Integer Programming
ISE 418

Lecture 18

Dr. Ted Ralphs

Reading for This Lecture

- “Selected Topics in Column Generation,” Lübbecke and Desrosiers

Column Generation

- The cutting plane method can be viewed as a technique for solving integer linear programs with a large number of constraints.
- In the context of integer programming, these large LPs arise as (partial) descriptions of the convex hull of feasible solutions to an integer program.
- By combining the cutting plane method with branch and bound, we obtain branch and cut.
- *Column generation*, on the other hand, is a method for solving LPs with a large number of potential columns.
- Theoretically, this is nothing more than a cutting plane method applied to the dual linear program, but it is useful to consider the method separately.
- When column generation is combined with branch and bound, we obtain a method called *branch and price*.

Formulations Involving Many Columns

- Formulations involving many columns can arise in a number of different ways.
  - Applying a decomposition method, such as Dantzig-Wolfe, to an existing formulation results in a reformulation with many columns.
  - Extended formulations can arise through some other reformulation technique that lifts the problem to a higher-dimensional space (lot-sizing).
  - Formulations with many columns may be the "natural" formulation for some problems.
- Even when we start natively with a formulation that has an exponential number of columns, there is often an underlying "compact formulation".
- Typically, we have a way of writing down the set of columns as the feasible set of a mathematical program of polynomial size.
- In such a case, we can often reformulate the problem in this lower-dimensional space.

Example: The Cutting Stock Problem

- The cutting stock problem was one of the first applications of column generation.
- We are selling rolls of paper in specified widths w_i, $i = 1, \ldots, m$.
- For each width i, we have a given demand d_i that must be satisfied.
- There are large rolls from which the smaller rolls are cut with width W.
- We want to minimize the total number of larger rolls we need to use.
- An IP formulation of this problem is

$$
\begin{aligned}
min \quad & \sum_{i=1}^{n} \lambda_i \\
s.t. \quad & \sum_{i=1}^{n} \lambda_i a^i \geq d \\
& \lambda_i \geq 0, i = 1, \ldots, n, \\
& \lambda_i \quad \text{integer}, i = 1, \ldots, n
\end{aligned}
$$

where the columns a^i represent the *feasible patterns*.

Basic Idea of Solution Method

- We solve the LP to optimality using simplex with only a subset of the columns.
- We then ask whether any column that has been left out has positive reduced cost—if so, that column is added and we reoptimize.
- The problem of determining the column with most positive reduced cost is an optimization problem.
- This is called the *column generation subproblem*.

Generic Column Generation Algorithm

- We are interested in solving an LP with a large number of columns.
- Consider the *restricted problem* obtained by considering only the subset of the columns indexed by set I.

$$\begin{aligned} \max \ & \sum_{i \in I} c_i x_i \\ s.t. \ & \sum_{i \in I} A_i x_i = b \\ & x \geq 0 \end{aligned}$$

- Solve this LP and calculate the optimal dual solution u.
- Now, we must generate a new column A_j for which $c_j - c_B B^{-1} A_j = c_j - u A_j > 0$.
- This can be done by solving the *column generation subproblem*

$$\max_{a \in C} c_a - ua,$$

where C is the global set of columns.

Pattern Generation for Cutting Stock

- The potential columns correspond to feasible *patterns*.
- A given column vector a corresponds to a feasible pattern if and only if

$$\sum_{i=1}^{m} a_i w_i \leq W$$

  and a contains only non-negative integers.
- The objective function coefficient of every pattern (column) is 1.
- Finding the column with the smallest reduced cost is a knapsack problem:

$$\begin{aligned} \max \ & \sum_{i=1}^{m} p_i a_i \\ s.t. \ & \sum_{i=1}^{m} w_i a_i \leq W \\ & a_i \geq 0 \\ & a_i \quad \text{integer} \end{aligned}$$

Example: Set Partitioning Models

- Recall the set partitioning problem.
- In this problem, A is a 0-1 matrix and we wish to find

$$\min\{cx \mid Ax = 1, x \in \mathbb{B}^n\}$$

- Examples of Set Partitioning Models
  - Airline Crew Scheduling
  - Winner Determination in Combinatorial Auctions
  - Vehicle Routing
- Note that in each case, the columns have to satisfy a particular structure that defines the *column generation subproblem*.
- One advantage of these formulations is that we can implicitly introduce constraints that are otherwise difficult to model.

Example: The Generalized Assignment Problem

- The problem is to assign m tasks to n machines subject to capacity constraints.
- An IP formulation of this problem is

$$
\begin{aligned}
\max \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} z_{ij} & \\
s.t. \quad \sum_{i=1}^{n} z_{ij} = 1, & \qquad i = 1, \ldots, m, \\
\sum_{i=1}^{m} w_{ij} z_{ij} \leq d_j, & \qquad j = 1, \ldots, n, \\
z_{ij} \in \{0,1\}, i = 1, \ldots, m, j = 1, \ldots, n, &
\end{aligned}
$$

Reformulating the Generalized Assignment Problem

- Let's rewrite the GAP using columns that represent feasible assignments of tasks to machines.
- Then we have

$$\max \sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij} \left(\sum_{k=1}^{K_j} \lambda_k^j y_{ik}^j \right)$$

$$\begin{aligned} s.t. \quad & \sum_{i=1}^{n} \sum_{k=1}^{K_j} \lambda_k^j y_{ik}^j = 1, & i = 1, \ldots, m, \\ & \sum_{k=1}^{K_j} \lambda_k^j = 1, & j = 1, \ldots, n, \\ & \lambda_k^j \in \{0, 1\}, j = 1, \ldots, n, k = 1, \ldots, K_j, \end{aligned}$$

- Note that this a set partitioning problem.

Branch and Price

- When a decomposition-based bounding method is used within branch and bound, the overall method is usually referred to as *branch and price*.
- Some of the computational issues that arise in branch and price are similar to those in branch and cut, but some are distinct.
  - Generating initial columns and achieving initial feasibility (Phase I).
  - Column management (Phase II).
    - Role of heuristics for solving subproblems (generating multiple solutions).
    - Obtaining true bounds by solving the subproblem exactly.
  - Primal heuristics
  - Convergence and stabilization
  - Branching (when and how?)
  - Algorithm for solving the RMP.

Generating Initial Columns

- One aspect that is very different from a typical cutting plane method is that we do not have a valid formulation to begin with.
- In fact, we may not even have a feasible relaxation.
- We need to generate some initial columns.
- How this is done has a big affect on the effectiveness of the overall algorithm.
- Options
  - Use knowledge of problem structure to generate solutions heuristically.
  - Use problem structure to generate a set of solutions guaranteed to be feasible.
  - Solve the subproblem with randomly perturbed objectives.
  - Do a few iterations of Lagrangian relaxation.
  - Use the generic separation algorithm discussed earlier (decompose and cut).

Phase I

- The initial set of columns may not yield a feasible relaxation.
- In this case, the convex hull of the columns has an empty intersection with $\mathcal{Q}''$ (the complicating constraints).
- The proof of this infeasibility is a dual ray that can be used as an objective in finding the next column.
- Essentially, we want to find a new column that eliminates this dual ray.
- This process continues until the relaxation is feasible.
- Alternatively, we can add artificial variables to ensure feasibility.
- These options mirror the ones we have in standard linear programming.

Obtaining True Bounds

- Since the RMP is a *restriction* of the relaxation we are trying to solve, it does not yield a true bound like the LP relaxation.
- The bound yielded by the RMP is a lower bound on the optimal value of the relaxation, which is itself an upper bound on the optimal value of the IP.
- Note, however, that in each iteration, the subproblem we solve is a Lagrangian relaxation and thus yields a true bound.
- In fact, the process of solving the RMP is to bring together the upper and lower bounds as in branch and bound itself.
- By tracking the current gap between the upper and lower bound, we can get an idea of how quickly we are converging.

Using Heuristic Methods to Solve the Subproblem

- Technically, all that is needed in each iteration is *some* column with positive reduced cost.
- It may not matter if the entering column is the one with the *most* positive reduced cost.
- We can thus use a quick and dirty heuristic method to generate columns initially.
- We can stop anytime after we find a column with positive reduced cost.
- Note, however, that we do not get a true bound in this case.

Column Management

- As in branch and cut, we need to be concerned about managing the set of active columns.
- We can manage the column set in a fashion similar to the way cuts are managed in branch and cut.
  - Maintain a separate columns pool.
  - Control the number of columns entering in each iteration.
  - Remove columns that have negative reduced cost.
- In general, it is advantageous to generate more than one column per iteration.
- This can usually be accomplished by mixing in a variety heuristic methods or generating suboptimal solutions.

Convergence and Tailing Off

- In practice, column generation methods are sometimes slow to converge.
- After a while, the bound may not change much after adding each new column.
- At this point, it may be better to branch than to continue to generate columns.
- Beware that the bound obtained by solving the LP relaxation is not a valid upper bound unless no columns can be generated!
- Note that the same phenomena can occur in branch and cut, but there we are able to stop generating cuts anytime and we still have a valid bound.
- It is possible to obtain a "true" upper bound using Lagrangian duality, even when column generation has not been completed.

Stabilization

- One of the well-known difficulties with column generation is the slow convergence.
- This can sometimes be due to wild fluctuations in the dual solution.
- There are a number of techniques for dealing with this phenomena.
  - Artificially bounding the dual variables.
  - Penalizing changes in the dual solution.
  - Trust region method.
  - Weighted Dantzig-Wolfe.
  - Using an interior point method to solve the RMP.
- These methods can have a big impact in practice.