Chapter 5: *The Network Simplex Algorithm*

In this chapter, we will consider the transshipment problem:

Given a network $G = (V, E)$, demands $b = (b_i : i \in V)$, and costs $c = (c_j : j \in E)$ (which in this chapter may be negative), find an $x = (x_j : j \in E)$ which:

(5.0) minimizes $cx = \sum_{j \in E} c_j \, x_j$ subject to

(5.1) $\forall i \in V, \ \sum_{t(j)=i} x_j - \sum_{h(j)=i} x_j = b_i;$

(5.2) $\forall j \in E, \ x_j \geq 0.$

An $x = (x_j : j \in E)$ satisfying (5.1) - (5.2) is called a *feasible* (transshipment) *flow*. An $x = (x_j : j \in E)$ satisfying (5.0) - (5.2) is called an *optimum (transshipment) flow*.

We will assume $\sum_{i \in V} b_i = 0$, since otherwise there is no x satisfying (5.1). (Be sure you can prove this.)

We will usually consider circuits with one of the two possible orientations specified. This classifies the edges of the circuit as *forward* or *backward* edges.

A network is *connected* if there is a path joining every pair of vertices. We will assume our networks are connected.

A *tree* is a connected network which contains no circuits.

A *spanning tree* in network G is a tree which meets every vertex of G.

Basic x's

An $x = (x_j : j \in E)$ which satisfies the demands $b = (b_i : i \in V)$ (i.e. satisfies (5.1)) but which may not satisfy $x \geq \underline{0}$ is called *basic* if there is a spanning tree T of G such that $x_j = 0$ for each edge j not in T. T is called a *basis*. If the basic x for T is non-negative, then T is called a *feasible basis*.

(5.3) **Theorem**. For any spanning tree T of a connected network G with $\sum_{i \in V} b_i = 0$, there is exactly one $x = (x_j : j \in E)$ which is basic for T.

Given a spanning tree T in a network G, one can determine the x basic for T by using the following:

(5.4) For any tree T, there is some vertex i such that exactly one edge of T meets i.

To find the x basic for T, set $x_j = 0$ for edges j not in T.

(5.5) Choose a vertex i such that exactly one edge, say e, of T meets i. Every edge j of G meeting i except edge e has its x_j already determined. Thus x_e is uniquely determined so the demand at vertex i will be satisfied.

Delete edge e from T, but not from G (you will need to remember its x_e value), and repeat from (5.5).

For any tree T, (the number of vertices of T) = (the number of edges of T) + 1. Above, we have repeatedly used a new vertex i to determine x_e for some $e \in T$, and it should be clear that our x satisfies the demand at vertex i. But when we are finished, there is one vertex, say u, which we have not considered. Convince yourself that (assuming $\sum_{i \in V} b_i = 0$) the demand at vertex u is also met by x.

Here is a way to describe the x basic for T. When an edge j is deleted from a tree T, T falls into two connected parts, called connected components. One of these components will contain $h(j)$, the other will contain $t(j)$.

$$(5.6)\quad x_j = \begin{cases} 0 \text{ for } j \notin T \\ \sum\{b_i : i \text{ is in the component of } T - j \text{ containing } t(j)\} \text{ for } j \in T \end{cases}$$

An x which satisfies the demands is basic for one or more spanning trees T of G if and only if there is no circuit P such that $x_j \neq 0$ for each edge j of P. This follows from the fact that if $\{j \in E : x_j \neq 0\}$ contains no circuit, then it can be extended (by adding more edges of G) to a spanning tree of G.

Each flow x appearing in the Network Simplex Algorithm is feasible and basic for a specified spanning tree T, called the basis. We next look at how any basic feasible flow can be replaced by a basic feasible flow which is at least as cheap.

Feasible Pivots

Let T be any spanning tree of G such that the basic solution x for T is feasible. Let $f \in E$ be any edge not in T. Let $P = P(T,\ f)$ be the circuit, oriented so that f is a forward edge, consisting of f and the unique path in T from $h(f)$ to $t(f)$. If P has no backward edge, then it is not possible to make a feasible pivot with f. If P has a backward edge, then define

(5.7) $\epsilon = \epsilon(P, x) \equiv$ minimum $\{x_j : j$ is a backward edge of $P.\}$ Note that $\epsilon(P,\ x) \geq 0$.

Define $x' = (x_j : j \in E)$ as follows:

$$(5.8)\quad x'_j = \begin{cases} x_j + \epsilon(P,\ x) & \text{for forward edges } j \text{ of } P \\ x_j - \epsilon(P,\ x) & \text{for backward edges } j \text{ of } P \\ x_j & \text{for edges } j \notin P. \end{cases}$$

Then for some backward edge g of P, $x'_g = 0$. $T' \equiv T\ \cup \{f\} - \{g\}$ is a spanning tree and x' is its basic solution.

This is called a *feasible pivot.*

Note that more than one backward edge j of P may have $x'_j = 0$. Any one but only one such edge is removed from T.

Be sure you see that x' is feasible. In fact, one can show more generally, the following:

(5.8) Let x be a feasible transshipment flow and P *any* circuit with specified orientation. If P has no backward edges, let ϵ be any non-negative number. If P has a backward edge, let ϵ be as defined in (5.7). Then where x' is as defined in (5.8), x' is feasible.

Simplex Pivots

Where $x,\ P,\ \epsilon$, and x' are as described in (5.8),

$$\begin{aligned} cx' &= cx + \sum_{\substack{j \text{ forward} \\ \text{edge of } P}} c_j\epsilon - \sum_{\substack{j \text{ backward} \\ \text{edge of } P}} c_j\epsilon \\ &= cx + \epsilon \underbrace{\left(\sum_{\substack{j \text{ forward} \\ \text{edge of } P}} c_j - \sum_{\substack{j \text{ backward} \\ \text{edge of } P}} c_j \right)}_{\equiv c(P)} \end{aligned}$$

(5.9) $cx' = cx + \epsilon \cdot c(P)$

In particular, (5.9) holds when we make a feasible pivot. Since $\epsilon \geq 0$, in order to try to make $cx' < cx$, we should choose an edge $j \notin T$ with $c(P(T,\ j)) < 0$.
Then if $\epsilon = \epsilon(P(T,\ j)) > 0,\ cx' < cx$, and if $\epsilon = \epsilon(P(T,\ j)) = 0,\ cx' = cx$.

If in the Network Simplex Method, f is an edge not in the tree with $c(P(T,\ f)) < 0$, but $P = P(T,\ f)$ has no backward edges, then we cannot make a feasible pivot with f. However, in this case it is easily seen (using (5.8) and (5.9)) that by increasing the flow around P by a larger and larger ϵ, we can make the total cost decrease as much as we want. Such a transshipment problem is called *unbounded*.

If there is no edge $j \notin T$ with $c(P(T,\ f)) < 0$, then we will see later that the current basic feasible flow is optimum.

A feasible pivot where we bring an edge $f \notin T$ into the basis with $c(P(T,\ f)) < 0$, is called a *simplex pivot*. The network simplex method starts with a spanning tree whose basic solution is non-negative, and makes feasible pivots until it discovers unboundedness or an optimum solution. That is:

Input: A transshipment problem
A spanning tree T whose basic solution x is $\geq \underline{0}$.

If for every edge $f \notin T$ $c(P(T,\ f)) < 0$,

then **stop**: x is an optimum transshipment flow.

Otherwise choose any edge $f \notin T$ with $c(P(T,\ f)) < 0$.

If there is no edge directed opposite to f in $(P(T,\ f))$

then **stop**: $(P(T,\ f))$ is a negative cost directed circuit and thus the given problem is unbounded.

Otherwise make a feasible pivot to bring f into the tree. Let T' be the new spanning tree and x' be the new basic solution. *Repeat* using T' and x' in place of T and x.

Vertex Numbers

We will now introduce vertex numbers $y = (y_i : i \in V)$ into the Network Simplex Algorithm. Vertex numbers serve two very valuable purposes:

(5.10) They simplify calculations in the algorithm.

(5.11) They provide a proof that for a basis T where $c(P(T,\ f)) \geq 0$ for every edge $f \notin T$, the basic feasible flow x is optimum.

Given a transshipment problem on network $G = (V,\ E)$, choose any vertex r to be the "root". Let T be a spanning tree in G. Let $y = (y_i : i \in V)$ be such that

(5.12) $y_r = 0$ and

(5.13) For every edge j of T, $\bar{c}_j = y_{t(j)} + c_j - y_{h(j)} = 0$.

Note that the y_i's are easily computed since knowing the y_i on $i = t(j)$, where $j \in T$, (5.13) says the y_i where $i = h(j)$ is:

(5.13) $y_{h(j)} = y_{t(j)} + c_j$ where $j \in T$

Similarly, knowing the y_i on $i = h(j)$, where $j \in T$, (5.13) says the y_i where $i = t(j)$ is:

(5.13) $y_{t(j)} = y_{h(j)} - c_j$ where $j \in T$

Thus since the value of one of the y_i's is known, (namely, $y_r = 0$), the values of the remaining y_i's are easily computed.

It is quite easy to see that:

(5.14) Where $y = (y_i : i \in V)$ is as defined as in (5.12) - (5.13), y is the unique vertex-numbering such that where P_{ri} denotes the unique path in T from r to i (oriented from r to i),

$$\begin{aligned} y_i &= \sum\{c_j : j \text{ is a forward edge of } P_{ri}\} - \sum\{c_j : j \text{ is a backward edge of } P_{ri}\} \\ &\equiv c(P_{ri}) \end{aligned}$$

It follows from (5.14) that:

(5.15) $\forall$ edge $j \notin T$,

$$\begin{aligned} \bar{c}_j &= c_j + y_{t(j)} + -y_{h(j)} \\ &= c_j + c(P_{h(j),t(j)}) \\ &= c(P(T,\ j)) \end{aligned}$$

where $P_{h(j),\, t(j)}$ denotes the path in T from $h(j)$ to $t(j)$ and $P(T,\ j)$ denotes the circuit consisting of $P_{h(j),\, t(j)}$ and j, oriented so that j is a forward edge.

Proof of (5.15). Let j be an edge, $j \notin T$. Let u be the last vertex common to $P_{r,\ h(j)}$ and $P_{r,t(j)}$.
Then $P(T,\ j)$ is j, followed by the reverse of $P_{u,\ h(j)}$, followed by $P_{u,t(j)}$. Thus

$$\begin{aligned} c(P(T,j)) &= c_j - c(P_{u,\ h(j)}) + c(P_{u,\ t(j)}). \\ &= c_j - c(P_{u,\ h(j)}) - c(P_{r,\ u}) + c(P_{r,\ u}) + c(P_{u,\ t(j)}). \\ &= c_j - c(P_{r,\ h(j)}) + c(P_{r,\ t(j)}). \\ &= c_j - y_{h(j)} + y_{t(j)} \ by\ (5.14) \\ &\equiv \bar{c}_j \end{aligned}$$

□

(5.16) *Optimality Theorem* Let T be a feasible basis with basic solution x. Let y be as in (5.12) - (5.13). Suppose that for every edge $j \notin T,\ c(P(T,\ j)) = \bar{c}_j \geq 0$. Then x and y satisfy the Magic Number Conditions (MNC) for transshipment, and hence x is a minimum cost transshipment flow.

Proof. Recall that the MNC for transshipment are:
(TMNC 1) $\forall j \in E,\ \bar{c}_j \equiv c_j + y_{t(j)} - y_{h(j)} \geq 0$ and
(TMNC 2) $x_j > 0, \Rightarrow\ \bar{c}_j \equiv c_j + y_{t(j)} - y_{h(j)} = 0.$

By assumption, for every edge $j \notin T,\ \bar{c}_j \geq 0$. (5.13) says that for $j \in T,\ \bar{c}_j = 0$. Thus (TMNC 1) holds. Now assume $x_j > 0$. Then $j \in T$. So $\bar{c}_j = 0$. I.e. (TMNC 2) holds.

□

This is how the y_i's make calculations in the Network Simplex Algorithm easier: To compute $c(P(T,\ f))$ directly, we must somehow "trace out" $P(T,\ f)$. We might have to do this for each edge $f \notin T$. To compute $c(P(T,\ f))$ using (5.15), we *once* "trace out" T. Then for any $f \notin T$ we only have to do an addition and a subtraction to get $\bar{c}_f = c(P(T,\ f))$.

The Network Simplex Algorithm Using Vertex Numbers
Input: A transshipment problem
A spanning tree T whose basic solution x is $\geq \underline{0}$.

Choose some vertex to be the root r. Usually we don't change r during the algorithm. Compute $y = (y_i : i \in V)$ using (5.11) - (5.12); that is: $y_r = 0$.

For $j \in T$, $\bar{c}_j \equiv y_{t(j)} + c_j - y_{h(j)} = 0$.

If for every edge $f \notin T$, $\bar{c}_j \equiv y_{t(f)} + c_f - y_{h(f)} = c(P(T, f)) \geq 0$.

then **stop**: x and y satisfy the MNC for transshipment flows, and thus x is an optimum transshipment flow.

Otherwise choose any edge $f \notin T$ with $\bar{c}_f \equiv y_{t(f)} + c_f - y_{h(f)} = c(P(T,\ f)) < 0$.
If there is no edge directed opposite to f in $P(T,\ f)$,

then **stop**: $P(T,\ f)$ is a negative cost directed circuit and thus the given problem is unbounded.

Otherwise make a feasible pivot to bring f into the tree. Let T' be the new spanning tree and x' be the new basic solution. *Repeat* using T' and x' in place of T and x.

Degeneracy and Finiteness

A simplex pivot where $\epsilon(P,\ x) = 0$ is called *degenerate*. It is possible that after a number of degenerate pivots, that the Network Simplex Algorithm will return to an earlier basis T, and then proceed to repeat itself without ever stopping. A sequence of pivots from a basis T back to T is called *cycling*.

In order to make sure that the Network Simplex Algorithm stops after a finite number of pivots, the rule for choosing a simplex pivot must be refined somehow so that each pivot goes to a pair $(T,\ x)$ not previously used. Since a basis T has at most one basic x, it is equivalent to make sure that each pivot goes to a T not previously used. Then since a network has a finite number of bases $\left(\leq \binom{|E|}{|V|-I} \text{bases}\right)$, the algorithm will stop after a finite number of pivots.

Perhaps later in the course we will have an opportunity to discuss ways to refine the pivot rule in order to guarantee the prevention of cycling. In real life, the Network Simplex Algorithm is often used without such a guarantee.

Final Bases

A basis T is called a final basis if either

(5.17) $c(P(T,\ f)) \geq 0$ for every edge $f \notin T$ or

(5.18) for some edge $f \notin T,\ c(P(T,\ f)) < 0$ and $P(T,\ f)$ has no backward edges.

(5.19) *Theorem.* For any transshipment problem with a feasible basis T, there exists a final basis. In fact, for any feasible basis T, there is some application of the Network Simplex Algorithm which takes you from T to a final basis.

(5.19) is proved using a refinement of the network simplex pivot rule which guarantees finiteness.

How to Find a Feasible Basis or Determine that There Isn't One

Given a transshipment problem on a network $G = (V,\ E)$, we will now describe another transshipment problem on a new network $\hat{G} = (V,\ \hat{E})$ such that

(5.20) the transshipment problem on $\hat{G}$ obviously has a feasible basis and

(5.21) given a basic feasible $\hat{x}$ and a $\hat{y}$ satisfying the transshipment MNC in $\hat{G}$, we can easily obtain

(5.21.1) a feasible basis for the given transshipment problem or

(5.21.2) a set $S \subseteq V$ such that $\sum\{b_i : i \in S\} > 0$ and no edge leaves S, which shows there is no feasible transshipment flow in G.

Given a transshipment problem on $G = (V,\ E)$. We assume, as usual, that $\sum\{b_i : i \in V\} = 0$. Create a new network $\hat{G} = (V,\ \hat{E})$ as follows. $\hat{G}$ has the same vertex-set as G. Choose any vertex $r \in V$ and call r the root. Add the following edges to E to get the edge-set $\hat{E}$ of $\hat{G}$.

For $i \in V - r$,

(5.22) If $b_i \leq 0$, add an edge $(r,\ i)$ with tail r and head i, unless there already is such an edge in E, and

(5.23) If $b_i > 0$, add an edge $(i,\ r)$ with tail i and head r, unless there already is such an edge in E.

We will sometimes call the edges of (5.22) - (5.23) *artificial* edges, and the edges of E *real* edges.

Let $\hat{G}$ have the same demands as G has. Assign costs $\hat{c}$ in $\hat{G}$ as follows:

$$\hat{c}_j = \begin{cases} 1 & if \quad j \in \hat{E} - E \quad (i.e \quad j \quad is \quad artificial) \\ 0 & if \quad j \in E \qquad (i.e \quad j \quad is \quad real) \end{cases}$$

Note that for any feasible flow $\hat{x}$ in $\hat{G}$, $\hat{c}\hat{x} \geq 0$.

Let T be the spanning tree of $\hat{G}$ with edges $\{(r,\ i) : b_i \leq 0\} \cup \{(i,\ r) : b_i > 0\}$. T is easily seen to be a feasible basis: the basic solution for T is:

$$\hat{x}_j = \begin{cases} |b_i| & if\ j = (r,i) \quad and \quad b_i \leq 0 \\ b_i & if\ j = (i,r) \quad and \quad b_i > 0 \\ 0 & otherwise \end{cases}$$

(5.24) *Lemma.* G has a feasible flow if and only if a minimum cost feasible flow $\hat{x}$ in $\hat{G}$ has $\hat{c}\hat{x} = 0$.

Proof. *"only if " part.* Suppose G has a feasible flow x. Then $\hat{x}_j = \begin{cases} x_j & if \quad j \in E \\ 0 & if \quad j \in \hat{E} - E \end{cases}$ is a feasible flow in $\hat{G}$ with $\hat{c}\hat{x} = 0$, which is clearly as small as possible. Note that if x is a basic flow, so is $\hat{x}$.

"if" part. Suppose $\hat{x}$ is a feasible flow in $\hat{G}$ with $\hat{x}$ in $\hat{G}$. Then no artificial edge has positive flow. So $x_j = \hat{x}_j$ for $j \in E$ is a feasible flow in G. Note that if $\hat{x}$ is basic, then x is basic for any tree obtained by extending $\{j \in E : x_j > 0\}$ to a spanning tree in G.

□

We now give an alternative proof of the "only if" part. Suppose $\hat{x}$ is a minimum cost feasible flow in $\hat{G}$ and $\hat{c}\hat{x} > 0$, and suppose also that there is a $\hat{y}$ satisfying the transshipment MNC (TMNC 1 and TMNC 2) with $\hat{x}$. Since $\hat{c}\hat{x} > 0$, there is some artificial edge a with $\hat{x}_a > 0$. Define:

$$S = \{i \in V : \hat{y}_i < \hat{y}_{h(a)}\}$$

(5.25) S satisfies:
(5.25.1) $\sum\{b_i : i \in S\} > 0$ and
(5.25.2) no edge of G leaves S

Thus there is no feasible transshipment flow in G.

Proof of (5.25)

(5.26) **Claim**: $t(a) \in S,\ h(a) \in V - S$ (i.e. a leaves S):

Proof: By (TMNC 2), $\hat{x}_a > 0 \Rightarrow \hat{y}_{t(a)} + \hat{c}_a - \hat{y}_{h(a)} = 0$. Since $\hat{c}_a = 1$, this says $\hat{y}_{t(a)} < \hat{y}_{h(a)}$, and thus by the definition of S, $t(a) \in S$. Clearly, $h(a) \in V - S$.

(5.27) **Claim**: No real edge leaves S:

Proof: Suppose j is a real edge and $t(j) \in S$. By (TMNC 1), $\hat{y}_{t(j)} + \hat{c}_j - \hat{y}_{h(j)} \geq 0$, and since $\hat{c}_j = 0$, this says $\hat{y}_{t(j)} \geq \hat{y}_{h(j)}$. Then by definition of $S,\ h(j) \in S$. So j does not leave S.

(5.28) **Claim**: If j enters S, then $\hat{x}_j = 0$:

Proof: If j enters S, i.e. if $t(j) \in V - S$, $h(j) \in S$, then by definition of S, $\hat{y}_{h(j)} \geq \hat{y}_{h(a)}$ and $\hat{y}_{t(j)} \geq \hat{y}_{h(a)}$; so $\hat{y}_{t(j)} > \hat{y}_{h(j)}$. Since $\hat{c}_j \geq 0$, we get $\hat{y}_{t(j)} + \hat{c}_j - \hat{y}_{h(j)} > 0$, which by (TMNC 2) implies $\hat{x}_j = 0$. Now

$$\begin{aligned}
\sum\{b_i : i \in S\} &= \sum\{\hat{x}_j : j \text{ leaves } S\} - \sum\{\hat{x}_j : j \text{ enters } S\} \\
&\quad \text{(you should know why)} \\
&= \sum\{\hat{x}_j : j \text{ leaves } S\} \quad \text{by (5.28)} \\
&\geq \hat{x}_a \text{ by (5.26) and since } \hat{x}_j \geq 0\ \forall j \\
&> 0 \text{ by choice of } a.
\end{aligned}$$

□

(I hope you appreciate what a nice application of the transshipment MNT this proof is.)

The Two-Phase Network Simplex Algorithm

Input: A transshipment problem on network $G = (V,\ E)$ with costs c_j and demands b_i; $\sum\{b_i : i \in V\} = 0$.

Phase I: To find a feasible basis for the given problem or decide that there isn't one.

Formulate the transshipment problem on network $\hat{G} = (V,\ \hat{E})$ as described on page 19. Let T be the feasible basis for this problem as described on page 19 and $\hat{x}$ its basic solution. Input the transshipment problem on $\hat{G}$, as well as T and $\hat{x}$ to the Network Simplex Algorithm described on page 18. (Note that this problem can not be unbounded.) Output from

the algorithm will be a spanning tree $\hat{\hat{T}}$, its basic solution $\hat{\hat{x}} \geq 0$, and a $\hat{\hat{y}}$ which satisfies the transshipment MNC with $\hat{\hat{x}}$. ($\hat{\hat{x}}$ is an optimum solution for the transshipment problem on $\hat{\hat{G}}$.)

If $\hat{\hat{c}}\hat{\hat{x}} > 0$

then there is some artificial edge a with $\hat{\hat{x}}_a > 0$. Choose any such edge a. Define $S = \{i \in V : \hat{\hat{y}}_i < \hat{\hat{y}}_{h(a)}\}$. Then S satisfies (5.25.1) and (5.25.2), so there is no feasible flow in G. **Stop**.

Otherwise $\hat{\hat{c}}\hat{\hat{x}}_a = 0$. For every $j \in E$, let $x_j = \hat{\hat{x}}_j$. Extend $\{j : x_j > 0\}$ to a spanning tree T in G by adding edges if necessary. x is the basic solution for T, and $x \geq \underline{0}$. Go to phase II.

Phase II: To find an optimum solution for the given problem or decide that it is unbounded. Input the given transshipment problem together with the spanning tree T and its basic solution $x \geq \underline{0}$ obtained in phase I to the Network Simplex Algorithm as described on p. 16-17. Output will be a spanning tree T^*, its basic solution x^* and a y^* which satisfies the transshipment MNC with x^* (thus x^* is an optimum solution for the given problem) or a negative cost directed circuit (thus the given problem is unbounded).
Note: Once an artificial edge leaves the basis in Phase I, you may erase it — you never want it to come back into the basis.

Using the Network Simplex Algorithm to Prove the Converses of the Obstruction Theorem for Feasible Transshipment Flows and of the MNT for Transshipment

(5.29) **Theorem**. If

(5.29.1) $\sum\{b : i \in V\} = 0$
and

(5.29.2) For every $S \subseteq V$ such that no edge leaves S, $\sum\{b_i : i \in S\} \leq 0$,
then there exists a feasible transshipment flow.

This can be proved using the Network Simplex Algorithm, assuming the algorithm is finite (and as we have said, there exist ways to make the algorithm finite).

Assume (5.29.1) and (5.29.2) are satisfied. Apply phase I of the Network Simplex Algorithm. It can't stop with $\hat{\hat{c}}\hat{\hat{x}} > 0$, since then there would be an $S \subseteq V$ not satisfying (5.29.2). So the algorithm produces a (basic) feasible transshipment flow.

(5.30) **Theorem**. Suppose $x^* = (x^*_j : j \in E)$ is a minimum cost feasible transshipment flow

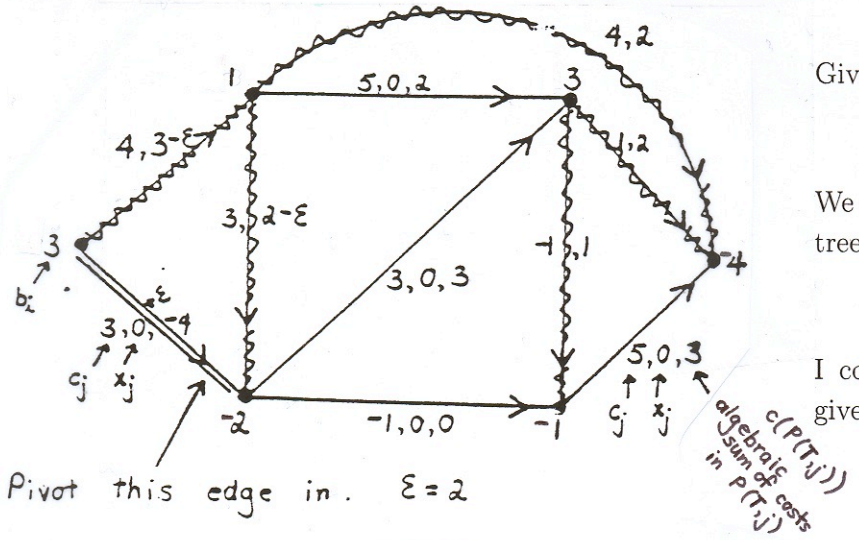
in network $G = (V,\ E)$. Then there exists $y = (y_i : i \in V)$ such that x^* and y satisfy the MNC for transshipment, i.e.:

(TMNC 1) $\forall j \in E,\ \bar{c}_j = c_j + y_{t(j)} - y_{h(j)} \geq 0$ and
(TMNC 2) $x^*_j > 0 \Rightarrow \bar{c}_j = c_j + y_{t(j)} - y_{h(j)} = 0$.

Sketch of proof. Since x^* is an optimum transshipment flow, if we apply the network simplex method to G, it will stop in phase II with an x and a y which satisfy the MNC for transshipment. It can be proved that this y and the x^* of the theorem also satisfy the MNC for transshipment.

□

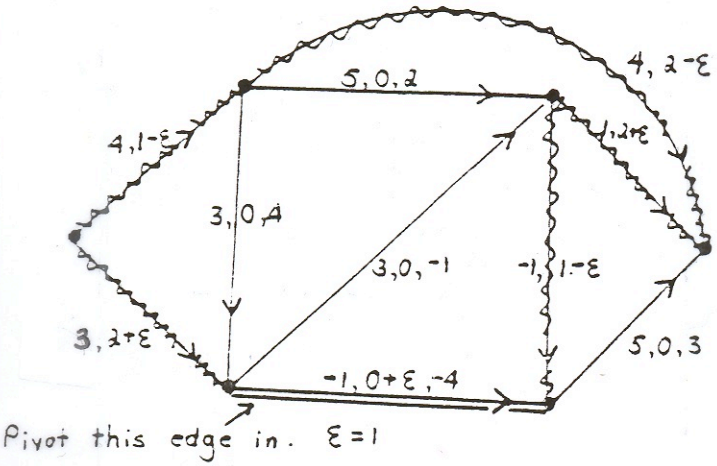
The Network Simplex Method : An Example



Given: network, costs c_j, demands b_i

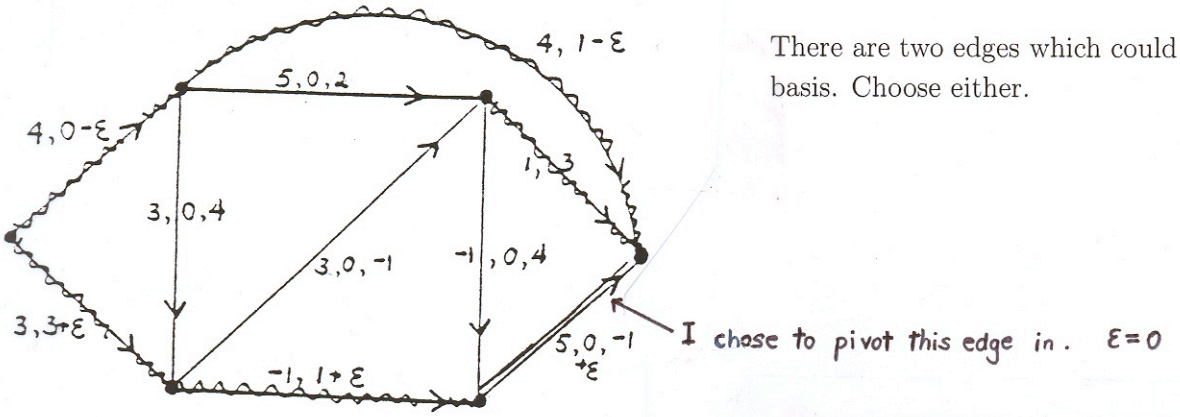
We also assume we are given a spanning tree T whose basic solution is $\geq \underline{0}$.

I computed the basic solution x for the given tree, and $c(P(T,\ j))$ for each $j \notin T$.

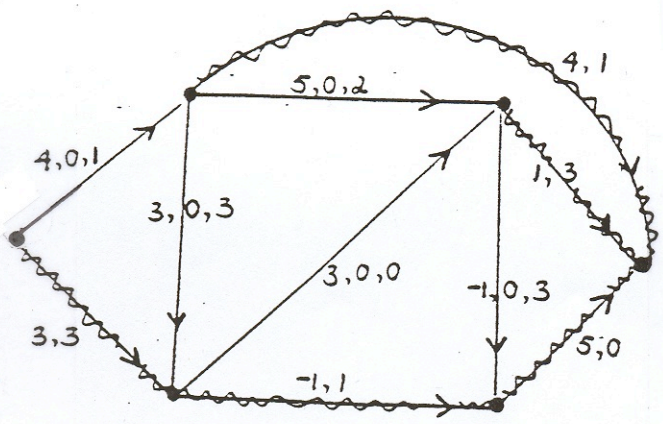


There are two edges which could leave the tree. Choose either.

(I did not rewrite the b_i's here because we no longer need them.)

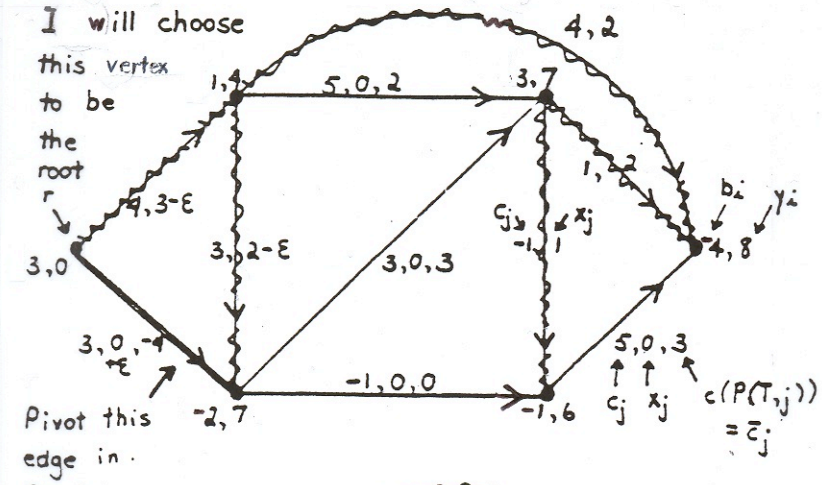


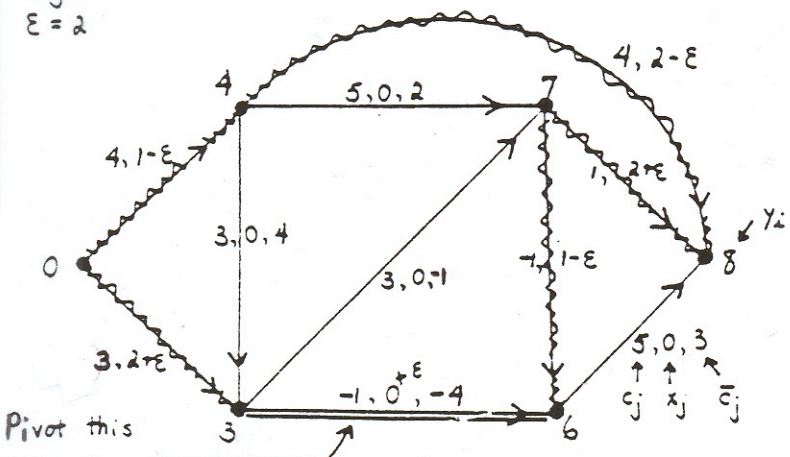
There are two edges which could enter the basis. Choose either.

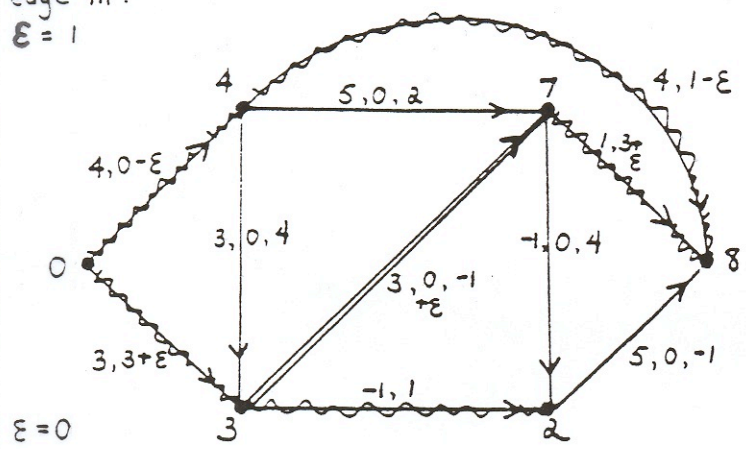


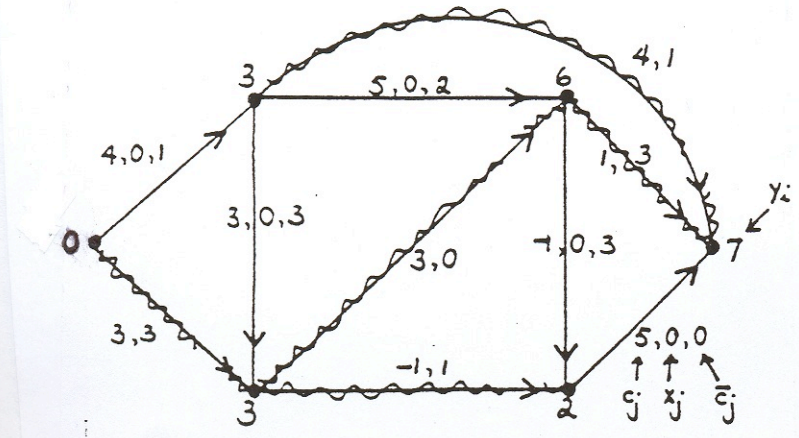
Now for each edge $j \notin T$, $c(P(T,\ j)) \geq 0$. Thus the x shown is an optimum flow. Its objective value $cx = 15$

The Network Simplex Method : An Example Using Vertex Numbers









Given: network, costs c_j, demands b_i

We also assume we are given a spanning tree T whose basic solution is $\geq \underline{0}$.

I computed the basic solution x for the given tree. I calculated y_i for each vertex i. Then I calculated $c(P(T,\ j)) = \bar{c}_j = y_{t(j)} + c_j - y_{h(j)}$ for each $j \notin T$.

(I did not rewrite the b_i's here because we no longer need them.)

Here I chose to pivot in a different edge from previous example.

Now for the each edge $j \notin T$, $\bar{c}_j = y_{t(j)} + c_j - y_{h(j)} \geq 0$. $\therefore$ Using the MNT for Transshipment Flows, we can see that x is optimum.
Note that the x here is the same as the final x on the previous page although the spanning trees are different. Two different spanning trees may have the same basic x.