Mathematical Programming	**Lecture 17 OR 630 Fall 2005**	**October 25, 2005**
Notes by Ilya Alexandrovich Sheynzon		

Network problems and the Network Simplex Algorithm

(Chapter 7 of Bertsimas-Tsitsiklis, Chapter 19 of Chvatal).

A directed graph is a pair $G = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ (usually $\{1, 2, 3, .., n\}$) is a finite set of *nodes* and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of *arcs*. *Loops* (arcs (i, i)), but not parallel edges (two arcs (i, j)) (easy to adapt), and opposite arcs ((i, j) and (j, i)) are allowed. Arc (i, j) is from i to j; i is its tail and j is its head; and i and j are its endpoints.

A *walk* in G is a sequence $i_0, e_1, i_1, e_2, .., e_k, i_k$, where each $i_l \in \mathcal{N}$ and each $e_l \in \mathcal{A}$, with either $e_l = (i_{l-1}, i_l)$ (a *forward* arc) *or* $e_l = (i_l, i_{l-1})$(a *reverse* arc). The walk is from i_0 to i_k. A walk is a *path* if all the i_l's are distinct. It is a *cycle* if $i_1, i_2, .., i_k$ are distinct, but $i_0 = i_k, k \geq 1$, and if $k = 2$, $e_1 \neq e_2$.

A walk, path, or cycle is *directed* if all its arcs are forward. G is *connected* if there is a walk (equivalently a path) from every node to every other node. G is *acyclic* if it has no cycle.

$G' = (\mathcal{N}, \mathcal{A}')$ with $\mathcal{A}' \subseteq \mathcal{A}$ is a (spanning) subgraph of G. A graph is a tree if it is connected and acyclic. A spanning subgraph of G that is itself a tree is called a *spanning* tree of G.

For $j \in \mathcal{N}$, the outdegree of j is $|\{k \in \mathcal{N} : (j, k) \in \mathcal{A}\}|$ and indegree of j is $|\{i \in \mathcal{N} : (i, j) \in \mathcal{A}\}|$. The degree is the sum of these.

A *network* is a directed graph G together with additional data associated to the nodes and arcs. We will consider a vector b indexed by the nodes (b_i is the net supply at node i) and a vector c of costs (and possibly u of capacities) indexed by $\mathcal{A}$ (c_{ij} is the cost of arc (i, j)).

Let $\mathbf{R}^{\mathcal{A}} := \left\{ w = (w_{ij})_{(i,j)\in\mathcal{A}},\ \text{all } w_{ij} \in \mathbf{R} \right\}$ (this can also be thought of as all functions from $\mathcal{A}$ to $\mathbf{R}$). So $c, u \in \mathbf{R}^{\mathcal{A}}$, $b \in \mathbf{R}^{\mathcal{N}}$ $(= \mathbf{R}^n$ if $\mathcal{N} = \{1, 2.., n\})$.

A *feasible flow* is a vector $x \in \mathbf{R}^{\mathcal{A}}$ satisfying

$$\sum_{k:(j,k)\in\mathcal{A}} x_{jk} - \sum_{i:(i,j)\in\mathcal{A}} x_{ij} = b_j$$

for all $j \in \mathcal{N}$, $x \geq 0$ (or $0 \leq x \leq u$) (flow conservation at node j).

$j \in \mathcal{N}$ is a *source* if $b_j > 0$, a *sink* if $b_j < 0$, and a *transshipment node* if $b_j = 0$. We want a feasible flow with minimal cost

$$c^T x = \sum_{(i,j)\in\mathcal{A}} c_{ij} x_{ij}.$$

Associated with G is its *node-arc incidence matrix* A with rows indexed by $\mathcal{N}$ and columns by $\mathcal{A}$ with

$$a_{i,(j,k)} = \begin{cases} 0, & \text{if} \quad i \notin \{j, k\} \quad \text{or} \quad j = k \\ +1, & \text{if} \quad i = j \neq k \\ -1, & \text{if} \quad i = k \neq j. \end{cases}$$

If G has n nodes and m arcs, A is $n \times m$, and each column has 2 nonzeroes (if not a loop), a $+1$ at its tail row and a -1 at its head row (similarly, such a matrix A defines a directed graph G). Then feasible flows are $\{x : Ax = b, x \geq 0\}$.

Note: the sum of all the rows of A is the zero vector. So $\text{rank}(A) < n =$ the number of rows, and also $\sum_{j \in \mathcal{N}} b_j = 0$ is a necessary condition for the existence of a feasible flow. Henceforth, assume $\sum_{j \in \mathcal{N}} b_j = 0$.

Examples

a) Shortest path.

Set $b_j = \begin{cases} +1, & \text{at the initial node} \\ -1, & \text{at the final node} \\ 0, & \text{elsewhere.} \end{cases}$

b) Max flow.

Try to maximize the flow from source s to a sink t, with capacity restrictions.

The max flow problem can be reformulated as a network flow problem, by adding the arc (t, s) with a cost $c_{ts} = -1$ and putting cost 0 on all other arcs, with $b_i = 0$ for all i and upper bounds equal to the capacity on all arcs except (t, s).

c) Transportation problem.

$$\mathcal{N} = \{1, 2, .., m, 1', 2', .., n'\},$$

$$\mathcal{A} = \{(i, j') : 1 \leq i \leq m, 1 \leq j \leq n\}, \ G \text{ is } \textit{bipartite},$$

$$b_i = s_i > 0, 1 \leq i \leq m,$$

$$b_j = -d_j < 0, 1 \leq j \leq n.$$

We want $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$. If $m = n$ and all s_i's, d_j's are equal to 1, this is the assignment problem. If we have a transportation problem where the total flow out of node i is at most s_i and the total flow into node j' is at least d_j, where the total supply exceeds the total demand and all costs c_{ij} are nonnegative, we can make this into a regular transportation problem by introducing a dummy demand of $d_0 = \sum_{i=1}^{m} s_i - \sum_{j=1}^{n} d_j$.

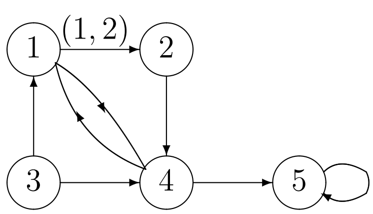

Figure 1: *A directed graph.*

$1, (1, 2), 2, (2, 4), 4$ *is a path from* 1 *to* 4.

$1, (1, 2), 2, (2, 4), 4, (4, 1), 1$ *is a directed cycle; so is* $5, (5, 5), 5$ *using the loop* $(5, 5)$.

$1, (1, 2), 2, (2, 4), 4, (3, 4), 3, (3, 1), 1$ *is a not directed cycle.*

$(3, 4)$ *is a reverse arc.*

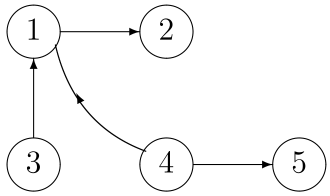

Figure 2: *A spanning tree.*

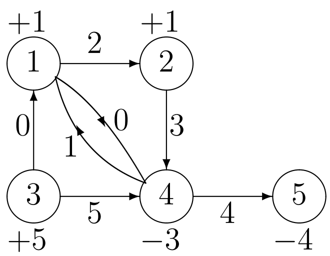

Figure 3: *A network corresponding to this directed graph with the loop omitted; a feasible flow is shown.*

Figure 4: *A directed graph corresponding to a transportation problem with* m *suppliers and* n *consumers. If the total demand is less than total supply, then we can create a dummy sink* $0'$ *to absorb the difference between the two. The transportation costs for the units sent to* $0'$ *should be set to zero. Using the same argument, we can deal with network problems with inequalities where the total demand is less than the total supply.*