Advanced Operations Research Techniques
IE316

Lecture 23

Dr. Ted Ralphs

Reading for This Lecture

- Bertsimas Sections 10.2, 10.3, 11.1, 11.2

The Importance of Formulation

- The most vital aspect of branch and bound is obtaining "good" lower bounds.
- In this respect, not all formulations are created equal.
- Choosing the right one is critical.
- A typical MILP can have many alternative formulations.
- Each formulation corresponds to a different polyhedron enclosing the integer points that are feasible for the problem.
- The more closely the polyhedron approximates the convex hull of the integer solutions, the better the bound will be.

Example: Facility Location Problem

- We are given n potential facility locations and m customers.
- There is a fixed cost c_j of opening facility j.
- There is a cost d_{ij} associated with serving customer i from facility j.
- We have two sets of binary variables.
  - y_j is 1 if facility j is opened, 0 otherwise.
  - x_{ij} is 1 if customer i is served by facility j, 0 otherwise.
- Here is one formulation:

$$min \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} x_{ij} = K \qquad \forall i$$

$$\sum_{i=1}^{m} x_{ij} \leq m y_j \qquad \forall j$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i, j$$

Example: Facility Location Problem

- Here is another formulation for the same problem:

$$min \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{n} x_{ij} = K \qquad \forall i$$

$$x_{ij} \leq y_j \qquad \forall i, j$$

$$x_{ij}, y_j \in \{0, 1\} \qquad \forall i, j$$

- Notice that the set of integer solutions contained in each of the polyhedra is the same (why?).
- However, the second polyhedra strictly includes the first one.
- Therefore, the second polyhedra will yield better lower bounds and be better for branch and bound.
- Notice that the second formulation includes more constraints, but will likely solve more quickly.

Formulation Strength and Ideal Formulations

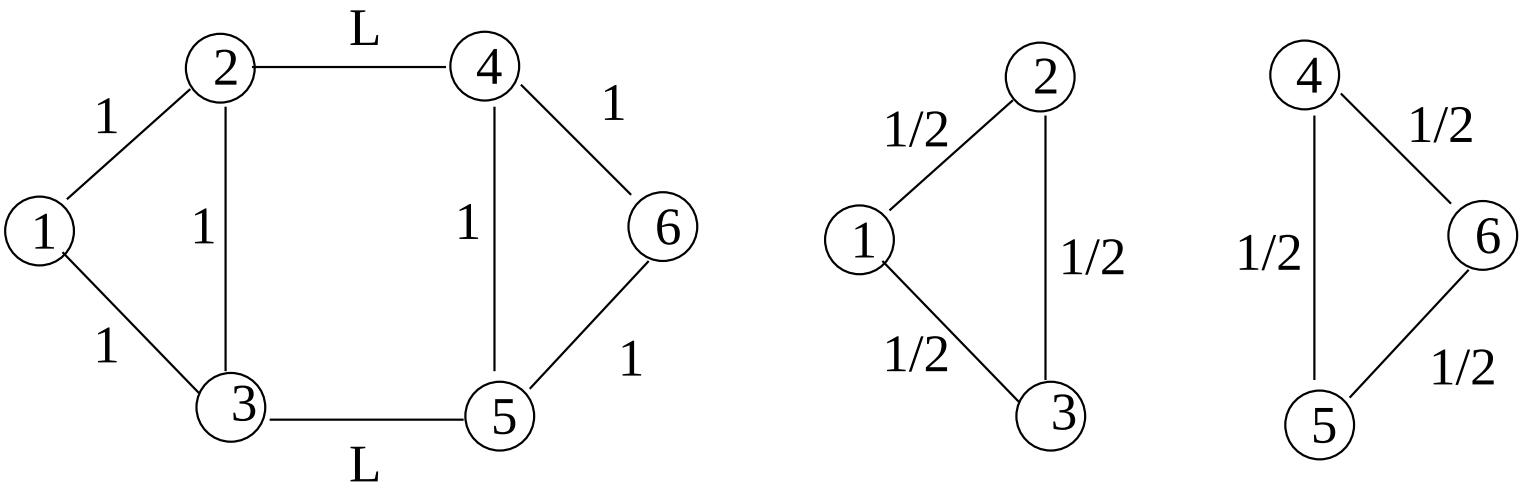
- Consider two formulations A and B for the same ILP.
- Denote the corresponding feasible regions for their LP relaxations as P_A and P_B.
- Formulation A is said to be *at least as strong as* formulation B if $P_A \subseteq P_B$.
- If the inclusion is strict, then A is *stronger than* B.
- If F is the set of all feasible integer solutions for the ILP, then we must have $conv(F) \subseteq P_A$ (why?).
- A is *ideal* if $conv(F) = P_A$

Strengthening Formulations

- Often, a given formulation can be strengthened with additional inequalities satisfied by all feasible integer solutions.
- Example: The Perfect Matching Problem
  - We are given a set of n people that need to paired in teams of two.
  - Let c_{ij} represent the "cost" of the team formed by person i and person j.
  - We wish to maximize efficiency over all teams.
  - We can represent this problem on an undirected graph $G = (N, E)$.
  - The nodes represent the people and the edges represent pairings.
  - We have $x_e = 1$ if the endpoints of e are matched, $x_e = 0$ otherwise.

$$
\begin{aligned}
min \quad & \sum_{e=\{i,j\}\in E} c_e x_e \\
s.t. \quad & \sum_{\{j|\{i,j\}\in E\}} x_{ij} = 1, \ \forall i \in N \\
& x_e \in \{0,1\}, \qquad \forall e = \{i,j\} \in E.
\end{aligned}
$$

Valid Inequalities for Matching



- Consider the graph on the left above.
- The optimal perfect matching has value $L + 2$.
- The optimal solution to the LP relaxation has value 3.
- This formulation can be extremely weak.
- Add the valid inequality $x_{24} + x_{35} \geq 1$.
- Every perfect matching satisfies this inequality.

The Odd Set Inequalities

- We can generalize the inequality from the last slide.
- Consider the cut S corresponding to any odd set of nodes.
- The *cutset* corresponding to S is

$$\delta(S) = \{\{i, j\} \in E | i \in s, j \notin S\}.$$

- An *odd cutset* is any $\delta(S)$ for which $|S|$ is odd.
- Note that every perfect matching contains at least one edge from every odd cutset.
- Hence, each odd cutset of induces a possible valid inequality.

$$\sum_{e \in \delta(S)} x_e \geq 1, S \subset N, |S| odd.$$

Using the New Formulation

- If we add all of the odd set inequalities, the new formulation is ideal.
- However, the number of inequalities is exponential in size.
- Only a small number of these inequalities will be active at the optimal solution.
- Recall the concept of a *constraint generation algorithm*.
- We can generate these inequalities on the fly.
- This can be done efficiently.

Constraint Generation Algorithm for Matching

1. Solve the initial LP relaxation.
2. If the solution is feasible, STOP.
3. Otherwise, look for a violated odd set inequality.
4. Add the inequality and reoptimize from the current basis.
5. Go to Step 2.

Branch and Cut Algorithms

- If we combine constraint generation with branch and bound, we get *branch and cut*.
- The relaxation at each node is strengthened using valid inequalities.
- This increases the lower bound and improves efficiency.
- Branch and cut is the current state of the art for solving ILPs.

The Traveling Salesman Problem

- We are given a set N of *customers*, along with a cost c_{ij} associated with traveling between customers i and j.
- We want to order the customers so that the cost of visiting all customers in the specified order and then returning to the starting point is minimized.
- We consider an undirected graph $G = (N, E)$ where each edge $\{i, j\}$ has associated cost c_{ij}.
- Our problem is to find a minimum cost *Hamiltonian tour* in this graph.
- Integer programming formulation:

$$min \sum_{e \in E} c_e x_e \tag{1}$$

$$s.t. \sum_{\{j|\{i,j\} \in E\}} x_e = 2 \; \forall i \in N, \tag{2}$$

$$\sum_{\substack{\{i,j\} \in E \\ i \in S, j \notin S}} x_e \geq 2 \; \forall S \subseteq N, |S| > 2, \tag{3}$$

$$x_e \in \{0, 1\} \; \forall e \in E. \tag{4}$$

Solving the Traveling Salesman Problem

- Constraints (3) are called the *subtour elimination constraints*.
- Once again, we see that the number of these constraints is exponential.
- In this case, however, the formulation is not ideal—we must use branch and cut.
- We can solve the LP relaxation by using constraint generation.
  - Solve the LP without constraints (3) to obtain $\hat{x}$.
  - Construct a network by associating the capacity $\hat{x}_e$ with each edge e.
  - If the minimum cut in this network has capacity < 2, this corresponds to a violated subtour elimination constraint. Add the constraint to the relaxation and resolve.
  - If the minimum cut in this network has capacity ≥ 2, then all constraints (3) are satisfied and the relaxation is solved.
- We can now embed this subroutine inside a branch and bound algorithm to solve the TSP.

A Branch and Cut Algorithm for the TSP

- At each node in the search tree, solve the relaxation (1)-(4) along with the constraints imposed by branching.
- This LP can be solved using the previously discussed constraint generation algorithm.
- If the optimal solution to the relaxation is not integral, then branch on some fractional variable and continue.
- This branch and cut algorithm will solve reasonably sized instances of the TSP.

Solving Linear Programs in Practice

- The practice of linear and integer programming is as much art as science.
- There are many tradeoffs and considerations in developing and solving a model for a complex system.
- Developing a Model
  - For large, complex systems, there may be a large number of possible models.
  - Real systems have many, many constraints.
  - Timing considerations will determine how many constraints can realistically be modeled.
  - It is important to limit the number of constraints and variables as much as possible.

Developing a Set of Variables

- It is easy to develop models with far too many variables.
- The variables should represent independent decisions that need to be made.
- If the value of a variable can be inferred from the values of other variables, it can sometimes be eliminated.
- Exception: complex cost structures.
- Examples
  - Inventory Models
  - Fixed Charge Network Flow Models
- Additional variables can mean additional *linking constraints*.
- If the number of variables in the model is still large, consider column generation or try alternative pricing rules.

Developing a Set of Constraints

- The number of constraints determines the size of the basis, which in turn affects the efficiency of the simplex algorithm (and others).
- It is therefore crucial to eliminate *redundant constraints*.
- Example: The subtour elimination constraints.
  - Are all the subtour constraints needed?
  - Which ones might be eliminated?
- "Soft" constraints can sometimes be incorporated into the objective function with a penalty (Lagrangian relaxation).
- In real problems, constraints may have to be left out to simplify the model.