \xi^s) \\ \text{s.t.}\ & Ax \geq b \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

where

$$\begin{aligned} Q(x, \xi^s) = \min\ & q_s^\top y \\ \text{s.t.}\ & W_s y = h_s - T_s x \\ & y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+ \end{aligned}$$

SMIP $\equiv$ Large-scale structured MIP

First option for solving a SMIP with finite scenarios

Extensive form (deterministic equivalent) of an SMIP

$$
\begin{aligned}
\min\ & c^\top x + \frac{1}{S}\sum_{s=1}^{S} (q_s)^\top y_s \\
\text{s.t. } & Ax \geq b \\
& T_s x + W_s y_s = h_s, \quad s = 1, \ldots, S \\
& x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \\
& y_s \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+, \quad s = 1, \ldots, S
\end{aligned}
$$

Give this to your favorite MIP solver: Gurobi, CPLEX, Xpress, SCIP,...

Example: Stochastic generator expansion

$$\min_{x,y,z} \sum_{i\in G} f_i x_i + \frac{1}{S}\sum_{s=1}^{S}\sum_{i\in G}\sum_{j\in J} c_{ij}^s y_{ijs} + \frac{1}{S}\sum_{s=1}^{S}\sum_{j\in J} q_j^s z_{js}$$

$$\text{s.t.} \sum_{i\in G} y_{ijs} + z_{js} \geq d_{sj}, \quad j \in J, s = 1,\ldots,S$$

$$\sum_{j\in J} y_{ijs} \leq C_{si} x_i, \quad i \in G, s = 1,\ldots,S$$

$$x_i \in \{0,1\}, \quad i \in I$$

$$z_{js} \geq 0, y_{ijs} \geq 0, \quad i \in G, j \in J, s = 1,\ldots,S$$

Solving via Deterministic Equivalent Form

Advantages

- Straightforward to implement
- State-of-the-art MIP solvers are able to solve impressive size problems

Limitation

- Size can still become too large

Idea: Decomposition algorithms – solve sequence of smaller problems

- To understand these, we need to know a little integer programming methodology

Branch-and-bound

Basic idea behind most algorithms for solving integer programming problems

- Solve a *relaxation* of the problem
  - Some constraints are ignored or replaced with less stringent constraints
- Gives a lower bound on the true optimal value
- If the relaxation solution is feasible, it is optimal
- Otherwise, divide the feasible region (branch) and repeat

How long does branch-and-bound take?

Simple approximation:

$$\text{Total time} = (\text{Time to process a node}) \times (\text{Number of nodes})$$

Both can be very important:

- For very large instances (as in stochastic programming), solving a single relaxation can be too time-consuming
- Number of nodes can grow exponentially in number of decision variables if do not prune often enough

Keys to success

- Solve relaxations fast (enough)
- Obtain strong relaxations so that can prune high in tree

Valid inequalities

Let $X = \{x \in \mathbb{R}^n_+ : Ax \leq b, x_j \in \mathbb{Z}, j \in J\}$

Definition

An inequality $\pi x \leq \pi_0$ is a valid inequality for X if $\pi x \leq \pi_0$ for all $x \in X$. $(\pi \in \mathbb{R}^n, \pi_0 \in \mathbb{R})$

- Valid inequalities are also called "cutting planes" or "cuts"
- Goal of adding valid inequalities to a formulation: improve relaxation bound $\Rightarrow$ explore fewer branch-and-bound nodes

Key questions

- How to find valid inequalities?
- How to use them in a branch-and-bound algorithm?

Finding Valid Inequalities

Generator Expansion Example

- $x_i = 1$ if generator i is open, y_{ij} = demand at location j served from generator i
- Formulation we used earlier:

$$\sum_{j \in J} y_{ijs} \leq C_{si} x_i, \quad \forall i \in G$$

- Valid inequalities:

$$y_{ijs} \leq \min\{d_{sj}, C_{si}\} x_i, \quad \forall i \in I, j \in J$$

- Set of integer feasible points satisfying these are the same
- But many fractional points that satisfy original formulation do not satisfy the redundant constraints

Finding valid inequalities in general

- HUGE topic of research $\Rightarrow$ Power of commercial MIP solvers

Branch-and-cut

At each node in branch-and-bound tree

1. Solve current LP relaxation $\Rightarrow$ $\hat{x}$
2. Attempt to generate valid inequalities that cut off $\hat{x}$
3. If cuts found, add to LP relaxation and go to step 1

Why branch-and-cut?

- Reduce number of nodes to explore with improved relaxation bounds
- Add inequalities required to define feasible region (relevant for Benders for SMIP)

This approach is the heart of all modern MIP solvers

Case 1: SMIP with Continuous Recourse

$$\begin{aligned} \min\ & c^\top x + \sum_{s=1}^{S} p_s \theta_s \\ \text{s.t. } & Ax \geq b \\ & \theta_s \geq Q_s(x), \quad s = 1, \ldots, S \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \end{aligned}$$

where for $s = 1, \ldots, S$

$$\begin{aligned} Q_s(x) = \min\ & q_s^\top y \\ \text{s.t. } & W_s y = h_s - T_s x \\ & y \in \mathbb{R}_+^{n_2} \times \mathbb{R}_+^{p_2} \end{aligned}$$

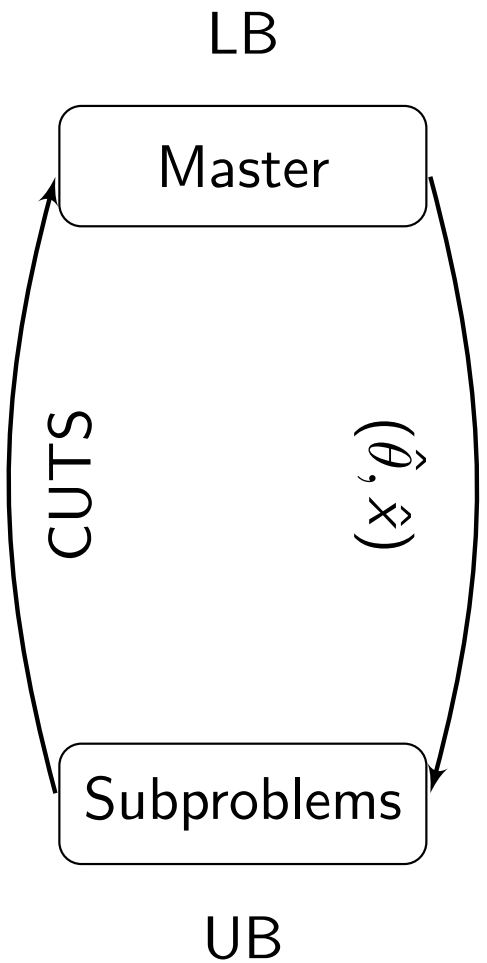
$Q_s(\cdot)$: Piecewise-linear convex function

- Supporting cuts via dual solution of the second-stage LP

Method 1: Basic Benders decomposition

$$(\mathrm{MP})_t^{LP} : \min_{\theta,x} \; c^T x + \sum_{s=1}^{S} p_s \theta_s$$
$$\text{s.t. } Ax \geq b, x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}$$
$$e\theta_s \geq d_{s,t} + B_{s,t} x, \; s = 1, \ldots, S,$$

$$(\mathrm{SP})^s : Q_s(\hat{x}) := \min_{y_s} \; q_s^T y_s$$
$$\text{s.t. } W_s y_s \geq h_s - T_s \hat{x}$$
$$y \in \mathbb{R}_+^{n_2}$$



- Converges after finitely many iterations
- Master problem is a mixed-integer program

Basic Benders algorithm

Limitation

Solving the master MIP can become very time-consuming

- Tends to become more difficult as more cuts are added
- Unlike an LP, MIP master cannot be very effectively warm-started
  $\Rightarrow$ Significant "redundant" work

Alternative

Add Benders cuts as needed during a single branch-and-cut process.

Method 2: Branch-and-cut with Benders cuts

Initalize Benders master problem with Benders cuts

- E.g., solve the LP relaxation via Benders and keep cuts

Begin branch-and-cut algorithm. At each node in the search tree:

- Solve LP relaxation $\Rightarrow$ $(\hat{x}, \hat{\theta})$
- If LP bound exceeds known incumbent, prune.
- If $\hat{x}$ is integer feasible: $(\hat{x}, \hat{\theta})$ might not be feasible!
  - Solve scenario subproblems to generate Benders cuts
  - If $(\hat{\theta}_s, \hat{x})$ violates any Benders cut, add cut to LP relaxation and re-solve.
- If $\hat{x}$ not integer feasible:
  - Optional: Solve scenario subproblems and add Benders cuts if violated
  - Else: Branch to create new nodes

Cuts added when $\hat{x}$ is integer feasible are known as lazy cuts in MIP solvers (add via cut callback routine).

Example: After solving master LP relaxation

Master linear problem

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_1 \geq 179 - 52x_1 - 72x_3 \\
& \ldots \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \ldots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

Optimal solution: $\hat{x} = (0.5, 0.76, 0.33)$, $\hat{\theta} = (129, 112)$

Optimal value (lower bound on SMIP): 286.5

- Subproblems yield no more violated Benders cuts
- Solution is optimal to the LP relaxation

Example: Branch-and-cut phase

Current master problem

$$
\begin{aligned}
\min \quad & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} \quad & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_1 \geq 179 - 52x_1 - 72x_3 \\
& \dots \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \dots \\
& \cancel{0 \leq x_i \leq 1, i = 1, 2, 3} \\
& x_i \in \{0, 1\}, i = 1, 2, 3
\end{aligned}
$$

Load this (partial) formulation to the MIP solver and start solution process

- Let's first suppose MIP solver adds no cuts of its own
- What will it do?

Example: Branch-and-cut phase

Initial master linear program relaxation

$$
\begin{aligned}
\min \quad & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} \quad & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_1 \geq 179 - 52x_1 - 72x_3 \\
& \dots \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \dots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{aligned}
$$

Optimal solution: $\hat{x} = (0.5, 0.76, 0.33)$, $\hat{\theta} = (129, 112)$

Branch!

And keep branching until prune node, or find integer feasible solution

Example: After some branches

$$
\begin{aligned}
\min \quad & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} \quad & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_1 \geq 179 - 52x_1 - 72x_3 \\
& \dots \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \dots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{aligned}
$$

Node 3: Fix $x_1 = 1$, $x_2 = 0$

Optimal solution: $\hat{x} = (1, 0, 0.42)$, $\hat{z} = 355.2$

Node 4: Fix $x_1 = 1$, $x_2 = 1$

Optimal solution: $\hat{x} = (1, 1, 0)$, $\hat{z} = 345.5$

Node 4 yields integer feasible solution!

- But $(\hat{x}, \hat{\theta})$ is not necessarily feasible! (if $\hat{\theta}_s < Q_s(\hat{x})$ for some s)
- We MUST check if there are any violated Benders cuts

Scenario subproblems at Node 4

Subproblems with $\hat{x} = (1, 1, 0)$:

$$\begin{aligned} \min \quad & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} \quad & \sum_{i=1}^{4} y_{ij} + z_j = d_{1j}, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \leq 26 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \leq 25 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \leq 18 \cdot 0 \\ & y_{ij} \geq 0, z_j \geq 0 \end{aligned}$$

Yields violated Benders cut:

$$\theta_1 \geq 141 - 36x_3$$

$$\begin{aligned} \min \quad & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} \quad & \sum_{i=1}^{4} y_{ij} + z_j = d_{2j}, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \leq 26 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \leq 25 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \leq 18 \cdot 0 \\ & y_{ij} \geq 0, z_j \geq 0 \end{aligned}$$

Does not yield a violated Benders cut

Upper bound (because $\hat{x}$ is integer feasible!):
$\sum_i f_i \hat{x}_i + \sum_s p_s Q_s(\hat{x}) = 220 + 1/2(141 + 124) = 352.5$

Branch-and-cut results

Algorithm eventually terminates after 8 nodes

- Not very efficient for a 3-variable binary problem!

What went wrong?

- Poor LP relaxations!

What to do?

- Add Benders cuts at fractional LP solutions
- Use integrality to add stronger cuts (valid inequalities not implied by LP relaxation)

Two options for using integrality to add stronger cuts

- Generate cuts directly in the master problem
- Generate cuts in the subproblems

Master problem cuts

Idea

Derive valid inequalities for the mixed-integer set:

$$\{(x,\theta) : Ax \geq b,$$
$$e\theta_s \geq d_{s,t} + B_{s,t}x,\ s = 1,\ldots,S,$$
$$x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, \theta \in \mathbb{R}^S\}$$

where the constraints in second row are some Benders cuts

- E.g., split cuts, mixed-integer rounding, Gomory mixed-integer cuts,...,
- Ideally, would have all Benders cuts defining $\{(x,\theta) : \theta_s \geq Q_s(x)\}$ but in general too many to enumerate

Master problem cuts: Help the MIP solver help you

Master Problem Cuts

Derive valid inequalities for the mixed-integer set:

$$\{(x, \theta) : Ax \geq b,$$
$$e\theta_s \geq d_{s,t} + B_{s,t}x, \ s = 1, \ldots, S,$$
$$x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, \theta \in \mathbb{R}^S\}$$

where the constraints in second row are some Benders cuts

- MIP solvers will (try to) do this for you if Benders cuts are given to the solver as part of the formulation!
- Facility location example: Gurobi improves root node relaxation from 286.5 to 326.8 (compared to 352 opt)

Many MIP solvers do not derive cuts based on cuts you add in a callback

Master problem cuts: Help the MIP solver help you

Master Problem Cuts

Derive valid inequalities for the mixed-integer set:

$$\{(x,\theta) : Ax \geq b,$$
$$e\theta_s \geq d_{s,t} + B_{s,t}x,\ s = 1,\ldots,S,$$
$$x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, \theta \in \mathbb{R}^S\}$$

where the constraints in second row are some Benders cuts

Takeaway

Phase 0 (solve LP relaxation with Benders, include cuts in formulation) can be very important for effective branch-and-cut implementation

- Don't just add cuts in a callback

Cuts in the subproblems: Help yourself!

Key Idea

Use valid inequalities to obtain stronger LP relaxation of each scenario mixed-integer set:

$$X_s := \{(x, y) : Ax \geq b,\ T_s x + W_s y = h_s \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+\}$$

- NB: So far, we have only seen a convergent algorithm for the case y is continuous
- But subproblem approach for generating cuts is valid and useful for y mixed-integer
- See: [Sen and Higle, 2005, Sen and Sherali, 2006, Gade et al., 2014, Zhang and Küçükyavuz, 2014, Ntaimo, 2013]

Cuts generated for a single scenario $\Rightarrow$ Can still apply Benders decomposition

Cuts in subproblem

$$(\text{MP})_t^{LP} : \min_{\theta,x} \; c^T x + \sum_{s=1}^{S} p_s \theta_s$$

$$\text{s.t. } Ax \geq b, x \in \mathbb{R}_+^{n_1} \times \mathbb{R}_+^{p_1}$$

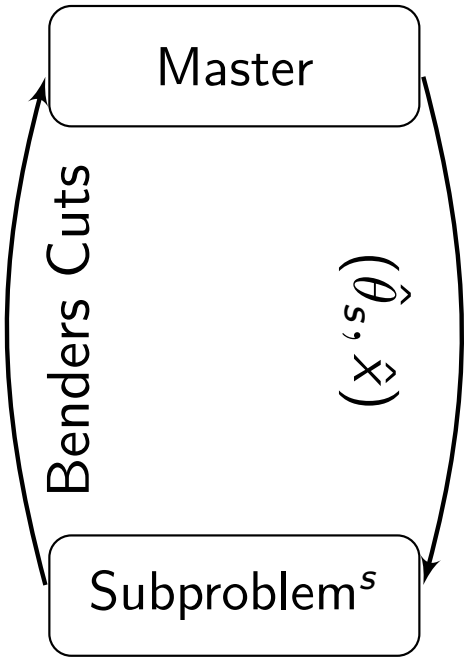
$$e\theta_s \geq d_{s,t} + B_{s,t} x, \; \forall s,$$

$$\theta \in \mathbb{R}^S$$

$$(\text{SP})^s : Q_s(\hat{x}) := \min_{y_s} \; q_s^T y_s$$

$$\text{s.t. } W_s y_s \geq h_s - T_s \hat{x}$$

$$y \in \mathbb{R}_+^{n_2} \times \mathbb{R}_+^{p_2}$$



Cuts in subproblem

$$(\text{MP})_t^{LP} : \min_{\theta,x} \; c^T x + \sum_{s=1}^{S} p_s \theta_s$$

$$\text{s.t. } Ax \geq b, x \in \mathbb{R}_+^{n_1} \times \mathbb{R}_+^{p_1}$$

$$e\theta_s \geq d_{s,t} + B_{s,t} x, \; \forall s,$$

$$\theta \in \mathbb{R}^S$$

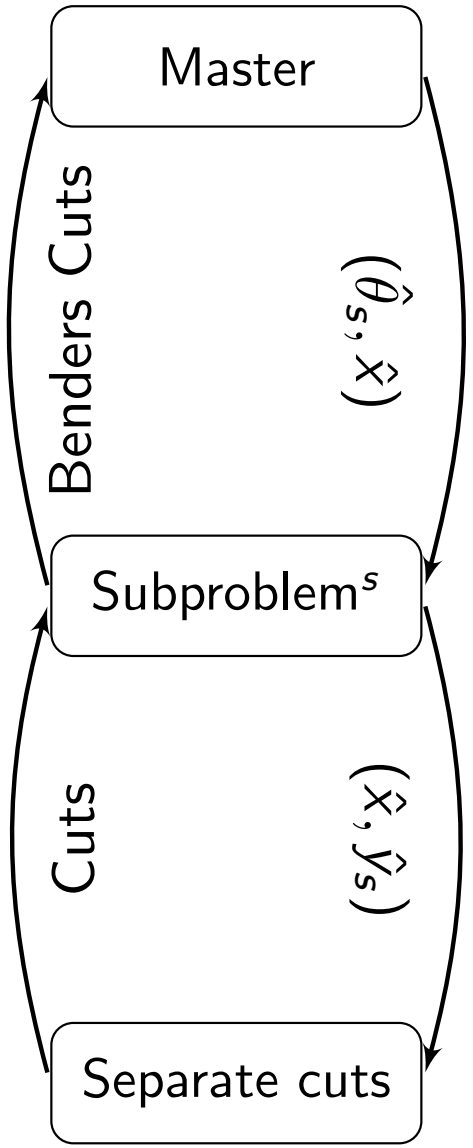
$$(\text{SP})^s : Q_s(\hat{x}) := \min_{y_s} \; q_s^T y_s$$

$$\text{s.t. } W_s y_s \geq h_s - T_s \hat{x}$$

$$\boxed{C_s y_s \geq g_s - D_s \hat{x}}$$

$$y \in \mathbb{R}_+^{n_2} \times \mathbb{R}_+^{p_2}$$

Add cuts to $(\text{SP})^s$, even if it's originally an LP ($p_2 = 0$)



Cuts in the subproblems: Help yourself!

Question

How to generate valid inequalities for each scenario mixed-integer set?

$$X_s := \{(x, y) : Ax \geq b,\ T_s x + W_s y = h_s \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+\}$$

Generating such cuts might require expertise in general integer programming cuts

- Split cuts, Gomory mixed-integer cuts, Chvátal-Gomory cuts,...

But it also might not...

- Use problem-specific cuts, or a better formulation
- E.g., facility location problem

Facility location: Subproblem cuts

Feasible region for a scenario s:

$$\{(x,y,z): \sum_{i\in G} y_{ij} + z_j \geq d_{sj}, \quad j \in J$$
$$\sum_{j\in J} y_{ij} \leq C_{is}x_i, \quad i \in G$$
$$y_{ij} \geq 0, z_j \geq 0, \quad i \in G, j \in J$$
$$x_i \in \{0,1\}, \quad i \in G \quad \}$$

Recall: Valid inequalities

$$y_{ij} \leq \min\{d_{sj}, C_{si}\}x_i, \quad i \in G, j \in J$$

Two options for using them (because there is a "small" number of them)

- Directly add them to the scenario subproblem formulations ✓
- Add them as cuts when solving scenario subproblems

Branch-and-cut again

Initialization phase: Solve new LP relaxation via Benders

Iteration 1: Master linear problem (no θ variable yet)

$$\min\ 120x_1 + 100x_2 + 90x_3$$
$$\text{s.t. } 0 \leq x_i \leq 1, i = 1, 2, 3$$

Optimal solution: $\hat{x} = (0, 0, 0)$
Optimal value (lower bound on SMIP): 0

Example: Iteration 1

Subproblems with $\hat{x} = (0, 0, 0)$:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} \quad & \sum_{i=1}^{4} y_{ij} + z_j = d_{1j}, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \leq 26 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \leq 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \leq 18 \cdot 0 \\
& y_{11} \leq 12 \cdot 0 \\
& \ldots \\
& y_{34} \leq 11 \cdot 0 \\
& y_{ij} \geq 0, z_j \geq 0
\end{aligned}
$$

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} \quad & \sum_{i=1}^{4} y_{ij} + z_j = d_{2j}, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \leq 26 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \leq 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \leq 18 \cdot 0 \\
& y_{11} \leq 8 \cdot 0 \\
& \ldots \\
& y_{34} \leq 6 \cdot 0 \\
& y_{ij} \geq 0, z_j \geq 0
\end{aligned}
$$

Valid inequalities in Benders subproblem $\Rightarrow$ Better cuts in master problem...

Example after root LP solved

Final master linear program

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 923x_1 - 922x_2 - 864x_3 \\
& \ldots \\
& \theta_2 \geq 990 - 794x_1 - 812x_2 - 758x_3 \\
& \ldots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

Optimal solution: $\hat{x} = (0.56, 0.93, 0)$, $\hat{\theta} = (190.3, 152.4)$
Recall: Original LP relaxation bound $= 286.5$

- Bound using this formulation: 332.4 (recall opt $= 352$)
- After Gurobi master cuts on this: 350.5

Recap: SMIP with continuous recourse

Two general approaches:

1. Benders with MIP master problem
2. Branch-and-cut adding Benders cuts (and others) in tree

Which is better?

1. Sequence of MIPs

- Easy to implement
- Tends to solve fewer scenario subproblems
- Master MIP problems may become bottleneck
- Takes full advantage of MIP solver

2. Branch-and-cut

- More difficult to implement
- Single tree eliminates redundant work
- Allows exploiting subproblem cuts, e.g., based on problem structure

My advice: Try simpler option first!

Mixed-integer recourse

What goes wrong with Benders approach with mixed-integer recourse?

Stochastic MIP

$$\min\ c^\top x + \sum_{s=1}^{S} p_s \theta_s$$

$$\text{s.t. } Ax \geq b$$

$$\theta_s \geq Q_s(x), \quad s = 1, \ldots, S$$

$$x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+$$

Where for $s = 1, \ldots, S$

$$Q_s(x) = \min\ q_s^\top y$$

$$\text{s.t. } W_s y = h_s - T_s x$$

$$y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+$$

- $Q_s(x)$: Value function of a mixed-integer program
- Benders cuts (including strengthened with subproblem cuts) still valid
- But master problem constraints $\theta_s \geq Q_s(x)$ cannot be enforced with Benders cuts alone!

Special methods for many cases

Key ingredient in each case

Use cuts/branching to enforce constraint $\theta_s \geq Q_s(x)$ for x feasible to first-stage problem

- *All* can be improved using master cuts and subproblem cuts

Pure binary first stage:

- Integer L-shaped cuts: [Laporte and Louveaux, 1993]
- Split cuts, transfer from one scenario to another: [Sen and Higle, 2005]
- Gomory cuts: [Gade et al., 2014]
- Fenchel cuts: [Ntaimo, 2013]
- Coordination branching: [Alonso-Ayuso et al., 2003]
- Lagrangian cuts: [Zou et al., 2019]

Pure integer first and second-stage:

- Gomory cuts [Zhang and Küçükyavuz, 2014]

Other cases (cont'd)

Mixed binary in first and second-stage:

- Lift-and-project (split) cuts: [Carøe, 1998, Tanner and Ntaimo, 2008]
- Reformulation linearization technique: [Sherali and Zhu, 2007]
- Disjunctive cuts from branch-and-cut tree: [Sen and Sherali, 2006]

Pure integer second-stage:

- Reformulation, integer subproblems, specialized branching: [Ahmed et al., 2004]

General mixed-integer both stages:

- Scaled cuts (Benders master cuts included in scenario subproblems): [van der Laan and Romeijnders, 2020]

Dual Decomposition

LP relaxation is not the only possible relaxation for MIP

- [Carøe and Schultz, 1999] introduced a method for using *Lagrangian relaxation* on a particular reformulation of SMIP
- Reformulation is based on variable splitting

This method, plus extensions, is basis for SMIP solvers

- DSP `dsp.readthedocs.io` [Kim and Zavala, 2018]
- DDSIP [Märkert and Gollmer, 2016]

Variable Splitting

Idea

- Create copies of the first-stage decision variables for each scenario

Recall: Extensive form

$$
\begin{aligned}
z^{SMIP} = \min\ & c^\top x + \sum_{s=1}^{S} p_s q_s^\top y_s \\
\text{s.t. } & Ax \geq b \\
& T_s x + W_s y_s = h_s \qquad s = 1, \ldots, S \\
& x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \\
& y_s \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \qquad s = 1, \ldots, S
\end{aligned}
$$

Variable Splitting

Idea

- Create copies of the first-stage decision variables for each scenario

Copy first-stage variables

$$
\begin{aligned}
z^{SMIP} = \min \sum_{s=1}^{S} p_s (c^T x_s + q_s^T y_s) & \\
A x_s \geq b & \qquad s = 1, \ldots, S \\
T_s x_s + W_s y_s = h_s & \qquad s = 1, \ldots, S \\
x_s = \sum_{s'=1}^{S} p_{s'} x_{s'} & \qquad s = 1, \ldots, S \\
x_s \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, & \qquad s = 1, \ldots, S \\
y_s \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, & \qquad s = 1, \ldots, S
\end{aligned}
$$

Relax nonanticipativity

- The constraints $x_s = \sum_{s'=1}^{S} p_{s'} x_{s'}$ are called *nonanticipativity constraints.*
- Relax these constraints using Lagrangian Relaxation with dual vectors $\lambda = (\lambda_1, \ldots, \lambda_S)$:

$$\mathcal{L}(\lambda) := \min \sum_{s=1}^{S} p_s (c^T x_s + q_s^T y_s) + \sum_{s=1}^{S} p_s \lambda_s^\top \left(x_s - \sum_{s'=1}^{S} p_{s'} x_{s'} \right)$$

$$\begin{aligned}
& Ax_s \geq b && s = 1, \ldots, S \\
& T_s y_s + W_s y_s = h_s && s = 1, \ldots, S \\
& x_s \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, && s = 1, \ldots, S \\
& y_s \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, && s = 1, \ldots, S
\end{aligned}$$

Relax nonanticipativity

- After simplifying Lagrangian relaxation problem has the form: $\mathcal{L}(\lambda) = \sum_s p_s D_s(\lambda_s)$ where

$$\begin{aligned} D_s(\lambda_s) := \quad & \min \ (c + \lambda_s)^\top x + q_s^\top y_s \\ & \text{s.t. } Ax \geq b, \ T_s x + W_s y = h_s \\ & x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, y \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2} \end{aligned}$$

- Each subproblem is a deterministic mixed-integer program

Lagrangian dual problem

- For any $\lambda = (\lambda_1, \ldots, \lambda_S)$ with $\sum_s p_s \lambda_s = 0$,

$$\mathcal{L}(\lambda) \leq z^{SMIP}$$

Lagrangian dual

Find best lower bound:

$$w^{LD} := \max\left\{\mathcal{L}(\lambda) : \sum_{s=1}^{S} p_s \lambda_s = 0\right\}$$

- In general $w^{LD} < z^{SMIP}$
- But $w^{LD} \geq z^{SLP}$ (the usual LP relaxation)
- w^{LD} at least as good as any bound obtained using cuts in single scenario subproblems
- In many test instances, w^{LD} is very close to z^{SMIP}

Solving the Lagrangian dual

Lagrangian dual

Find best lower bound:

$$w^{LD} := \max\left\{\mathcal{L}(\lambda) : \sum_{s=1}^{S} p_s \lambda_s = 0\right\}$$

- λ: High-dimensional ($S \times n$)
- $\mathcal{L}(\lambda)$: Non-smooth, concave function of λ
- Subgradients of $\mathcal{L}(\lambda)$: Solve S deterministic MIP problems

Convex program, but challenging!

Options for Solving Lagrangian dual

Classic approaches

- Subgradient
- Cutting plane algorithm (like Benders, but solve MIP to generate cuts)

Improvements: Add objective term or constraint to encourage/require RMP solutions to not move "too far" in consecutive iterations

- Proximal bundle, bundle level, trust region
- See [Lubin et al., 2013] for numerical comparison

Progressive hedging

- Introduced for SLP case: [Rockafellar and Wets, 1991]
- Extended to SMIP: [Gade et al., 2016, Boland et al., 2018]

New work on parallel implementations:
[Kim et al., 2019, Aravena and Papavasiliou, 2020, Lim et al., 2020]

Closing the gap

Suppose we have (approximately) solved Lagrangian dual

- $\hat{\lambda} = (\hat{\lambda}^1, \ldots, \hat{\lambda}^S)$ and $\mathcal{L}(\hat{\lambda}) \leq z^{SMIP}$
- Primal subproblem solutions: $\hat{x}_1, \ldots, \hat{x}_S$
- Problem: $\hat{x}_s$ possibly not all equal!

To find optimal solution

- Dual decomposition [Carøe and Schultz, 1999]: Use Lagrangian dual in branch-and-bound algorithm
- Requires *spatial* branching – may branch on continuous variables

---

Advantages

- Exploit ever-improving MIP solvers for solving single-scenario MIP's
- Strong relaxations $\Rightarrow$ Often small trees

Limitations

- Even root Lagrangian dual can be slow to solve
- Solving many MIP's to evaluate Lagrangian can still be major bottleneck

Best of both worlds?

Main advantage of Benders branch-and-cut

- LP relaxations solve quickly in branch-and-bound

Limitations

- Extensions to mixed-integer recourse are challenging
- Bounds can be weak (even with cut techniques)

Idea: Solve MIP subproblems like in dual decomposition to generate stronger cuts within Benders

- Early works: [Zou et al., 2019, Rahmaniani et al., 2020]
- Recent: [Chen and Luedtke, 2020] – Speed-up cut generation by working in restricted space of cut coefficients
- Very nice! [van der Laan and Romeijnders, 2020] – Share strength of cuts between scenarios to yield new convergent cutting-plane algorithm

Parting thoughts

The future is bright for stochastic mixed-integer programming!

- Many important applications
- Significant and steady progress in methods

But we have work to do

- SMIP still not "routine", even in simplest case of continuous recourse
- Many current test instances still small
- Mulsti-stage SMIP remains VERY challenging

Questions?

- `jim.luedtke@wisc.edu`


Ahmed, S. and Shapiro, A. (2002).
The sample average approximation method for stochastic programs with integer recourse.
Preprint available at www.optimization-online.org.


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


Alonso-Ayuso, A., Escudero, L. F., and no, M. T. O. (2003).
Bfc, a branch-and-fix coordination algorithmic framework for solving some types of stochastic pure and mixed 0–1 programs.
European Journal of Operational Research, 151(3):503 – 519.


Aravena, I. and Papavasiliou, A. (2020).
Asynchronous lagrangian scenario decomposition.
Mathematical Programming Computation, pages 1–50.


Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Linderoth, J., Luedtke, J., and Oliveira, F. (2018).
Combining progressive hedging with a frank-wolfe method to compute lagrangian dual bounds in stochastic mixed-integer programming.
SIAM Journal on Optimization, 28:1312–1336.


Carøe, C. C. (1998).
Decomposition in Stochastic Integer Programming.
PhD thesis, Department of Operations Research, University of Copenhagen, Denmark.


Carøe, C. C. and Schultz, R. (1998).
A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system.


Carøe, C. C. and Schultz, R. (1999).
Dual decomposition in stochastic integer programming.
Operations Research Letters, 24(1-2):37–45.


Chen, R. and Luedtke, J. (2020).

Approximating the lagrangian dual of a stochastic integer program via fenchel cuts.


Cheung, K., Gade, D., Silva-Monroy, C., Ryan, S. M., Watson, J.-P., Wets, R. J.-B., and Woodruff, D. L. (2015).
Toward scalable stochastic unit commitment.
Energy Systems, 6(3):417–438.


Gade, D., Hackebeil, G., Ryan, S. M., Watson, J., Wets, R. J., and Woodruff, D. L. (2016).
Obtaining lower bounds from the progressive hedging algorithm for stochastic mixed-integer programs.
Mathematical Programming, 157(1):47–67.


Gade, D., Küçükyavuz, S., and Sen, S. (2014).
Decomposition algorithms with parametric Gomory cuts for two-stage stochastic integer programs.
Mathematical Programming, 144(1-2):39–64.


Gendreau, M., Laporte, G., and Séguin, R. (1996).

Stochastic vehicle routing.
European Journal of Operational Research, 88(1):3–12.


Kim, K., Petra, C. G., and Zavala, V. M. (2019).
An asynchronous bundle-trust-region method for dual decomposition of stochastic mixed-integer programming.
SIAM Journal on Optimization, 29(1):318–342.


Kim, K. and Zavala, V. M. (2018).
Algorithmic innovations and software for the dual decomposition method applied to stochastic mixed-integer programs.
Mathematical Programming Computation, 10(2):225–266.


Kleywegt, A. J., Shapiro, A., and Homem-de Mello, T. (2002).
The sample average approximation method for stochastic discrete optimization.
SIAM Journal on Optimization, 12(2):479–502.


Laporte, G. and Louveaux, F. (1993).

The integer L-shaped method for stochastic integer programs with complete recourse.
Operations Research Letters, 13(3):133–142.


Lim, C., Linderoth, J. T., Luedtke, J. R., and Wright, S. J. (2020).
Parallelizing subgradient methods for the lagrangian dual in stochastic mixed-integer programming.
INFORMS Journal on Optimization.


Lubin, M., Martin, K., Petra, C. G., and Sandikci, B. (2013).
On parallelizing dual decomposition in stochastic integer programming.
Operations Research Letters, 41(3):252 – 258.


Mak, W.-K., Morton, D., and Wood, R. (1999).
Monte Carlo bounding techniques for determining solution quality in stochastic programs.
Operations Research Letters, 24:47–56.


Märkert, A. and Gollmer, R. (2016).

User's guide to ddsip – a C package for the dual decomposition of two-stage stochastic programs with mixed-integer recourse.
www.uni-due.de/~hn215go/software/ddsip-man.pdf.


Ntaimo, L. (2013).
Fenchel decomposition for stochastic mixed-integer programming.
Journal of Global Optimization, 55:141–163.


Ntaimo, L., Arrubla, J. A. G., Stripling, C., Young, J., and Spencer, T. (2012).
A stochastic programming standard response model for wildfire initial attack planning.
Canadian Journal of Forest Research, 42(6):987–1001.


Rahmaniani, R., Ahmed, S., Crainic, T. G., Gendreau, M., and Rei, W. (2020).
The benders dual decomposition method.
Operations Research, 68(3):878–895.


Rockafellar, R. and Wets, R.-B. (1991).

Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147.


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


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


Shapiro, A. and Homem-de-Mello, T. (2000).
On the rate of convergence of optimal solutions of Monte Carlo approximations of stochastic programs.
SIAM Journal on Optimization, 11:70–86.


Sherali, H. and Zhu, X. (2007).

On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables.
Mathematical Programming, 108:597–616.


Takriti, S., Birge, J. R., and Long, E. (1996).
A stochastic model for the unit commitment problem.
IEEE Transactions on Power Systems, 11(3):1497–1508.


Tanner, M. and Ntaimo, L. (2008).
Computations with disjunctive cuts for two-stage stochastic mixed 0-1 integer programs.
Journal of Global Optimization, 58:365–384.


van der Laan, N. and Romeijnders, W. (2020).
A converging benders' decomposition algorithm for two-stage mixed-integer recourse models.


Zhang, M. and Küçükyavuz, S. (2014).
Finitely convergent decomposition algorithms for two-stage stochastic pure integer programs.

SIAM Journal on Optimization, 24:1933–1951.

Zou, J., Ahmed, S., and Sun, X. A. (2019).
Stochastic dual dynamic integer programming.
Mathematical Programming, 175:461–502.