Mathematical Programming	**Lecture 16**
OR 630 Fall 2005	**October 20, 2005**
Notes by Arijit Chakrabarty	

Dual Simplex Algorithm concluded and Extensions to the simplex method

We are not going to completely prove validity of the Dual Simplex Algorithm, but state three key theorems analogous to those for primal simplex method.
Assume (P) is a standard form problem, B is a basis matrix corresponding to basic indices β and non-basic indices ν.

$$\begin{aligned} \text{Let } \bar{x} = \begin{bmatrix} \bar{x}_B \\ \bar{x}_N \end{bmatrix} &:= \begin{bmatrix} \bar{b} \\ 0 \end{bmatrix} \\ \text{with } \bar{b} &:= B^{-1}b \\ \text{and } \bar{y} &:= B^{-T}c_B \end{aligned}$$

be corresponding basic solutions to (P) and (D).
Assume that $\bar{y}$ is feasible in (D), so

$$\bar{c} = c - A^T\bar{y} \geq 0$$

(*i.e.*, B is dual feasible. $\bar{x}$ is not necessarily feasible in (P)).

Theorem 1 *With the hypotheses above, if $\bar{b} \geq 0$, then $\bar{x}$ is feasible, and $\bar{x}$ and $\bar{y}$ are optimal for (P) and (D) respectively.*

Theorem 2 *With the hypotheses above, suppose $\bar{b}_p < 0$, and x_k is the $p-th$ basic variable. Compute $z = B^{-T}e_p$. If $a_j^T z \geq 0$ for all $j \in \nu$, then (D) is unbounded and (P) is infeasible and $-z$ is a certificate of infeasibility of (P).*

Theorem 3 *With the hypotheses above, suppose $\bar{b}_p < 0$ and $z = B^{-T}e_p$ has $a_j^T z < 0$ for some $j \in \nu$. Then let $q \in \nu$ satisfy*

$$\begin{aligned} a_q^T z &< 0 \\ \textit{and } \frac{\bar{c}_q}{-a_q^T z} &= \min\{\frac{\bar{c}_j}{-a_j^T z} : j \in \nu \textit{ with } a_j^T z < 0\} \end{aligned}$$

Then, if we replace the $p-th$ basic index by q, the resulting matrix B_+ is non-singular, the corresponding solution $\bar{y}_+$ is feasible in (D), and

$$b^T\bar{y}_+ = b^T\bar{y} + \frac{\bar{b}_p\bar{c}_q}{a_q^T z} \geq b^T\bar{y}$$

with strict inequality if $\bar{c}_q > 0$.

Proofs: From complementary slackness, the discussion last time and checking that $\bar{c}_+ \geq 0$ if choose q as in Theorem 3.

Remark 1 *The work per iteration is comparable to that for the primal simplex algorithm. Computing $a_j^T z$ for each $j \in \nu$ is equivalent to computing $a_j^T z$ (to get $\bar{c}$) for each $j \in \nu$. The difference is: Primal Simplex has (n-m) comparisons to find q and $\leq m$ divisions in the minimum ratio test, whereas Dual Simplex has m comparisons to find p and $\leq (n-m)$ divisions in the minimum ratio test.*

Remark 2 *Dual degeneracy is equivalent to one or more $\bar{c}_j, j \in \nu$, being zero. In the event of no dual degeneracy, the objective function value increases at each iteration and so the algorithm terminates finitely (since there are only finitely many basic solutions).*

Remark 3 *If we employ the least-index rule to choose the leaving variable x_k and the entering variable x_q, then we get finite termination even in the presence of dual degeneracy.*

Simplex algorithm for problems not in standard form

Note in this context that commercial software accepts problems in any LP form.

0.1 Free variables simplex method

$$
(P) \qquad \begin{array}{rrl}
\min_x & c^T x & \\
& Ax & = b, \\
& x_j & \text{free}, \quad j \in \phi \\
& x_j & \geq 0, \quad j \in \gamma
\end{array}
$$

$$
(D) \qquad \begin{array}{rrl}
\max_y & b^T y & \\
& a_j^T y & = c_j, \quad j \in \phi \\
& a_j^T y & \leq c_j, \quad j \in \gamma
\end{array}
$$

Again we have a basis matrix B, and corresponding primal solution

$$
\bar{x} = \begin{bmatrix} \bar{x}_B \\ \bar{x}_N \end{bmatrix} = \begin{bmatrix} \bar{b} \\ 0 \end{bmatrix},
$$

$$
\bar{b} = B^{-1} b.
$$

It is a basic feasible solution if

$$
\bar{x}_j \geq 0, j \in \beta \cap \gamma.
$$

Is it optimal? Compute $\bar{y} = B^{-T} b$ and consider $\bar{c} = c - A^T \bar{y}$. We are optimal (look at (D)) if

$$
\begin{array}{rcll}
\bar{c}_j & = & 0, & j \in \nu \cap \phi, \\
\bar{c}_j & \geq & 0, & j \in \nu \cap \gamma.
\end{array}
$$

If not, if there exists $j \in \nu$ with $\bar{c}_j < 0$, increase x_j as usual (choose x_q). If instead, $\bar{c}_j > 0$ for $j \in \nu \cap \phi$, then *decrease* x_j (choose x_q).
Increase or decrease x_q and see the effect on the basic variables. We need to look at negative $\bar{a}_{iq}$'s if x_q is decreasing. Also ignore i if the $i-th$ basic variable is in $\beta \cap \phi$. This gives possible unboundedness and changed rules for choosing p. Note that, once a free variable becomes basic, it never leaves the basis.

0.2 Bounded variable simplex method

$$
\begin{array}{cccccc}
 & \min_x & c^T x & & \\
(P) & & Ax & = & b. \\
 & 0 \leq & x & \leq & u.
\end{array}
$$

Assume $0 < u_j < +\infty \, \forall j$. We could also deal with the more general setup $l \leq x \leq u$. The dual problem is

$$
\begin{array}{cccccccc}
 & \max_{y,z} & b^T y & - & u^T z & & \\
(D) & & A^T y & - & z & \leq & c, \\
 & & & & z & \geq & 0.
\end{array}
$$

Note that (D) is always feasible. If we write the constraints as

$$
\begin{array}{rcl}
A^T y - z + s & = & c, \\
z, s & \geq & 0,
\end{array}
$$

for any $\bar{y}$ we can choose

$$s = \max\{0, c - A^T y\}$$

$$z = \max\{0, A^T y - c\}$$

with the "max"s taken componentwise to get a feasible solution. We try to handle the upper bounds without increasing the number of equations or the size of the basis matrix from m to $m + n$. Consider a partition of the indices into basic (β) and nonbasic (ν), and also of ν into nonbasic at lower bound (ν_l) and nonbasic at upper bound (ν_u).
The corresponding basic solution has

$$
\begin{array}{rcll}
\bar{x}_j & = & 0, & j \in \nu_l, \\
\bar{x}_j & = & u_j, & j \in \nu_u, \\
\bar{x}_B & = & B^{-1}(b - N\bar{x}_N). &
\end{array}
$$

It's feasible if $0 \leq \bar{x}_B \leq u_B$.
Let $\bar{y} = B^{-T} c_B$ and compute $\bar{c} = c - A^T \bar{y}$. Look at $\bar{c}_j, j \in \nu$. We are optimal (using complementary slackness and checking the constraints in (D)) if

$$\bar{c}_j \geq 0 \text{ for } j \in \nu_l$$

$$(\bar{x}_j = 0 \Rightarrow \bar{x}_j < u_j \Rightarrow z_j = 0 \Rightarrow s_j = \bar{c}_j), \text{ and}$$

$$\bar{c}_j \le 0 \text{ for } j \in \nu_u$$

$$(\bar{x}_j = u_j \Rightarrow \bar{x}_j > 0 \Rightarrow s_j = 0 \Rightarrow z_j = -\bar{c}_j).$$

Otherwise, choose x_q if $\left\{ \begin{array}{c} \bar{c}_q < 0 \& q \in \nu_l \\ \bar{c}_q > 0 \& q \in \nu_u \end{array} \right\}$ to $\left\{ \begin{array}{c} \text{increase} \\ \text{decrease} \end{array} \right\}$. Compute $\bar{a}_q = B^{-1} a_q$. Choose p by examining the effects on the basic variables.

If x_q is increasing, we find a limit on its increase as follows:

$$\begin{aligned} \bar{a}_{iq} > 0 \quad &\Rightarrow \quad \text{limited by } \frac{(\bar{x}_B)_i}{\bar{a}_{iq}} \\ \bar{a}_{iq} < 0 \quad &\Rightarrow \quad \text{limited by } \frac{(u_B)_i - (\bar{x}_B)_i}{-\bar{a}_{iq}}. \end{aligned}$$

Also there is a limit u_q on the change in x_q. This means that it can be the case that $p = q$, *i.e.*, both entering and leaving variables are x_q!! There are corresponding limits if x_q is decreasing.

Bounded variables can be used to model decreasing returns to scale, or convex cost functions. Suppose that the cost associated with a variable x_j is 2 per unit for the first 10 units, and 3 per unit thereafter. The cost function is then convex and piecewise-linear. We can convert this into a linear programming problem as follows: we use two variables, x_{j1} and x_{j2}, with costs of 2 and 3 respectively. Both variables appear in the constraints in the same way. We put an upper bound on x_{j1} of 10; both variables are restricted to be nonnegative. Then the algorithm will choose to use x_{j1} first, at a unit cost of 2, up to its upper bound of 10; then x_{j2} will be used at a unit cost of 3. The resulting cost will exactly mirror the desired convex cost.

If instead there were increasing returns to scale (like quantity discounts, as opposed to limited supplies from the cheap suppliers), we would want to model a piecewise-linear *concave* cost like 3 per unit for the first 10 units, and 2 per unit thereafter. If we tried the same trick as above, the algorithm would try to use the cheap variable first, and the result would not correctly represent the concave cost.

In HW2, we saw a nice feature of minimizing *concave* cost functions over a polyhedron; here we see an advantage for *convex* cost functions. Another important property of minimizing a convex function over a convex set is that local minimizers are global minimizers. Overall, convexity seems more useful than concavity: but linear objective functions are both concave and convex, so we get all the advantages of both.