CS/ISyE 719: Stochastic Integer Programming

Jim Luedtke

Department of Industrial and Systems Engineering
University of Wisconsin-Madison

October 25, 2016

Outline

Outline

Introduction

Stochastic Mixed Integer Program (SMIP)

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

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

Example applications

Stochastic service system design (facility location)

- Random customer demands
- First-stage decisions: which service centers to open (binary)
- Second-stage decisions: which customers served by which servers (binary or continuous)

Stochastic vehicle routing (a priori routing)

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

Stochastic unit commitment

- 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)

Finite scenario model

We assume the random data ξ is represented by a finite set of scenarios:

- $(q^k, h^k, T^k, W^k), k = 1, \ldots, K$
- Scenario k occurs with probability p_k
- K should not be too large!

Sample Average Approximation

- Medium accuracy approximation can be obtained by Monte Carlo sampling
- Required sample size grows linearly with number of first-stage variables
- Replicating SAA problems yields statistical estimates of optimality
- Key challenge: Solving the SAA problem for "large enough" K

Example: Stochastic facility location

Problem setup

A firm is deciding which facilities to open to serve customers with random demands. Goal is to minimize total (expected) cost.

Notation:

- I: Set of possible facilities to open
- J: Set of customers
- f_i: Fixed cost for opening facility i
- C_i: Capacity of facility i
- c_{ij}: Unit cost for serving customer j demand at facility i
- q_j: Penalty per unit of unmet demand of customer j
- p_k: Probability of scenario k, $k = 1, \ldots, K$
- d_j^k: Demand of customer demand j in scenario k

Example: Stochastic facility location

First-stage integer variables: $x_i = 1$ if facility i is open, 0 otherwise

$$\min_x \sum_{i \in I} f_i x_i + \sum_{k=1}^{K} p_k Q_k(x)$$
$$\text{s.t. } x_i \in \{0,1\}, \quad i \in I$$

where

$$Q_k(x) = \min_{y,z} \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} + \sum_{j \in J} q_j z_j$$
$$\text{s.t. } \sum_{i \in I} y_{ij} + z_j \geq d_j^k, \quad j \in J$$
$$\sum_{j \in J} y_{ij} \leq C_i x_i, \quad i \in I$$
$$y_{ij} \geq 0, z_j \geq 0, \quad i \in I, j \in J$$

SMIP ≡ Large-scale structured MIP

First option for solving a SMIP with finite scenarios

Extensive form (deterministic equivalent) of an SMIP

$$
\begin{aligned}
\min\ & c^\top x + \sum_{k=1}^{K} p_k q_k^\top y_k \\
\text{s.t. } & Ax = b \\
& T_k x + W_k y_k = h_k, \quad k = 1, \ldots, K \\
& x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1} \\
& y_k \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \quad k = 1, \ldots, K
\end{aligned}
$$

Example: Stochastic facility location

$$\min_{x,y,z} \sum_{i \in I} f_i x_i + \sum_{k=1}^{K} p_k \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ijk} + \sum_{k=1}^{K} p_k \sum_{j \in J} q_j z_{jk}$$

$$\text{s.t.} \sum_{i \in I} y_{ijk} + z_{jk} \geq d_j^k, \quad j \in J, k = 1, \ldots, K$$

$$\sum_{j \in J} y_{ijk} \leq C_i x_i, \quad i \in I, k = 1, \ldots, K$$

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

$$z_{jk} \geq 0, y_{ijk} \geq 0, \quad i \in I, j \in J, k = 1, \ldots, K$$

What makes SMIP hard?

Stochastic integer programming combines challenges from Integer programming and Stochastic Programming

Integer programming challenges

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

Stochastic programming challenges

- Evaluating expectation
- Huge size even with finite scenario approximation

Key questions

- Approximate expected value (SAA)
- Obtain strong relaxations
- Decompose large problem into smaller subproblems
- Preserve relaxation strength when doing decomposition
- Converge to optimal solution

Outline

Outline

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

Linear programming relaxation

Mixed-integer program:

$$z_{IP} = \min\ c^\top x$$
$$Ax = b$$
$$x \in \mathbb{R}^n_+ \times \mathbb{Z}^p_+$$

Linear programming relaxation:

$$z_{LP} = \min\ c^\top x$$
$$Ax = b$$
$$x \geq 0$$

Simple observation

$z_{LP} \leq z_{IP}$

Branching: The "divide" in "Divide-and-conquer"

Generic optimization problem:

$$z^* = \min\{c^\top x : x \in S\}$$

Consider subsets $S_1, \ldots S_k$ of S which cover S: $S = \bigcup_i S_i$. Then

$$\min\{c^\top x : x \in S\} = \min_{1 \le i \le k} \left\{ \min\{c^\top x : x \in S_i\} \right\}$$

In other words, we can optimize over each subset separately.

- Usually want S_i sets to be disjoint ($S_i \cap S_j = \emptyset$ for all $i \neq j$)

Dividing the original problem into subproblems is called branching

Bounding: The "conquer" in "Divide-and-conquer"

Any feasible solution to the problem provides an upper bound U on the optimal solution value. $(\hat{x} \in S \Rightarrow z^* \leq c^\top \hat{x})$.

- We can use heuristics to find a feasible solution $\hat{x}$

After branching, for each subproblem i we solve a *relaxation* yielding a lower bound $\ell(S_i)$ on the optimal solution value for the subproblem.

- Overall Bound: $L = \min_i \ell(S_i)$

Key: If $\ell(S_i) \geq U$, then we don't need to consider subproblem i.

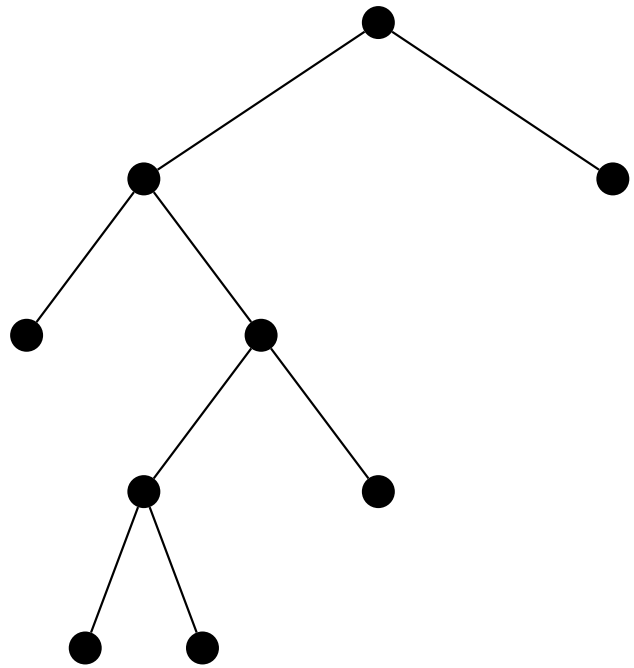
In deterministic MIP, we usually get the lower bound by solving the LP relaxation, but there are other ways too.

Branch and bound for MIP

- Let z_{IP} be the optimal value of the MIP
- Solve the relaxation of the original problem:
  1. If unbounded $\Rightarrow$ the MIP is unbounded or infeasible.
  2. If infeasible $\Rightarrow$ MIP is infeasible.
  3. If we obtain a feasible solution for the MIP $\Rightarrow$ it is an optimal solution to MIP. $(L = z_{IP} = U)$
  4. Else $\Rightarrow$ Lower Bound. $(L = z_{Rel})$.
- In the first three cases, we are finished.
- In the final case, we must **branch** and recursively solve the resulting subproblems.

Terminology

- Subproblems (nodes) form a **search tree**
- Eliminating a problem from further consideration is called **pruning**
- The act of bounding and then branching is called **processing**
- A subproblem that has not yet been processed is called a **candidate**
- The set of candidates is the **candidate list**



Branch and bound algorithm

1. Derive an bound U using a heuristic method (if possible).
2. Put the original problem on the candidate list.
3. Select a problem S from the candidate list and solve the relaxation to obtain the bound $\ell(S)$
   - Relaxation infeasible $\Rightarrow$ node can be pruned.
   - $\ell(S) < U$ and the solution is feasible for the MIP $\Rightarrow$ set $U \leftarrow \ell(S)$.
   - $\ell(S) \geq U \Rightarrow$ node can be pruned.
   - Otherwise, branch. Add the new subproblems to the list.
4. If the candidate list in nonempty, go to Step 3. Otherwise, the algorithm is completed.

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

Also important (but less so):

- Make good branching decisions (limits size of tree)
- Obtain good feasible solutions early (e.g., good heuristics)

Outline

Formulations in MIP

Integer programs can often be formulated in multiple ways

- E.g., facility location problem
- $x_i = 1$ if facility i is open, $y_{ij} =$ customer j demand served from facility i
- Formulation we used earlier:

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

- Redundant constraints:

$$y_{ij} \leq \min\{d_j^k, C_i\} x_i, \quad \forall i \in I, j \in J$$

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

Formulations in MIP

Given two formulations, which is better?

- Option 1: Fewer constraints $\Rightarrow$ Faster LP relaxation solution

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

- Option 2: Better LP relaxations $\Rightarrow$ Prune more often

$$\sum_{j \in J} y_{ij} \leq C_i x_i, \quad \forall i \in I, \qquad y_{ij} \leq \min\{d_j^k, C_i\} x_i, \quad \forall i \in I, j \in J$$

Using the formulation with better bound is almost always (much) better.

- May need to use specialized techniques to solve larger relaxation

Example: Stochastic facility location revisited

$$\min_{x,y,z} \sum_{i\in I} f_i x_i + \sum_{k=1}^{K} p_k \sum_{i\in I}\sum_{j\in J} c_{ij} y_{ijs} + \sum_{k=1}^{K} p_k \sum_{j\in J} q_j z_{jk}$$

$$\text{s.t.} \sum_{i\in I} y_{ijk} + z_{jk} \geq d_j^k, \quad j \in J, k = 1,\ldots,K$$

$$\sum_{j\in J} y_{ijk} \leq C_i x_i, \quad i \in I, k = 1,\ldots,K$$

$$y_{ijk} \leq \min\{d_j^k, C_i\} x_i, \quad i \in I, j \in J, k = 1,\ldots,K$$

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

$$z_{jk} \geq 0, y_{ijk} \geq 0, \quad i \in I, j \in J, k = 1,\ldots,K$$

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?

Example: Matching

Matching problem:

- Given a graph $G = (V, E)$
- A matching is a set of edges that share no endpoint
- Given weights $w : E \to \mathbb{R}$, find maximum weight matching

IP formulation:

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(i)} x_e \leq 1, \quad i \in V \\ & x_e \in \{0, 1\}, \quad e \in E \end{aligned}$$

Matching valid inequalities

IP formulation:

$$\begin{aligned} \max \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(i)} x_e \leq 1, \quad i \in V \\ & x_e \in \{0,1\}, \quad e \in E \end{aligned}$$

Let $S \subseteq V$ with $|S|$ odd. The inequality

$$\sum_{e=ij : i,j \in S} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor$$

is valid

- Why? Every edge in a matching "covers" two nodes. So at most $|S|/2$ edges can be chosen in set S.

Using valid inequalities

1. Add them to the initial formulation
   - Creates a formulation with better LP relaxation
   - Feasible only when you have a "small" set of valid inequalities
   - Easy to implement

---

2. Add them only as needed to cut off fractional solutions
   - Solve LP relaxation, cut off solution with valid inequalities, repeat
   - Cut-and-branch: Do this *only* with the initial LP relaxation (root node)
   - Branch-and-cut: Do this at all nodes in the branch-and-bound tree

Trade-off with increasing effort generating cuts

\+ Fewer nodes from better bounds

\– More time finding cuts and solving LP

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

This approach is the heart of all modern MIP solvers

Deriving valid inequalities

General purpose valid inequalities

- Assume only that you have a (mixed or pure) integer set described with inequalities

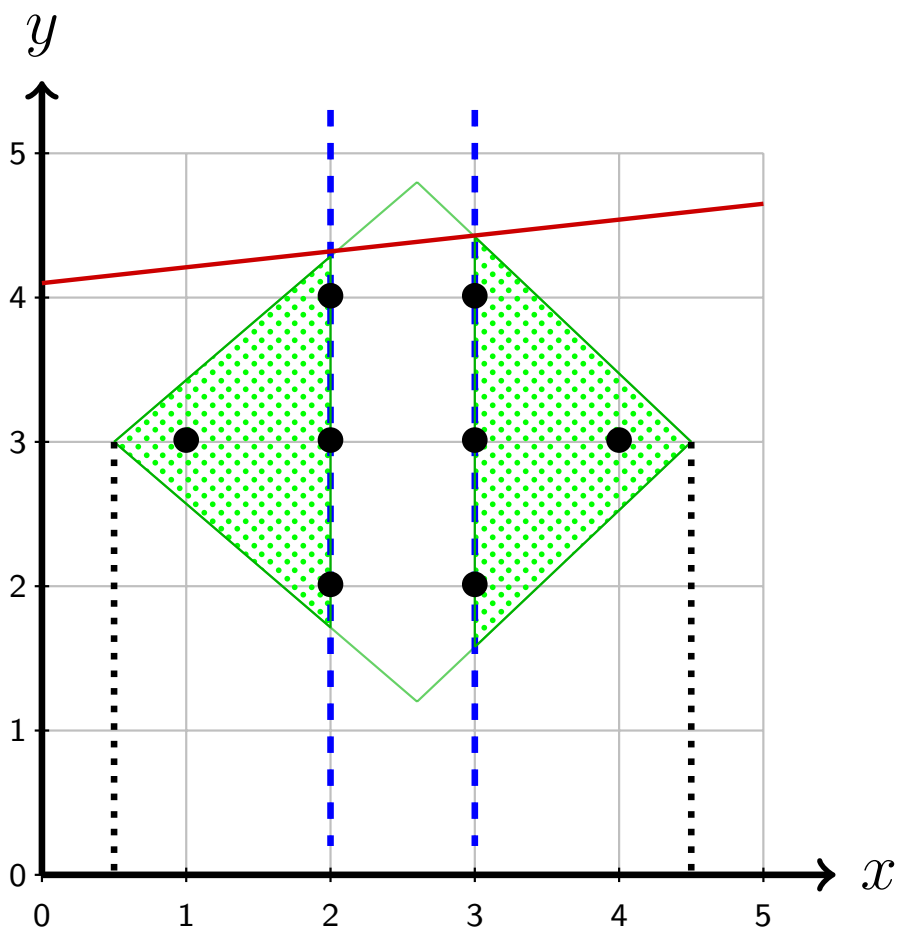
$$X = \{x \in \mathbb{R}^n_+ : Ax \leq b, x_j \in \mathbb{Z}, j \in J\}$$

- Critical to success of deterministic MIP solvers

Structure-specific valid inequalities

- Rely on particular structure that appears in a problem
- Combinatorial optimization problems: matching, traveling salesman problem, set packing, ...
- Knapsack constraints: $\{x \in \mathbb{Z}^n_+ : ax \leq b\}$
- Flow balance with variable upper bounds, etc.

Deriving valid inequalities: Split cuts



- Let $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$ and $X = \{x \in P : x_j \in \mathbb{Z}, \text{ for } j \in J\}$
- Let $\pi \in \mathbb{Z}^n$ be such that $\pi_j = 0$ for all $j \notin J$, and $\pi_0 \in \mathbb{Z}$
- πx is integer for all $x \in X$
- So $X \subset P_0 \cup P_1$ where

$$P_0 = \{x \in P : \pi x \leq \pi_0\}$$
$$P_1 = \{x \in P : \pi x \geq \pi_0 + 1\}$$

- An inequality valid for $P_0 \cup P_1$ is called a split cut

Finding violated split cuts

Separation problem

Given an LP relaxation solution $\hat{x}$, find a valid inequality that "cuts off" $\hat{x}$, or prove none exists.

When the split disjunction $\pi x \leq \pi_0$, $\pi x \geq \pi_0 + 1$ is fixed

- A split cut, if one exists, can be obtained by solving a linear program

How to choose the split disjunction?

- Difficult problem in general
- Lift-and-project cuts: Restrict to $x_j \leq 0$ and $x_j \geq 1$ for binary variables x_j
- Many other heuristics

Split cuts closely related to Gomory mixed-integer cuts and mixed-integer rounding cuts

Outline

Outline

LP relaxation of SMIP

Stochastic MIP

$$\min\ c^\top x + \sum_{k=1}^{K} p_k \theta_k$$

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

$$\theta_k \geq Q_k(x), \quad k = 1, \ldots, K$$

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

where for $k = 1, \ldots, K$

$$Q_k(x) = \min\ q_k^\top y$$

$$\text{s.t. } W_k y = h_k - T_k x$$

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

LP Relaxation

$$\min\ c^\top x + \sum_{k=1}^{K} p_k \theta_k$$

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

$$\theta_k \geq Q_k^{LP}(x), \quad k = 1, \ldots, K$$

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

where for $k = 1, \ldots, K$

$$Q_k^{LP}(x) = \min\ q_k^\top y$$

$$\text{s.t. } W_k y = h_k - T_k x$$

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

Benders Decomposition (L-Shaped Method)

$$(\text{MP})_t^{LP} : \min_{\theta,x} \; c^\top x + \sum_{k=1}^{K} p_k \theta_k$$

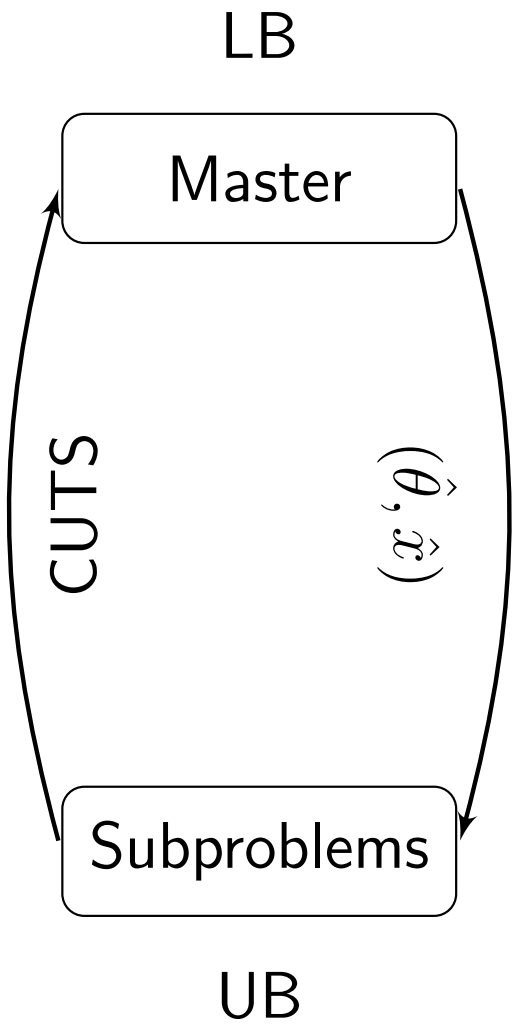
$$\text{s.t. } Ax = b, x \in \mathbb{R}^{n_1}_+ \times \mathbb{R}^{p_1}_+$$

$$e\theta_k \geq d_{k,t} + B_{k,t}x, \;\; k = 1, \ldots, K,$$

$$(\text{SP})^k : Q_k^{LP}(\hat{x}) := \min_{y_k} \; q_k^\top y_k$$

$$\text{s.t. } W_k y_k \geq h_k - T_k \hat{x}$$

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



- Converges after finitely many iterations
- Improvements: trust region/level stabilization, modified subproblem

Outline

Simplest case: Continuous recourse

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

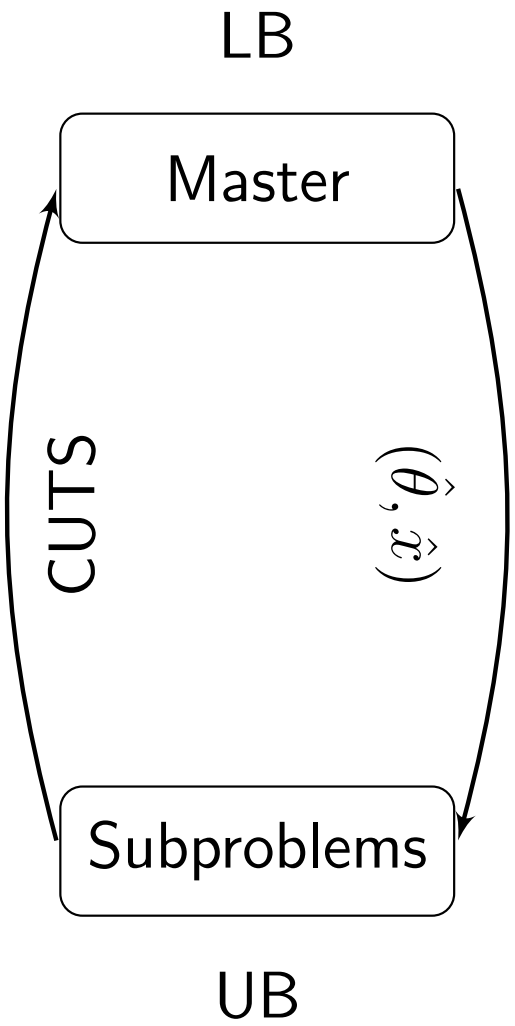
where for $k = 1, \ldots, K$

$$\begin{aligned}
Q_k(x) = \min\ & q_k^\top y \\
\text{s.t. } & W_k y = h_k - T_k x \\
& y \in \mathbb{R}^{n_2}_+
\end{aligned}$$

Method 1: Basic Benders decomposition

$$(\text{MP})_t^{LP}: \min_{\theta,x} c^\top x + \sum_{k=1}^{K} p_k \theta_k$$
$$\text{s.t. } Ax = b, x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}$$
$$e\theta_k \geq d_{k,t} + B_{k,t}x, \ k = 1, \ldots, K,$$

$$(\text{SP})^k: Q_k^{LP}(\hat{x}) := \min_{y_k} q_k^\top y_k$$
$$\text{s.t. } W_k y_k \geq h_k - T_k \hat{x}$$
$$y \in \mathbb{R}_+^{n_2}$$



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

Example: Facility location

Example data:

- Three possible facilities and four customers
- Fixed costs: $f = [120, 100, 90]$
- Capacity: $C = [26, 25, 18]$
- Two equally likely scenarios: $d^1 = [12, 8, 6, 11]$, $d^2 = [8, 11, 7, 6]$
- Penalty for unmet demand: $q_j = 20$

Iteration 1: Master problem (no θ variable yet)

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

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

Example: Iteration 1

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

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

Yields Benders cut:

$$\theta_1 \ge 1140 - 728x_1 - 675x_2 - 468x_3$$

Yields Benders cut:

$$\theta_2 \ge 990 - 728x_1 - 675x_2 - 468x_3$$

Upper bound: $\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} p_k Q_k(\hat{x}) = 0 + 1/2(1140 + 990) = 1065$

Example: Iteration 2

Updated master problem

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

Optimal solution: $\hat{x} = (0, 1, 1)$, $\hat{\theta} = (0, 0)$
Optimal value (lower bound on SMIP): 190

Example: Iteration 2

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

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 1 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 1 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 1 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

Yields Benders cut:

$$\theta_1 \ge 200 - 130x_1 - 18x_3$$

Yields Benders cut:

$$\theta_2 \ge 142 - 104x_1$$

Upper bound: $\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} Q_k(\hat{x}) = 190 + 1/2(182 + 142) = 352$

Example: Iteration 3

Updated master problem

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_1 \geq 200 - 130x_1 - 18x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 142 - 104x_1 \\
& x_i \in \{0, 1\}, i = 1, 2, 3
\end{array}
$$

Optimal solution: $\hat{x} = (1, 0, 1)$, $\hat{\theta} = (52, 38)$
Optimal value (lower bound on SMIP): 255

Example: Iteration 3

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

$$
\begin{array}{ll}
\min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 1 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 1 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
\qquad
\begin{array}{ll}
\min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 1 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 1 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
$$

Yields Benders cut:

$$\theta_1 \ge 237 - 26x_1 - 125x_2$$

Yields Benders cut:

$$\theta_2 \ge 208 - 26x_1 - 125x_2$$

Upper bound: $\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} Q_k(\hat{x}) = 210 + 1/2(211 + 182) = 406.5$

Example: Iteration 5 (skipped one!)

Updated master problem

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \vdots \\
& \theta_1 \geq 141 - 36x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \vdots \\
& \theta_2 \geq 124 - 36x_3 \\
& x_i \in \{0,1\}, i = 1,2,3
\end{array}
$$

Optimal solution: $\hat{x} = (0,1,1)$, $\hat{\theta} = (182, 142)$

Optimal value (lower bound on SMIP): 352

- Matches upper bound from Iteration 2 $\Rightarrow$ Optimal
- Subproblems yield no violated cuts

Recap: Basic Benders algorithm

Basic Benders for SMIP with continuous recourse

Repeat until no cuts found:

1. Solve Benders master mixed-integer program
2. Solve scenario LP subproblems, generate cuts, and add to Benders master.

Limitation

Solving the MIP in step 1 can become very time-consuming

- Tends to become more difficult as more cuts are added
- Unlike an LP, MIP master cannot be 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 $(\hat{\theta}_k, \hat{x})$ 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 revisited: Initialization

First, solve the LP relaxation via Benders

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

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

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

Example: Iteration 1

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

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

$$\begin{array}{ll} \min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\ \text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\ & \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\ & \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\ & y_{ij} \ge 0, z_j \ge 0 \end{array}$$

Yields Benders cut:

$$\theta_1 \ge 1140 - 728x_1 - 675x_2 - 468x_3$$

Yields Benders cut:

$$\theta_2 \ge 990 - 728x_1 - 675x_2 - 468x_3$$

Upper bound (because $\hat{x}$ is integer feasible!):
$\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} Q_k(\hat{x}) = 0 + 1/2(1140 + 990) = 1065$

Example: Iteration 2

Updated master linear program

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

Optimal solution: $\hat{x} = (0.639, 1, 0)$, $\hat{\theta} = (0, 0)$
Optimal value (lower bound on SMIP): 176.6

Example: Iteration 2

Subproblems with $\hat{x} = (0.639, 1, 0)$:

$$
\begin{array}{ll}
\min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0.639 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 1 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
$$

$$
\begin{array}{ll}
\min & \sum_{i=1}^{3}\sum_{j=1}^{4} c_{ij}y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0.639 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 1 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
$$

Yields Benders cut:

$$\theta_1 \ge 179 - 52x_1 - 72x_3$$

Yields Benders cut:

$$\theta_2 \ge 124 - 36x_3$$

No upper bound because $\hat{x}$ is not integer feasible

Example: Iteration 6 (skipped several steps)

Updated master linear problem

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

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

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

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

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

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

Branch!

Let's branch on x_1:

Example: First two nodes

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

Node 1: Fix $x_1 = 0$

Optimal solution: $\hat{x} = (0, 1, 0.66)$, $\hat{z} = 326.3$

Node 2: Fix $x_1 = 1$

Optimal solution: $\hat{x} = (1, 0.46, 0)$, $\hat{z} = 304.9$

Neither can be pruned. Neither yields an upper bound

$\Rightarrow$ Further subdivide each. Start with node 2.

Example: More nodes

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

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 $\hat{\theta}_k < Q_k(\hat{x})$ for some k)
- We MUST check if there are any violated Benders cuts

Scenario subproblems at Node 4

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

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

Yields violated Benders cut:

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

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

Does not yield a violated Benders cut

Upper bound (because $\hat{x}$ is integer feasible!):
$\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} Q_k(\hat{x}) = 220 + 1/2(141 + 124) = 352.5$

Example: Updated master LP relaxation

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \dots \\
& \theta_1 \geq 141 - 36x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \dots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

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

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

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

New optimal solution: $\hat{x} = (1, 1, 0)$, $\hat{z} = 352.5$

Re-solve master problem at Node 4: Again integer feasible

- No more violated Benders cut $\Rightarrow$ 352.5 is a valid upper bound
- Both Nodes 3 and 4 can be pruned!

Example: Back to node 1

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \ldots \\
& \theta_1 \geq 141 - 36x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \ldots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

Node 1: Fix $x_1 = 0$

Optimal solution: $\hat{x} = (0, 1, 0.66)$, $\hat{z} = 326.3$

Subdivide again: $x_3 = 0$ or $x_3 = 1$

Example: More nodes

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \ldots \\
& \theta_1 \geq 141 - 36x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \ldots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

Node 5: Fix $x_1 = 0$, $x_3 = 0$

Optimal solution: $\hat{x} = (0, 1, 0)$, $\hat{z} = 507$

Node 6: Fix $x_1 = 0$, $x_3 = 1$

Optimal solution: $\hat{x} = (0, 0.75, 1)$, $\hat{z} = 326.3$

- Node 5 can be pruned (bound is worse than upper bound of 352.5)
- Node 6 must be subdivided again: $x_2 = 0$ or $x_3 = 1$

Example: More nodes

$$
\begin{array}{ll}
\min & 120x_1 + 100x_2 + 90x_3 + 1/2(\theta_1 + \theta_2) \\
\text{s.t.} & \theta_1 \geq 1140 - 728x_1 - 675x_2 - 468x_3 \\
& \dots \\
& \theta_1 \geq 141 - 36x_3 \\
& \theta_2 \geq 990 - 728x_1 - 675x_2 - 468x_3 \\
& \theta_2 \geq 124 - 36x_3 \\
& \dots \\
& 0 \leq x_i \leq 1, i = 1, 2, 3
\end{array}
$$

Node 7: Fix $x_1 = 0$, $x_3 = 1$, $x_2 = 0$

Optimal solution: $\hat{x} = (0, 0, 1)$, $\hat{z} = 687$

Node 8: Fix $x_1 = 0$, $x_3 = 1$, $x_2 = 1$

Optimal solution: $\hat{x} = (0, 1, 1)$, $\hat{z} = 350.5$

- Node 7 can be pruned (bound is worse than upper bound of 352.5)
- Node 8 is integer feasible: Must check for Benders cuts

Scenario subproblems at Node 8

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

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

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

Yields Benders cut:

$$\theta_1 \geq 200 - 130 x_1 - 18 x_3$$

Yields Benders cut:

$$\theta_2 \geq 142 - 104 x_1$$

Upper bound (because $\hat{x}$ is integer feasible):
$\sum_i f_i \hat{x}_i + \sum_{k=1}^{K} Q_k(\hat{x}) = 190 + 1/2(182 + 142) = 352$

- After re-solving, node 8 can be pruned $\Rightarrow$ We are done!

That was not very efficient!

What went wrong?

- Poor LP relaxations!

What to do?

- Add Benders cuts at fractional LP solutions
- Use integrality to add stronger cuts (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:

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

where the constraints in second row are some Benders cuts

- E.g., split cuts, mixed-integer rounding (Bodur and Luedtke, 2016), Gomory mixed-integer cuts,...,
- Ideally, would have all Benders cuts defining $\{(x, \theta) : \theta_k \geq Q_k(x)\}$ but in general too many to enumerate

Master problem cuts: Help MIP solver help you

Master Problem Cuts

Derive valid inequalities for the mixed-integer set:

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

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)

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

Master problem cuts: Help MIP solver help you

Master Problem Cuts

Derive valid inequalities for the mixed-integer set:

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

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:

$$X_k := \{(x, y) : Ax = b,\ T_k x + W_k y = h_k \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+\}$$

- 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

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

Strength of split cuts: Master vs. subproblem

$$E_k := \{(x, y, \theta) : Ax = b,\ T_k x + W_k y = h_k, \theta = q_k^\top y$$
$$x \in \mathbb{R}^{n_1}_+ \times \mathbb{R}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{R}^{p_2}_+\}$$
$$P_k := \{(x, \theta) : \quad \exists y \text{ with } (x, y, \theta) \in Q_k\}$$

- E_k is the subproblem formulation, P_k is projection based on all Benders cuts from one scenario

Theorem Bodur et al. (2016)

Relaxation obtained using all split cuts on E_k is at least as good as that using all split cuts on P_k, and difference can be large.

Adding cuts in subproblems can be much more effective:

- Power of extended variable space
- Can also use second-stage integrality

Also implies that split cuts in master problem can be more effective when using multi-cut ($\theta_k \geq \ldots$) vs. single-cut ($\theta \geq \sum_k \ldots$)

Cuts in the subproblems: Help yourself!

Question

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

$$X_k := \{(x, y) : Ax = b,\ T_k x + W_k y = h_k \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+\}$$

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 k:

$$\{(x,y,z): \sum_{i\in I} y_{ij} + z_j \geq d_j^k, \quad j \in J$$
$$\sum_{j\in J} y_{ij} \leq C_i x_i, \quad i \in I$$
$$y_{ij} \geq 0, z_j \geq 0, \quad i \in I, j \in J$$
$$x_i \in \{0,1\}, \quad i \in I \quad \}$$

Recall: Valid inequalities

$$y_{ij} \leq \min\{d_j^k, C_i\} x_i, \quad i \in I, j \in J$$

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)

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

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

Example: Iteration 1

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

$$
\begin{array}{ll}
\min & \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^1, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\
& y_{11} \le 12 \cdot 0 \\
& \dots \\
& y_{34} \le 11 \cdot 0 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
\qquad
\begin{array}{ll}
\min & \sum_{i=1}^{3} \sum_{j=1}^{4} c_{ij} y_{ij} + \sum_{j=1}^{4} 30 z_j \\
\text{s.t.} & \sum_{i=1}^{4} y_{ij} + z_j = d_j^2, \ \forall j \\
& \sum_{j=1}^{4} y_{ij} \le 26 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 25 \cdot 0 \\
& \sum_{j=1}^{4} y_{ij} \le 18 \cdot 0 \\
& y_{11} \le 8 \cdot 0 \\
& \dots \\
& y_{34} \le 6 \cdot 0 \\
& y_{ij} \ge 0, z_j \ge 0
\end{array}
$$

Each yields a Benders cut which is added to master problem...

Example: Iteration 7 (skipped many steps)

Updated master linear program

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

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!

Outline

Mixed-integer recourse

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

Stochastic MIP

$$\min\ c^\top x + \sum_{k=1}^{K} p_k \theta_k$$

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

$$\theta_k \geq Q_k(x), \quad k = 1, \ldots, K$$

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

Where for $k = 1, \ldots, K$

$$Q_k(x) = \min\ q_k^\top y$$

$$\text{s.t. } W_k y = h_k - T_k x$$

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

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

Pure binary first-stage

Suppose all first-stage variables are binary

$$\begin{aligned} \min\ & c^\top x + \sum_{k=1}^{K} p_k \theta_k \\ \text{s.t.}\ & Ax = b \\ & \theta_k \geq Q_k(x), \quad k = 1, \ldots, K \\ & x \in \{0,1\}^{n_1} \end{aligned}$$

Integer L-shaped cuts

Assume $Q_k(x) \geq L_k$ for all feasible x. Let $\hat{x} \in \{0,1\}^{n_1}$. Then the following inequality is valid:

$$\theta_k \geq Q_k(\hat{x}) - (Q_k(\hat{x}) - L_k)\left(\sum_{j:\hat{x}_j=1} (1 - x_j) + \sum_{j:\hat{x}_j=0} x_j \right)$$

Pure binary first-stage

Suppose all first-stage variables are binary

Integer L-shaped cuts

Assume $Q_k(x) \geq L_k$ for all feasible x. Let $\hat{x} \in \{0,1\}^{n_1}$. Then the following inequality is valid:

$$\theta_k \geq Q_k(\hat{x}) - (Q_k(\hat{x}) - L_k)\left(\sum_{j:\hat{x}_j=1} (1 - x_j) + \sum_{j:\hat{x}_j=0} x_j \right)$$

- NB: Calculating an integer L-shaped cut requires solving MIP to evaluate $Q_k(\hat{x})$
- Let's see validity proof

Pure binary first-stage

Integer L-shaped cuts can be used exactly as Benders cuts were used in case of continuous recourse

- Solve sequence of MIP master problems
- Use them as cuts within branch-and-cut algorithm

But integer L-shaped should not be used alone!

- VERY WEAK cuts. Cut defined by $\hat{x}$ probably only useful for $x = \hat{x}$
- $\Rightarrow$ May require one cut for every feasible solution

Improving on integer L-shaped cuts

Combine integer L-shaped cuts with all the techniques we discussed for continuous recourse

- Benders cuts from LP relaxation: see, e.g., Angulo et al. (2016)
- Master problem cuts based on integrality of first-stage variables and subset of Benders cuts
- Subproblem cuts based on integrality of first and second-stage variables

Stronger integer L-shaped cuts can be derived by considering all "neighbors" of a solution $\hat{x}$

- Solve scenario surpoblem at all n solutions in which one "bit" of $\hat{x}$ is changed
- Optimal values (or lower bounds) can be combined to yield an improved cut
- See Birge and Louveaux (1997) for details

Outline

Many other cases

Key ingredient in each case

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

Pure binary first stage:

- 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. (2016)

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)

Simple integer second-stage recourse:

- Cuts and reformulation: Louveaux and van der Vlerk (1993)

General mixed-integer first and second-stage: Qi and Sen (2016)

Outline

Two types of cuts

User cuts: These never cut off an integer solution that is feasible to the formulation loaded to the solver

- These are valid inequalities in the traditional sense
- Only purpose of these cuts is to improve relaxation bound

Lazy cuts: These inequalities help to *define the feasible region*

- They might cut off an integer solution feasible to the formulation originally loaded to the solver
- Examples: Benders cuts, subtour elimination constraints in TSP, ...
- Solvers will let you attempt to generate lazy cuts *at every integer solution*

You must tell Gurobi which type of cut you are adding.

Using lazy cuts

- Using lazy cuts to solve a problem with exponentially constraints is sometimes referred to *delayed constraint genreation*
- If you plan to do this, you must turn off some presolve reductions
  - These reductions are based on feasibility considerations of the problem, so are not valid if you will change the feasible region
- In Gurobi: m.params.lazyConstraints $= 1$

Gurobi callbacks

First, declare and implement callback function like this:

```
def mycallback(model, where):
    # Implement the callback routine here
```

Instruct Gurobi to use the callback when you optimize the model:

```
m.optimize(mycallback)
```

If you need to access data or objects from within the callback, you may add them as private members of the model object, e.g.,

```
m._vars = vars
```

- vars is a list created earlier in the code
- m._vars then stores a pointer to that list for use in the callback
- Names of user data must begin with an underscore

Can also use global variables declared above callback function

Within the callback function

Gurobi provides special functions that can be called within the callback function

- model.cbGetSolution, model.cbGetNodeRel, model.cbLazy, model.cbCut, model.cbGet, model.cbSetSolution

The *where* parameter of the callback routine tells you what step of the solution process Gurobi is in, and limits which functions you can call

```
def mycallback(model, where):
    if where == GRB.callback.SIMPLEX:
        # maybe query something
    if where == GRB.callback.MIP:
        # maybe query something
    if where == GRB.callback.MIPNODE:
        # maybe add user cuts, or set a heuristic solution
    if where == GRB.callback.MIPSOL:
        # maybe lazy cuts
```

Getting information in the callback function

Most information on current status can be queried with the function:

- mode.cbGet(what)
- 'what' is the code for the information desired
- Allowed values for 'what' depend on the value of 'where'

```
def mycallback(model, where):
    if where == GRB.callback.SIMPLEX:
        print model.cbGet(GRB.callback.SPX_ITRCNT)
if where == GRB.callback.MIPSOL:
print model.cbGet(GRB.callback.MIPSOL_OBJ)
```

Gurobi Example: tsp.py

This example finds and adds subtour elimination constraints *only* at integer solutions

```
def subtourelim(model, where):
    if where == GRB.callback.MIPSOL:
        selected = []
        # make a list of edges selected in the solution
        for i in range(n):
            sol = model.cbGetSolution(
                  [model._vars[i,j] for j in range(n)])
            selected += [(i,j) for j in range(n) if sol[j] > 0.1]
        # find the shortest cycle in the selected edge list
        tour = subtour(selected)
```

Example: tsp.py (cont'd)

```
if len(tour) < n:
    # add a subtour elimination constraint
    expr = 0
    for i in range(len(tour)):
        for j in range(i+1, len(tour)):
            expr += model._vars[tour[i], tour[j]]
    model.cbLazy(expr <= len(tour)-1)
```

Possible variation: Also search for subtour elimination constraints at *fractional* solutions

Example:

Outline

Outline

Lagrangian relaxation

Consider the following IP

$$\begin{aligned} z^{IP} = \max\ & cx \\ \text{s.t. } & Dx = d \\ & Ax \leq b \\ & x \in \mathbb{Z}^n_+ \end{aligned}$$

Lagrangian relaxation: relax the constraints $Dx = d$ by dualizing them – adding them to the objective with a penalty for violation

- Problem with just constraints $Ax \leq b$ should be easier to solve
- We'll discuss how to choose the constraints to dualize later
- Only for simplicity we assume $x \in \mathbb{Z}^n_+$

Lagrangian relaxation

First, re-write our IP, with $X := \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$

$$\begin{aligned} z^{IP} = \max\ & cx \\ \text{subject to } & Dx = d \\ & x \in X \end{aligned}$$

Let $u \in \mathbb{R}^m$, and define the following Lagrangian Relaxation problem:

$$IP(u): \quad z(u) = \max\{c^\top x + u^\top(d - Dx) : x \in X\}$$

Theorem

For any $u \in \mathbb{R}^m$, $\mathcal{L}(u) \geq z^{IP}$.

Let's see the proof.

Lagrangian dual

For $u \in \mathbb{R}^m$,

$$\mathcal{L}(u) = \max\{c^\top x + u^\top (d - Dx) : x \in X\}$$

Definition

The problem:

$$w^{LD} = \min\{\mathcal{L}(u) : u \in \mathbb{R}^m\}$$

is called a Lagrangian dual.

Properties

- $w^{LD} \geq z^{IP}$
- $w^{LD} \leq \mathcal{L}(u)$ for all $u \in \mathbb{R}^m$

Modification with equality constraints $Dx \leq d$: variables $u \geq 0$

Lagrangian dual

$$IP(u): \quad \mathcal{L}(u) = \max\{c^\top x + u^\top(d - Dx) : x \in X\}$$

Theorem

Let $u \in \mathbb{R}^m$ and $\hat{x}$ be an optimal solution to $IP(u)$. If, $D\hat{x} = d$ then $\hat{x}$ is an optimal solution to IP.

Note: If constraints are of the form $Dx \leq d$, with $u \geq 0$, need two conditions

(1) $D\hat{x} \leq d$, and

(2) $u^\top(d - D\hat{x}) = 0$,

How good is the bound from the Lagrangian dual?

The integer program we're trying to solve

$$z^{IP} = \max\{cx : Dx \leq d, x \in X\} \qquad \text{(IP)}$$

where $X = \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$

Theorem

$$w^{LD} = \max\{cx : Dx \leq d, x \in \text{conv}(X)\}$$

Let $P = \{x \in \mathbb{R}^n_+ : Ax \leq b\}$

$$z^{LP} = \max\{cx : Dx \leq d, x \in P\}$$

- $P \supseteq \text{conv}(X)$, so $w^{LD} \leq z^{LP}$
- If $P = \text{conv}(X)$, then $w^{LD} = z^{LP}$
- $w^{LD} < z^{LP}$ is only possible if $P \neq \text{conv}(X)$

Let's prove it!

Subgradients

Subgradients of $\mathcal{L}$

Let $x^t \in X$ be an optimal solution to calculation of $\mathcal{L}(u^t)$. Then

$$\gamma(u^t) = d - Dx^t$$

is a subgradient of z at u^t.

Proof: Let $v \in \mathbb{R}^m$.

$$\begin{aligned}
\mathcal{L}(v) &= \max\{cx + v(d - Dx) : x \in X\} \\
&\geq cx^t + v(d - Dx^t) \\
&= cx^t + u^t(d - Dx^t) + v(d - Dx^t) - u^t(d - Dx^t) \\
&= \mathcal{L}(u^t) + (v - u^t)(d - Dx^t) \\
&= \mathcal{L}(u^t) + (v - u^t)\gamma(u^t)
\end{aligned}$$

Outline

Variable Splitting

Idea

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

Recall: Extensive form

$$
\begin{aligned}
z^{SMIP} = \min\ & c^\top x + \sum_{k=1}^{K} p_k q_k^\top y_k \\
\text{s.t. } & Ax = b \\
& T_k x + W_k y_k = h_k \quad k = 1, \ldots, K \\
& x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \\
& y_k \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+, \quad k = 1, \ldots, K
\end{aligned}
$$

Variable Splitting

Idea

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

Copy first-stage variables

$$
\begin{aligned}
z^{SMIP} = \min \ & \textstyle\sum_{k=1}^{K} p_k (c^\top x_k + q_k^\top y_k) & \\
& Ax_k = b & k = 1, \ldots, K \\
& T_k x_k + W_k y_k = h_k & k = 1, \ldots, K \\
& x_k = \sum_{k'=1}^{K} p_{k'} x_{s'} & k = 1, \ldots, K \\
& x_k \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, & k = 1, \ldots, K \\
& y_k \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, & k = 1, \ldots, K
\end{aligned}
$$

Relax nonanticipativity

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

$$\begin{aligned}
\mathcal{L}(\lambda) \stackrel{\text{def}}{=} \min \; & \sum_{k=1}^{K} p_k (c^\top x_k + q_k^\top y_k) + \sum_{k=1}^{K} p_k \lambda_k^\top \left(x_k - \sum_{k'=1}^{K} p_{k'} x_{k'} \right) \\
& Ax_k = b && k = 1, \ldots, K \\
& T_k y_k + W_k y_k = h_k && k = 1, \ldots, K \\
& x_k \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, && k = 1, \ldots, K \\
& y_k \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, && k = 1, \ldots, K
\end{aligned}$$

- Rewrite the objective ($\bar{\lambda} = \sum_{k=1}^{K} p_k \lambda_k$):

$$\sum_{k=1}^{K} p_k \left(\left(c + (\lambda_k - \bar{\lambda}) \right)^\top x_k + q_k^\top y_k \right)$$

Relax nonanticipativity

- Rewritten objective ($\bar{\lambda} = \sum_{k=1}^{K} p_k \lambda_k$):

$$\sum_{k=1}^{K} p_k \Big(\big(c + (\lambda_k - \bar{\lambda})\big)^\top x_k + q_k^\top y_k \Big)$$

- Normalize λ_k so that $\bar{\lambda} = 0$
- Lagrangian relaxation problem decomposes: $\mathcal{L}(\lambda) = \sum_{k=1}^{K} D_k(\lambda_k)$ where

$$\begin{aligned} D_k(\lambda_k) \stackrel{\text{def}}{=} \min_{x,y} \; & (c + \lambda_k)^\top x + q_k^\top y_k \\ \text{s.t.} \; & Ax = b, \; T_k x + W_k y = h_k \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+ \end{aligned}$$

- Each subproblem is a deterministic mixed-integer program

Lagrangian dual problem

- For any $\lambda = (\lambda_1, \ldots, \lambda_K)$ with $\sum_{k=1}^{K} \lambda_k = 0$,

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

Lagrangian dual

Find best lower bound:

$$w^{LD} \stackrel{\text{def}}{=} \max\left\{ \mathcal{L}(\lambda) : \sum_{k=1}^{K} p_k \lambda_k = 0 \right\}$$

Strength of Lagrangian dual

Theorem

The Lagrangian dual bound satisfies

$$w^{LD} = \min \left\{ c^\top x + \sum_{k=1}^{K} p_k y_k : (x, y_k) \in \text{conv}(X_k), k = 1, \ldots, K \right\}$$

where for $k = 1, \ldots, K$

$$X_k := \{(x, y) : Ax = b,\ T_k x + W_k y = h_k \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+\}$$

- 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

Solving the Lagrangian dual

Lagrangian dual

Find best lower bound:

$$w^{LD} \stackrel{\text{def}}{=} \max\left\{\mathcal{L}(\lambda) : \sum_{k=1}^{K} \lambda_k = 0\right\}$$

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

Convex program, but challenging!

Review

Many methods same as for two-stage stochastic *linear* programs:

- Subgradient algorithm ✓
- Cutting plane algorithm (including regularized versions) ✓

Only difference: Evaluating subgradient requires solving K MIP problems

Progressive Hedging?

- Convergence depends on convexity!

Review: Progressive hedging

Progresive hedging: Rockafellar and Wets (1991)

- Elegant algorithm for solving primal and dual for convex stochastic programs
- Application of ADMM

Iteration t

- Solve augmented scenario problems and obtain solution x_k^t, $k = 1, \ldots, K$:

$$D_k^\rho(\lambda_k; \bar{x}^{t-1}) \stackrel{\text{def}}{=} \min_{x,y} (c + \lambda_k^t)^\top x + q_k^\top y + (\rho/2)\|x - \bar{x}^{t-1}\|_2^2$$
$$\text{s.t. } Ax = b, T_k x + W_k y = h_k$$
$$x \in \mathbb{R}_+^{n_1}, y \in \mathbb{R}_+^{n_2}$$

- Calculate $\bar{x}^t = \sum_k p_k x_k^t$
- Update λ_k, $k = 1, \ldots, K$ (ρ is a fixed step size)

$$\lambda_k^{t+1} \leftarrow \lambda_k^t + \rho(x_k^t - \bar{x}^t)$$

Progressive hedging for SMIP?

What if we just solve quadratic MIP subproblems?

$$D_k^\rho(\lambda_k; \bar{x}^{t-1}) := \min\ (c + \lambda_k^t)^\top x + q_k^\top y + (\rho/2)\|x - \bar{x}^{t-1}\|_2^2$$
$$\text{s.t. } (x, y) \in X_k$$

where (recall):

$$X_k = \{(x, y) : Ax = b, T_k x + W_k y = h_k$$
$$x \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, y \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}\}$$

No convergence theory, but:

- Watson et al. (2010) Basis of heuristic for primal feasible solutions
- Gade et al. (2016) Heuristic search for good dual solutions λ_k (sensitive to choice of ρ)

Progressive hedging for SMIP?

Suppose we could solve this subproblem:

$$\min\left\{(c+\lambda_k)^\top x + q_k^\top y + (\rho/2)\|x-\bar{x}^{t-1}\|_2^2 : (x,y) \in \text{conv}(X_k)\right\}$$

Problem is convex $\Rightarrow$ Progressive hedging works!

- Dual solution λ_k converges to optimum of Lagrangian dual
- Challenge: $\text{conv}(X_k)$ not known explicitly

Boland et al. (2016): Use inner approximation of $\text{conv}(X_k)$

- Update inner approximation by solving standard MILP subproblems

$$D_k(\lambda_k) = \min\left\{(c+\lambda_k)^\top x + q_k^\top y : (x,y) \in X_k\right\}$$

- Each iteration requires solving K "one scenario" QP's and MILP's

Modified PH for SMIP

Initialize: $V_k \subseteq X_k \neq \emptyset$, $|V_k| < \infty$ and "small"

- Requirement: $\bigcap_{k=1,\ldots,K} \text{proj}_x(\text{conv}(V_k)) \neq \emptyset$

Example: Under relatively complete recourse:

- Choose any $\hat{x} \in X$
- Let $y_k \in \arg\min_y \{d_s^\top y : (\hat{x}, y) \in X_k\}$ for $k = 1, \ldots, K$
- Let $V_k = \{(\hat{x}, y_k)\}$, for $k = 1, \ldots, K$

Modified PH for SMIP (cont'd)

Iteration t

- For each $k = 1, \ldots, K$, let (x_k^t, y_k^t) be an optimal solution to MILP:

$$\min_{x,y}\{(c + \lambda_k^t)^\top x + q_k^\top y : (x, y) \in X_k\}$$

- Update $V_k \leftarrow V_k \cup \{(x_k^t, y_k^t)\}$, for $k = 1, \ldots, K$
- Solve augmented scenario problems and obtain solution x_k^t, $k = 1, \ldots, K$:

$$D_k^\rho(\lambda_k; \bar{x}^{t-1}) \stackrel{\text{def}}{=} \min_{x,y} (c + \lambda_k^t)^\top x + q_k^\top y + (\rho/2)\|x - \bar{x}^{t-1}\|_2^2$$
$$\text{s.t. } (x, y) \in \text{conv}(V_k)$$

- Calculate $\bar{x}^t = \sum_k p_k x_k^t$
- Update λ_k, $k = 1, \ldots, K$ (ρ is a fixed step size)

$$\lambda_k^{t+1} \leftarrow \lambda_k^t + \rho(x_k^t - \bar{x}^t)$$

Implementing QP

How to implement constraint?

$$(x, y) \in \text{conv}(V_k)$$

Add convex combination variables $\lambda \in \mathbb{R}_+^{|V_k|}$:

$$\sum_{i=1}^{|V_k|} \lambda_i (x^i, y^i) = (x, y)$$

$$\sum_{i=1}^{|V_k|} \lambda_i = 1$$

Convergence

It works theorem

Assume the initialization requirement is satisfied. Iterates $\bar{x}^t$ converge to a primal optimal solution of Lagrangian dual problem, and λ^t converges to a dual optimal solution.

Here the "primal problem" is the relaxation

$$\min \left\{ c^\top x + \sum_{k=1}^{K} p_k y_k : (x, y_k) \in \text{conv}(X_k), k = 1, \ldots, K \right\}$$

where for $k = 1, \ldots, K$

$$X_k := \{(x, y) : Ax = b,\ T_k x + W_k y = h_k \\ x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+, y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+ \}$$

Convergence

It works theorem

Assume the initialization requirement is satisfied. Iterates $\bar{x}^t$ converge to a primal optimal solution of Lagrangian dual problem, and λ^t converges to a dual optimal solution.

Proof sketch:

- Inner approximation is always updated by MILP if the QP is not "right"
- Only finitely many extreme point solutions of MILP $\Rightarrow$ eventually no more updates
- At that point, QP always "right", so algorithm reduces to PH applied to convex program

Outline

Closing the gap

Suppose we have (approximately) solved Lagrangian dual

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

Options:

- Find a heuristic solution $\Rightarrow z^{UB}$, compare to $\mathcal{L}(\hat{\lambda})$
- Dual decomposition Carøe and Schultz (1999): Use Lagrangian dual in branch-and-bound algorithm
  - Implemented in DDSIP Märkert and Gollmer (2016)

Simple heuristic (works assuming relatively complete recourse):

- Choose any first-stage scenario solution $\hat{x}_k$
- Fix $x = \hat{x}_k$, and solve all second-stage subproblems

Branch-and-bound

Given approximate Lagrangian dual solution

- $\hat{\lambda} = (\hat{\lambda}^1, \ldots, \hat{\lambda}^K)$ and $\mathcal{L}(\hat{\lambda}) \leq z^{SMIP}$
- Primal subproblem solutions: $\hat{x}_1, \ldots, \hat{x}_K$, $\bar{x} = \sum_{k=1}^{K} \hat{x}_k$

How to branch?

- Solution is infeasible $\Leftrightarrow \exists i, k$ with $\hat{x}_{k,i} \neq \bar{x}_i$
- If x_i is an integer decision variable:
  - Branch 1: $x_i \leq \lfloor \bar{x}_i \rfloor$, Branch 2: $x_i \geq \lceil \bar{x}_i \rceil$
- Else:
  - Branch 1: $x_i \leq \bar{x}_i$, Branch 2: $x_i \geq \bar{x}_i$
- Enforce branching constraints in all scenario subproblems
  - At least one subproblem solution must change!
  - Algorithm is finite if accept solutions as "feasible" when $|\hat{x}_{k,i} - \bar{x}_i| \leq \delta \ \forall i, k$ for some tolerance $\delta > 0$

Binary first-stage variables: Fewer nonanticiptivity constraints?

Suppose x_i is a binary variable: Clever modeling trick?

- Standard nonanticipitivity constraints: K constraints

$$x_{k,i} = \sum_{k'} x_{k',i} \quad \forall k = 1, \ldots, K$$

- As $x_i \in \{0, 1\}$, can be enforced more compactly, e.g.,

$$\sum_{k=2}^{K} x_{k,i} = (K-1)x_{1i}$$

- $x_{1i} = 1 \Leftrightarrow \sum_{k=2}^{K} x_{k,i} = K - 1 \Leftrightarrow x_{k,i} = 1, k = 2, \ldots, K$

n binary variables $\Rightarrow$ Only n Lagrangian dual variables instead of nK!

Almost surely a BAD IDEA!

- Strength of Lagrangian dual bound can be (significantly) weaker $\Rightarrow$ More branching!
- Yet calculating Lagrangian dual still requires solving K MIP

Outline

Outline

Scenario Bundling

Simple Idea

- Partition scenario set as: $\{1, \ldots, K\} = \bigcup_{i=1}^{I} B_i$, $B_i \cap B_{i'} = \emptyset$
- When doing decomposition, treat scenarios within each "bundle" as a single scenario

- Idea can be applied in either LP or Lagrangian based approaches
- See, e.g., Wets (1988), Gade et al. (2016),Escudero et al. (2013), Crainic et al. (2014)

Scenario bundling in LP based methods

Scenario decomposition

$$\begin{aligned} \min\ & c^\top x + \textstyle\sum_{k=1}^{K} p_k Q_k(x) \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

where for $k = 1, \ldots, K$

$$\begin{aligned} Q_k(x) = \min\ & q_k^\top y \\ \text{s.t. } & W_k y = h_k - T_k x \\ & y \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+ \end{aligned}$$

Bundle decomposition

$$\begin{aligned} \min\ & c^\top x + \textstyle\sum_{i=1}^{I} \bar{Q}_i(x) \\ \text{s.t. } & Ax = b \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

where for $i = 1, \ldots, I$

$$\begin{aligned} \bar{Q}_i(x) = \min\ & \textstyle\sum_{k \in B_i} p_k q_k^\top y_k \\ \text{s.t. } & W_k y_k = h_k - T_k x, \quad k \in B_i \\ & y_k \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+, \quad k \in B_i \end{aligned}$$

- Just apply all methods to "Bundle" formulation, with bundles treated like large scenarios

Scenario bundling in LP based methods

Cost of bundle formulation

- Bundle subproblems are larger

Potential benefits

- Faster convergence to solve relaxations (fewer θ_i vars $\Rightarrow$ fewer cuts)
- Fewer subproblems to solve per iteration
- Subproblem cuts derived from bundles can be stronger

Questions (answer empirically?)

- How much bundling gives best trade-off?
- Which scenarios bundle?

Scenario bundling in Lagrangian approach

Create one copy of first-stage variables per bundle

$$z^{SMIP} = \min \sum_{i=1}^{I} \sum_{k \in B_i} p_k (c^\top x_i + q_k^\top y_k)$$

$$Ax_i = b \qquad i = 1, \ldots, I$$

$$T_k x_i + W_k y_k = h_k \qquad k \in B_i, i = 1, \ldots, I$$

$$x_i = \sum_{i'=1}^{K} x_{i'} \qquad i = 1, \ldots, I$$

$$x_i \in \mathbb{R}_+^{n_1} \times \mathbb{Z}_+^{p_1}, \qquad i = 1, \ldots, I$$

$$y_k \in \mathbb{R}_+^{n_2} \times \mathbb{Z}_+^{p_2}, \qquad k = 1, \ldots, K$$

Scenario bundling in Lagrangian approach

Relax the bundle nonanticipitivity constraints with dual variables λ_i, $i = 1, \ldots, I$

$$x_i = \sum_{i'=1}^{I} x_{i'}, \quad i = 1, \ldots, I$$

Bundle subproblems become:

$$\begin{aligned} D_i(\lambda_i) := \ & \min \ (c + \lambda_i)^\top x + \textstyle\sum_{k \in B_i} p_k q_k^\top y_k \\ & \text{s.t. } Ax = b, \ T_k x + W_k y = h_k, \ k \in B_i \\ & y_k \in \mathbb{R}^{n_2}_+ \times \mathbb{Z}^{p_2}_+, \quad k \in B_i \\ & x \in \mathbb{R}^{n_1}_+ \times \mathbb{Z}^{p_1}_+ \end{aligned}$$

Similar trade-offs:

- Fewer dual variables $\Rightarrow$ Lagrangian dual may be solved in fewer iterations
- Fewer MIP subproblems, but each is larger
- Better Lagrangian bound (possibly much better!)

Parting thoughts

The future is bright for stochastic mixed-integer programming!

- Many important applications
- Significant and steady progress in methods

[Cameron Luedtke]

But we have work to do

- SMIP not well solved, even in simplest case of continuous recourse
- Many current test instances: $10 - 20$ first-stage variables, ≤ 50 scenarios

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.

Angulo, G., Ahmed, S., and Dey, S. (2016). Improving the integer L-shaped method. *INFORMS Journal on Computing*, 28:483–499.

Birge, J. R. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer, New York.

Bodur, M., Dash, S., Günlük, O., and Luedtke, J. (2016). Strengthened Benders cuts for stochastic integer programs with continuous recourse. *IJOC*.

Bodur, M. and Luedtke, J. (2016). Mixed-integer rounding enhanced Benders decomposition for multiclass service system staffing and scheduling with arrival rate uncertainty. *Mananagement Science*.

Boland, N., Christiansen, J., Dandurand, B., Eberhard, A., Linderoth, J., Luedtke, J., and Oliveira, F. (2016). Progressive hedging with a Frank-Wolfe method for computing stochastic mixed-integer programming Lagrangian dual bounds. `optimization-online.org`.

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. (1999). Dual decomposition in stochastic integer programming. *Oper. Res. Lett.*, pages 37–45.

Crainic, T. G., Hewitt, M., and Rei, W. (2014). Scenario grouping in a progressive hedging-based meta-heuristic for stochastic network design. *Comput. Oper. Res.*, 43:90–99.

Escudero, L., Araceli GarÃn, M., PÃ©rez, G., and Unzueta, A. (2013). Scenario cluster decomposition of the lagrangian dual in two-stage stochastic mixed 0-1 optimization. *Computers and Operations Research*, 40(1):362–377.

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.

Louveaux, F. and van der Vlerk, M. (1993). Stochastic programming with simple integer recourse. *Mathematical Programming*, 61:301–325.

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.

Qi, Y. and Sen, S. (2016). The ancestral benders' cutting plane algorithm with multi-term disjunctions for mixed-integer recourse decisions in stochastic programming. *Mathematical Programming*. DOI 10.1007/s10107-016-1006-6.

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.

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.

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.

Watson, J.-P., Wets, R. J.-B., and Woodruff, D. L. (2010). Scalable heuristics for a class of chance-constrained stochastic programs. *INFORMS Journal on Computing*, 22(4):543–554.

Wets, R. (1988). Large-scale linear programming techniques in stochastic programming. In Ermoliev, I. and Wets, R., editors, *Numerical techniques for stochastic optimization*, pages 61–89. Springer-Verlag, New York.

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. (2016). Nested decomposition of multistage stochastic integer programs with binary state variables. `optimization-online.org`.