Mathematical Programming — **Lecture 19**
OR 630 Fall 2005 — **November 1, 2005**
Notes by Nico Diener

Column and Constraint Generation

Idea The column and constraint generation algorithms exploit the fact that the revised simplex algorithm only needs very limited access to data of the LP problem to solve some large-scale LPs.

Illustration The one-dimensional cutting-stock problem (Chapter 6 of Bertsimas and Tsitsiklis, Chapters 13 and 26 of Chvatal).

Problem A paper manufacturer produces paper in large rolls of width W and has a demand for narrow rolls of widths say $w_1, ..., w_n$, where $0 < w_i < W$. The demand is for b_i rolls of width w_i.

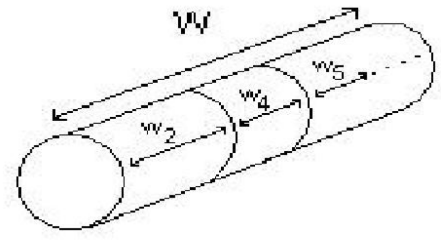

Figure 1:

We want to use as few large rolls as possible to satisfy the demand. We consider patterns, i.e., ways of cutting a large roll into smaller rolls: each pattern j corresponds to a vector $a_j \in \mathbf{R}^m$ with a_{ij} equal to the number of rolls of width w_i in pattern j.

Example: $a = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix}$.

If we could list all patterns say $1, 2, ..., N$ (N can be very large), then the problem would become:

$$\begin{array}{rrcl} \min & \sum_{j=1}^N x_j & & \\ & \sum_{j=1}^N a_j x_j & = & b, \\ & x_j & \geq & 0 \quad \forall j. \end{array} \tag{1}$$

Actually: We want x_j, the number of rolls cut in pattern j, integer but we are just going to consider the LP relaxation.

Problem N is large and A (the matrix made of columns a_j) is known only implicitly:
Any vector $a \in \mathbf{R}^m$ satisfying $\sum_{i=1}^N w_i a_i \leq W$, $a_i \geq 0$ and integer, defines a column of A.

Remark The columns a_j in (1) can be viewed as variables in the above problem.

We consider applying the revised simplex method to (1) by generating columns as needed. We can find an initial basic feasible solution by considering for $a_{j_1}, ..., a_{j_m}$:

$$\begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix}, ..., \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} \text{ or } \begin{pmatrix} \lfloor W/w_1 \rfloor \\ 0 \\ 0 \\ \vdots \end{pmatrix}, ..., \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \lfloor W/w_m \rfloor \end{pmatrix}.$$

At any time, we have a basic feasible solution using $a_{j_1}, ..., a_{j_m}$ and the corresponding dual solution $\overline{y} = B^{-T} e$ with $e = (1 \cdots 1)^T$.

This solution is optimal if all the reduced costs $\overline{c}_j = c_j - a_j^T \overline{y}$ are non-negative. We want to solve:

$$\min\{\overline{c}_j : j = 1, 2, ..., N\}. \tag{2}$$

In our case, we want to see if $a^T \overline{y} \leq 1$ for all columns a of A, so we solve:

$$\max_a \{\overline{y}^T a : w^T a \leq W, a \geq 0, a \in \mathbf{N}^m\}. \tag{3}$$

This is called the knapsack problem (cf. question 4 on HW2: the fractional knapsack problem).
If we can solve (3) and obtain an optimal value at most 1, then the current basic feasible solution is optimal.
If not, its solution $a = a_q$ gives a new column with $\overline{c}_q < 0$ to enter the basis.

Question How to solve (3)?

One way is by using Dynamic Programming. We assume that all w_i's and W are integer and we solve the problem with a recursive algorithm.

Let $F(v)$ be the optimal objective value of (3) with W being replaced by $v \geq 0$ and integer. For $v < \min_i w_i$, $F(v) = 0$, else
$F(v) = \max\{0, \max\{\overline{y}_i + F(v - w_i) : i = 1...m, w_i \leq v\}\}$
Note that the optimal knapsack with weight at most v is obtained by first filling the knapsack with weight at most $v - w_i$ in an optimal way and then adding weight w_i, for the best i, or by not putting anything in at all.

<u>Illustration</u>

- These equations calculate $F(0)$, $F(1)$, ..., $F(W)$ recursively and are called Bellman's equations.
- The computational complexity of the algorithm is $O(mW)$.
- If we remove the integer constraints, this problem becomes obvious.
- We can obtain a lower bound by rounding down the solution without integer constraints. Chvatal describes a branch and bound method for the subproblem based on these ideas.

<u>Remark</u> We have described a method where we only keep the basic columns, but we could also keep some subset $J \subseteq \{1, 2, ..., N\}$ of the previously generated columns including the current basic indices, and then solve:

$$\begin{array}{rrcl} \min & \sum_{j \in J} x_j & & \\ & \sum_{j \in J} a_j x_j & = & b, \\ & x_j & \geq & 0 \quad \forall j \in J, \end{array} \tag{4}$$

in order to obtain an optimal solution, and then use the dual solution to define (3) and then solve it.
The solution is either optimal or we get a new column a_q.
Then we can discard an arbitrary set of non-basic columns from J, add q and get a new problem of the form (4) to solve.
If we keep all the old columns, we can implement Bland's least index rule by numbering the new generated columns.

<u>Note</u> The column generation algorithm can be applied anytime (2): $\min \overline{c}_j : j = 1...N$ can be solved efficiently.

We can also apply these ideas to the dual of (1):
$\max\{b^T y : y \in \mathbf{Q}\}$ where $\mathbf{Q} = \{y \in \mathbf{R}^m : a_j^T y \leq c_j, j = 1...N\}$ is non-empty.
The idea now is that we do not know all the constraints defining $\mathbf{Q}$ ahead of time, but we want to generate them as needed.

We have the original problem:

$$\max\{b^T y : a_j^T y \leq c_j, j = 1...N\} \tag{5}$$

and the current relaxation:

$$\max\{b^T y : a_j^T y \leq c_j, j \in J\} \tag{6}$$

We assume that this problem is bounded.

Motivation The Traveling Salesman Problem: the convex hull of the incidence vectors of tours in a graph is a polytope and therefore a polyhedron. Because we don't know all the constraints, we generate them as needed.

We solve (6) for some $J \subseteq \{1, 2, ..., N\}$ to get an optimal $\overline{y}$.
If $\overline{y}$ is feasible in all other constraints, then $\overline{y}$ is optimal in (5).
Otherwise we try to generate a violated constraint by solving:

$$\min\{\overline{c}_j = c_j - a_j^T \overline{y} : j = 1..N\}. \tag{7}$$

We obtain a new column a_q and then we can add q to J and continue. We are searching for a cutting plane, valid for all $y \in \mathbf{Q}$ but violated by $\overline{y}$.

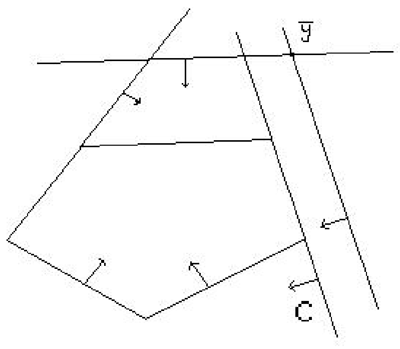

Figure 2: *We try to find a constraint C that is violated*

This approach is hugely successful in solving (a certain class of) combinatorial optimization problems.

Note This method is the dual of the column generation method for (1).