CS 524: Introduction to Optimization
Lecture 36 : Discrete SP

Michael Ferris

Computer Sciences Department
University of Wisconsin-Madison

November 29, 2023

Building the Supermodel (MIP `sfl.gms`)

Weird Science

A general technique for creating two-stage resource problems.

1. Write a nominal (one scenario) model
2. Decide which variables are first stage, and second stage
3. Give s scenario index to all second stage variables and random parameters
4. "Give context" to all scenarios

Facility Location and Distribution

- Facilities: I
- Customers: J
- Fixed cost f_i, capacity u_i for facility $i \in I$
- Demand d_j: for $j \in J$
- Per unit delivery cost: $c_{ij} \quad \forall i \in J, j \in J$

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

$$\sum_{i \in I} y_{ij} \geq d_j \quad \forall j \in J$$

$$\sum_{j \in J} y_{ij} - u_i x_i \leq 0 \quad \forall i \in I$$

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

Evolution of Information

1. Build facilities now
2. Demand becomes known. One of the scenarios $S = \{d^1, d^2, \dots d^{|S|}\}$ happens
3. Meet demand from open facilities

- First stage variables: x_i
- Second stage variables: y_{ijs}

The SuperModel

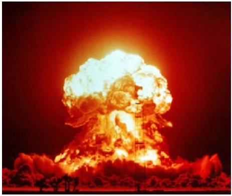
$$\min \sum_{i\in I} f_i x_i + \sum_{s\in S} p_s \sum_{i\in I}\sum_{j\in J} c_{ij} y_{ijs}$$

$$\sum_{i\in I} y_{ijs} \geq d_{js} \quad \forall j \in J\ \forall s \in S$$

$$\sum_{j\in J} y_{ijs} - u_i x_i \leq 0 \quad \forall i \in I,\ \forall s \in S$$

$$x_i \in \{0,1\}, y_{ijs} \geq 0 \qquad \forall i \in I, \forall j \in J, \forall s \in S$$

Modeling Discussion



- Do we always want to meet demand?
  - Regardless of the outcome d^s?
- What happens on the off chance that our product is so popular that we can't possibly meet demand, even if we opened all of the facilities?
  - Does the world end?

Two Ideas

1. We could penalize not meeting demand of customers.
2. We only want to meet demand "most of the time"

Penalize Shortfall: A Recourse Formulation

$$\min \sum_{i \in I} f_i x_i + \sum_{s \in S} p_s \left[\sum_{i \in I} \sum_{j \in J} c_{ij} y_{ijs} + \lambda e_{js} \right]$$

$$\sum_{i \in I} y_{ijs} + e_{js} \geq d_{js} \quad \forall j \in J \; \forall s \in S$$

$$\sum_{j \in J} y_{ijs} - u_i x_i \leq 0 \quad \forall i \in I, \; \forall s \in S$$

$$x_i \in \{0, 1\}, y_{ijs}, e_{js} \geq 0 \qquad \forall i \in I, \forall j \in J, \forall s \in S$$

Can set up SAA model by selecting subset of S and choosing $p_s = 1/|S|$

Handling Uncertainty Another Way

- Suppose instead of penalizing demand shortfall, we would like to enforce that the probability that we meet demand is sufficiently high – say 95%.
- Can we do this?

Probabilistic Constraints (assume y is first-stage decision)

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

$$P\Big(\sum_{i \in I} y_{ij} \geq d_j \ \forall j \in J\Big) \geq 1 - \alpha$$

$$\sum_{j \in J} y_{ij} - u_i x_i \leq 0 \quad \forall i \in I$$

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

How to Model a Probabilistic Constraint

- Suppose a finite number of scenarios $s \in S$ with probability p_s.
- Let $z_s \in \{0, 1\}$ be a binary decision variable with the property that

$$\exists j \in J : \sum_{i \in I} y_{ij} < d_{js} \Rightarrow z_s = 1$$

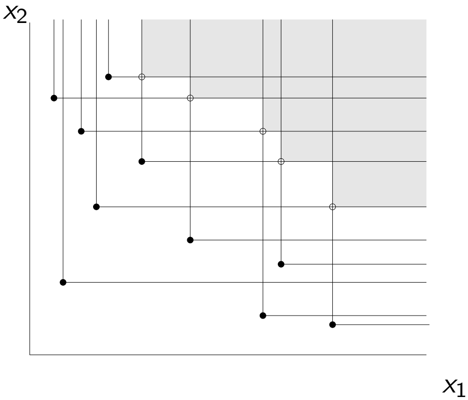
- Then $P(\sum_{i \in I} y_{ij} \geq d_j \; \forall j \in J) = 1 - \sum_{s \in S} p_s z_s$
- The logical conditions, along with the constraint

$$\sum_{s \in S} p_s z_s \leq \alpha$$

  will ensure our probabilistic constraint
- Note: This modeling means that a scenario is "good" only if we meet demand for each customer

Do we NEED integer variables?

- Yes: the feasible region is not convex
- Consider: $P(x_1 \geq \xi_1, x_2 \geq \xi_2) \geq 0.6$
  Each dot: a realization of ξ which occurs with probability $1/10$



Appropriate Trick

- $f(y) > 0 \implies z_s = 1$ with $f(y) = d_{js} - \sum_{i \in I} y_{ij}$
- This is the key constraint
- Note that upper bound M on $f(y)$ is d_{js}
- Plugging it all in yields the (obvious)

$$\begin{aligned} d_{js} > \sum_{i \in I} y_{ij} \Rightarrow z_s = 1 \Leftrightarrow & \qquad f(y) > 0 \Rightarrow z_s = 1 \\ \Leftrightarrow & \qquad f(y) \leq d_{js} z_s \\ \Leftrightarrow & \qquad d_{js} \leq \sum_{i \in I} y_{ij} + d_{js} z_s \end{aligned}$$

Probabilistic Constraints (`sfl-cc.gms`)

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

$$\sum_{i \in I} y_{ij} + d_{js} z_s \geq d_{js} \ \forall j \in J \ \forall s \in S$$

$$\sum_{s \in S} p_s z_s \leq \alpha$$

$$\sum_{j \in J} y_{ij} - u_i x_i \leq 0 \quad \forall i \in I$$

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

$$z_s \in \{0,1\} \qquad \forall s \in S$$

Probabilistic Constraints - y is second stage decision (`sfl-altcc.gms`)

$$\min \sum_{i \in I} f_i x_i + \sum_{s} p_s \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ijs}$$

$$\sum_{i \in I} y_{ijs} + d_{js} z_s \geq d_{js} \; \forall j \in J \; \forall s \in S$$

$$\sum_{s \in S} p_s z_s \leq \alpha$$

$$\sum_{j \in J} y_{ijs} - u_i x_i \leq 0 \quad \forall i \in I, \forall s \in S$$

$$x_i \in \{0,1\}, y_{ijs} \geq 0 \qquad \forall i \in I, \forall j \in J, \forall s \in S$$

$$z_s \in \{0,1\} \qquad \forall s \in S$$

Computation of SAA

- Can use CPLEX (or Gurobi) options to solve using barrier method, e,g,

```
advind 0
lpmethod 4
solutiontype 2
names no
threads 2
```

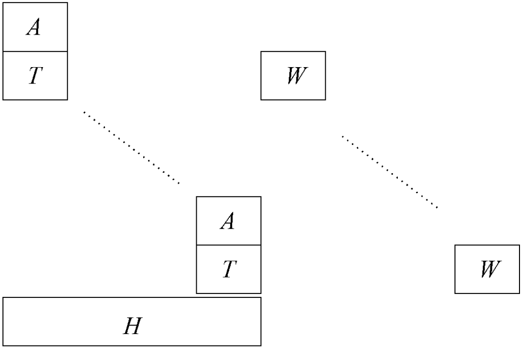
- Can generate samples in four different ways:
  - Directly in GAMS - see `furnsaa0.gms`
  - Using python and GAMS API - see `furnsaa.gms` (API not covered in this class)
  - Using python and gamstransfer - see `furnsaa2.gms` (slightly slower)
  - Using jupyter - see `35sampling.ipynb` (slower due to jupyter overhead)

Summary and Other Extensions

- Sampling: SAA and out of sample testing (see newsvendor example)
- Multi-stage problems: decisions at more than 2 stages
- Robust optimization: extensions of robust linear programming using SOCP
- Risk measures: techniques to value future outcomes differently
- Algorithms to exploit structure of problems:
  - Dynamic programming and approximations - rolling horizon, ADP, SDDP
  - Decomposition methods: Benders, L-shaped method, proximal algorithms
  - Large scale computational schemes: parallel implementations, etc

Key-idea: Non-anticipativity constraints

- Replace x with $x_1, x_2, \ldots, x_K$
- Non-anticipativity: $(x_1, x_2, \ldots, x_K) \in L$ (a subspace) - the H constraints



Computational methods exploit the separability of these constraints, essentially by dualization of the non-anticipativity constraints.

- Primal and dual decompositions (Lagrangian relaxation, progressive hedging , etc)
- L shaped method (Benders decomposition applied to det. equiv.)
- Trust region methods and/or regularized decomposition