SE524/EC524 Optimization Theory and Methods

Yannis Paschalidis
yannisp@bu.edu, http://ionia.bu.edu/

Department of Electrical and Computer Engineering,
Division of Systems Engineering,
and Center for Information and Systems Engineering,
Boston University

Lecture 13: Outline

1. Column generation.
2. Application: The cutting stock problem.
3. Cutting plane methods.

Column generation: Key Idea

Why ? **"Real" Problems are large !**

Consider the LP:

$$\begin{array}{rll} \text{minimize} & \mathbf{c}'\mathbf{x} & \\ \text{s.t.} & \mathbf{A}\mathbf{x} = & \mathbf{b} \\ & \mathbf{x} \geq & \mathbf{0}, \end{array}$$

$\mathbf{A}$: $m \times n$ and lin. ind. rows.

When n is large, use revised simplex and generate columns when they are needed. Can be solving the problem

$$\min \bar{c}_i$$

to find pivot column.

The Master Iterations method

1. (**Master Iteration**) Search for j with $\bar{c}_j < 0$. Form a **restricted** problem by considering set of columns I containing: basic columns, j, and potentially some additional nonbasic columns, i.e.,

$$\begin{array}{rll} \text{minimize} & \sum_{i\in I} c_i x_i & \\ \text{s.t.} & \sum_{i\in I} \mathbf{A}_i x_i = & \mathbf{b} \\ & \mathbf{x} \geq & \mathbf{0}, \end{array}$$

2. Solve to optimality.
3. Go to Step 1.

Variants:

1. $I = \{\text{basic variables}, j\}$, i.e., drop columns that exit the basis.
2. Never drop anything. I grows as we add new columns.
3. Keep "recent" columns in I.

The Cutting Stock Problem

Paper company produces rolls of paper of width W.
Customers request b_i rolls of width $w_i < W$ for $i = 1, \ldots, m$.
Pattern $\mathbf{A}_j$, with a_{ij} being # of rolls of width w_i produced by the pattern.

Feasible patterns satisfy:

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

Number of feasible patterns can be **HUGE**.

Objective: Minimize # of large rolls used to satisfy demand.
Decision variables: x_j: # of large rolls cut according to pattern j.

The Cutting Stock Problem (cont.)

$$\begin{array}{lrcll} \text{minimize} & \sum_{j=1}^{n} x_j & & & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j & = & b_i, & i = 1, \ldots, m \\ & x_j & \geq & 0 & j = 1, \ldots, n, \end{array}$$

x_j should be integer, LP relaxation yields good solution for b_i large.
Initial bfs: For $j = 1, \ldots, m$, a feasible pattern $\mathbf{A}_j$ is to cut only one roll w_j, i.e., columns $1, \ldots, m$ of $\mathbf{A}$ form identity matrix.
Use Revised simplex. Generate columns to:

$$\begin{aligned} \min_j \bar{c}_j &\Leftrightarrow \min_j (1 - \mathbf{c}_B' \mathbf{B}^{-1} \mathbf{A}_j) \\ &\Leftrightarrow \min_j (1 - \mathbf{p}' \mathbf{A}_j) \\ &\Leftrightarrow \max_j \mathbf{p}' \mathbf{A}_j \end{aligned}$$

If $\max_j \mathbf{p}' \mathbf{A}_j > 1$ then $\bar{c}_j < 0$ discovered, otherwise all reduced costs ≥ 0, i.e., reached optimality.

Column generation subproblem: Knapsack

$$\begin{array}{lll} \text{maximize} & \sum_{i=1}^m p_i a_i & \\ \text{s.t.} & \sum_{i=1}^m a_i w_i \leq & W, \\ & a_i \geq 0 & i = 1, \ldots, m, \\ & a_i : \text{integer} & i = 1, \ldots, m, \end{array}$$

Can use **dynamic programming (DP)** to solve it. Let $J(r)$ the "reward-to-go" for filling knapsack of capacity r. Note that $J(r) = 0$ for $r < \min_i w_i$.

$$J(r) = \max_{\substack{i=1,\ldots,m \\ w_i \leq r}} [p_i + J(r - w_i)]$$

solve moving forward $r = \min_i w_i, \min_i w_i + 1, \ldots, W$, recover solution moving backwards. Complexity $O(mW)$.

Cutting plane methods

Let us consider the dual analog of column generation methods.

$$\begin{array}{lll} \text{maximize} & \mathbf{p}'\mathbf{b} & (A) \\ \text{s.t.} & \mathbf{p}'\mathbf{A}_i \leq c_i, & i = 1, \ldots, n, \end{array}$$

Let $I \subset \{1, \ldots, n\}$. The **relaxed dual** is:

$$\begin{array}{lll} \text{maximize} & \mathbf{p}'\mathbf{b} & (B) \\ \text{s.t.} & \mathbf{p}'\mathbf{A}_i \leq c_i, & i \in I \end{array}$$

Solve (B) to optimality $\Rightarrow \mathbf{p}^*$.

1. If $\mathbf{p}^*$ is feasible for (A) then optimal for (A).
2. If $\mathbf{p}^*$ is infeasible for (A) find violated constraint, add to I, and resolve (B).

Separation Problem

If $\mathbf{p}^*$ is infeasible for (A), how do we find violating constraint ?
We could solve:

$$\min_i (c_i - (\mathbf{p}^*)' \mathbf{A}_i)$$

If optimal value is ≥ 0, then $\mathbf{p}^*$ is feasible, thus, optimal for (A).
Otherwise, minimizing i gives a violating constraint.

Variants:

1. $I = \{\text{active constraints}\}$, i.e., drop constraints that become inactive.
2. Never drop anything. I grows as we add new constraints.
3. Keep constraints that have been active "recently".