Contents

Chapter 8

Generalized Network Flows

In flow problems discussed so far, we assumed that if f_{ij} units of a commodity enter an arc (i, j) at its tail node i and travel across the arc, then exactly the same f_{ij} units will reach its head node j. This assumption may not hold in some flow models. For example, in a water distribution network, if some quantity of water is shipped across an open canal linking two nodes, some is lost due to evaporation and seepage during transit, and the amount reaching the destination will only be a fraction of the amount that left the origin. The same phenomenon takes place in the transmission of electric power through high voltage transmission lines, because of transmission losses. On the other hand, if we transmit money from one period to the next by holding it in a bank account (cash flow), because of the interest earned, the amount reaching the destination will be more than the amount that left the origin. In all these examples there exists a positive **multiplier** p_{ij} associated with arc (i, j) such that if a packet of f_{ij} units of the commodity enter the arc (i, j) at node i, and travels through the arc, then by the time it reaches node j, the packet contains $p_{ij}f_{ij}$ units of the commodity. If $0 < p_{ij} < 1$, arc (i, j) is said to be **lossy**; and if $1 < p_{ij} < \infty$, it is said to be **gainy**. In pure networks studied so far, $p_{ij} = 1$ for all arcs (i, j), flow problems on them have been called **pure network flow problems**. If $p_{ij} \neq 1$ for at least one arc, the network is called a **generalized network,** *or a* **network with multipliers,** *or a* **network with gains or losses**, and flow problems on it are called **generalized**

network flow problems.

We assume that the flow variable f_{ij} associated with an arc (i, j) in a generalized network always refers to the amount of material entering this arc at its tail node i for transit to node j. We also assume that all the data on this arc (lower bound, capacity, cost coefficient, multiplier) applies to this variable. Let $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell = (\ell_{ij}), k = (k_{ij}), p = (p_{ij}), c = (c_{ij}), \breve{s}, \breve{t})$ be a connected directed generalized network with $|\mathcal{N}| = n \geqq 2, |\mathcal{A}| = m$, p as the vector of multipliers associated with the arcs in $\mathcal{A}$, and the usual meaning for the other symbols. Let $v_{\breve{s}}, v_{\breve{t}}$ denote the amounts of material leaving the source node $\breve{s}$, and arriving at the sink node $\breve{t}$ respectively. Then the flow vector $f = (f_{ij})$ is feasible in G if it satisfies

$$
\begin{aligned}
-\sum_{j \in \mathbf{A}(i)} f_{ij} + \sum_{j \in \mathbf{B}(i)} p_{ji} f_{ji} \quad &= \quad \begin{cases} -v_{\breve{s}} & \text{if } i = \breve{s} \\ \\ 0 & \text{if } i \neq \breve{s} \text{ or } \breve{t} \\ \\ v_{\breve{t}} & \text{if } i = \breve{t} \end{cases} \\
\ell_{ij} \leqq f_{ij} \leqq k_{ij} \quad &, \quad \text{for all } (i, j) \in \mathcal{A}
\end{aligned}
\tag{8.1}
$$

where $\mathbf{A}(i)$, $\mathbf{B}(i)$ are the after i, and before i sets in G. Because of the multipliers, $v_{\breve{t}}$ may not be equal to $v_{\breve{s}}$ in (8.1). In the coefficient matrix of the equality constraints in (8.1), each column has exactly 2 nonzero entries, one of them a "-1," and the other the positive multiplier associated with the arc in G corresponding to this column. We may be interested in maximizing $v_{\breve{t}}$ given $v_{\breve{s}}$ subject to (8.1), this problem is known as the **minimum loss problem**. A feasible flow vector which maximizes $v_{\breve{t}}$ is known as a **maximum feasible flow vector**. Among all maximum feasible flow vectors, the one which is associated with the smallest value of $v_{\breve{s}}$ is known as an **optimum maximum feasible flow vector**. The minimum cost flow problem in G deals with minimizing $\sum(c_{ij} f_{ij}$: over $(i, j) \in \mathcal{A})$ subject to (8.1), given $v_{\breve{t}}$ or $v_{\breve{s}}$. This is the general problem that we will consider.

Sometimes there may be gains or losses occurring at the nodes, on the material passing through them. On such networks, replace each node i associated with a gain or loss factor, by an arc (i_1, i_2) with the new nodes i_1, i_2 representing the receiving and departing ends of the original node i as discussed in Section 2.1, and make the multiplier of this arc (i_1, i_2) equal to the gain or loss factor at i. In the modified network only arcs have multipliers.

THEOREM 8.1 *Let A be the $n \times m$ coefficient matrix of the system of equality constraints in (8.1). Each row (column) is associated with a node in $\mathcal{N}$ (arc in $\mathcal{A}$). If* G *is connected, the rank of A is $n-1$ or n.*

Proof Let $\mathbb{T}$ be a spanning tree in G. Draw $\mathbb{T}$ as a rooted tree with any arbitrary node selected as the root node. Let B be the square matrix of order $n-1$ consisting of the columns in A associated with the arcs in $\mathbb{T}$, with the row corresponding to the root node struck off. We will now prove that the determinant of B is nonzero. Since $n \geqq 2$, there exists at least one nonroot terminal node in $\mathbb{T}$, say i_1. So, in the row corresponding to node i_1 there exists a single nonzero entry in B. Hence the determinant of B is a nonzero multiple of the determinant of the matrix obtained by striking off the row associated with node i_1, and the column of the nonzero entry in it, from B. The resulting matrix is of order $n-2$, and it is associated with the tree obtained by deleting node i_1 and the unique arc incident at it, from $\mathbb{T}$. The same process can be repeated on this matrix, and continued. After $n-2$ repetitions it leads to the conclusion that the determinant of B is nonzero. Hence the rank of A is $n-1$ or n. ∎

THEOREM 8.2 *If* G *is connected, and the rank of A, the coefficient matrix of the system of equality constraints in (8.1), is $n-1$, then (8.1) can be transformed into a pure network flow system.*

Proof Suppose the rank of A is $n-1$. So, there exists $\alpha = (\alpha_1, \ldots, \alpha_n) \neq 0$ such that

$$\alpha_1 A_{1.} + \ldots + \alpha_n A_{n.} = 0 \tag{8.2}$$

We will now show that in any α satisfying (8.2), $\alpha_i \neq 0$ for all i, by contradiction. Suppose $\alpha_n = 0$. Select a spanning tree, $\mathbb{T}$, in G and draw it as a rooted tree with n as the root node. Let B be the square matrix of order $n-1$, consisting of columns in A corresponding to arcs in $\mathbb{T}$, with the row corresponding to the root node n struck off. Since $\alpha_n = 0$, from (8.2), we have $\alpha' B = 0$, where $\alpha' = (\alpha_1, \ldots, \alpha_{n-1})$. The column in B corresponding to any in-tree arc incident at node n contains only a single nonzero entry (it lies in the row corresponding to the other node on that arc). This, and $\alpha' B = 0$ together imply that $\alpha_i = 0$ for all nodes i which are immediate successors of n in $\mathbb{T}$. In the same manner, going down $\mathbb{T}$ one level at a time, we conclude that α_i must be 0 for all i, a contradiction. Thus α_n could not have been 0. Similarly, $\alpha_i \neq 0$ for all i.

For $i = 1$ to n, multiply the equation in (8.1) corresponding to node i by α_i on both sides, and let A' be the matrix of coefficients of the modified system. From (8.2), the sum of the row vectors of A' is 0. Since there are exactly two nonzero entries in each column of A', this implies that the two nonzero entries in any column of A' must have the same absolute value, and opposite signs. Let γ_{ij} denote the absolute value of the nonzero entries in the column of A' corresponding to the arc $(i, j) \in \mathcal{A}$. Transform the variables using the linear transformation $f_{ij} = f'_{ij}/\gamma_{ij}$, for each $(i, j) \in \mathcal{A}$. So, if A'' is the coefficient matrix of the equality constraints after these transformations, each of its columns contains exactly two nonzero entries, a "-1" and a "$+1$." Thus A'' is the node-arc incidence of G, and therefore these transformations have converted (8.1) into a pure network flow system. ∎

Therefore, in the sequel, we will assume that the rank of A is n. Let $\mathbb{C}$ be an oriented simple cycle in G. The quantity

$$\frac{\text{Product of multipliers associated with reverse arcs in } \mathbb{C}}{\text{Product of multipliers associated with forward arcs in } \mathbb{C}} \tag{8.3}$$

is known as the **loop factor** of the cycle $\mathbb{C}$ under this orientation. If the orientation of $\mathbb{C}$ is reversed its loop factor gets inverted. As an example, consider the cycle in the subnetwork on the left in Figure 8.1 oriented so that (1, 2) is a forward arc. The numbers on the cycle arcs are the multipliers. Arcs (1, 2), (3, 4), (4, 5) are the forward arcs; and arcs (3, 2), (1, 5) are the reverse arcs. The loop factor of this cycle is $(5(4))/(2(1/2)1) = 20$. If the orientation of this cycle is reversed, its loop factor becomes $1/20$.

Similarly, let $\mathcal{P}_{ji}$ be a simple path from i to j in G. Then the following quantity is known as the **path factor** of $\mathcal{P}_{ji}$ in G

$$\frac{\text{Product of multipliers associated with reverse arcs on } \mathcal{P}_{ji}}{\text{Product of multipliers associated with forward arcs on } \mathcal{P}_{ji}} \tag{8.4}$$

THEOREM 8.3 *The generalized network flow system (8.1) can be transformed into a pure network flow system iff there exists nonzero node weights d_i for $i \in \mathcal{N}$, such that*

$$(d_i p_{ij})/d_j = 1 \text{ for each } (i,j) \in \mathcal{A} \tag{8.5}$$

Proof Define $V_i = 0$ for $i \neq \breve{s}$ or $\breve{t}$, $-V_{\breve{s}}$ for $i = \breve{s}$, and $-V_{\breve{t}}$ for $i = \breve{t}$. If node weights d_i satisfying (8.5) exist, multiply the equation corresponding to node i by $1/d_i$ on both sides. Since $p_{ij} = d_j/d_i$ by (8.5), this leads to

$$-\sum_{j \in \mathbf{A}(i)} (f_{ij}/d_i) + \sum_{j \in \mathbf{B}(i)} (f_{ji}/d_j) = (V_i/d_i), \text{ for } i \in \mathcal{N} \tag{8.6}$$

Let $f'_{ij} = (f_{ij}/d_i)$, for $(i,j) \in \mathcal{A}$. In terms of the new variables f'_{ij}, (8.6) are the conservation equations in a pure network flow problem.

Conversely, suppose (8.1) can be transformed into a pure network flow system. Then, the rank of A, the coefficient matrix of the system of equality constraints in (8.1), must be $n-1$. So, there must exist $\alpha = (\alpha_1, \ldots, \alpha_n) \neq 0$ such that when you multiply the ith equation in (8.1) by α_i and add over $i \in \mathcal{N}$, we get " $0 = 0$," i.e., $\alpha_j p_{ij} - \alpha_i = 0$, for each $(i,j) \in \mathcal{A}$. As in the proof of Theorem 8.2, it can be shown that $\alpha_i \neq 0$ for each $i \in \mathcal{N}$ in such a vector α. So, $(\alpha_j p_{ij})/\alpha_i = 1$, for each $(i,j) \in \mathcal{A}$. Hence, selecting $d_i = 1/\alpha_i$ for each $i \in \mathcal{N}$, we have the node weights d_i satisfying (8.5). ∎

THEOREM 8.4 *Node weights satisfying (8.5) exist iff the loop factor associated with every cycle in* G *is equal to 1.*

Proof Suppose node weights d_i satisfying (8.5) do exist. Since $p_{ij} = d_j/d_i$ for all $(i,j) \in \mathcal{A}$, by (8.5), the loop factor of any cycle $\mathbb{C}$ in G is 1.

To prove the converse, suppose the loop factor of every cycle in G is 1. Select a spanning tree ,$\mathbb{T}$, in G. Set $d_1 = 1$, and determine d_i for $i \neq 1$ from $(d_i p_{ij})/d_j = 1$ for each in-tree arc (i,j), uniquely. If (r,u) is an out-of-tree arc, by the hypothesis, the loop factor of the fundamental cycle of (r,u) wrt $\mathbb{T}$ is also 1, this implies $(d_r p_{ru})/d_u = 1$. Hence (8.5) holds for all $(i,j) \in \mathcal{A}$ when d is determined as here. ∎

In general, the constraints in a generalized network flow problem may be inequalities. For example, the net shipment out of a source node could be $\leqq$ the amount available at it. Or, the net shipment reaching a sink node could be $\geqq$ the requirement there. Each inequality constraint leads to a slack variable in the system when all the constraints are written as equations, and the column associated with that slack variable contains only one nonzero entry, a $+1$ or -1. If this nonzero coefficient is in row i, that column corresponds to the **self loop** (i,i) at node i. It is said to be a **surplus self loop** if the nonzero coefficient in its column is -1, **slack self loop** if that entry is $+ 1$. The multiplier associated with a self loop is always taken to be $+ 1$, whether it is a surplus or a slack self loop. Hence, for all self loops (i,i) in the network, $p_{ii} = +1$.

Thus the generalized network G $= (\mathcal{N}, \mathcal{A})$ on which our problem is defined is a directed network which may have self loops, and multiple arcs joining the same pair of nodes with the same or different orientations. To handle the self loops, we modify the definitions of the before and after sets for any node i in G to be the following:

$$\begin{aligned} \mathbf{A}(i) &= \text{Set of head nodes on arcs incident out of } i\text{; and } i \\ &\quad\ \text{if there is a surplus self loop at } i \\ \mathbf{B}(i) &= \text{Set of tail nodes on arcs incident out of } i\text{; and } i \\ &\quad\ \text{if there is a slack self loop at } i \end{aligned} \tag{8.7}$$

We will consider the problem with the general objective function in G, it is of the form (8.8) given below, and is called the **minimum cost generalized network flow problem**. In (8.8), b_i represents the requirement (negative of the exogenous flow) at node i. This problem encompasses all capacitated or uncapacitated LPs in which the coefficient matrix contains at most two nonzero entries of opposite signs in each column (the negative entry can be transformed into -1 by scaling the associated variable appropriately, after these transformations the LP assumes the form (8.8)).

$$\begin{array}{rrcl}
\text{Minimize } z(f) = \sum(c_{ij} f_{ij} : \text{ over } (i,j) & \in & \mathcal{A}) & \\
\text{subject to } \quad -\sum_{j \in \mathbf{A}(i)} f_{ij} + \sum_{j \in \mathbf{B}(i)} p_{ji} f_{ji} & = & b_i, \text{ for } i \in \mathcal{N} & (8.8) \\
\ell_{ij} \leqq f_{ij} \leqq k_{ij}, \text{ for all } \quad (i,j) & \in \mathcal{A} & &
\end{array}$$

If $\ell_{ij} = k_{ij}$ for some $(i,j) \in \mathcal{A}$, then f_{ij} could be fixed equal to their common value and eliminated from the problem. So, in the sequel we assume that $\ell < k$.

THEOREM 8.5 *Let* $\mathbb{C}$ *be a simple cycle in* G *which is not a self loop. Let* $\mathbf{\Gamma}$ *be the set of column vectors associated with the variables* f_{ij} *for arcs* (i,j) *on* $\mathbb{C}$*, in (8.8).* $\mathbf{\Gamma}$ *is linearly dependent iff* Δ*, the loop factor of* $\mathbb{C}$ *is 1.*

Proof Select an arc, say $e_1 = (1,\ 2)$, on $\mathbb{C}$ and orient it so that this arc is a forward arc. Suppose under this orientation, the sequence of arcs on $\mathbb{C}$ is $e_1, \dots, e_r$, and the sequence of nodes is 1, 2, $\dots$, r. $\mathbf{\Gamma}$ is linearly dependent iff there exists $(\alpha_1, \dots, \alpha_r) \neq 0$ such that $\sum_{t=1}^{r} \alpha_t($ column of e_t in (8.8)$) = 0$. This equation holds iff when we make the flow amount on e_t equal to α_t for $t = 1$ to r, conservation holds at all the nodes $1, \dots, r$. To check whether we can find a nonzero flow vector on $\mathbb{C}$ that maintains flow conservation at all the nodes, we select the initial arc, $e_1 = (1,\ 2)$, and fix the flow amount on it to be 1. This brings the amount p_{12} to node 2. If the next arc e_2 is a forward arc (i.e., $e_2 = (2,\ 3)$), for conservation to hold at node 2, the flow on it must be equal to p_{12}, and this brings $p_{12}p_{23}$ to node 3. On the other hand, if e_2 is the reverse arc (3, 2), for conservation to hold at node 2, the flow on it must be $-p_{12}/p_{32}$, this leaves an amount of p_{12}/p_{32} at node 3 which has to be shipped out of the other arc incident at it for conservation to hold. Continuing this procedure, we can determine the flows on all the arcs on $\mathbb{C}$. When this procedure is completed by going around $\mathbb{C}$ once, and we come back to the initial node 1, we determine the flow on the initial arc (1, 2) for conservation to hold at node 1; that flow amount, β, say, must turn out to be 1, the same quantity that we fixed it to be at the beginning of this procedure. Otherwise there is no nonzero flow on $\mathbb{C}$ that maintains conservation at all the nodes on $\mathbb{C}$.

Let $\mathcal{P}_{i1}$ denote the path from 1 to i as we travel along the oriented cycle $\mathbb{C}$ from node 1 to node i.

Let e_t be any arc on $\mathbb{C}$, i its tail node, and let $\gamma_t = 1/$(path factor of $\mathcal{P}_{i1}$). It can be verified that the flow on e_t obtained in this procedure is $-\gamma_t$ if e_t is a reverse arc on $\mathbb{C}$, or γ_t if e_t is a forward arc on $\mathbb{C}$. So, $\beta = 1/\Delta$. Hence, a nonzero flow vector maintaining conservation at all the nodes exists in $\mathbb{C}$ iff $1/\Delta = 1$, i.e., iff $\Delta = 1$. Therefore, $\mathbf{\Gamma}$ is linearly dependent iff $\Delta = 1$. ∎

There are algorithms for solving the maximum flow problem in generalized networks, or the optimum maximum flow problem, based on FAPs obtained by using a labeling method or the shortest chain method, but we will not discuss these special methods. Instead we discuss an implementation of the primal simplex algorithm to solve (8.8) using tree labels without the need for computing the basis inverse in any step. This algorithm is general enough to handle any of these generalized network flow problems.

8.1 The Primal Simplex Method for Generalized Network Flow Problems

A **basis** for (8.8) is a square nonsingular submatrix of order n of the coefficient matrix of the system of equality constraints in it. The **basis network** corresponding to a basis B, denoted by G_B, is the subnetwork of G consisting of the arcs corresponding to columns of B. Arcs in the basis network are called **basic arcs**, those not in the basis network are called **nonbasic arcs**.

THEOREM 8.6 *A basis network* G_B *for (8.8) may consist of several connected components. Each connected component of* G_B *consists of a tree plus an additional arc which may be a self loop, and hence contains a unique loop or cycle.*

Proof There are n columns in B, so if G_B is connected it is a connected network consisting of a spanning tree with one extra arc, and hence the statement of the theorem holds in this case.

Suppose G_B consists of t connected components, with the gth one consisting of r_g nodes for $g = 1$ to t. Since the gth component is connected, it must contain at least $r_g - 1$ arcs. The set of row vectors, $\mathbf{S}$, of the basis B corresponding to the r_g nodes in this component, must be linearly independent, as B is a basis. If this component contains only $r_g - 1$ arcs, there are only $r_g - 1$ columns of B which contain nonzero entries in rows of this set $\mathbf{S}$, this implies that the rank of $\mathbf{S}$ is $\leqq r_g - 1$, a contradiction. So, each connected component of G_B must contain at least as many arcs as nodes. Since B is square, the number of nodes and arcs in G_B are equal. These facts imply that each connected component of G_B has the same number of arcs as nodes, and hence it is a spanning tree in these nodes with one additional arc. This additional arc may either be a self loop or an out-of-tree arc. Hence each connected component of G_B contains a unique loop or cycle. ∎

$$B = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_t \end{pmatrix}$$

$$B_g = \begin{pmatrix} a_1 & & & & & \\ -1 & a_2 & & & & \\ & -1 & & & & \\ & & \ddots & & & \\ & & & -1 & a_{r_g-1} & \\ & & & & -1 & \boxed{a_{r_g}} \\ & & & & & \\ & & & & & \text{self loop} \\ & & & & & \text{portion} \end{pmatrix}$$

Rearrange the rows of the basis B so that the rows corresponding to nodes in each connected component of G_B appear consecutively. Now rearrange the columns of B so that the columns associated with arcs in each connected component of G_B appear consecutively; and in the same order in which these components appear among the rows. Then clearly B takes the block diagonal form shown above.

Here B_g is a square nonsingular matrix of order r_g for $g = 1$ to t, and all elements in B outside these diagonal blocks are 0. B_g is itself a triangular matrix if the loop in the connected component of G_B corresponding to it is a self loop. In this case, the rows and columns of B_g can be rearranged so that it has the structure given above, where a_{r_g} is either $+1$ or -1, and is the unique nonzero entry in the column in B corresponding to this self loop. $a_1, \ldots, a_{r_g-1}$ are the multipliers associated with the other arcs in this connected component.

If the loop in the gth connected component of G_B is a cycle which is not a self loop, the rows and columns of B_g can be rearranged so that it has the following structure.

$$B_g = \left(\begin{array}{cccc|ccccc|}
a_1 &&&&&&&& \\
-1 & a_2 &&&&&&& \\
 & -1 &&&&&&& \\
 && \ddots &&&&&& \\
 &&& a_{w-1} &&&&& \\ \cline{5-9}
 &&& -1 & a_w &&&& -1 \\
 &&&& -1 & a_{w+1} &&& \\
 &&&&& -1 && & \\
 &&&&&& \ddots && \\
 &&&&&&& -1 & a_{r_g} \\ \cline{5-9}
 &&&& \multicolumn{5}{c}{\text{The cycle portion}}
\end{array}\right) \qquad (8.9)$$

Columns 1 to $w-1$ in (8.9) correspond to arcs in the gth connected component of G_B which are not on the cycle, the remaining columns w to r_g correspond to arcs on the cycle. If the "-1" entry in row w and the last column is made 0 in (8.9), then B_g becomes triangular. Hence in this case B_g is said to be **near triangular**.

A **quasitree** in G is a partial subnetwork $(\hat{\mathcal{N}}, \hat{\mathcal{A}})$ with $\hat{\mathcal{N}} \subseteq \mathcal{N}$, and $\hat{\mathcal{A}}$ consisting of the arcs in a tree spanning the nodes in $\hat{\mathcal{N}}$, plus either a self loop at one of the nodes in $\hat{\mathcal{N}}$, or an arc joining two nodes in $\hat{\mathcal{N}}$. By Theorem 8.6 every connected component in the basis network G_B is a quasitree. Thus a basis network for the generalized network flow problem (8.8) is not a spanning tree as in pure network flow problems, but a set of disjoint quasitrees. We will now discuss how quasitrees can be stored and manipulated using node labels just as trees were. Node labels for storing each quasitree are derived separately as described below.

When all the cycle arcs are deleted from a quasitree leaving all the nodes as they are, we are left with a set of **tributary trees**, some of which may consist of a single node on the cycle. See Figures 8.1, 8.2. For each tributary tree choose the node in it that belongs to the cycle as the root, and generate the predecessor, successor, and brother labels for all the nonroot nodes exactly as in Section 1.2. If the cycle is a self loop, at node i say, make the predecessor index of i to be $+i$ or $-i$ depending on whether the nonzero entry in the column corresponding to it is $+1$ or -1. If the cycle is not a self loop, let the nodes on it be $i_1, \ldots, i_h$ in this order when the cycle is written as a path from some node i_1 back to itself with some orientation. Then for $r = 1$ to h make the predecessor index of node i_r either $+i_{r-1}$ or $-i_{r-1}$ depending on whether the arc joining them is a forward or reverse arc under the orientation chosen for the cycle, where $i_0 = i_h$. Thus each node on the cycle is its own ancestor. The predecessor indices of nodes on the cycle can be used to trace these nodes uniformly in a clockwise or counterclockwise direction. Also, each node in the cycle has an equal status as an ancestor of all nodes in the quasitree. For each $r = 1$ to h, list i_r as the youngest son of i_{r-1} (so, i_r is younger to every immediate successor of i_{r-1} in the tributary tree rooted at it in the quasitree).

These labels are said to define the **rooted loop labeling** of the quasitree. In this labeling, every node has a predecessor, and hence the predecessor index of no node is empty. As an example consider the basis network given in Figure 8.1. There are two quasitrees in it. In the left hand quasitree there is a cycle consisting of nodes 1, 2, 3, 4, 5, and of these only 1, 3 have tributary trees with one or more arcs (see Figure 8.2). The second quasitree on the right has a self loop at node 17 which is a slack self loop. The node labels corresponding to

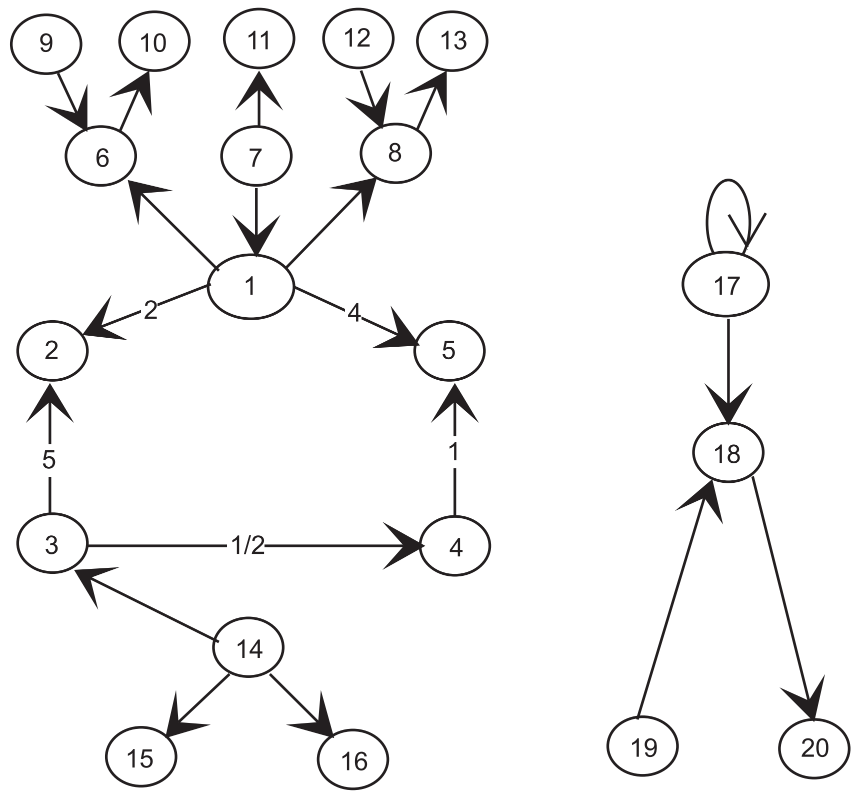

Figure 8.1: A basis network with two quasitrees. The numbers on the cycle arcs are the arc multipliers.

this basis are given in the following table (empty labels are left blank in the table).

To find the cycle or the loop in any quasitree using the predecessor indices, select any node, say i, in this quasitree and trace its predecessor path. Keep tracing this path until at some stage a node on this path, say i_1, appears for a second time (this will always happen). The cycle in this quasitree consists of all the arcs and nodes obtained in this process after i_1 has been obtained for the first time. The entire path traced in this process beginning with node i is known as the **predecessor path** of i, it duplicates no arcs, and it contains the cycle or loop in this quasitree. As an example, the predecessor path of node 12 in the

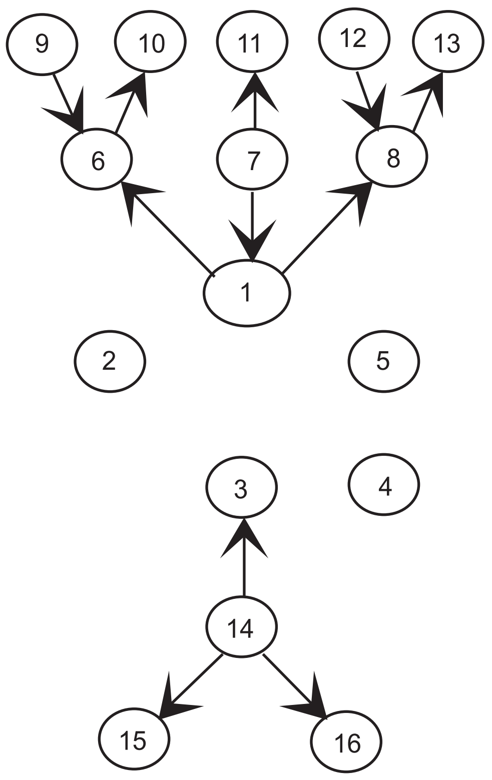

Figure 8.2: The tributary trees in the quasitree on the left of Figure 8.1.

Node	1	2	3	4	5	6	7	8	9	10
PI	− 5	1	− 2	3	4	1	− 1	1	− 6	6
SI	− 8	3	14	− 5	1	−10	−11	−13		
EBI		6		14		7	8		10	
YBI						2	6	7		9

Node	11	12	13	14	15	16	17	18	19	20
PI	7	− 8	8	− 3	14	14	17	17	−18	18
SI				−15			−18	19		
EBI		13				15	18			19
YBI			12	4	16			17	20	

quasitree on the left hand side in Figure 8.1 is: 12, (12, 8), 8, (1, 8), 1, (1, 5), 5, (4, 5), 4, (3, 4), 3, (3, 2), 2, (1, 2), 1, and this path contains the unique cycle in this quasitree.

Thus predecessor paths in a basis network for a generalized network flow problem are elementary but not simple paths.

Node labels other than the predecessor, successor, and brother indices are sometimes used in generalized network codes to store and manipulate quasitrees. Among these are the distance or depth label, thread label, etc. The distance or depth label is defined to be 0 for every node in the cycle or self loop in a quasitree. The depth label for a non-cycle node is exactly its depth in the tributary tree in which it is contained as defined in Section 1.2.2. So, the depth label of any node in a quasitree is the number of non-cycle arcs in its predecessor path. The thread label is defined exactly as in Section 1.2.2. As an example, consider the quasitree in Figure 8.3 (arc orientations are not shown for simplicity), the depth and thread labels for nodes on it are shown in the table following the figure. With the thread label defined this way, it can be verified that the set of descendents of any node can be obtained by recursive applications of the maps $t^r(i)$ exactly as discussed in Section 1.2.2.

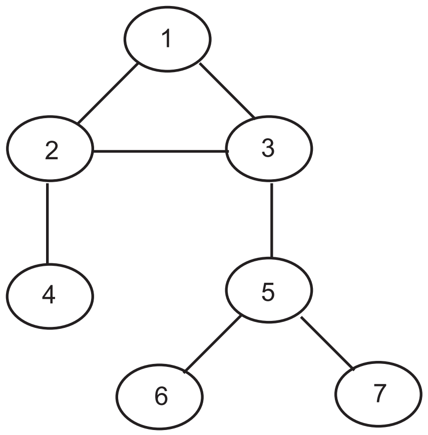

Figure 8.3:

Node	1	2	3	4	5	6	7
Predecessor label	3	1	2	2	3	5	5
Depth label	0	0	0	1	1	2	2
Thread label	2	4	5	3	6	7	1

Let A be the coefficient matrix of the system of equality constraints in (8.8). Since G is a connected generalized network with $|\mathcal{N}| = n$, rank of A is n. A basic solution for (8.8) corresponds to a partition of the arcs in G into (**Q, L, U**) where **Q** is the set of basic arcs, and **L**, **U** are the sets of nonbasic arcs on which the flows are fixed equal to the lower, upper bounds respectively. In any partition (**Q, L, U**), $|\mathbf{Q}| = n$, the matrix B consisting of the column vectors in A corresponding to arcs in **Q** is a basis for A, and every arc in **U** has finite capacity. By Theorem 8.5, the loop factors of all the cycles in the basis network ($\mathcal{N}$, **Q**) which are not self loops, are different from 1.

How To Compute The Primal Basic Solution Corresponding To A Given Partition (Q, L, U)

Let B be the basis for (8.8) corresponding to **Q**, and let $\bar{f} = (\bar{f}_{ij})$ denote the basic solution of (8.8) corresponding to the partition (**Q, L, U**). By definition, $\bar{f}_{ij} = \ell_{ij}$ if $(i, j) \in \mathbf{L}$, and $= k_{ij}$ if $(i, j) \in \mathbf{U}$. For each $(i, j) \in \mathbf{L} \cup \mathbf{U}$, multiply the column vector corresponding to it in (8.8) by $\bar{f}_{ij}$ and transfer it to the right hand side, and suppose this changes the right hand side constants vector in (8.8) to b'. b' is the vector of remaining requirements at the nodes after the flows on arcs in **L, U** are fixed at their lower bounds, capacities respectively. Now only the flow amounts on the basic arcs remain to be computed in order to satisfy these remaining requirements. These are obtained by solving

$$B(\bar{f}_{ij} : (i, j) \in \mathbf{Q}) = b' \qquad (8.10)$$

Since B has a block diagonal structure as discussed earlier, we can solve (8.10) by finding the flow amounts on arcs in each quasitree in the basis network ($\mathcal{N}$, **Q**) separately.

Consider the gth quasitree in $(\mathcal{N}, \mathbf{Q})$. Computing the flow amounts on arcs in this quasitree can be conveniently carried out in two stages. In Stage 1 flows on arcs on the tributary trees in this quasitree are determined using back substitution as discussed in Section 5.4. The Stage 1 procedure processes each non-cycle node on the tributary tree exactly once. Once a node is processed, the flow amounts on all the arcs incident at it are known. At any time in Stage 1, $\mathbf{Y}$ denotes the set of processed nodes at that time. This procedure also maintains another set $\mathbf{X}$ of non-cycle nodes in the quasitree satisfying the property that for each node in $\mathbf{X}$, the flow amounts on all the arcs incident at it are already known at this time except on the arc joining it to its immediate predecessor. After the flow amount on each arc is obtained, the requirements at the nodes on it are updated to cancel the requirements fulfilled by it, and obtain the remaining requirements still to be fulfilled, denoted by (b'_i) itself.

PROCEDURE FOR DETERMINING THE FLOW AMOUNTS ON BASIC ARCS IN A QUASITREE IN $(\mathcal{N}, \mathbf{Q})$ IN THE BASIC SOLUTION ASSOCIATED WITH THE PARTITION $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$

Stage 1 Initiate with $\mathbf{X} =$ set of non-cycle leaf nodes in the tributary trees in this quasitree, $\mathbf{Y} = \emptyset$.

In a general step, select and delete a node from $\mathbf{X}$, say i. Let j be the immediate predecessor of i. Let b'_i, b'_j be the remaining requirements at nodes i, j at this time. If the basic arc joining i and j is (i, j), make $\bar{f}_{ij} = -b'_i$ and change the remaining requirement at j to $b'_j - \bar{f}_{ij}p_{ij}$. If the basic arc joining i, j is (j, i), make $\bar{f}_{ji} = b'_i/p_{ji}$ and change the remaining requirement at j to $b'_j + b'_i/p_{ji}$. Include i in $\mathbf{Y}$. If j is a non-cycle node, and if all the brothers of i in its tributary tree are already in $\mathbf{Y}$, include j in $\mathbf{X}$.

If $\mathbf{X} \neq \emptyset$ repeat the general step above. If $\mathbf{X} = \emptyset$, go to Stage 2.

Stage 2 If the cycle in the quasitree is a self loop, suppose it is (i, i). Let b'_i be the remaining requirement at node i at this time. Then $\bar{f}_{ii} = -b'_i$ or $+b'_i$ depending on whether the self loop at i is a surplus or slack self loop.

Suppose the cycle is not a self loop. Suppose the nodes in it are 1, 2, $\ldots, t$ in this order when it is oriented with an arc incident at 1, say $e_1 = (1, 2)$, as a forward arc. Let the remaining arcs on the cycle in this order be $e_2, \ldots, e_t$; so, e_r is either $(r, r+1)$ or $(r+1, r)$ depending on whether it is a forward or a reverse arc. Let p_r be the multiplier associated with e_r, and let a_r be p_r or $1/p_r$ depending on whether e_r is a forward or reverse arc, for $r = 1$ to t. Then $\Delta = a_1 a_2 \ldots a_t$ is the inverse of the loop factor of this cycle under the present orientation, $\Delta \neq 1$ by Theorem 8. 5. Denote the flow amount on e_r in the solution we are computing by $\bar{f}_r$. Let b'_r be the remaining requirement at node r at this time, for $r = 1$ to t. Then from (8.9) we see that $(\bar{f}_r : r = 1 \text{ to } t)$ is the solution of a system of equations of the form

$$\begin{pmatrix} a_{11} & a_{12} & & & & \\ & a_{22} & a_{23} & & & \\ & & a_{33} & & & \\ & & & \ddots & & \\ & & & & a_{t-1,t-1} & a_{t-1,t} \\ a_{t1} & & & & & a_{tt} \end{pmatrix} \begin{pmatrix} \bar{f}_1 \\ \bar{f}_2 \\ \bar{f}_3 \\ \vdots \\ \bar{f}_{t-1} \\ \bar{f}_t \end{pmatrix} = \begin{pmatrix} b'_1 \\ b'_2 \\ b'_3 \\ \vdots \\ b'_{t-1} \\ b'_t \end{pmatrix} \tag{8.11}$$

where for each r, $\{a_{rr}, a_{r,r+1}\} = \{-1, p_{r+1}\}$ in (8.11). This system is *near triangular*, in the sense that if the flow amount on one of the arcs in the cycle is given, say α, the flow amounts on the other arcs can be computed by back substitution from (8.11) as affine functions of α. Also, by going around the cycle once we get an equation for α from which α can be uniquely determined.

If $\bar{f}_r$ is given, to determine the flow amount on the next arc in the cycle, $\bar{f}_{r+1}$, four cases arise corresponding to the orientations of the two arcs e_r, e_{r+1}. If both e_r, e_{r+1} are forward arcs (see Figure 8.4), for conservation to hold at node $r+1$, we clearly have $\bar{f}_{r+1} = -b'_{r+1} + p_r f_r$.

Similarly, if e_r is a forward arc and e_{r+1} is a reverse arc, $\bar{f}_{r+1} = (b'_{r+1} - p_r \bar{f}_r)/p_{r+1}$. If e_r is a reverse arc and e_{r+1} is a forward

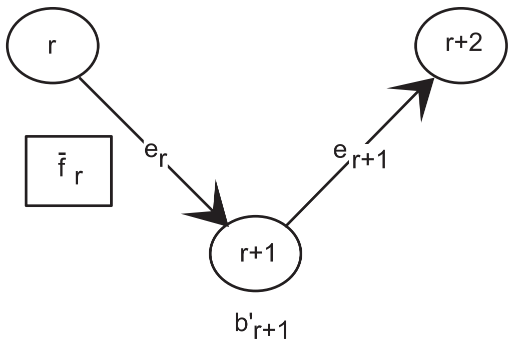

Figure 8.4: Flow amount on e_r is $\bar{f}_r$.

arc, $\bar{f}_{r+1} = -b'_{r+1} - \bar{f}_r$. If both e_r, e_{r+1} are reverse arcs, $\bar{f}_{r+1} = (b'_{r+1} + \bar{f}_r)/p_{r+1}$.

Assume that the flow amount on $e_1, \bar{f}_1$, is α. By using the formulas discussed above, all $\bar{f}_r$ can be obtained as functions of α. By going around the cycle once completely we get an expression for α in terms of itself, which leads to

$$\alpha = (-b'_1 - a_t b'_t - a_t a_{t-1} b'_{t-1} - \ldots - a_t a_{t-1} \ldots a_2 b'_2)/(1 - \Delta) \quad (8.12)$$

Get $\alpha = \bar{f}_1$ from (8.12). Then, compute the flow amounts on the other arcs in the cycle using the formulas developed above.

A partition (**Q, L, U**) for (8.8) is said to be *primal feasible* if the primal basic solution corresponding to it, $\bar{f} = (\bar{f}_{ij})$, is feasible to (8.8), i.e., if $\ell_{ij} \leqq \bar{f}_{ij} \leqq k_{ij}$ for all $(i, j) \in \mathbf{Q}$; otherwise it is said to be *primal infeasible.*

How to Compute the Dual Basic Solution Associated with a Partition

Let (**Q, L, U**) be a given partition for (8.8). The dual basic solution associated with this partition is the node price vector $\pi = (\pi_i)$ obtained from

$$\begin{aligned} p_{ij}\pi_j - \pi_i &= c_{ij}, \text{ for each } i \neq j, (i,j) \in \mathbf{Q} \\ a_{ii}\pi_i &= c_{ii}, \text{ for each self loop } (i,i) \in \mathbf{Q} \end{aligned} \tag{8.13}$$

where $a_{ii} = \pm 1$ is the single nonzero entry in the column of f_{ii} in (8.8), and c_{ii} the cost coefficient of f_{ii}. The node price computation in each quasitree of the basis network can be carried out separately.

If a quasitree contains a self loop, at node i say, then from (8.13), $\pi_i = c_{ii}/a_{ii}$. Using this value of π_i, the node prices for the other nodes in this quasitree can be computed from the remaining equations in (8.13) by back substitution, going down this quasitree beginning at i level by level.

Suppose a quasitree contains a cycle consisting of t nodes; $1, \ldots, t$ say; in this order, when it is oriented with an arc incident at 1, $e_1 = (1,2)$ as a forward arc. Let the remaining arcs on the cycle in this order be $e_2, \ldots, e_t$; so, e_r is either $(r, r+1)$ or $(r+1, r)$ depending on whether it is a forward or a reverse arc. Let c_r, p_r be the cost coefficient, multiplier associated with e_r, $r = 1$ to t. For $r = 1$ to t define $a_r = p_r$, $\gamma_r = 1/p_r$ if e_r is a forward arc on the cycle; otherwise $a_r = 1/p_r, \gamma_r = -1$. $\Delta = a_1 a_2 \ldots a_t$ is the inverse of the loop factor of this cycle under the present orientation, $\Delta \neq 1$ by Theorem 8.5. Applying (8.13) to e_r on the cycle we get

$$\pi_{r+1} = \begin{cases} \frac{\pi_r}{p_r} + \frac{c_r}{p_r}, \text{ if } e_r \text{ is a forward arc} \\ \\ p_r\pi_r - c_r, \text{ if } e_r \text{ is a reverse arc} \end{cases} \tag{8.14}$$

By applying (8.14) once around the cycle we get

$$\pi_1 = \left(\frac{\Delta}{\Delta - 1}\right)\left(c_t\gamma_t + \frac{c_{t-1}\gamma_{t-1}}{a_t} + \frac{c_{t-2}\gamma_{t-2}}{a_t a_{t-1}} + \cdots + \frac{c_2\gamma_2}{a_t \cdots a_3} + \frac{c_1\gamma_1}{a_t \cdots a_2}\right)$$

Having obtained π_1, the value of π_j for each node j on the cycle can be computed using (8.14) by going around the cycle. Once node prices for all nodes on the cycle are known, the node prices of all the non-cycle nodes in this quasitree can be computed using (8.13) by going down each tributary tree beginning with the cycle node on it level by level.

The Primal Simplex Optimality Criterion

Given a primal feasible partition (**Q, L, U**) for (8.8) associated with the BFS $\bar{f}$, and the dual basic solution π, the relative cost coefficient of arc $(i,j) \in \mathcal{A}$ wrt this partition is $\overline{c}_{ij} = c_{ij} - (p_{ij}\pi_j - \pi_i)$ if $i \neq j$, or $= c_{ii} - a_{ii}\pi_i$ if $i = j$, where a_{ii} is the single nonzero entry in the column associated with f_{ii} in (8. 8). (**Q, L, U**) is an optimum partition for (8.8), and $\bar{f}$ is an optimum solution for it if

$$\begin{array}{lll} \overline{c}_{ij} > 0 & \text{implies} & \bar{f}_{ij} = \ell_{ij},\ \text{i. e.,}(i,j) \in \mathbf{L} \\ \overline{c}_{ij} < 0 & \text{implies} & k_{ij} \text{ is finite and } \bar{f}_{ij} = k_{ij},\ \text{i.e.,}(i,j) \in \mathbf{U} \end{array} \qquad (8.15)$$

If (8.15) does not hold, define

$$\mathbf{E} = \{(i,j) : (i,j) \in \mathbf{L} \cup \mathbf{U} \text{ and violates (8.15) } \} \qquad (8.16)$$

E is the set of nonbasic arcs eligible to be an entering arc for a pivot step in the present primal feasible partition to continue the primal simplex algorithm.

Basis Representation of an Entering Arc

Let **Q, L, U** be the present primal feasible partition for (8.8), and suppose the nonbasic arc (g,h) has been selected as the entering arc for a pivot step in it. To carry out the computations in the ensuing pivot step we need the updated column of the entering variable f_{gh} wrt the present basic set **Q**, which is the pivot column for the pivot step. Let $H(g,h)$ denote the original column of f_{gh} in (8.8). $H(g,h) = -I_{.g} + p_{gh}I_{.h}$ (here I is the unit matrix of order n) if $g \neq h$, or $= a_{gg}I_{.g}$ if $g = h$ ($a_{gg} = \pm 1$ is the unique nonzero entry in the column of f_{gg} in (8.8)). The updated column of f_{gh} wrt **Q** is the vector of coefficients in the representation of $H(g,h)$ as a linear combination of the basic columns, hence it is called the **basis representation of the entering arc** wrt the basic set **Q**. If B is the present basis (the matrix consisting of the columns corresponding to the basic arcs in **Q** in (8.8)), this representation, denoted by $\overline{y}$ is the solution of

$$B\overline{y} = H(g,h) \tag{8.17}$$

So, $\overline{y}$ is the vector of flow values on the basic arcs in $\mathbf{Q}$ to meet a requirement of -1 unit at node g, and p_{gh} units at node h, while the flow amounts on all the nonbasic arcs are 0. So, $\overline{y}$ can be computed directly using the procedure described earlier for computing the flows on basic arcs to meet specified requirements at the nodes. For any $i \in \mathcal{N}$, let $\mathcal{P}_i(\mathbf{Q})$ denote both the predecessor path of i in its quasitree in $\mathbf{Q}$, or the set of arcs on this path. Obviously, nonzero entries in the basis representation $\overline{y}$ of (g,h) correspond to arcs in $\mathcal{P}_g(\mathbf{Q}) \cup \mathcal{P}_h(\mathbf{Q})$.

Selection of the Dropping Arc and Updating the Primal Solution in a Pivot Step

Let $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$ be the present primal feasible partition for (8.8) associated with the BFS $\bar{f} = (\bar{f}_{ij})$ in which the optimality criterion (8.15) is violated. Suppose the nonbasic arc (g,h) has been selected as the entering arc in this partition for a primal simplex pivot step. Let $\overline{y} = (\overline{y}_{ij} : (i,j) \in \mathbf{Q})$ be the updated column of the entering variable f_{gh}. Let $\mathcal{P}_g(\mathbf{Q})$, $\mathcal{P}_h(\mathbf{Q})$ denote the predecessor paths (or the set of arcs on these paths) of g, h in their quasitrees in $\mathbf{Q}$. In this pivot step we keep the flow amounts on all the nonbasic arcs other than (g,h) at their present values, change the flow amount on the entering arc (g,h) from its present $\bar{f}_{gh}$ to $\bar{f}_{gh} + \delta\lambda$, where $\delta = +1$ if $(g,h) \in \mathbf{L}$, or -1 if $(g,h) \in \mathbf{U}$; and reevaluate the flow amounts on the basic arcs in $\mathbf{Q}$ as functions of λ. This leads to the solution $\hat{f}(\lambda) = (\hat{f}_{ij}(\lambda))$ where

$$\hat{f}_{ij}(\lambda) = \begin{cases} \bar{f}_{ij} & \text{for } (i,j) \neq (g,h) \text{ in } \mathbf{L} \cup \mathbf{U} \\ \bar{f}_{ij} + \delta\lambda & \text{for } (i,j) = (g,h) \\ \bar{f}_{ij} - \delta\lambda\overline{y}_{ij} & \text{for } (i,j) \in \mathbf{Q} \end{cases} \tag{8.18}$$

Thus $\hat{f}_{ij}(\lambda)$ differs from $\bar{f}_{ij}$ only if $(i,j) \in \{(g,h)\} \cup (\mathcal{P}_g(\mathbf{Q}) \cup \mathcal{P}_h(\mathbf{Q}))$. The minimum ratio in this pivot step is the maximum value for λ that keeps $\hat{f}(\lambda)$ primal feasible, it is

$$\theta = \text{ Min. } \{k_{gh} - \ell_{gh}\} \;\cup\; \left\{\frac{k_{ij} - \bar{f}_{ij}}{-\delta\overline{y}_{ij}} : (i,j) \in \mathbf{Q} \text{ and } \delta\overline{y}_{ij} < 0\right\} \cup \left\{\frac{\bar{f}_{ij} - \ell_{ij}}{\delta\overline{y}_{ij}} : (i,j) \in \mathbf{Q} \text{ and } \delta\overline{y}_{ij} > 0\right\} \tag{8.19}$$

In computing θ we adopt the convention that the minimum in the empty set is ∞. θ is always nonnegative. If $\theta = \infty$, the unboundedness criterion is satisfied, $\hat{f}(\lambda)$ remains feasible for all $\lambda \geqq 0$ and its objective value $\to -\infty$ as $\lambda \to +\infty$, we terminate the algorithm. If θ is finite, let

$$\mathbf{D} = \{(i,j) : (i,j) \in \mathbf{Q} \cup \{(g,h)\} \text{ ties for the minimum in (8.19)}\} \tag{8.20}$$

$\mathbf{D} \subset \{(g,h)\} \cup (\mathcal{P}_g(\mathbf{Q}) \cup \mathcal{P}_h(\mathbf{Q}))$ is the set of **blocking arcs**, i.e., arcs which are among those first driven to an upper or lower bound by an attempted change in the value of f_{gh} away from the bound it currently equals. Arcs in $\mathbf{D}$ are those which are eligible to be the **dropping** *or* **leaving arc** in this pivot step. Let $(r,s) \in \mathbf{D}$ denote the arc chosen as the dropping arc in this pivot step.

This pivot step is **degenerate** if $\theta = 0$, **nondegenerate** if $\theta > 0$. $\hat{f}(\theta)$ is the new BFS obtained after this pivot step. Notice that there is no change in the BFS in a degenerate pivot step.

If $(r,s) = (g,h)$, (g,h) is moved from $\mathbf{L}$ or $\mathbf{U}$ where it is presently contained, into the other set. There is no change in $\mathbf{Q}$. This gives the new partition. Since there is no change in $\mathbf{Q}$, there are no changes in the node labels or the dual solution π.

If $(r,s) \neq (g,h)$, delete (g,h) from $\mathbf{L}$ or $\mathbf{U}$ which contains it presently. Drop (r,s) from $\mathbf{Q}$, and include (g,h) in $\mathbf{Q}$ in its place. Include (r,s) in $\mathbf{L}$ or $\mathbf{U}$ depending on whether $\hat{f}_{rs}(\theta) = \ell_{rs}$ or k_{rs} respectively. Since there is a change in the basic set of arcs, the node labels and the dual basic solution have to be updated as described below.

After all the necessary updatings are carried out, the primal simplex algorithm moves to the next step with the new partition.

Updating the Node Labels and Node Prices in a Pivot Step

Let $(\mathbf{Q}, \mathbf{L}, \mathbf{U})$ be the present partition, (g, h) the entering arc, and $(r, s) \neq (g, h)$ the leaving arc. Let $\mathcal{P}_g(\mathbf{Q})$, $\mathcal{P}_h(\mathbf{Q})$ be the predecessor paths (or the set of arcs on these paths) of g, h in their quasitrees in $\mathbf{Q}$. $\hat{\mathbf{Q}} = \mathbf{Q} \cup \{(g, h)\} \backslash \{(r, s)\}$ is the new basic set of arcs in the next partition. $\hat{\mathbf{Q}}$ contains a new cycle that $\mathbf{Q}$ does not contain iff $(r, s) \in \mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q})$.

If $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) = \emptyset$, nodes g, h belong to different quasitrees in $\mathbf{Q}$. If $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) \neq \emptyset$, nodes g, h belong to the same quasitree, and both $\mathcal{P}_g(\mathbf{Q})$, $\mathcal{P}_h(\mathbf{Q})$ contain the cycle in this quasitree.

For each $i \in \mathcal{N}$ define $\mathcal{P}_{ig}(\mathbf{Q})$ to be the empty path if i is not an ancestor of g, or the portion of the predecessor path of g up to node i the first time that i appears on the path if i is an ancestor of g. In the latter case $\mathcal{P}_{ig}(\mathbf{Q})$ is a simple path beginning with g and ending with i. We will also use the same symbol $\mathcal{P}_{ig}(\mathbf{Q})$ to denote the set of arcs on this path.

Let $\pi = (\pi_i), \hat{\pi} = (\hat{\pi}_i)$ denote the dual basic solutions wrt $\mathbf{Q}$, $\hat{\mathbf{Q}}$ respectively. Removing the outgoing arc (r, s) from $\mathbf{Q}$ without adding the incoming arc (g, h) creates a unique tree in $\mathcal{P}_g(\mathbf{Q}) \cup \mathcal{P}_h(\mathbf{Q})$, which we denote by $\mathbb{T}$. It is possible that $\mathbb{T}$ may consist of a single node only. See Figures 8.5, 8.6, 8.7. $\hat{\pi}_i = \pi_i$ for all nodes i except those on $\mathbb{T}$, as can be verified from the manner in which the dual solution is computed by the procedure discussed above. So, it is necessary to update the node prices only for the nodes in $\mathbb{T}$ in this step.

When $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) \neq \emptyset$, we define

$$\begin{array}{lll} x & : & \text{the node on } \mathcal{P}_g(\mathbf{Q}) \text{ such that } \mathcal{P}_{xg}(\mathbf{Q}) = \mathcal{P}_g(\mathbf{Q}) \backslash \mathcal{P}_h(\mathbf{Q}) \\ y & : & \text{the node on } \mathcal{P}_h(\mathbf{Q}) \text{ such that } \mathcal{P}_{yh}(\mathbf{Q}) = \mathcal{P}_h(\mathbf{Q}) \backslash \mathcal{P}_g(\mathbf{Q}) \end{array} \tag{8.21}$$

If g and h belong to the same tributary tree, then $x = y$, and this is the first common node on $\mathcal{P}_g(\mathbf{Q})$ and $\mathcal{P}_h(\mathbf{Q})$. See Figure 8.5. If g and h belong to different tributary trees of the same quasitree, then x, y are the roots (i.e., cycle nodes) of these quasitrees. See Figure 8.6. We now discuss the procedures for updating the node labels and node prices under several cases separately.

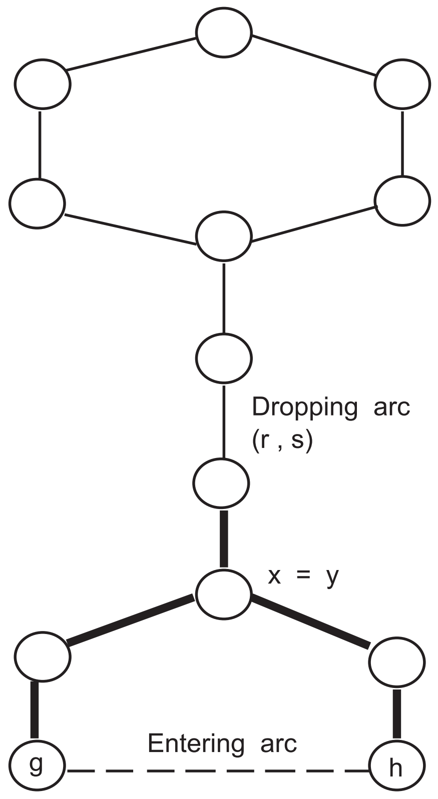

Figure 8.5: The tree $\mathbb{T}$ is marked with thick lines.

In Figures 8.5 to 8.8 orientations of the arcs are not shown. On each non-cycle arc, the node at the top is the predecessor of the node at the bottom. Cycles are drawn in such a way that as you move in the clockwise direction, you move from a node to its predecessor.

CASE 1 : $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) = \emptyset$, AND THE DROPPING ARC $(r, s) \in \mathcal{P}_g(\mathbf{Q})$.

Let i, j be the predecessor, successor nodes respectively, among r, s. So, $\text{PI}(j) = \pm i$ and (r, s) is either (i, j) or (j, i). Make $\text{PI}(g) = -h$, $\text{SI}(h) = g$, $\text{SI}(i) = \text{SI}(j) = \emptyset$. Reverse the predecessor, successor relationships on each arc along the path $\mathcal{P}_{jg}(\mathbf{Q})$. Make corresponding changes in the

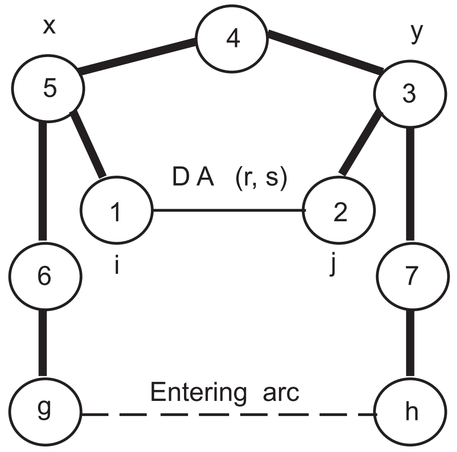

Figure 8.6: The dropping arc, DA, is on $\mathcal{P}_{xy}(\mathbf{Q})$. The tree $\mathbb{T}$ is marked with thick lines. Position after this pivot step is indicated in Figure 8.8(b).

successor index and brother indices of nodes along this path, and their brothers, as in Section 5.4. See Figures 8.7, 8.8 (a). The updating of other node labels such as the depth label, thread label, etc. can be carried out in a similar way. We leave these to the reader, in this and in the following cases.

Node prices change only for nodes in $\mathbb{T}$. After updating all the node labels, the values of $\hat{\pi}_i$ for $i \in \mathbb{T}$ are computed using (8.13) beginning with the known value of $\hat{\pi}_h = \pi_h$.

CASE 2 : $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) = \emptyset$, AND THE DROPPING ARC $(r, s) \in \mathcal{P}_h(\mathbf{Q})$.

Define nodes i, j as in Case 1. Make PI$(h) = g$, SI$(g) = -h$, SI$(i) =$ SI$(j) = \emptyset$. Reverse the predecessor, successor relationships on each arc along the path $\mathcal{P}_{jh}(\mathbf{Q})$. Make corresponding changes in the successor index and brother indices of nodes along this path, and their brothers, as in Section 5.4.

In this case node prices are updated for nodes on $\mathbb{T}$ using the known value of $\hat{\pi}_g = \pi_g$ as mentioned in Case 1.

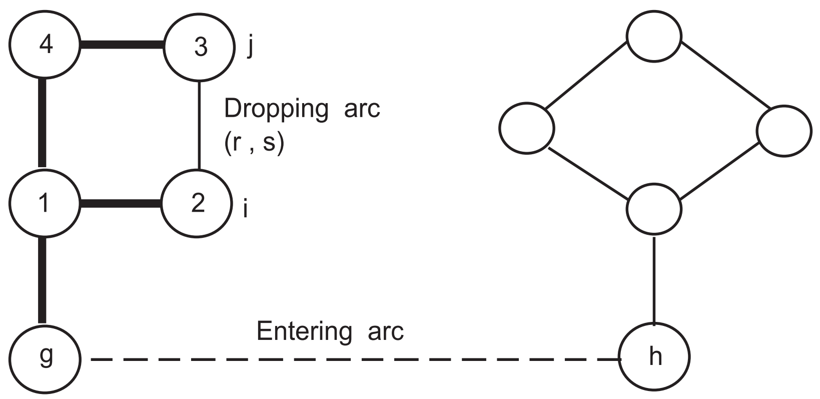

Figure 8.7: Illustration of Case 1. The tree $\mathbb{T}$ is marked with thick arcs. Position after the pivot step is indicated in Figure 8.8(a).

CASE 3 : $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) \neq \emptyset$, AND THE DROPPING ARC (r, s) IS NOT IN $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q})$.

If $(r, s) \in \mathcal{P}_g(\mathbf{Q}) \backslash \mathcal{P}_h(\mathbf{Q})$, update the node labels and node prices as in Case 1. If $(r, s) \in \mathcal{P}_h(\mathbf{Q}) \backslash \mathcal{P}_g(\mathbf{Q})$ update the node labels and node prices as in Case 2.

CASE 4 : THE DROPPING ARC $(r, s) \in \mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q})$, BUT NOT ON THE CYCLE IN THIS QUASITREE.

See Figure 8.5. In this case nodes g, h belong to the same tributary tree in $\mathbf{Q}$ and nodes x, y defined in (8.21) are the same. $\mathbb{T} \cup \{(g, h)\}$ forms a new quasitree in the new basic set $\hat{\mathbf{Q}}$. The cycle in this new quasitree is a new cycle that is created, and it contains the entering arc (g, h). Update the node labels as in Case 1. Compute the new node prices $\hat{\pi}_i$ for nodes i in this new quasitree by the procedure discussed earlier. Node prices at all other nodes remain unchanged.

CASE 5 : THE DROPPING ARC (r, s) IS ON THE CYCLE IN $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q})$.

See Figure 8.6. In this case $\mathbb{T} \cup \{(g, h)\}$ forms a new quasitree in $\hat{\mathbf{Q}}$. Define nodes i, j as in Case 1. If $(r, s) \in \mathcal{P}_{xy}(\mathbf{Q})$, make PI$(h) = g$, SI$(g)$

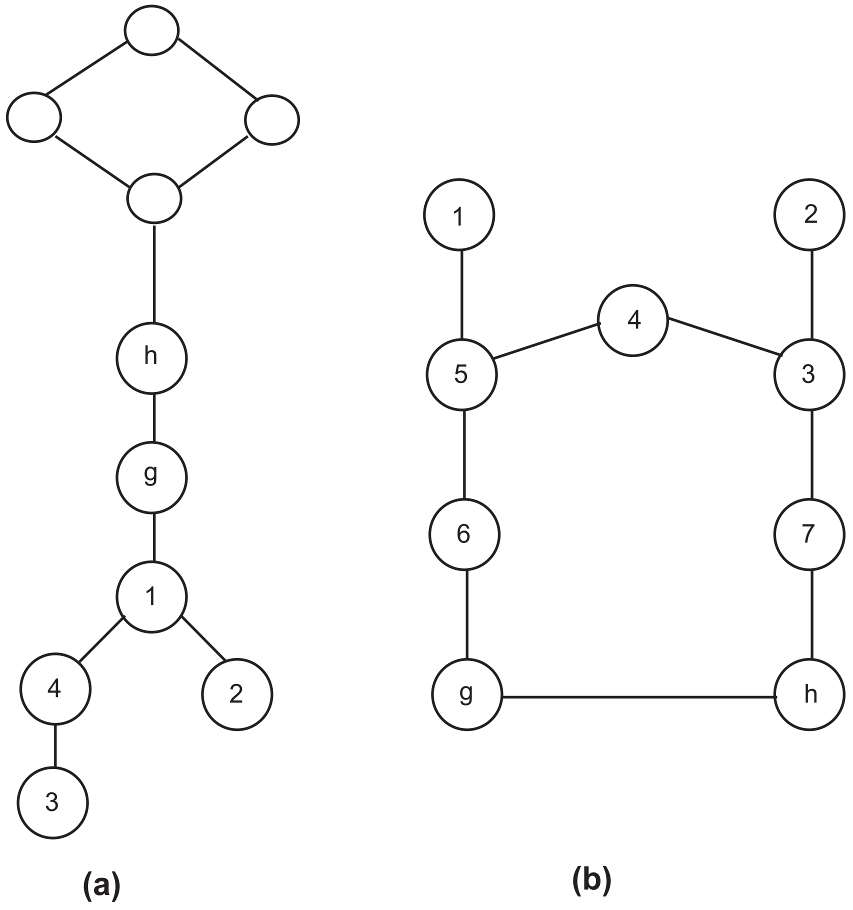

Figure 8.8:

$= -h$; and reverse the predecessor, successor relationships on each arc along the path $\mathcal{P}_{jh}(\mathbf{Q})$. See Figures 8.6, 8.8 (b). If $(r, s) \in \mathcal{P}_{yx}(\mathbf{Q})$, make $\mathrm{PI}(g) = -h$, $\mathrm{SI}(h) = g$; and reverse the predecessor, successor relationships on each arc along the path $\mathcal{P}_{jg}(\mathbf{Q})$. Make corresponding changes in the successor index and brother indices of nodes along these paths, and their brothers, as in Section 5.4.

Compute the new node prices $\hat{\pi}_i$ for nodes i in this new quasitree by the procedure discussed earlier. Node prices at all other nodes remain unchanged.

The Primal Simplex Method for (8.8)

If a primal feasible partition is known, (8.8) can be solved by initiating the primal simplex algorithm with it. Each primal feasible partition obtained in this algorithm is checked for optimality. If the optimality criterion (8.15) is satisfied, the present BFS is optimal and the algorithm terminates. Otherwise, an entering arc is selected among those nonbasic arcs eligible to enter, and a pivot step is carried out using the procedures discussed above. The same process is then repeated with the new partition.

If a primal feasible partition for (8.8) is not known, select a spanning tree in G, and make it into a basic set **Q** for (8.8) by including an additional arc in this spanning tree. Partition the nonbasic arcs into **L, U** arbitrarily and generate the initial partition (**Q, L, U**). If it is primal feasible, use it to initiate the primal simplex algorithm. If the initial partition is primal infeasible, alter the lower bounds and capacities on basic arcs and construct a Phase I problem exactly as discussed in Section 5.4 for the pure network flow problem. Solve the Phase I problem by the primal simplex algorithm beginning with this initial feasible partition for it. At Phase I termination, we will either obtain a feasible partition for the original problem (8.8), or conclude that it is infeasible.

8.2 Resolution of Cycling Under Degeneracy in the Primal Simplex Algorithm for Generalized Network Flows

The potential theoretical consequences of degeneracy (cycling and stalling) in the primal simplex method have already been explained in Chapter 5 and possible remedies for them in the minimum cost pure network flow problem have been discussed. Similar techniques are needed in the generalized network flow problem too under degeneracy. In this section we discuss a method developed by Elam, Glover, and Klingman [1979] for resolving cycling under degeneracy in the primal simplex algorithm for the minimum cost generalized network flow problem (8.8) when the multiplier vector $(p_{ij}) > 0$. It is called the **method of strongly convergent partitions** *or the* **strongly convergent primal simplex algorithm** *or the* **SC method** in short. It can be viewed as an extension of the method of strongly feasible partitions for the minimum cost pure network flow problem discussed in Section 5.4, to the generalized network case.

The main idea behind the SC method is the following. A subset of feasible partitions for (8.8) called **strongly convergent feasible partitions** *or* **SC partitions** is defined. It is shown that (8.8) always has an SC partition when it is feasible. The method is initiated with an SC partition. A dropping arc choice rule which determines the dropping arc uniquely and unambiguously, and maintains the SC-character in the next partition, is developed. So, when the primal simplex algorithm is initiated with an SC partition and executed using this special dropping arc choice rule, it is restricted to SC partitions throughout. Finally it is shown that under this restriction it cannot cycle, and hence has to terminate in a finite number of pivot steps, and it can only do so by satisfying either the unboundedness or the optimality criterion.

Let (**Q, L, U**) be a feasible partition for (8.8). In this partition, a basic arc $(i, j) \in \mathbf{Q}$ is said to be a **forward basic arc** if node i is a predecessor of node j, or if $i = j$ and the arc is a slack self loop; a **reverse basic arc** otherwise. For any subset of basic arcs $\mathbf{A} \subset \mathbf{Q}$, FOR(**A**), REV(**A**) denote the sets of forward and reverse basic arcs in **A** respectively. We have already defined loop factors of cycles, and path factors of simple paths, which depend on the orientation selected for those objects. Extending this concept, here we define a **collective gain/loss factor** for any subset of basic arcs $\mathbf{A} \subset \mathbf{Q}$ to be $\psi(\mathbf{A})$ given by

$$\psi(\mathbf{A}) = \frac{\text{Product of multipliers associated with arcs in REV}(\mathbf{A})}{\text{Product of multipliers associated with arcs in FOR}(\mathbf{A})}$$

where the product of multipliers in a set is defined to be 1 if that set is $\emptyset$.

In a feasible flow vector $\bar{f} = (\bar{f}_{ij})$ for (8.8), an arc (i, j) is said to have **lower leeway** if $\bar{f}_{ij} > \ell_{ij}$, **upper leeway** if $\bar{f}_{ij} < k_{ij}$, **both lower and upper leeways** *or* **double leeway** if $\ell_{ij} < \bar{f}_{ij} < k_{ij}$.

In a primal feasible partition ($\mathbf{Q}$, $\mathbf{L}$, $\mathbf{U}$) for (8.8), a basic arc $(i, j) \in \mathbf{Q}$ is said to be an **SC arc** *or* **to have SC leeway** if it is a forward basic arc with a lower leeway or a reverse basic arc with an upper leeway. The primal feasible partition ($\mathbf{Q}$, $\mathbf{L}$, $\mathbf{U}$) for (8.8) is said to be a **strongly convergent partition** *or* **SC partition** if: (1) all basic arcs have SC leeway in it, and (2) the collective gain/loss factor of every cycle in $\mathbf{Q}$ that is not a self loop is < 1. It is always possible to construct an initial SC partition which may involve some artificial variables, since a basic set consisting of slack or surplus self loops, possibly artificial, satisfies the condition for being an SC partition for the appropriately set Phase I problem.

In a pivot step of the primal simplex algorithm in a feasible partition ($\mathbf{Q}$, $\mathbf{L}$, $\mathbf{U}$) for (8.8), let (g, h) be the nonbasic arc selected as the entering arc, and let $(\overline{y}_{ij} : (i, j) \in \mathbf{Q})$ be its basis representation. Define

$$\begin{aligned}
\mathbf{B}^{+} &= \{(i,j) : (i,j) \in \mathbf{Q} \text{ and } \overline{y}_{ij} > 0\} \\
\mathbf{B}^{-} &= \{(i,j) : (i,j) \in \mathbf{Q} \text{ and } \overline{y}_{ij} < 0\} \\
\mathbf{B} &= \mathbf{B}^{+} \cup \mathbf{B}^{-} \qquad\qquad (8.22)\\
\mathbf{H} &= \{(i,j) : \text{ either } (i,j) \in \text{FOR}(\mathbf{B}) \text{ and } \overline{y}_{ij} < 0, \\
&\qquad \text{or } (i,j) \in \text{REV}(\mathbf{B}) \text{ and } \overline{y}_{ij} > 0\} \\
\mathbf{J} &= \{(i,j) : \text{ either } (i,j) \in \text{FOR}(\mathbf{B}) \text{ and } \overline{y}_{ij} > 0, \\
&\qquad \text{or } (i,j) \in \text{REV}(\mathbf{B}) \text{ and } \overline{y}_{ij} < 0\}
\end{aligned}$$

Hence $\mathbf{H}$ is the set of all forward basic arcs whose flow change has the same sign, and all reverse basic arcs whose flow change has the opposite sign, to that of the entering arc (g, h) in this pivot step. A similar symmetric interpretation holds for $\mathbf{J}$. Elam, Glover, and Klingman [1979] have proved that $\mathbf{H} = \mathcal{P}_g(\mathbf{Q}), \mathbf{J} = \mathcal{P}_h(\mathbf{Q})$, if $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) = \emptyset$; and that in general even when $\mathcal{P}_g(\mathbf{Q}) \cap \mathcal{P}_h(\mathbf{Q}) \neq \emptyset$, $\mathbf{H}$ consists of consecutive arcs in $\mathcal{P}_g(\mathbf{Q})$ and $\mathbf{J}$ consists of consecutive arcs in $\mathcal{P}_h(\mathbf{Q})$. Index the arcs in $\mathbf{H}$, $\mathbf{J}$ in ascending order, in the same sequence as they are encountered by a trace of the predecessor paths of g and h starting at these nodes. Thus if $\mathbf{H} \neq \emptyset$, the arc joining g and its predecessor PI(g) is the first in $\mathbf{H}$; and if $\mathbf{J} \neq \emptyset$, the arc joining h and its predecessor PI(h) is the first in **J**, etc. In addition, include (g, h) in **H** at the beginning of this set.

The set of arcs eligible to be the dropping arc in this pivot step is the set **D** defined in (8.20). The **SC dropping arc selection rule** *or the* **SC pivot rule** specifies that the dropping arc (r, s) in this pivot step should be selected from **D** according to the following:

> if $(g, h) \in \mathbf{L}$, (r, s) is the last arc on **J** in the set $\mathbf{J} \cap \mathbf{D}$ if this set is nonempty, otherwise it is the first arc in $\{(g, h)\} \cup \mathbf{H}$ that is in **D**
>
> if $(g, h) \in \mathbf{U}$, (r, s) is the last arc in the set $(\{(g, h)\} \cup \mathbf{H}) \cap \mathbf{D}$ if this set is nonempty, otherwise it is the first arc in **J** that is in **D**

Elam, Glover, and Klingman [1979] have shown that if the feasible partition (**Q, L, U**) at the beginning of this pivot step is an SC partition, and if the dropping arc in this pivot step is selected by this special rule, then the next partition will also be an SC partition. Furthermore, any choice of the dropping arc other than that specified by the above rule destroys the SC property in the next partition.

The method of strongly convergent partitions is the primal simplex algorithm initiated with an SC Partition, and executed using the special dropping arc choice rule given above, in each pivot step. In each pivot step the entering arc can be selected by any rule, or even arbitrarily from the set of those eligible (this is the set **E** of (8.16)). All partitions obtained in the algorithm will be SC partitions. It has also been shown that in any sequence of consecutive degenerate pivot steps in this SC algorithm, any node price that changes always increases, i.e., they always change in a uniform direction throughout the entire sequence of degenerate pivot steps. This is the reason for the phrase **strongly convergent** in the name. A similar property was shown to hold in Section 5.4 in the method of strongly feasible partitions for pure network flow problems. This property guarantees that the SC algorithm cannot cycle, and hence must terminate in a finite number of pivot steps by satisfying either the unboundedness or the optimality criteria. Orlin [1985] has shown that the SC pivot rule is equivalent to lexicography in its choice of the exiting arc. Thus the SC algorithm extends the method of strong feasibility to generalized network flow problems with positive multipliers.

A Phase I problem, which is in the same form but with a readily available artificial SC partition, can be formulated for (8.8). When the SC algorithm is applied to solve this Phase I problem beginning with this initial SC partition, it will terminate finitely with either a proof of infeasibility of the original problem, or with an SC partition for it. If, feasible, beginning with the SC partition provided by Phase I, the original problem can be solved finitely by the SC algorithm in Phase II.

8.3 The Most General Generalized Network Flow Problem

The generalized network flow model discussed so far can accommodate any LP in which the coefficient matrix has the property that each column has at most two nonzero entries, and if there are two nonzero entries in a column they have opposite signs. This sign condition guarantees that the multiplier associated with every arc in the corresponding generalized network is always a positive number.

The most general problem of this type is any LP in which every column in the coefficient matrix has at most two nonzero entries, without any restriction on the sign of these nonzero entries in any column. Given such an LP, scale each column with two nonzero entries so that one of these entries becomes equal to -1. After scaling, associate a node with each row of the coefficient matrix. If a column has a single nonzero entry, say in row i, associate that column with a self loop at node i. If a column has two nonzero entries, say a "-1" in row i and an entry of $p \neq 0$ in row j, associate it with arc (i, j) and make p the multiplier of that arc. Here p may be positive or negative. The LP is equivalent to the generalized network flow problem of the form (8.8) on this network, but some of the multipliers may be negative. The results at the beginning of this chapter continue to hold for this problem, and it can be solved by the implementation of the primal simplex algorithm discussed in Section 8.1. However, the SC algorithm of Section 8.2 and all the special convergence properties mentioned there are based on the assumption that all the arc multipliers are positive, and these properties and results may not hold for the general problem with some negative multipliers. But recently, Arantes, Birge, and Murty [1992] developed a special method for resolving the problem of cycling under degeneracy in the primal simplex algorithm applied to this general generalized network flow problem. This method is also a specialization of lexicography using the special structure of this problem.

8.4 Exercises

8.1 Let ($\mathbf{Q}$, $\mathbf{L}$, $\mathbf{U}$) be an SC partition for (8.8). Prove that the pivot step in this partition with (g, h) as the incoming arc is nondegenerate iff either the dropping arc $(r, s) = (g, h)$, or $(r, s) \in \mathbf{J}$ if $(g, h) \in \mathbf{L}$, or $(r, s) \in \mathbf{H}$ if $(g, h) \in \mathbf{U}$; where $\mathbf{J}$, $\mathbf{H}$ are the sets defined in Section 8.2.

8.2 In Section 3.3 we discussed a method for transforming a single commodity minimum cost flow problem in a pure non-bipartite network, into a similar problem on a pure bipartite network. Show that the minimum cost flow problem (8.8) in a non-bipartite generalized network G can similarly be transformed into a problem on a bipartite generalized network (Malek-Zavarei and Aggarwal [1972]).

8.3 Consider the mixed 0-1 integer programming problem of minimizing $z(x) = cx$ subject to, $d \leqq Ax \leqq b$, $x_j \in \{0, 1\}$ for all $j \in \mathbf{N}_1$, $0 \leqq x_j \leqq u_j$ for all $j \in \mathbf{N} \backslash \mathbf{N}_1$; where A is of order $m \times n$, $\mathbf{N} = \{1, \cdots, n\}$, $\mathbf{N}_1 \subset \mathbf{N}$, and $A_{.j}$ has at most two nonzero entries for each $j \in \mathbf{N} \backslash \mathbf{N}_1$. Show that this problem can be transformed into a mixed 0-1 generalized network flow problem (i.e., a generalized network flow problem in which the flow on a specified subset of arcs is required to be 0 or 1). In addition, for each $j \in \mathbf{N} \backslash \mathbf{N}_1$ suppose the following holds: If $A_{.j}$ contains a single nonzero entry, it is $+1$ or -1; if it contains two nonzero entries, one is $+ 1$ and the other is -1. In this case show that this problem can be transformed into a mixed $0 - U$ pure network flow problem (i.e., one in which the flow on a specified subset of arcs is required to be either the lower bound 0 or the capacity on that arc).

8.4 Let G = $(\mathcal{N}, \mathcal{A}, \ell, k, p)$ be a generalized network with p as the multiplier vector. Prove that a necessary and sufficient condition for the existence of a feasible circulation in G is that the following condition holds for every tree (N, A) in G, and any node price vector $\pi = (\pi_i)$ in G satisfying: $\pi_i = 0$ for $i \in \mathcal{N} \setminus$ N, and $\pi_i - \pi_j p_{ij} = 0$ for all $(i, j) \in$ A.

$$\sum_{(i,j)\in\mathcal{A}} k_{ij}(\text{ max. } \{0,\ \pi_i - p_{ij}\pi_j\}) \geqq \sum_{(i,j)\in\mathcal{A}} \ell_{ij}(\text{ max. } \{0, p_{ij}\pi_j - \pi_i\})$$

Verify that no feasible circulation exists in the network in Figure 8.9, where the data on arc (i, j) is $\ell_{ij}, k_{ij}, p_{ij}$ in that order; and that the above condition fails to hold for the tree with arcs (1, 2), (3, 1) and the vector π = (1, 1/2, 1).

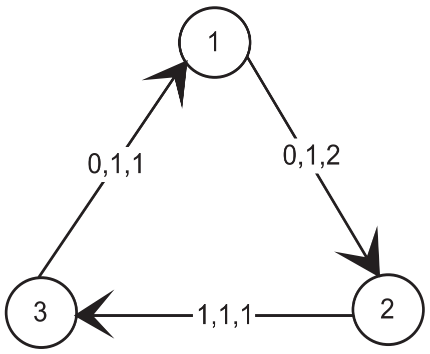

Figure 8.9:

(Hassin[1981])

8.5 Cash Flow Management Model Consider a discrete time deterministic cash flow problem. There are n time periods in the planning horizon. The supply and demand for cash in each time period is known in advance. Spare cash can be saved from one period to the next in a savings account at fraction yield of α per period, or invested in a bond portfolio at fraction yield of β per period. All transactions occur at the beginning of a period, and returns are obtained at the end of a period. c_1, c_2 are the per unit fraction cost of converting cash into bond portfolio, a bond back into cash, respectively. Conversions are instantaneous and can be carried out either way in any quantity. $S(t)$ is the new supply (incoming) of cash in addition to any return from investments, and $D(t)$ is the demand for cash (to be paid out) at the beginning of period $t = 1$ to n. There is an initial inventory in bond portfolio of y_0 at the start.

(i) It is required to determine the amounts to be invested in each asset in each period so as to maximize the return from the final planning period. Formulate this as a generalized network flow problem on a network with $2n+3$ nodes. Construct this network model for the numerical problem in which $n = 3$, $(S(1), S(2), S(3)) = (12, 8, 5)$, $(D(1), D(2), D(3)) = (4, 10, 10)$ in million\$, $\alpha = .05, \beta = .08, c_1 = c_2 = .02$ and $y_0 = 1$ in million\$ units; and obtain the optimum solution.

(ii) Consider the same problem with two additional features. One is a minimum cash balance specification, i.e., the firm is required by policy to put down a minimum of L units in cash in the savings account from each period to the next. The other is a borrowing capability at a fraction interest of γ per period up to an upper limit of U units in any period. Discuss the modifications to be made in the network model of (i) to incorporate these features. Construct this model for the numerical problem in which $n = 4$, $(S(1)$ to $S(4)) = (5, 10, 20, 15)$, $(D(1)$ to $D(4)) = (16, 12, 8, 8)$, $\alpha = .05$, $\beta = .08$, $\gamma = .10$, $L = 5$, $U = 10$, $y_0 = 10$. The cash unit is one million \$. Find the optimum cash flow policy. Solve the same numerical problem with all the data the same, except $\gamma = .09, .1025, .12$ respectively. Verify that as γ increases from .09 to .10, and from .1025 to .12, the optimum policy actually borrows more, a counterintuitive phenomenon. Explain.

(iii) Assume $\beta > \alpha$. N^* is said to be a forecast horizon for period t if demands in periods beyond N^* have no effect on amounts converted from cash into bonds or vice versa at period t in an optimum policy. If N' is the smallest positive integer $\mathbf{N}$ satisfying $(1 - c_1)(1 - c_2)(1 + \beta)^{\mathbf{N}} \geqq (1 + \alpha)^{\mathbf{N}}$, then prove that min. $\{n, t + N' - 1\}$ is a forecast horizon for period t.

(iv) Instead of a single bond-portfolio, suppose there are r different assets ($r \geqq 2$) in which cash can be invested in each period. Given corresponding data for each asset, discuss the modifications to be made in the network model to handle this change.

(Golden, Liberatore, and Lieberman [1979])

8.6 Consider the minimum cost flow problem (8.8) on the generalized network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$, with the multiplier vector $(p_{ij} : (i, j) \in \mathcal{A}) > 0$, and $|\mathcal{N}| = n$. Let (**Q, L, U**) be a feasible partition for this problem with the arcs in **Q** arranged and numbered as $e_1, \ldots, e_n$. Let B be the basis for (8.8) corresponding to **Q**. Let e_{n+1} be a nonbasic arc with $(\overline{y}_t : t = 1 \text{ to } n)$ as its basis representation wrt **Q**. Let $\beta = B^{-1}$. For each $t = 1$ to n such that $\overline{y}_t \neq 0$ define $S_{t.} = (\beta_{t.})/\overline{y}_t$, and let S be the matrix with all these rows $S_{t.}$. So, if $r = |\{t : \overline{y}_t \neq 0\}|$, then $S = (s_{tj})$ is of order $r \times n$. Prove the following:

(i) For each $t = 1$ to n, all the nonzero elements of $\beta_{t.}$ have the same sign. This shows that the unisign property discussed in Theorem 1.8 also holds in generalized networks in which all multipliers are positive.

(ii) For some t_1, t_2, if $S_{t_1.}$ is lexicographically greater than $S_{t_2.}$, then $s_{t_1 j} \geqq s_{t_2 j}$ for all $j = 1$ to n.

(iii) Suppose the partition (**Q, L, U**) was obtained in the process of solving (8.8) by the primal simplex algorithm, and that e_{n+1} is the arc selected to be the entering arc into it. Suppose the dropping arc in this pivot step has been selected by the special dropping arc choice rule specified by the SC algorithm, and it is e_r. Then show that $S_{r.}$ is the lexico maximum row among the rows of S.

Comment 8.1 Bhaumik [1973], Glover, Hultz, Klingman, and Stutz [1978], Golden, Liberatore, and Lieberman [1979], Hamacher and Tufekci [1987], are some references discussing applications of generalized network flow models.

In general, any LP in which the coefficient matrix of constraints other than individual bound restrictions on single variables contains at most two nonzero entries per column, can be treated as a generalized network flow problem. In this chapter we discussed an implementation of the primal simplex method based on rooted loop labelings of quasitrees, for the generalized minimum cost flow problem. Although total unimodularity is not there, this implementation makes it possible to execute the primal simplex method on this problem in a purely combinatorial mode without the use of basis inverses. Brown and Mcbride [1984], Elam, Glover, and Klingman [1979], Glover and Stutz [1973], Nulty and Trick [1988] are some references on this topic. Elam, Glover, and Klingman [1979] report that this implementation gave excellent performance in computational tests. For solving randomly generated generalized minimum cost flow problems it was far superior than simplex based general purpose LP codes. However it takes about 3 times the computer time than a corresponding code does to solve a pure minimum cost flow problem of comparable size.

Several other algorithms are available for this problem. One of the earliest is a primal-dual algorithm due to Jewell [1962]. Jensen and Bhaumik [1977] describe a shortest augmenting path type approach for this problem. Bertsekas and Tseng[1988] have extended their relaxation methods (discussed in Section 5.6 for pure minimum cost flow problems) to handle minimum cost flow problems in generalized networks.

8.5 References

J. C. ARANTES, 1991, "Resolution of Degeneracy in Generalized Networks and Penalty Methods for Linear Programming," Ph. D. dissertation, University of Michigan, Ann Arbor, MI.

J. C. ARANTES, J. R. BIRGE, and K. G. MURTY, 1992, "Studies of Lexicography in the Generalized Network Simplex Method," Tech. report, University of Cincinnati, Cincinnati, Ohio.

D. BERTSEKAS and P. TSENG, Jan. - Feb. 1988, "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems," *OR*, 36, n0. 1(93-114).

G. BHAUMIK, 1973, "Optimum Operating Policies of a Water Distribution System with Losses," Ph. D. dissertation, University of Texas at Austin, TX.

G. G. BROWN and R. MCBRIDE, 1984, "Solving Generalized Networks," *MS*, 30(1497-1523).

V. CHANDRU, C. R. COULLARD, and D. K. WAGNER, July 1985, "On the Complexity of Recognizing a Class of Generalized Networks," *OR Letters*, 4, n0. 2(75-78).

J. ELAM, F. GLOVER, and D. KLINGMAN, Feb. 1979, "A Strongly Convergent Primal Simplex Algorithm for Generalized Networks," *MOR*, 4, no. 1(39-59).

F. GLOVER, J. HULTZ, D. KLINGMAN, and J. STUTZ, Aug. 1978, "Generalized Networks: A Fundamental Computer Based Planning Tool," *MS*, 24, no. 12(1209-1220).

F. GLOVER and D. KLINGMAN, 1973, "On the Equivalence of Some Generalized Network Problems to Pure Network Problems," *MP*, 4(369-378).

F. GLOVER and J. MULVEY, May-June 1980, "Equivalence of the 0-1 Integer Programming Problem to Discrete Generalized and Pure Networks," *OR*, 28, no. 3(829-836).

F. GLOVER and J. STUTZ, 1973, "Extensions of the Augmented Predecessor Index Method to Generalized Network Problems," *TS*, 7(377-384).

B. GOLDEN, M. LIBERATORE, and C. LIEBERMAN, 1979, "Models and Solution Techniques for Cash Flow Management," *COR*, 6, no. 1(13-20).

H. W. HAMACHER and S. TUFEKCI, 1987, "On the Use of Lexicographic Minimum Cost Flows in Evacuation Modeling," *NRLQ*, 34(487-503).

R. HASSIN, 1981, "Generalizations of Hoffman's Existence Theorem for Circulations," *Networks*, 11, no. 3(243-254).

P. A. JENSEN and G. BHAUMIK, 1977, "A Flow Augmentation Approach to the Network with Gains Minimal Cost Flow Problem," *MS*, 23(631-643).

W, S, JEWELL, 1962, "Optimal Flow Through Networks with Gains," *OR*, 10(476-499).

M. MALEK-ZAVAREI and J. K. AGGARWAL, 1972, "Optimal Flow in Networks with Gains and Costs," *Networks*, 1(355-365).

J. MAURRAS, 1972, "Optimization of the Flow Through Networks with Gains," *MP*, 3(135-144).

R. D. MCBRIDE, July 1985, "Solving Embedded Generalized Network Problems," *EJOR*, 21, n0. 1(82-92).

W. G. NULTY and M. A. TRICK, April 1988, "GNO/PC Generalized Network

Optimization System," *OR Letters*, 7, no. 2(101-102).

J. B. ORLIN, 1985, "On the Simplex Algorithm for Networks and Generalized Networks," *MPS*, 24(166-178).

K. TRUEMPER, 1977, "On Max Flow with Gains and Pure Min-Cost Flows," *SIAM J. Applied Mathematics*, 32(450-456).

Index

For each index entry we provide the page number where it is defined or discussed first.