Contents

Chapter 5

Algorithms for Minimum Cost Flow Problems in Pure Networks

This chapter will consider algorithms for minimum cost flow problems in pure networks. We begin by considering pure single commodity linear static minimum cost flow problems in Sections 5.1 to 5.8. The assignment, transportation, and shortest chain problems discussed in Chapters 3, 4 are special cases of these problems. Section 5.9 treats the case of a piecewise linear convex cost function. In Section 5.10 we consider dynamic flow problems in pure networks briefly. Finally, in Section 5.11 we present the arc-chain approach for solving multicommodity flow problems in pure networks.

As discussed in Chapter 1, we assume without any loss of generality that our problems are defined on the directed network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell = (\ell_{ij}), k = (k_{ij}), c = (c_{ij}))$, with ℓ, k, c as the lower bound, capacity, and unit cost coefficient vectors for flows on the arcs in $\mathcal{A}$. We assume that $k \geqq \ell \geqq 0$. Some, or all, of the entries in k may be ∞. If a feasible flow vector exists in G, it will be shown later that the cost function is unbounded below on the set of feasible solutions iff there exists an uncapacitated negative cost circuit (i.e., a negative cost circuit consisting only of arcs (i,j) with $k_{ij}=\infty$). By forcing a flow amount of ∞ around an uncapacitated negative cost circuit, the cost can be driven to $-\infty$. We saw this phenomenon also in the shortest chain problem in Chapter 4.

5.1 Different Types of Single Commodity Minimum Cost Flow Models

A common problem, occurring on a directed network $\mathrm{G}=(\mathcal{N},\mathcal{A},\ell,k,c,\breve{s},\breve{t},\overline{v})$, is to ship a specified quantity, $\overline{v}$ units of the commodity from the specified sources $\breve{s}$, to the specified sink $\breve{t}$ in G at minimum cost. It is to find $f=(f_{ij}:(i,j)\in\mathcal{A})$ to

$$\begin{array}{rcl} \text{Minimize } \sum(c_{ij}f_{ij}: \text{ over } (i,j) & \in & \mathcal{A}) \\ \text{Subject to } \; -f(i,\mathcal{N})+f(\mathcal{N},i) & = & \left\{\begin{array}{rl} -\overline{v} & \text{if } i=\breve{s} \\ 0 & \text{if } i\neq\breve{s},\breve{t} \\ \overline{v} & \text{if } i=\breve{t} \end{array}\right. \\ \ell_{ij} \leqq f_{ij} \leqq k_{ij}, \text{ for all } (i,j) & \in & \mathcal{A} \end{array} \tag{5.1}$$

A special case of (5.1) has $\ell=0$, and $k_{ij}=\infty$ for all $(i,j)\in\mathcal{A}$. If it is feasible, and there are no negative cost circuits in G, an optimum flow for this special problem is obtained by sending all the $\overline{v}$ units along a shortest chain from $\breve{s}$ to $\breve{t}$ with c as the arc length vector. So, this special case is equivalent to a shortest chain problem.

Minimum Cost Circulation Problem

Another problem, known as the **minimum cost circulation problem** , is that of finding a minimum cost circulation in a directed network $\mathrm{G}=(\mathcal{N},\mathcal{A},\ell,k,c)$. It is to find $f=(f_{ij})$ to

$$\begin{array}{rcl} \text{Minimize } \sum(c_{ij}f_{ij}: \text{ over } (i,j) & \in & \mathcal{A}) \\ \text{Subject to } \; -f(i,\mathcal{N})+f(\mathcal{N},i) & = & 0, \text{ for each } i\in\mathcal{N} \end{array} \tag{5.2}$$

$$\ell_{ij} \leqq f_{ij} \leqq k_{ij}, \text{ for all } (i,j) \in \mathcal{A}$$

(5.1) is equivalent to a minimum cost circulation problem on an augmented network G$'$ obtained by including in G a new arc $(\breve{t}, \breve{s})$ associated with lower bound and capacity both equal to $\overline{v}$, and unit cost coefficient of 0.

Minimum Cost Flow Model with Exogenous Flow Values

Another model, (5.3), is to find a feasible flow vector $f = (f_{ij})$ in the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c, V = (V_i))$, which is a minimum cost flow satisfying specified exogenous flows at the nodes, given by the vector V. In this model, node i is a **source node** if $V_i > 0$, a **sink node** if $V_i < 0$, and an **intermediate** *or* **transit node** if $V_i = 0$. (5.3) can easily be transformed into a problem of type (5.1) on an augmented network obtained by including a super source and a super sink in G as in Section 2.1.

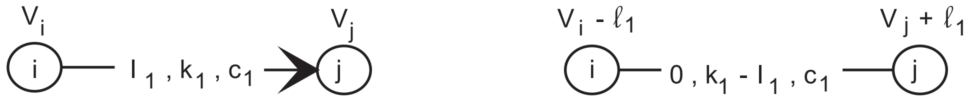
$$\begin{array}{rcl} \text{Minimize } \sum(c_{ij}f_{ij} : \text{ over } (i,j) & \in & \mathcal{A}) \\ \text{Subject to } -f(i,\mathcal{N}) + f(\mathcal{N}, i) & = & -V_i, \text{ for each } i \in \mathcal{N} \\ \ell_{ij} \leqq f_{ij} \leqq k_{ij}, \text{ for all } (i,j) & \in & \mathcal{A} \end{array} \qquad (5.3)$$

The transportation problem is a special case of (5.3) in which the network is bipartite, every node is either a source or a sink node, and all the arcs are directed from a source to a sink node. In a directed network, a node i is called a **shipping node** if $\mathbf{B}_i = \emptyset$, a **receiving node** if $\mathbf{A}_i = \emptyset$, and a **transshipment node** if both $\mathbf{B}_i, \mathbf{A}_i \neq \emptyset$. The transportation problem is a minimum cost flow problem on a network containing no transshipment nodes and vice versa. A minimum cost flow problem on a directed network containing some transshipment nodes is called a **transshipment problem** in the literature.

A necessary condition for (5.3) to be feasible, obtained by summing all the equality constraints in it, is

$$\sum(V_i : \text{ over } i \in \mathcal{N}) = 0 \qquad (5.4)$$

In this model, finite nonzero lower bounds on the variables can easily be transformed into zeros. If $\ell_{ij} \neq 0$ and finite for some $(i,j) \in \mathcal{A}$, transform f_{ij} by substituting $f_{ij} = f'_{ij} + \ell_{ij}$, where f'_{ij} is the new variable replacing f_{ij}. This leads to the transformation of the data on arc (i, j) as shown in Figure 5.1. Verify that in the transformed problem, (5.4) continues to hold if it does so in the original problem. A similar transformation can be carried out for every arc with finite nonzero lower bound.



V_i
V_j
i
l_1, k_1, c_1
j
$V_i - \ell_1$
$V_j + \ell_1$
i
$0, k_1 - l_1, c_1$
j
Original arc with data. Lower bound is finite.
Transformed arc with 0 as lower bound.

Figure 5.1:

Suppose (5.4) does not hold and $U = \sum(V_i :$ over $i \in \mathcal{N}) \neq 0$. In this case we have **excess supply** if $U > 0$, **shortage** otherwise, and the equality constraints in (5.3) cannot be satisfied exactly. If there is excess supply, introduce an artificial sink node $\breve{t}$ with exogenous flow value, $V_{\breve{t}} = -U$; and artificial arcs $(i, \breve{t})$ with lower bound, capacity, unit cost coefficient equal to 0, ∞, 0 respectively, for each source node i. Flows on each of these artificial arcs represent the amount of material unutilized in G at the source nodes on those arcs. Data on the augmented network now satisfy (5.4). We can find an optimum flow vector in G which meets all the requirements at the sink nodes exactly at minimum cost, while leaving U units of material unutilized at the source nodes, by solving the problem of type (5.3) in the augmented network.

If there is shortage, there will be an unfulfilled demand of $|U|$ units. Introduce an artificial source node $\breve{s}$ with an exogenous flow of $|U|$ units; and artificial arcs $(\breve{s}, j)$ with lower bound, capacity, unit cost coefficient equal to 0, ∞, $c_{\breve{s}j}$ respectively for each sink node j. Flow on the artificial arc $(\breve{s}, j)$ represents the unfulfilled requirement at node j. All $c_{\breve{s}j}$ are made 0 if it is just required to find how the existing supply can be used to meet as much of the demand as possible at minimum shipping cost, without giving any special preference to any of the sink nodes. Otherwise $c_{\breve{s}j}$ can be taken to be the per-unit shortage at sink j if such data is available and it is desired to minimize the sum of the shipping costs and the costs of shortage. Or, we can make $c_{\breve{s}j}$ to be suitable positive weights to reflect the priorities for fulfilling the demands at sinks j. With these modifications, data on the augmented network satisfy (5.4), and we can solve the problem of type (5.3) on it.

A General Minimum Cost Flow Model

In this model **S, T**, the sets of source, sink nodes in G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$ are specified. For each $i \in \mathbf{S}$, we are given numbers $a'_i \geqq a_i > 0$, and the net amount of material shipped out of i is required to be between a_i and a'_i. For each $j \in \mathbf{T}$, we are given numbers $b'_j \geqq b_j > 0$, and the net amount of material reaching j is required to be between b_j and b'_j. So, this model is a generalization of the model (5.3), with the exogenous flow amounts V_is being themselves variables subject to lower and upper bound constraints. Introduce two new nodes, $\breve{s}, \breve{t}$, a supersource and supersink. For each $i \in \mathbf{S}$, introduce the arc $(\breve{s}, i)$ with lower bound, capacity, cost coefficient equal to $a_i, a'_i, 0$ respectively. For each $j \in \mathbf{T}$, introduce the arc $(j, \breve{t})$ with lower bound, capacity, cost coefficient equal to $b_j, b'_j, 0$ respectively. Introduce the arc $(\breve{t}, \breve{s})$ with lower bound, capacity, cost coefficient equal to 0, ∞, 0 respectively. Clearly, this general model is equivalent to the minimum cost circulation problem of the form (5.2) in the augmented network.

The Maximum Profit Flow Problem

This problem, (5.5), is the same as (5.1), with the exception that each unit shipped to the sink can be sold there at a **premium** (this is the difference between the selling prices per unit of the material at the sink and the source) of λ. In this problem the data typically satisfies $\ell = 0, k > 0, c \geqq 0$. The objective is to maximize the net profit which is the total premium minus the shipping cost. The flow value v is also a variable in this problem, and typically it is required to be solved as a parametric problem with λ as a nonnegative parameter. For any λ, (5.5) can be posed as a minimum cost circulation problem of the form

(5.2) by introducing the arc $(\breve{t}, \breve{s})$ into the network and making v the flow variable associated with it.

$$\begin{aligned} \text{Maximize } P(f,\lambda) &= \lambda v - \sum (c_{ij} f_{ij} : \text{ over } (i,j) \in \mathcal{A}) \\ \text{Subject to} \quad - & f(i,\mathcal{N}) + f(\mathcal{N},i) = \begin{cases} -v & \text{if } i = \breve{s} \\ 0 & \text{if } i \neq \breve{s}, \breve{t} \\ v & \text{if } i = \breve{t} \end{cases} \quad (5.5) \\ & 0 \leqq f_{ij} \leqq k_{ij}, \text{ for all } (i,j) \in \mathcal{A} \end{aligned}$$

In each of the flow models discussed above on the directed network G with $c = (c_{ij})$ as the cost vector, c can be replaced by the reduced cost vector $\bar{c} = (\bar{c}_{ij} = c_{ij} - (\pi_j - \pi_i))$, where $\pi = (\pi_i)$ is any node price vector in G. For any feasible flow vector f, $cf = \bar{c}f+$ a constant, where the constant is independent of f, but depends only on π and the data in the problem such as V etc. Hence replacing c by $\bar{c}$ does not change the set of optimum solutions, and thus leads to an equivalent problem. Some algorithms make use of this idea.

We have seen that the various flow models discussed above are equivalent. In presenting algorithms, this gives us the freedom to select any of these models, and describe the algorithm as it applies to that model.

5.2 Optimality Conditions

Complementary Slackness Optimality Conditions

Consider the problem (5.3) in the network G. The equality constraints in it are $-Ef = -V$, where E is the node-arc incidence matrix of G. Each of these constraints corresponds to a node, and so the dual variable associated with that constraint can be interpreted as the price of that node in the dual problem. So, the variables in the dual problem are node prices. Given the node price (row) vector π, the complementary slackness optimality conditions (see (1.10) in Chapter 1) for (5.3) and its dual are stated in terms of f and the vector $c - (-\pi E)$. From the definition of E, it can be seen that $-\pi E$ is the row vector of tensions $(\pi_j - \pi_i : (i,j) \in \mathcal{A})$ on the arcs in $\mathcal{A}$ wrt π. From this, it can be verified that these conditions (1.10) simplify to (5.6) given below for this primal dual pair. In the same manner, in each of the problems (5.1), (5.2), or (5.3) on G, a feasible flow vector $f = (f_{ij})$ is optimal iff there exists a vector π of dual variables (or node prices, or node potentials) such that f, π together satisfy: for each $(i,j) \in \mathcal{A}$

$$\begin{aligned} &\text{if } \pi_j - \pi_i > c_{ij} \text{ then } k_{ij} \text{ is finite and } f_{ij} = k_{ij} \\ &\text{if } \pi_j - \pi_i < c_{ij} \text{ then } f_{ij} = \ell_{ij} \end{aligned} \tag{5.6}$$

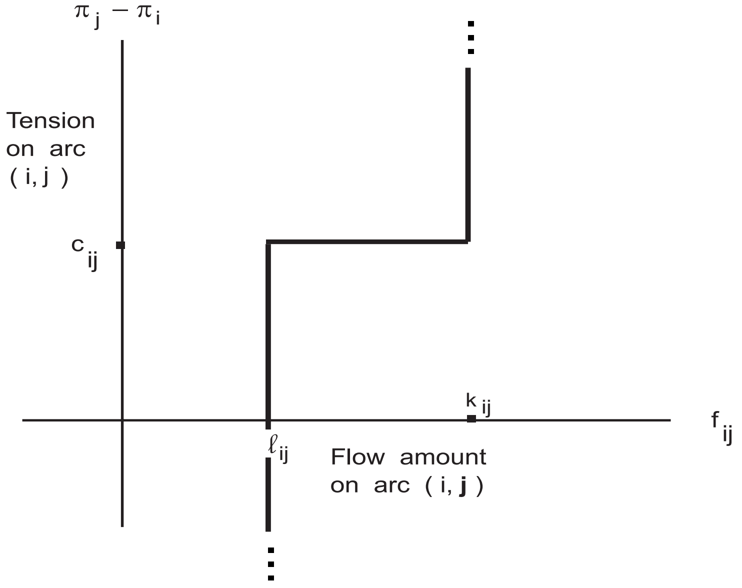


Figure 5.2: C.S. diagram for an arc (i,j) with finite capacity.

These conditions, known as the **complementary slackness optimality conditions**, or **c.s. conditions** in short, can be illustrated in a diagram known as the **complementary slackness diagram** *or* **c.s. diagram** for arc (i,j). See Figures 5.2, 5.3, in which f_{ij} is plotted on the horizontal axis, and the tension $\pi_j - \pi_i$ is plotted on the

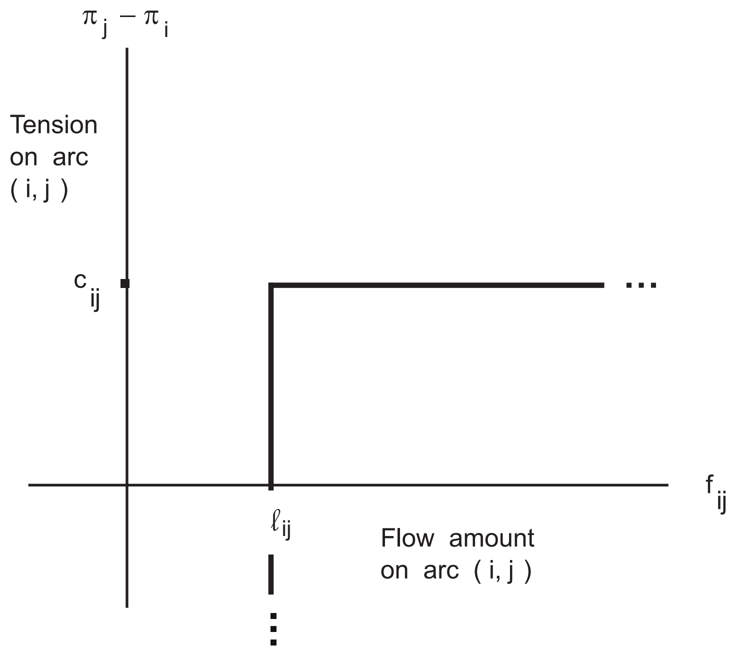


Figure 5.3: C.S. diagram for an arc (i, j) with infinite capacity.

vertical axis. The feasible flow vector-dual vector pair (f, π) satisfies the c.s. conditions for arc (i, j) iff the point $(f_{ij}, \pi_j - \pi_i)$ lies on the chair-shaped curve in Figures 5.2, 5.3.

Optimality Conditions in Terms of Negative Cost Residual Cycles

Consider any of the single commodity minimum cost flow problems (5.1), (5.2), or (5.3) in the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$. Given any path, or an oriented cycle in G, define its cost to be = (the sum of the costs of forward arcs in it) − (the sum of the costs of the reverse arcs in it). Theorem 5.1 given below establishes that a feasible flow vector for this problem is optimal iff there exists no negative cost residual cycle wrt it. The proof of this theorem requires a couple of lemmas which we state and prove first.

LEMMA 5.1 G *is a directed network with* c *as the vector of cost coefficients on the arcs, and* $g^0 = (g^0_{ij}) \geqq 0$ *is a circulation in it satisfying* $cg^0 < 0$. $\mathbf{W}_0 = \{(i,j) : g^0_{ij} > 0\}$. *Then there exists a negative cost simple circuit among the set of arcs* $\mathbf{W}_0$.

Proof Since g^0 is a circulation, if node i lies on an arc in $\mathbf{W}_0$, there must exist at least one arc in $\mathbf{W}_0$ in both the forward and reverse stars of i. Start with such a node i and arcs of the form $(j_1, i), (i, p_1)$ in $\mathbf{W}_0$. Using the same statement again and again trace a chain of the form $i, (i, p_1), p_1, (p_1, p_2), p_2, (p_2, p_3), \ldots$ and a path of the form $i, (j_1, i), j_1, (j_2, j_1), j_2, (j_3, j_2), \ldots$, both beginning at i and consisting of arcs from $\mathbf{W}_0$ only; until either the chain and the path have a common node, say s; or a node is repeated in the chain or the path. The path traced from i to s is actually a chain from s to i in reverse order. When either of these events occur, either the chain, or the path, or both of them put together, have a simple circuit, say $\vec{\mathbb{C}}_0$ from the set of arcs $\mathbf{W}_0$. If the cost of $\vec{\mathbb{C}}_0$ is < 0, we are done. Otherwise let $\alpha = \min. \{ g^0_{ij} : (i,j)$ on $\vec{\mathbb{C}}_0 \}$, subtract α from g^0_{ij} for each arc (i,j) on $\vec{\mathbb{C}}_0$ and let $g^1 = (g^1_{ij})$ be the resulting flow vector in G. Verify that g^1 is again a circulation in G and $g^1 \geqq 0$. The cost of g^1 is $cg^1 = cg^0 - \alpha$ (cost of $\vec{\mathbb{C}}_0) < 0$ since $cg^0 < 0, \alpha > 0$ and the cost of $\vec{\mathbb{C}}_0$ is $\geqq 0$. Also, $\mathbf{W}_1 = \{(i,j) : g^1_{ij} > 0\} \subset \mathbf{W}_0$ and $|\mathbf{W}_1| \leqq |\mathbf{W}_0| - 1$. Now apply the same procedure on $\mathbf{W}_1$, and repeat in the same way. This leads to a sequence of nonnegative circulations $g^1, g^2, \ldots$ in G, all of negative cost with $\mathbf{W}_r = \{(i,j) : g^r_{ij} > 0\}$ satisfying $|\mathbf{W}_r| \leqq |\mathbf{W}_{r-1}| - 1$ for all r, and $\mathbf{W}_0 \supset \mathbf{W}_1 \supset \mathbf{W}_2 \supset \ldots$. So, for some $r \leqq |\mathbf{W}_0| - 2$, the simple circuit found when the procedure is applied on $\mathbf{W}_r$ must have negative cost. This circuit is a negative cost simple circuit in $\mathbf{W}_0$, proving the lemma. ∎.

LEMMA 5.2 *Let G be a directed network with c as the vector of arc cost coefficients, and* $\overline{g} = (\overline{g}_{ij})$ *a negative cost circulation in G (i.e.,* $\overline{g}$ *satisfies flow conservation at all the nodes), but* $\overline{g}$ *may not be* $\geqq 0$. $\mathbf{W} = \{ (i,j) : \overline{g}_{ij} \neq 0 \}$. *Then there exists a negative cost oriented cycle* $\mathbb{C}$ *among the set of arcs* $\mathbf{W}$ *such that* $\overline{g}_{ij} > 0$ *on all forward arcs* (i,j) *on* $\mathbb{C}$, *and* $\overline{g}_{ij} < 0$ *on all reverse arcs* (i,j) *on* $\mathbb{C}$.

Proof If arc (i,j) is such that $\overline{g}_{ij} > 0$, label it with $+$ and leave it as it is. If (i,j) is such that $\overline{g}_{ij} < 0$ reverse its orientation (i.e., replace it with (j,i)), change the cost coefficient on it to $-c_{ij}$, and the flow on it to $-\overline{g}_{ij}$, and label it with $-$. Let the resulting network and the flow vector on it be $\hat{\mathrm{G}}, \hat{g}$. Then $\hat{g} \geqq 0$, it is a circulation in $\hat{\mathrm{G}}$, and its cost in $\hat{\mathrm{G}}$ = the cost of $\overline{g}$ in G, which is < 0. So, by Lemma 5.1 there exists a negative cost simple circuit $\vec{\mathbb{C}}$ in $\hat{\mathrm{G}}$ among the set of arcs on which $\hat{g}$ has positive flow. Verify that changing the orientations of the $-$ labeled arcs on $\vec{\mathbb{C}}$ converts it into a simple cycle $\mathbb{C}$ in G. Orient $\mathbb{C}$ so that the $+$ labeled arcs on it are forward arcs, and verify that it satisfies all the properties stated in the lemma. ∎

THEOREM 5.1 *Let $\overline{f} = (\overline{f})$ be a feasible flow vector in $G = (\mathcal{N}, \mathcal{A}, \ell, k, c)$ for a minimum cost flow problem of the form (5.1), (5.2), or (5.3). $\overline{f}$ is a minimum cost feasible flow vector for this problem iff there exists no negative cost residual cycle* wrt *$\overline{f}$ in G.*

Proof Suppose $\mathbb{C}_0$ is a negative cost residual cycle wrt $\overline{f}$ in G. Let ϵ_0 be the residual capacity of $\mathbb{C}_0$. Let $\hat{f}$ be the flow vector obtained by increasing (decreasing) the flow amount in $\overline{f}$ on forward (reverse) arcs of $\mathbb{C}_0$ by ϵ_0. Since these flow changes are made along the arcs on a simple cycle, we have $\hat{f}(i,\mathcal{N}) - \hat{f}(\mathcal{N},i) = \overline{f}(i,\mathcal{N}) - \overline{f}(\mathcal{N},i)$ for all $i \in \mathcal{N}$. From this and the definition of ϵ_0, it follows that $\hat{f}$ is a feasible flow vector and $c\hat{f} = c\overline{f} + \epsilon_0$ (cost of $\mathbb{C}_0$) $< c\overline{f}$ since $\epsilon_0 > 0$ and the cost of $\mathbb{C}_0$ is < 0, so $\overline{f}$ is not a minimum cost flow for the problem.

To prove the "only if" part, suppose $\overline{f}$ is a feasible but not a minimum cost flow vector for the problem. Then there must exist a feasible flow vector f^0 whose cost is strictly less than that of $\overline{f}$. Define $g = (g_{ij}) = f^0 - \overline{f}$. g is a negative cost circulation in G, but it may not be nonnegative or satisfy the lower or upper bound conditions on the arcs. Let $\mathbf{W}^+ = \{(i,j) : g_{ij} > 0\}$, $\mathbf{W}^- = \{(i,j) : g_{ij} < 0\}$, $\mathbf{W} = \mathbf{W}^+ \cup \mathbf{W}^-$. Since both f^0 and $\overline{f}$ are feasible flow vectors in G, we have

$$\begin{aligned} k_{ij} &\geqq f^0_{ij} > \overline{f}_{ij}, \text{ for all } (i,j) \in \mathbf{W}^+ \\ \ell_{ij} &\leqq f^0_{ij} < \overline{f}_{ij}, \text{ for all } (i,j) \in \mathbf{W}^- \end{aligned}$$

By applying Lemma 5.2 to g in G, we conclude that there must exist a negative cost oriented cycle $\mathbb{C}_0$ consisting of arcs from $\mathbf{W}$ satisfying: $g_{ij} > 0$ for forward arcs on $\mathbb{C}_0$ and $g_{ij} < 0$ for reverse arcs on $\mathbb{C}_0$. Verify that $\mathbb{C}_0$ is a negative cost residual cycle wrt $\overline{f}$. ∎

An equivalent statement to Theorem 5.1 is that a feasible flow vector f in G for problems (5.1), (5.2), or (5.3) is optimal iff the residual networks G(f) or G(f, π) for any node price vector π contain no negative cost circuits.

Canceling a Residual Cycle

Let f be a feasible flow vector in the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$ for (5.1), (5.2), or (5.3). Let $\mathbb{C}$ be a residual cycle wrt f of residual capacity α. Let $\hat{f}$ be the flow vector in G obtained by increasing (decreasing) the flow amount in f on forward (reverse) arcs of $\mathbb{C}$ by α. Clearly $\hat{f}$ is also a feasible flow vector for the problem. The operation of obtaining $\hat{f}$ from f is called **canceling the residual cycle $\mathbb{C}$ in f.** $c\hat{f} = cf + \alpha$ (cost of $\mathbb{C}$). Since $\alpha > 0$, canceling a negative cost residual cycle strictly reduces the cost. As an example consider the network in Figure 5.7 later on. The data on the arcs is the lower bound, capacity, cost coefficient, in that order. A feasible flow vector of value 12 and cost 126 is marked in Figure 5.7 with the flow on each arc entered inside a box by the side of the arc if it is nonzero. The cycle 1, (1, 3), 3, (3, 5), 5, (2, 5), 2, (1, 2), 1 with (1, 2), (2, 5) as reverse arcs and (1, 3), (3, 5) as forward arcs, is a negative cost residual cycle wrt this flow vector. Its residual capacity is min. $\{10 - 7, 4 - 3, 6, 5\} = 1$, and its cost is -7. Canceling this residual cycle leads to the feasible flow vector marked in Figure 5.8 with cost 119.

Many of the algorithms for solving minimum cost flow problems, such as the out-of-kilter algorithm (Section 5.3), the primal simplex algorithm (Section 5.4), the Goldberg-Tarjan algorithm (Section 5.8), are all based on the operation of finding and canceling negative cost residual cycles repeatedly.

Some Results on Optimum Solutions

LEMMA 5.3 *In (5.1) suppose $\ell = 0, k > 0$ and there exists no neg-*

ative cost circuit in G. Let δ = min. $\{k_{ij} : (i,j) \in \mathcal{A}\}$. If there is a chain from $\breve{s}$ to $\breve{t}$ in G, any flow vector which sends a flow amount of $\overline{v}$ along all the arcs of a shortest chain from $\breve{s}$ to $\breve{t}$ with c as the vector of arc lengths, is an optimum flow for (5.1) for all $0 \leqq \overline{v} \leqq \delta$.

Proof Let $\mathcal{C}$ be any such shortest chain from $\breve{s}$ to $\breve{t}$. By the results in Chapter 4, there exists a node price vector $\tilde{\pi} = (\tilde{\pi}_i)$ such that

$$\tilde{\pi}_j - \tilde{\pi}_i \begin{cases} = c_{ij} \text{ for all arcs } (i,j) \text{ on } \mathcal{C} \\ \leqq c_{ij} \text{ for all other arcs.} \end{cases} \tag{5.7}$$

Define $\tilde{f}_{ij} = \overline{v}$ for (i,j) on $\mathcal{C}$, $= 0$ otherwise, and let $\tilde{f} = (\tilde{f}_{ij})$. Then $\tilde{f}$ is feasible to (5.1) and by (5.7), $(\tilde{f}, \tilde{\pi})$ satisfy (5.6). So, $\tilde{f}$ is optimal to (5.1) for this value $\overline{v}$. ∎

Under the hypothesis in Lemma 5.3, one is tempted to think of the following scheme for solving (5.1) in G for any $\overline{v} \geqq 0$. The scheme begins with an initial optimum flow vector of small value obtained as in the proof of Lemma 5.3. Then it tries to augment flow successively on the cheapest available chain from $\breve{s}$ to $\breve{t}$ in each step, dropping arcs from further consideration once they become saturated, until the value reaches $\overline{v}$. Since flow augmentation is carried out only along chains, from Chapter 2 we know that when (5.1) is feasible, this scheme may not even find a feasible flow vector at termination. However, it seems highly intuitive that if a feasible flow vector is obtained at termination of this scheme, it will be a minimum cost flow. Unfortunately, this may not be true, as in the network in Figure 5.4 constructed by Mike Plantholt. All lower bounds are 0, and the data on the arcs is the capacity, unit cost coefficient, in that order. We require a minimum cost flow of value 2. With the cost coefficients as the lengths, the shortest chain from 1 to 6 is 1, (1, 2), 2, (2, 5), 5, (5, 6), 6. The capacity of this chain is 1, so, applying the above scheme, we get the initial flow vector of $f^1 = (f^1_{12}, f^1_{13}, f^1_{24}, f^1_{25}, f^1_{34}, f^1_{35}, f^1_{46}, f^1_{56}) = (1, 0, 0, 1, 0, 0, 0, 1)$ of value $v^1 = 1$. In all the flow vectors, we will order the arcs in the same order as in f^1. In f^1 arcs (1, 2), (2, 5), (5, 6) are saturated, which we eliminate from further consideration. The shortest chain from 1 to 6 in the remaining network is 1, (1, 3), 3, (3, 4), 4, (4, 6), 6 with a cost of 1003. Augmenting the flow on each of the arcs of this chain by 1 leads to the flow vector $f^2 = (1, 1, 0, 1, 1, 0, 1, 1)$, of value 2 and cost 1006. f^2 is not a minimum cost flow vector for this problem since the flow vector $f = (1, 1, 1, 0, 0, 1, 1, 1)$ is feasible and has a cost of only 8.

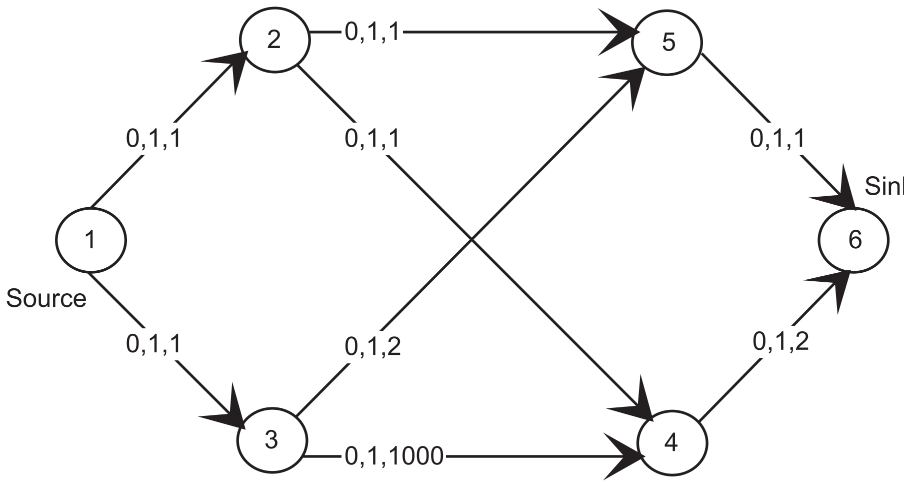


Figure 5.4:

Given a minimum cost feasible flow vector f of value v, to get minimum cost flow vectors of value $> v$ one may have to reduce the flow amounts in f on some of the arcs, and reroute those amounts. The above scheme never reduces the flow amount on any arc and that's why it didn't work. However, see Exercise 5.21 for minimum cost flow problems on a very special class of networks, for which this scheme works.

We have seen that flow augmentation along a minimum-cost FAC may not preserve optimality. However, we now show that flow augmentation along any minimum cost FAP does always preserve optimality.

THEOREM 5.2 *Let $\overline{f}$ be a minimum cost flow vector in $G = (\mathcal{N}, \mathcal{A}, \ell, k, c, \breve{s}, \breve{t})$ of value $\overline{v}$. $G(\overline{f}) = (\mathcal{N}, \mathcal{A}(\overline{f}), 0, \kappa, c')$ is the residual network* wrt *$\overline{f}$.*

(i) *The cost of any chain from $\breve{s}$ to $\breve{t}$ in $G(\overline{f})$ is the same as the cost of the corresponding FAP in G. So, every shortest chain from $\breve{s}$*

to $\breve{t}$ in $G(\overline{f})$ with c' as the arc length vector, corresponds to a minimum cost FAP from $\breve{s}$ to $\breve{t}$ wrt *$\overline{f}$ in G.*

(ii) *If there exists no chain from $\breve{s}$ to $\breve{t}$ in $G(\overline{f})$, $\overline{f}$ is a maximum value flow in G.*

(iii) *Let $\mathcal{C}_0$ be a shortest chain from $\breve{s}$ to $\breve{t}$ in $G(\overline{f})$. Its capacity is δ = min. $\{\kappa_{pq} : (p,q)$ on $\mathcal{C}_0\}$. Define a flow vector $f(\lambda) = (f_{ij}(\lambda) : (i,j) \in \mathcal{A})$ in G, where*

$$f_{ij}(\lambda) = \begin{cases} \overline{f}_{ij} & \text{if } (i,j) \text{ does not correspond to an arc on } \mathcal{C}_0 \\ \overline{f}_{ij} + \lambda & \text{if } (i,j) \text{ corresponds to a + arc on } \mathcal{C}_0 \\ \overline{f}_{ij} - \lambda & \text{if } (i,j) \text{ corresponds to a - arc on } \mathcal{C}_0 \end{cases}$$

Then $\delta > 0$, and $f(\lambda)$ is a minimum cost feasible flow vector in G of value $\overline{v} + \lambda$, for all $0 \leqq \lambda \leqq \delta$.

Proof (i) follows directly from the definitions. If there is no chain from $\breve{s}$ to $\breve{t}$ in G($\overline{f}$), there exists no FAP from $\breve{s}$ to $\breve{t}$ wrt $\overline{f}$ in G, hence $\overline{f}$ is a maximum value feasible flow vector in G by Theorem 2.3, establishing (ii).

Clearly $f(\lambda)$ is a feasible flow vector in G of value $\overline{v} + \lambda$, for all $0 \leqq \lambda \leqq \delta$. Since $\mathcal{C}_0$ is a shortest chain in G($\overline{f}$), by the results in Chapter 4 there must exist a node price vector $\mu = (\mu_i : i \in \mathcal{N})$ satisfying

$$\mu_q - \mu_p \leqq c'_{pq}, \text{ for all } (p,q) \in \mathcal{A}(\overline{f}) \tag{5.8}$$

and (5.8) holds as an equation for each arc on $\mathcal{C}_0$.

If $(i,j) \in \mathcal{A}$ is such that $\ell_{ij} < \overline{f}_{ij} < k_{ij}$, both (i,j) and (j,i) are in $\mathcal{A}(\overline{f})$; by applying (5.8) to both these arcs we have $\mu_j - \mu_i = c_{ij}$.

If (i,j) is such that $\overline{f}_{ij} = \ell_{ij}$, and $\ell_{ij} < k_{ij}$, then (i,j) is in $\mathcal{A}(\overline{f})$ but not (j,i), and from (5.8) we have $\mu_j - \mu_i \leqq c_{ij}$. Similarly, if (i,j) is such that $\overline{f}_{ij} = k_{ij}$, and $\ell_{ij} < k_{ij}$, then (j,i) is in $\mathcal{A}(\overline{f})$ but not (i,j), and from (5.8) we have $\mu_j - \mu_i \geqq c_{ij}$. For all arcs $(i,j) \in \mathcal{A}$

which correspond to an arc in $\mathcal{C}_0, \mu_j - \mu_i = c_{ij}$, since (5.8) holds as an equation for those arcs.

These facts together imply that $f(\lambda), \mu$ together satisfy the complementary slackness optimality conditions (5.6). So, $f(\lambda)$ is an optimum feasible flow vector for all $0 \leqq \lambda \leqq \delta$. ∎

Since shortest chains in $\mathrm{G}(\overline{f})$ and $\mathrm{G}(\overline{f}, \pi)$ for any node price vector π are the same, the results in Theorem 5.2 continue to hold if we replace $\mathrm{G}(\overline{f})$ by $\mathrm{G}(\overline{f}, \pi)$. Given a minimum cost flow vector for (5.1) for some $\overline{v}$, these results can be used to find minimum cost flow vectors of higher values by augmenting successively along cheapest FAPs until the desired value is reached. This is the **incremental**, *or* **build-up approach** for solving this problem. It is the basis for the shortest augmenting path method for minimum cost flows (Sections 5.3, 5.5). This approach is also useful when it is required to solve (5.1) parametrically, treating $\overline{v}$ as a parameter. Also, since the maximum profit flow problem (5.5) is basically a parametric problem, the algorithm for it discussed in Section 5.3 is derived by applying this approach to that problem.

There are three approaches on which most of the practical minimum cost flow algorithms are based. They are: (1) the shortest augmenting path, or the incremental approach, (2) the negative cost residual cycle approach, and (3) the primal-dual approach. We discuss several of these algorithms next.

Exercises

5.1 Let $\overline{f}$ be a minimum cost feasible flow vector for (5.1) in G, and $\mathcal{P}$ a shortest (i.e., minimum cost) augmenting path wrt $\overline{f}$ from $\breve{s}$ to $\breve{t}$ of residual capacity ϵ. Let $f(\lambda)$ be the flow vector obtained by augmenting the flow by λ along $\mathcal{P}$, for $0 \leqq \lambda \leqq \epsilon$. Let $\mathcal{C}$ be the shortest chain in $\mathrm{G}(\overline{f})$ corresponding to $\mathcal{P}$. If the residual network $\mathrm{G}(f(\lambda))$ has a negative cost circuit $\vec{\mathbb{C}}$, show that $\vec{\mathbb{C}}$ must have some common arcs with $\mathcal{C}$ (use Theorem 5.1) and that $\vec{\mathbb{C}} \cup \mathcal{C}$ contains a chain in $\mathrm{G}(\overline{f})$ shorter than $\mathcal{C}$, a contradiction. Hence provide an alternate proof of (iii) of Theorem 5.2 using Theorem 5.1.

5.2 Show that the minimum cost flow problem (5.3) can be transformed into a problem of the same type on an augmented network in which the lower bounds associated with all the arcs are 0, and all the capacities are all ∞.

5.3 G $= (\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$ is a directed connected single commodity flow network, with the following additional features. On each arc $(i, j) \in \mathcal{A}$, any flow amount $\leqq$ the capacity k_{ij} goes through absolutely free of cost. It is possible however to send on this arc any amount of flow $>$ the capacity, at a cost of $\$d_{ij}$ per additional unit. $d = (d_{ij})$ is given. The second feature is that we get a reward of $\$\mu$ for each unit of material reaching $\breve{t}$ from $\breve{s}$. It is required to find a conservative flow vector (i.e., one satisfying flow conservation equations at every node) in G that maximizes the net return. Formulate this as a minimum cost circulation problem.

5.4 Consider the minimum cost flow problem (5.1). $[\mathbf{X}, \overline{\mathbf{X}}]$ is an arbitrary cut separating $\breve{s}$ and $\breve{t}$ in G. Determine the new cost vector $c' = (c'_{ij})$, where $c'_{ij} = c_{ij} + \alpha$ if $(i, j) \in (\mathbf{X}, \overline{\mathbf{X}})$, $c_{ij} - \alpha$ if $(i, j) \in (\overline{\mathbf{X}}, \mathbf{X})$, or c_{ij} otherwise, for some α. Consider the same problem with c changed to c'. Is there any relationship between the sets of optimum solutions for the two problems? Explain why.

5.3 The Out-of-Kilter Algorithm

Consider any of the minimum cost flow problems discussed in Section 5.1 on the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$. To solve it, the OK method can be initiated with an arbitrary flow vector, node price vector pair (f, π). The practical efficiency of the algorithm improves considerably if the initial f is a feasible flow vector; this can be obtained using the methods discussed in Chapter 2. If initiated with a feasible flow vector, all flow vectors in the algorithm will be feasible. The method alternates between a flow change subroutine (during which the node price vector remains unchanged) and a node price change subroutine (during which the flow vector remains unchanged). So, in each step, on each arc (i, j), the point $(f_{ij}, \pi_j - \pi_i)$ either moves horizontally (flow change), or vertically (node price change), and it always moves closer to the chair-shaped c.s. diagram for that arc. We first discuss the version of the algorithm that begins with an initial feasible flow vector.

The **kilter status** of an arc (i, j) wrt a feasible flow vector, node price vector pair $(f = (f_{ij}), \pi = (\pi_i))$ is determined by the position of the point $(f_{ij}, \pi_j - \pi_i)$ on the c.s. diagram. There are five possible states which are determined by the following conditions. (i, j) is an:

$$
\begin{array}{lll}
\alpha - \text{arc} & if & \pi_j - \pi_i < c_{ij} \text{ and } f_{ij} = \ell_{ij} \\
\beta - \text{arc} & if & \pi_j - \pi_i = c_{ij} \text{ and } \ell_{ij} \leqq f_{ij} \leqq k_{ij} \\
\gamma - \text{arc} & if & \pi_j - \pi_i > c_{ij} \text{ and } k_{ij} \text{ is finite and } f_{ij} = k_{ij} \\
a - \text{arc} & if & \pi_j - \pi_i < c_{ij} \text{ and } f_{ij} > \ell_{ij} \\
b - \text{arc} & if & \pi_j - \pi_i > c_{ij} \text{ and } f_{ij} < k_{ij}
\end{array}
$$

See Figure 5.5. The pair (f, π) satisfies the c.s. conditions (5.6) corresponding to $\alpha-, \beta-, \gamma-$arcs, hence these arcs are said to be **in kilter**. It violates the c.s. conditions (5.6) corresponding to a-, b-arcs; hence these arcs are said to be **out-of-kilter**.

The β-arcs can be further classified into 3 distinct classes by the flow amount on them. In the pair (f, π) a β-arc (i, j) is: an **upper boundary** *or* **saturated β-arc** if k_{ij} is finite and $f_{ij} = k_{ij}$; an **interior β-arc** if $\ell_{ij} < f_{ij} < k_{ij}$; a **lower boundary β-arc** if $f_{ij} = \ell_{ij}$.

For each arc (i, j) we define a number called its **kilter number**, denoted by KN(i, j), wrt a pair (f, π), to measure how far away the point $(f_{ij}, \pi_j - \pi_i)$ is from satisfying the c.s. condition. KN(i, j) is always > 0 if (i, j) is out-of-kilter, 0 if it is in-kilter. The kilter numbers are not used in the execution of the algorithm, but only in proving its finite termination property. One possible definition is:

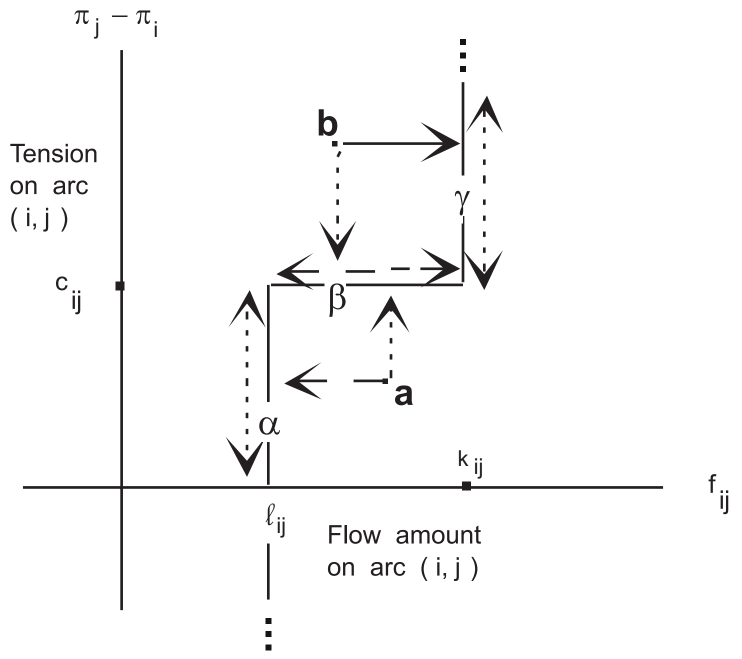


Figure 5.5: Kilter classes and possible changes in f_{ij}, $\pi_j - \pi_i$ in each class for a capacitated arc (i, j).

$$\text{KN}(i,j) = \begin{cases} 0 & \text{if } (i,j) \text{ is an } \alpha\text{-, } \beta\text{-, or } \gamma\text{-arc} \\ f_{ij} - \ell_{ij} & \text{if } (i,j) \text{ is an } a\text{-arc} \\ k_{ij} - f_{ij} & \text{if } (i,j) \text{ is a capacitated } b\text{-arc} \end{cases} \tag{5.9}$$

With this definition, at any stage of the algorithm, the sum of the kilter numbers on all the arcs is a measure (in terms of flow units) of the extent of nonoptimality of the current pair (f, π). Other definitions of kilter numbers are sometimes used; these are given later. In this algorithm, the kilter number of every arc will be monotone nonincreasing, and the algorithm terminates whenever all of them become 0.

Only certain types of flow and tension changes are permitted in this algorithm in order to make sure that in-kilter arcs always stay in-kilter, and out-of-kilter arcs always move closer to the in-kilter status. These permissible changes are summarized below (also see Figure 5.5).

$a-$ arc	f_{ij} can only decrease, up to ℓ_{ij},
	$\pi_j - \pi_i$ can increase only, up to c_{ij}
$b-$ arc	f_{ij} can only increase, up to k_{ij},
	$\pi_j - \pi_i$ can decrease only, up to c_{ij}
$\beta-$ arc	f_{ij} can change freely within ℓ_{ij} to k_{ij} ,
	$\pi_j - \pi_i$ cannot change for interior β-arcs
	can decrease arbitrarily for lower boundary β-arcs
	can increase arbitrarily for upper boundary β-arcs
$\alpha-$ arc	f_{ij} can't change, $\pi_j - \pi_i$ can decrease arbitrarily, or increase up to c_{ij}
$\gamma-$ arc	f_{ij} can't change, $\pi_j - \pi_i$ can increase arbitrarily, or decrease up to c_{ij}

Notice that an uncapacitated b-arc can never be brought any closer to in-kilter status by flow changes only; it can come closer to kilter only by reducing its tension up to its cost coefficient. For this reason, when there are uncapacitated arcs in G, we select the initial node price vector by a special procedure to guarantee that there will never be any uncapacitated b-arcs.

In each stage the algorithm selects an out-of-kilter arc and tries to bring it closer to kilter; this arc is called the **distinguished arc** in that stage. Suppose it is (p, q). The algorithm first tries to bring it into kilter through flow change. If (p, q) is an a-arc, the flow on it, f_{pq}, has to be decreased. In order to maintain feasibility, any decrease in f_{pq} has to be sent from p to q by a different route. Hence, this flow change operation is called **flow rerouting**, p is the **rerouting source** and q is the **rerouting sink** for it.

If the distinguished arc (p, q) is a b-arc, f_{pq} has to be increased, and so the rerouting source, sink are q, p respectively. See Figure 5.6.

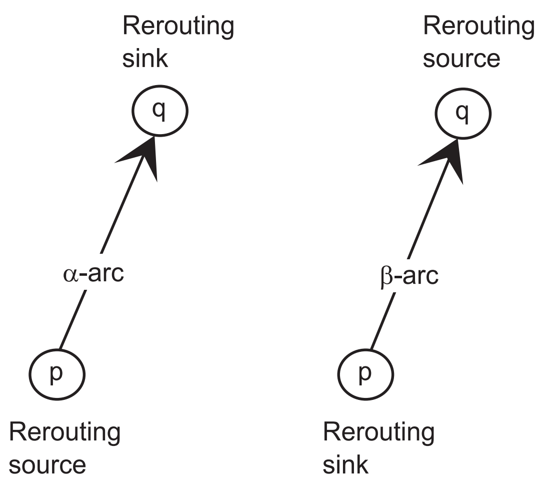


Figure 5.6: Definition of rerouting source and sink on distinguished arc (p, q).

Let $(f = (f_{ij}), \pi)$ be the current pair and (p, q) the distinguished arc. Denote the rerouting source, sink respectively by $\overline{s}, \overline{t}$. Define $\nu = f_{pq} - \ell_{pq}$ $(k_{pq} - f_{pq})$ if (p, q) is an a-arc (b-arc). To bring (p, q) into kilter by flow change alone we need to reroute ν units from $\overline{s}$ to $\overline{t}$. For this, we need to find an FRP (**flow rerouting path**) from $\overline{s}$ to $\overline{t}$ in G which is an admissible path in the sense that flow changes on it are permissible as described above; i.e., all forward (reverse) arcs on it, in which the flow amount increases (decreases) are b- or β-arcs (a- or β-arcs). The algorithm tries to find a shortest (in terms of the number of arcs on it) FRP from $\overline{s}$ to $\overline{t}$ using the tree growth routine described in Section 2.3.3, paying attention to allowable flow changes on the various arcs while growing the tree.

So, any FRP that is obtained in the algorithm will be a simple path, and it can be verified that the distinguished arc (p, q) cannot be contained on it. If an FRP, $\mathcal{P}$ say, is obtained, its **capacity** is $\epsilon_1 =$ min. $\{k_{ij} - f_{ij} : (i, j)$ a forward arc on $\mathcal{P}\} \cup \{f_{ij} - \ell_{ij} : (i, j)$ a reverse arc on $\mathcal{P}\}$, and let $\epsilon_0 =$ min. $\{\nu, \epsilon_1\}$. Flow rerouting using $\mathcal{P}$ consists of adding ϵ_0 to the flow amounts on all the forward arcs on $\mathcal{P}$ and (p, q) if it is a b-arc, and subtracting ϵ_0 from the flow amounts on all the reverse arcs on $\mathcal{P}$ and (p,q) if it is an a-arc.

Since the distinguished arc joins the rerouting source and sink, when we combine $\mathcal{P}$ with the distinguished arc we get a simple cycle, $\mathbb{C}$ say. Orient $\mathbb{C}$ so that all the forward, reverse arcs on $\mathcal{P}$ remain forward, reverse arcs respectively. Then verify that $\mathbb{C}$ is a residual cycle wrt f, and that its residual capacity is ϵ_0 computed above. Also, all forward arcs on $\mathbb{C}$ are b- or β-arcs, and all reverse arcs on it are a- or β-arcs. So, we have

$$\pi_j - \pi_i \begin{cases} \geqq c_{ij} & \text{for forward arcs } (i,j) \text{ on } \mathbb{C} \\ \leqq c_{ij} & \text{for reverse arcs } (i,j) \text{ on } \mathbb{C} \end{cases} \qquad (5.10)$$

Multiplying (5.10) by $+1$ over forward arcs (i,j) on $\mathbb{C}$, and by -1 over reverse arcs (i,j) on $\mathbb{C}$, and summing, we get $0 >$ cost of $\mathbb{C}$. Hence, $\mathbb{C}$ is a negative cost residual cycle wrt f. It can also be verified that the operation of flow rerouting using $\mathcal{P}$ is exactly the same as canceling the negative cost residual cycle $\mathbb{C}$ in f.

If the tree growth routine is unable to find an FRP it will terminate with a set of labeled nodes $\mathbf{X}$ containing the rerouting source $\bar{s}$ and not $\bar{t}$. This implies that at this stage, it is impossible to bring the distinguished arc any closer to kilter by flow changes. So, we try to do it by node price changes. Let $\overline{\mathbf{X}}$ be the complement of $\mathbf{X}$. Forward arcs of the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ at this stage can be a-, α-, γ-, or saturated β-arcs, but cannot be b-arcs (otherwise the node on that b-arc in $\overline{\mathbf{X}}$ would have been labeled in the tree growth process). Similarly reverse arcs in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ can be b-, γ-, α-, or lower boundary β-arcs, but cannot be a-arcs for similar reasons. Define

$$\mathbf{A}^1 = \{(i,j) : (i,j) \text{ is a forward } a\text{-, or } \alpha\text{-arc in } (\mathbf{X}, \overline{\mathbf{X}})\ \} \qquad (5.11)$$

$$\mathbf{A}^2 = \{(i,j) : (i,j) \text{ is a reverse } b\text{-, or } \gamma\text{-arc in } (\overline{\mathbf{X}}, \mathbf{X})\ \} \qquad (5.12)$$

$$\delta = \min.\ \{|\pi_j - \pi_i - c_{ij}| : (i,j) \in \mathbf{A}^1 \cup \mathbf{A}^2\} \qquad (5.13)$$

It can be verified that the distinguished arc (p,q) is contained in $\mathbf{A}^1 \cup \mathbf{A}^2$, so $\mathbf{A}^1 \cup \mathbf{A}^2 \neq \emptyset$. Also, on each arc in $\mathbf{A}^1$ ($\mathbf{A}^2$) it is permissible to increase (decrease) the tension. These facts imply that δ is a finite positive quantity. It can also be verified that δ is the largest amount by which the tension on all the arcs in $\mathbf{A}^1$ can be increased, and that on all the arcs in $\mathbf{A}^2$ decreased simultaneously, while keeping all these changes permissible as defined above.

The new node price vector will be a vector of the form $\hat{\pi} = (\hat{\pi}_i)$ where $\hat{\pi}_i = \pi_i$ if $i \in \mathbf{X}$, or $= \pi_i + \theta$ if $i \in \overline{\mathbf{X}}$; for a positive quantity θ to be selected suitably. As a result of this change, the tension on an arc does not change if both nodes on it are either in $\mathbf{X}$ or in $\overline{\mathbf{X}}$, increases by θ if the arc is in $(\mathbf{X}, \overline{\mathbf{X}})$, and decreases by θ if that arc is in $(\overline{\mathbf{X}}, \mathbf{X})$. These changes are permissible as defined earlier, but we must make sure that the tension on α-, a-arcs does not increase beyond the cost coefficient, and that on γ-, b-arcs does not decrease below the cost coefficient. For this we need to have $\theta \leqq \delta$. So, δ is the largest value that θ can have, and giving it this value brings the distinguished arc closest to the in-kilter status possible by these changes; hence we choose $\theta = \delta$ for defining the new node price vector. Verify that in the new pair $(f, \hat{\pi})$, the kilter number of every arc in G is either the same as before, or smaller, and all in-kilter arcs remain in kilter. The kilter status of the arcs in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ may change; this is determined for all arcs in the cut. If (p, q) still remains out-of-kilter, it continues to be the distinguished arc, and the algorithm now resumes tree growth from where it was left earlier. When this is done, if there are any α-arcs or unsaturated a-arcs in $(\mathbf{X}, \overline{\mathbf{X}})$, or γ-arcs or b-arcs with flow $>$ lower bound in $(\overline{\mathbf{X}}, \mathbf{X})$, which tied for the minimum in (5.13) in the definition of δ, all the unlabeled nodes on them will get labeled. And the method continues.

We are now ready to describe the OK algorithm. We will begin with the routine for selecting the initial node price vector.

SELECTION OF THE INITIAL NODE PRICE VECTOR IN THE OK ALGORITHM

If there are no uncapacitated arcs, i.e., if k is finite, the initial node price vector π^0 can be selected arbitrarily. So, assume that $\mathbf{U} = \{(i, j)\colon k_{ij} = \infty\} \neq \emptyset$. In this case we will select π^0 so that

$$\pi_j^0 - \pi_i^0 \leqq c_{ij} \text{ for all } (i,j) \in \mathbf{U} \tag{5.14}$$

(5.14) guarantees that there will be no uncapacitated b-arcs in the algorithm. Let $\mathbf{V}$ be the set of nodes on arcs in $\mathbf{U}$, and $c^1 = (c_{ij} : (i,j) \in \mathbf{U})$. $\mathrm{P} = (\mathbf{V}, \mathbf{U}, c^1)$ is the partial subnetwork consisting of the uncapacitated arcs in G. If $c^1 \geqq 0$, select $\pi^0 = (\pi_i^0)$ where $\pi_i^0 = 0$ if $i \in \mathbf{V}$, and arbitrary if $i \notin \mathbf{V}$.

If $c^1 \ngeqq 0$, select any node in $\mathbf{V}$, say i_1, and find a shortest chain tree rooted at i_1 in P by any of the efficient algorithms of Section 4.3. If this algorithm terminates by discovering a negative cost circuit $\vec{\mathbb{C}}$ in P, our minimum cost flow problem in G is unbounded below (start with any feasible flow, and add an amount η to the flow amount on each arc of $\vec{\mathbb{C}}$, as $\eta \to \infty$ the cost of this feasible flow $\to -\infty$), so terminate.

If the unboundedness termination did not occur, the shortest chain algorithm will obtain a shortest chain tree rooted at i_1 in P spanning all the nodes that can be reached from i_1 by a chain in it. Let this tree be $\mathbb{T}_1$. For each node i in $\mathbb{T}_1$ select π_i^0 to be the shortest chain length (with c^1 as the arc length vector) from i_1 to i in $\mathbb{T}_1$. If $\mathbb{T}_1$ does not span all the nodes in $\mathbf{V}$, any arc in $\mathbf{U}$ joining a node in $\mathbf{V}$ outside of $\mathbb{T}_1$ and a node in $\mathbb{T}_1$ must be directed towards the node in $\mathbb{T}_1$ (since there are no chains from i_1 to any node outside of $\mathbb{T}_1$ in P). Delete all the nodes in $\mathbb{T}_1$ and all the arcs incident at them from P and suppose this leads to $\mathrm{P}^1 = (\mathbf{V}^1, \mathbf{U}^1, c'^1)$. Select any node i_2 in $\mathbf{V}^1$ and find a shortest chain tree rooted at i_2 in P^1; let it be $\mathbb{T}_2$. Let μ be the vector of shortest chain lengths from i_2 in $\mathbb{T}_2$. For each node i in $\mathbb{T}_2$ choose π_i^0 to be $\mu_i + \vartheta$, where ϑ is a positive quantity, same for all nodes in $\mathbb{T}_2$, selected so that $\pi_j^0 - \pi_i^0 \leqq c_{ij}$ holds for all arcs $(i,j) \in \mathbf{U}$ with i in $\mathbb{T}_2$ and j in $\mathbb{T}_1$. If $\mathbb{T}_2$ does not span the nodes in $\mathbf{V}^1$, repeat the same process. If $c^1 \ngeqq 0$, and there is no negative cost circuit in P, by a suitable choice of ϑ in each step of this procedure, it is possible to select a spanning tree in each connected component of P, so that the node price vector (π_j^0) satisfying (5.14) in P obtained in the procedure is the node price vector corresponding to these trees.

If $c^1 \ngeqq 0$, another method for obtaining an initial node price vector π^0 satisfying (5.14) is the following. Add an artificial origin node s to the set $\mathbf{V}$, and arcs (s, i) of cost 0 for each $i \in \mathbf{V}$. Let $\mathbf{V}^2 = \mathbf{V} \cup \{s\}$, $\mathbf{U}^2 = \mathbf{U} \cup \{(s, i) : i \in \mathbf{V}\}$, and c^2 the cost vector on $(\mathbf{V}^2, \mathbf{U}^2)$ with cost coefficients of arcs in $\mathbf{U}$ given by those in c^1 and for those of the form (s, i) given by 0. In $(\mathbf{V}^2, \mathbf{U}^2)$ with arc length vector c^2, find a shortest chain tree rooted at s. Since there is an arc of the form (s, i) for each $i \in \mathbf{V}$, every node in $(\mathbf{V}^2, \mathbf{U}^2)$ can be reached from s by a chain. So, the shortest chain algorithm on $(\mathbf{V}^2, \mathbf{U}^2)$ will terminate either by finding a negative cost circuit or a shortest chain tree rooted at s. In the former case, let the circuit be $\vec{\mathbb{C}}$. Since there are no arcs incident into s in $(\mathbf{V}^2, \mathbf{U}^2)$, $\vec{\mathbb{C}}$ cannot contain the node s, so it is a negative cost circuit in the subnetwork P. In the latter case, let $\nu_s = 0, \nu_i$ for $i \in \mathbf{V}$ be the shortest chain distances from s to i in the shortest chain tree obtained. From the optimality conditions for the shortest chain tree, we verify that selecting $\pi_i^0 = \nu_i$ for all $i \in \mathbf{V}$ satisfies (5.14).

Applying either of these methods, we will find a node price vector π^0 in G satisfying (5. 14), if there is no negative cost circuit consisting only of uncapacitated arcs in it. The version of the out-of-kilter algorithm discussed next is initiated with such a node price vector and a feasible flow vector.

THE OUT-OF-KILTER ALGORITHM INITIATED WITH A FEASIBLE FLOW VECTOR

Step 1 Initialization Let f^0 be a feasible flow vector for the problem. Select a node price vector by the routine described above. If a negative cost circuit consisting only of uncapacitated arcs is discovered in this process, the cost function is unbounded below in the problem; terminate. Otherwise, let π^0 be the node price vector obtained. (f^0, π^0) is the initial pair. If this pair satisfies the c.s. conditions, it is an optimum pair; terminate. Otherwise go to Step 2.

Step 2 Selecting a distinguished arc Select an out-of-kilter arc, say (p, q) as the distinguished arc. Go to Step 3.

Step 3 Labeling routine Identify the rerouting source $\overline{s}$ and label it with $\emptyset$. List $= \{\overline{s}\}$. Go to Substep 1.

Substep 1 Select the node to scan from list If list $= \emptyset$ go to Step 5. Otherwise select the node from the top of the list to scan, delete it from the list, and go to Substep 2.

Substep 2 Scanning Let i be the node to be scanned. Let $(f = (f_{ij}), \pi)$ be the present pair.

(i) Forward labeling Label each node j unlabeled at this stage, such that (i, j) is either a b- or β-arc and $f_{ij} < k_{ij}$ with $(i, +)$ and include it at the bottom of the list.

(ii) Reverse labeling Label each node j unlabeled at this stage, such that (j, i) is either an a- or β-arc and $f_{ji} > \ell_{ji}$ with $(i, -)$ and include it at the bottom of the list.
If the rerouting sink $\bar{t}$ is now labeled, there is a breakthrough, and an FRP from $\bar{s}$ to $\bar{t}$ has been identified; go to Step 4. Otherwise, go back to Substep 1.

Step 4 Flow rerouting Find the FRP, $\mathcal{P}$, from $\bar{s}$ to $\bar{t}$, by a backward trace of the labels beginning at $\bar{t}$. When $\mathcal{P}$ is combined with the distiguished arc we get a negative cost residual cycle, $\mathbb{C}$. Orient $\mathbb{C}$ so that all forward, reverse arcs on $\mathcal{P}$ remain forward, reverse arcs respectively on $\mathbb{C}$. Cancel $\mathbb{C}$ in f, leading to the new flow vector $\hat{f}$. Find the new kilter status of each arc on $\mathbb{C}$ wrt the new pair $(\hat{f}, \pi)$. If all the arcs in the network are now in-kilter, the present pair is an optimum pair; terminate. If there are still some out-of-kilter arcs, but the distinguished arc is now in-kilter, go back to Step 2. If the distinguished arc continues to be out-of-kilter, erase the labels on all the nodes and go back to Step 3 with the same distinguished arc.

Step 5 Node price vector change Let $\mathbf{X}$ be the set of labeled nodes at this stage, and $\overline{\mathbf{X}}$ its complement. Find the quantity δ as in (5.11), (5.12), (5.13). Define the new node price vector to be $\hat{\pi} = (\hat{\pi}_i)$ where $\hat{\pi}_i = \pi_i$ for $i \in \mathbf{X}$, and $= \pi_i + \delta$ for $i \in \overline{\mathbf{X}}$. Find the new kilter status of each arc in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ wrt the new pair $(f, \hat{\pi})$. If all the arcs in the network are now in-kilter, the present pair is an optimum pair; terminate. If there are still some out-of-kilter arcs, but the distinguished arc is now in-kilter, go back to Step 2. If the distinguished arc continues to be out-of-kilter, make the list = **X**, and resume tree growth by going to Substep 1 in Step 3.

EXAMPLE 5.1

The network G $= (\mathcal{N}, \mathcal{A}, \ell = 0, k, c, \breve{s} = 1, \breve{t} = 6, \overline{v} = 12)$ for a minimum cost flow problem is given in Figure 5.7. Data on each arc are lower bound, capacity, cost coefficient, in that order. A feasible flow vector of specified value 12 is entered inside little boxes by the side of the arcs with nonzero flow amounts. Node prices in an initial node price vector are entered by the side of the nodes. The kilter status of each arc is also marked on the arcs.

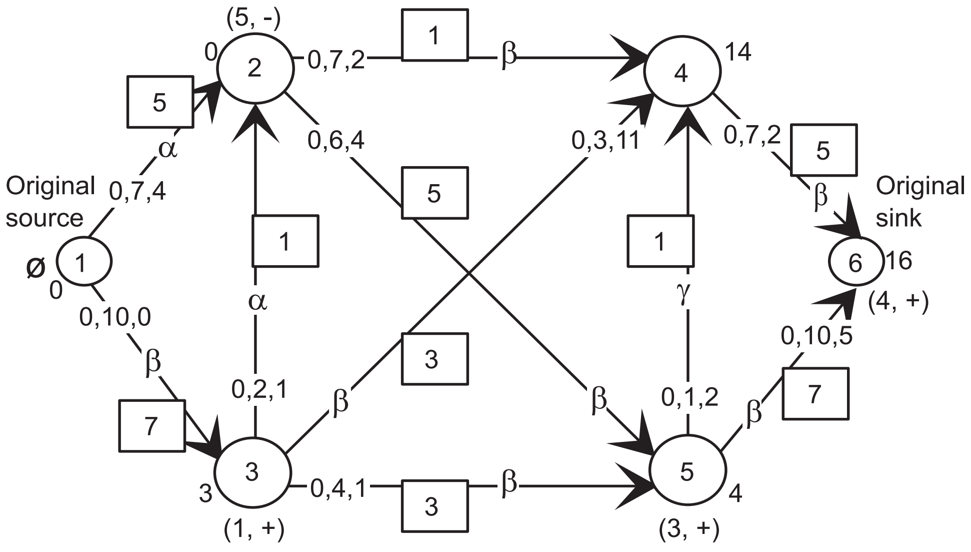


Figure 5.7: Arc (1, 2) is the distinguished arc. Nodes 1, 2 are the rerouting source, sink.

We select the a-arc (1, 2) as the distinguished arc. The rerouting source, sink are 1, 2 respectively. The labeling routine is applied and the labels are entered by the side of the nodes. Breakthrough has occurred, an FRP has been identified, and the negative cost residual cycle is 1, (1, 3), 3, (3, 5), 5, (2, 5), 2, (1, 2), 1, with residual capacity 1 and cost -7. The new flow vector obtained when this cycle is canceled is shown in Figure 5.8, together with the new kilter status of all the arcs. Notice that this step has resulted in a change in the objective value of -7.

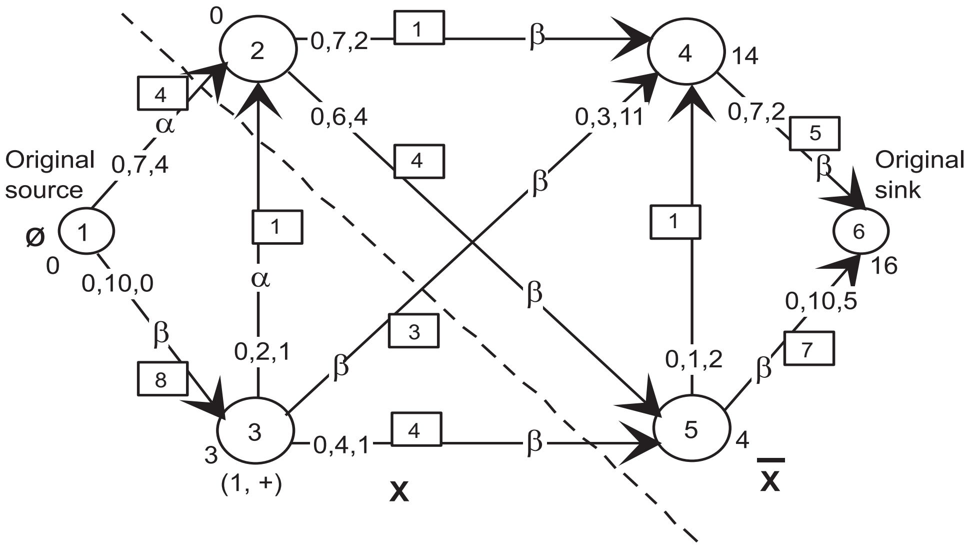


Figure 5.8: Arc (1, 2) is the distinguished arc. 1, 2 are the rerouting source, sink. Labeling ends in a nonbreakthrough with the cut [$\mathbf{X} = \{1, 3\}$, $\overline{\mathbf{X}} = \{2, 4, 5, 6\}$].

Arc (1, 2) continues to be an a-arc in Figure 5.8, so it remains the distinguished arc. Labeling routine is entered afresh resulting in the node labels shown in Figure 5.8. It ends in a nonbreakthrough, with the cut [$\mathbf{X} = \{ 1, 3 \}$, $\overline{\mathbf{X}} = \{ 2, 4, 5, 6 \}$]. $\mathbf{A}^1 = \{ (1, 2), (3, 2) \}$, $\mathbf{A}^2 = \emptyset$. So, $\delta = \text{min. } \{ 4, 4\} = 4$. The new node price vector and the new kilter status of each arc are shown in Figure 5.9. Arcs (2, 4), (5, 6) are still out-of-kilter. One of these can be selected as the distinguished arc and the algorithm continued.

An optimum flow vector, node price vector pair for this problem is

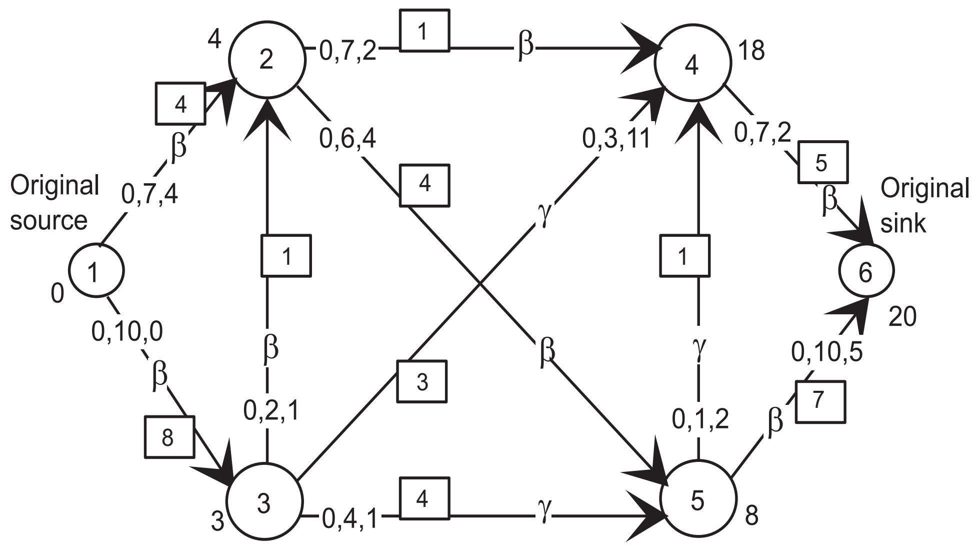


Figure 5.9: Arc (1, 2) is now in kilter.

($f = (f_{12}, f_{13}, f_{32}, f_{34}, f_{35}, f_{25}, f_{24}, f_{54}, f_{46}, f_{56})$, $\pi = (\pi_1$, to $\pi_6)) =$ ((6, 6, 2, 0, 4, 1, 7, 0, 7, 5), (11, 15, 11, 21, 19, 24)).

Discussion

When Step 4 occurs in the algorithm, if the cycle canceled is $\mathbb{C}$ with residual capacity ϵ and cost γ, the cost function changes by $\epsilon\gamma$ which is < 0 as $\epsilon > 0$ and $\gamma < 0$. Hence each time Step 4 occurs, the objective value strictly decreases.

Whenever Step 4 occurs, the kilter status of the arcs on the cycle $\mathbb{C}$ canceled may change (there won't be any changes for arcs not on $\mathbb{C}$ as the flow remains unaltered on them). KN(i, j) decreases by the residual capacity of $\mathbb{C}$ for all a-, b-arcs on $\mathbb{C}$. An a-arc on $\mathbb{C}$ may remain an a-arc, or become an α-arc. A b-arc may remain a b-arc or become a γ-arc. β-arcs on $\mathbb{C}$ remain β-arcs, but their classification into lower boundary, interior, saturated β-arcs may change.

Whenever Step 5 occurs, the kilter status for arcs in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ may change, but there won't be any change for arcs not in this cut. Tension increases by δ for each arc in $(\mathbf{X}, \overline{\mathbf{X}})$, and decreases by δ for each arc in $(\overline{\mathbf{X}}, \mathbf{X})$. An a-arc in $(\mathbf{X}, \overline{\mathbf{X}})$ may remain an a-arc or become a β-arc. An α-arc in $(\mathbf{X}, \overline{\mathbf{X}})$ may remain an α-arc or become a lower boundary β-arc. A γ-arc in $(\mathbf{X}, \overline{\mathbf{X}})$ remains a γ-arc, and a saturated β-arc in $(\mathbf{X}, \overline{\mathbf{X}})$ becomes a γ-arc. A b-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ may remain a b-arc or become a β-arc. A γ-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ may remain a γ-arc or become a saturated β-arc. A lower boundary β-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ becomes an α-arc.

Finiteness of the OK Algorithm

1. First consider the case where ℓ, k and the initial feasible flow vector f^0 are all integer vectors (c may be arbitrary). Let L denote the sum of the kilter nunbers KN(i,j) over all arcs $(i,j) \in \mathcal{A}$. Let Δ_0 be the value of L in the initial pair (f^0, π^0). By our assumption, if (f^0, π^0) is not an optimum pair, Δ_0 is a positive integer. Also, all flow vectors obtained will be integer vectors, KN(i,j) will be a positive integer for all out-of-kilter arcs (i,j), and whenever Step 4 is carried out in the algorithm, L decereases by at least 1.

   At an occurrence of Step 5 in this algorithm, there are two possibilities. One, denoted by P1, is that either an a-arc in $(\mathbf{X}, \overline{\mathbf{X}})$, or a b-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ ties for the minimum in (5.13) for the definition of δ in that step. Under this possibility these arcs become β-arcs and come into kilter as a result of the node price change in this step, and L undergoes a decrease of at least 1 right away. The second possibility, denoted by P2, is that the only arcs which tie for the minimum in (5.13) in this step are some α-arcs in $(\mathbf{X}, \overline{\mathbf{X}})$, or γ-arcs in $(\overline{\mathbf{X}}, \mathbf{X})$. Under this possibility, all unlabeled nodes on these arcs get labeled once tree growth is resumed after completing this step, and the number of labeled nodes increases by at least 1.

   Thus if Step 5 occurs with possibility P1, L undergoes a decrease of at least 1 right in this step. Step 5 with possibility P2 can occur consecutively at most n times; then it must be followed by

Step 4, which results in a decrease of at least 1 in L.

From these facts we verify that the algorithm is finite in this case, and that its overall complexity is at most $\mathrm{O}(n^{\mathbf{2}}\Delta_0)$, but this grows exponentially with the size of data in the worst case. See Zadeh [1973a, 1973b] for examples in which this exponential growth of computational effort occurs.

2. Now consider the case where c and the initial node price vector π^0 are integer vectors, but ℓ, k, f^0 may be real. In this case the quantity δ in Step 5 of the algorithm will always be a positive integer, and all the node price vectors obtained will be integral. Here we will define the kilter number of an arc (i,j) to be $\mathrm{KN}_1(i,j)$ where

$$\mathrm{KN}_1(i,j) = \begin{cases} 0, \text{ if } (i,j) \text{ is an } \alpha\text{-,}\beta\text{-, or } \gamma\text{-arc} \\ |c_{ij} - (\pi_j - \pi_i)|, \text{ if } (i,j) \text{ is an } a\text{-, or } b\text{-arc} \end{cases} \tag{5.15}$$

$\mathrm{KN}_1(i,j)$ is 0 for all in-kilter arcs, and a strict positive integer for all out-of-kilter arcs. Let L_1 denote the sum of $\mathrm{KN}_1(i,j)$ over all $(i,j) \in \mathcal{A}$, and Δ_1 the value of L_1 in the initial pair (f^0, π^0). Whenever Step 5 occurs in this algorithm, the distinguished arc at that stage is in the set $\mathbf{A}^1 \cup \mathbf{A}^2$ and its kilter number decreases by δ, so L_1 decreases by at least 1. Consider the interval between two consecutive occurrences of Step 5. Only flow changes take place in this interval. Two types of events can occur in this interval. One is the transformation of a b-arc into a γ-arc by flow increase until saturation. The other is the transformation of an a-arc into an α-arc by flow decrease to its lower bound. Both these events result in an out-of-kilter arc coming into kilter, and hence a strict decrease of L_1 by a positive integer. In between two consecutive events, or between the beginning of the interval and the first event, or between the last event and the end of the interval, the kilter status of every arc remains unchanged, and hence the subnetwork on which flow changes are permitted remains the same. The work during this time can be posed as a maximum value flow problem in that permissible subnetwork, which can be solved with at most $O(n^3)$ effort. So, between two consecutive instances of the value of L_1 decreasing (by either Step 5, or one of the two events mentioned above between two consecutive occurrences of Step 5) in this algorithm, the computational effort expended is at most $O(n^3)$. And each time L_1 decreases, it decreases by a positive integer. Hence the overall complexity of the algorithm in this case is at most $O(n^3\Delta_1)$.

3. We now consider the general case in which all the data may be real. For any feasible flow vector f, define $V_i = f(i, \mathcal{N}) - f(\mathcal{N}, i)$ for each $i \in \mathcal{N}$. In the problems (5.1), (5.2), (5.3) that we are considering, all these V_i are specified data and do not depend on the particular feasible f.

   Consider a stage during which a particular arc, (p, q) say, is the distinguished arc. Let f be the feasible flow vector, and $[\mathbf{X}, \overline{\mathbf{X}}]$ the cut when the algorithm has just reached Step 5 some time during this stage. So, at this time there are no b- or unsaturated β-arcs in $(\mathbf{X}, \overline{\mathbf{X}})$; or a- or non-lower boundary β-arcs in $(\overline{\mathbf{X}}, \mathbf{X})$ (otherwise labeling can continue and we would not have arrived at Step 5). Define $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3$ to be the set of α-, saturated β-, γ-arcs respectively in $(\mathbf{X}, \overline{\mathbf{X}})$; $\mathbf{H}_4, \mathbf{H}_5, \mathbf{H}_6$ to be the set of α-, lower boundary β-, γ-arcs respectively in $(\overline{\mathbf{X}}, \mathbf{X})$; $\mathbf{H}_7$ to be the set of a-arcs in $(\mathbf{X}, \overline{\mathbf{X}})$ other than (p, q); $\mathbf{H}_8$ to be the set of b-arcs in $(\overline{\mathbf{X}}, \mathbf{X})$ other than (p, q); and $\mathbf{H}_0 = \{(p, q)\}$. So, $\mathbf{H}_0$ to $\mathbf{H}_8$ is a partition of the cut $[\mathbf{X}, \overline{\mathbf{X}}]$.

   Let $g_0 = \sum(f_{ij} : \text{over } (i, j) \in \mathbf{H}_7) - \sum(f_{ij} : \text{over } (i, j) \in \mathbf{H}_8)$. Let $\omega = -1$ if (p, q) is an a-arc, $+ 1$ if (p, q) is a b-arc. We have

$$\begin{aligned}
\sum_{i \in \mathbf{X}} V_i = V(\mathbf{X}) &= f(\mathbf{X}, \overline{\mathbf{X}}) - f(\overline{\mathbf{X}}, \mathbf{X}) \\
&= g_0 + \sum(\ell_{ij} : \text{ over } (i, j) \in \mathbf{H}_1) \\
&\quad + \sum(k_{ij} : \text{ over } (i, j) \in \mathbf{H}_2 \cup \mathbf{H}_3) \\
&\quad - \sum(\ell_{ij} : \text{ over } (i, j) \in \mathbf{H}_4 \cup \mathbf{H}_5) \\
&\quad - \sum(k_{ij} : \text{ over } (i, j) \in \mathbf{H}_6) - \omega f_{pq}
\end{aligned}$$

Define $g_1 = V(\mathbf{X}) - \sum(\ell_{ij} :$ over $(i,j) \in \mathbf{H}_1) - \sum(k_{ij} :$ over $(i,j) \in \mathbf{H}_2 \cup \mathbf{H}_3) + \sum(\ell_{ij} :$ over $(i,j) \in \mathbf{H}_4 \cup \mathbf{H}_5) + \sum(k_{ij} :$ over $(i,j) \in \mathbf{H}_6)$. Then we have

$$g_0 = g_1 + \omega f_{pq} \tag{5.16}$$

g_1 depends only on the data in the problem, the set $\mathbf{X}$ of nodes, and the partition of the arcs in the set $[\mathbf{X}, \overline{\mathbf{X}}] \setminus (\mathbf{H}_0 \cup \mathbf{H}_7 \cup \mathbf{H}_8)$ into $\mathbf{H}_1$ to $\mathbf{H}_6$, and not on the flow vector. The quantity ωf_{pq} on the right-hand side of (5.16) strictly increases during flow change steps in this stage for bringing (p,q) into kilter. By the permissible flow changes in the algorithm, the quantity g_0 on the left hand side of (5.16) can only decrease during flow change steps in this stage. So, (5.16) implies that any given partition of the set of nondistinguished arcs in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ into $\mathbf{H}_1$ to $\mathbf{H}_8$ can appear at most once when the algorithm arrives at Step 5 during this stage. Since there are only a finite number of cuts separating the rerouting source and sink, and finite number of partitions of the nondistinguished arcs in each into the sets $\mathbf{H}_1$ to $\mathbf{H}_8$, this implies that the overall computational effort involved in this stage for bringing the distinguished arc (p,q) into kilter is finite. There are at most m out-of-kilter arcs to be brought into kilter, and once an arc comes into kilter, it remains in kilter subsequently. This shows that the overall computational effort in the algorithm is finite.

The Parametric Value Minimum Cost Flow Problem

Suppose it is required to solve the minimum cost flow problem (5.1) in G $= (\mathcal{N}, \mathcal{A}, \ell, k, c, \breve{s}, \breve{t}, v)$ treating the flow value v as a parameter for all possible values of v. Let $g(v)$ denote the minimum objective value in this problem as a function of v.

THEOREM 5.3 *Let $(\overline{f}, \overline{\pi})$ be an optimum pair of value $\overline{v}$ in G. So, all arcs in G are in-kilter (i.e., $\alpha-$, $\beta-$, or γ-arcs) in the pair $(\overline{f}, \overline{\pi})$. Any FAP, $\mathcal{P}_0$ say, from $\breve{s}$ to $\breve{t}$* wrt *$\overline{f}$ consisting of β-arcs only has cost $= \overline{\pi}_{\breve{t}} - \overline{\pi}_{\breve{s}}$, and is a least cost FAP from $\breve{s}$ to $\breve{t}$* wrt *$\overline{f}$.*

Proof Since all the arcs (i,j) on $\mathcal{P}_0$ are β-arcs, they satisfy $\overline{\pi}_j - \overline{\pi}_i = c_{ij}$. So, the cost of $\mathcal{P}_0 = \sum$ (c_{ij}: (i,j) forward on $\mathcal{P}_0$) - $\sum$(c_{ij}: (i,j) reverse on $\mathcal{P}_0$) = $\sum$(($\overline{\pi}_j - \overline{\pi}_i$): (i,j) forward on $\mathcal{P}_0$) - $\sum$ (($\overline{\pi}_j - \overline{\pi}_i$): (i,j) reverse on $\mathcal{P}_0$) = $\overline{\pi}_{\breve{t}} - \overline{\pi}_{\breve{s}}$.

If $\mathcal{P}$ is any other FAP from $\breve{s}$ to $\breve{t}$ wrt $\overline{f}$, all forward arcs (i,j) on it must be α- or β-arcs and hence satisfy $\overline{\pi}_j - \overline{\pi}_i \leqq c_{ij}$; and all reverse arcs (i,j) on it must be β- or γ-arcs and hence satisfy $\overline{\pi}_j - \overline{\pi}_i \geqq c_{ij}$. Substituting these in the expression for the cost of $\mathcal{P}$, we see that the cost of $\mathcal{P} \geqq \overline{\pi}_{\breve{t}} - \overline{\pi}_{\breve{s}}$ = the cost of $\mathcal{P}_0$. ∎

THE PARAMETRIC VALUE MINIMUM COST FLOW ALGORITHM

Step 1 Initialization Start with an optimum pair of some value, say (f^1, π^1) of value v^1. To find optimum pairs for $v > v^1$, define the flow changing source $\overline{s} = \breve{s}$, and flow changing sink $\overline{t} = \breve{t}$. Go to Step 2.

Step 2 Labeling routine Label $\overline{s}$ with $\emptyset$. Make list = $\{ \overline{s} \}$. Go to Step 3.

Step 3 Select a node for scanning If list = $\emptyset$ go to Step 6. Otherwise, select the topmost node from the list for scanning. Delete this node from the list and go to Step 4.

Step 4 Scanning Let $(f = (f_{ij}), \pi = (\pi_i))$ be the present optimum pair of value v. Let i be the node being scanned.

Substep 1 Forward labeling For each unlabeled node j such that (i,j) is a β-arc satisfying $f_{ij} < k_{ij}$, label it with $(i, +)$ and include it at the bottom of the list.

Substep 2 Reverse labeling For each unlabeled node j such that (j,i) is a β-arc satisfying $f_{ji} > \ell_{ji}$, label it with $(i, -)$ and include it at the bottom of the list.

If $\overline{t}$ is labeled there is a breakthrough; go to Step 5. Otherwise, go back to Step 3.

Step 5 Flow augmentation Trace the FAP from $\overline{s}$ to $\overline{t}$ using the labels. Suppose it is $\mathcal{P}$ with residual capacity ϵ. For $0 \leqq \theta \leqq \epsilon$

define the flow vector $f(\theta) = (f_{ij}(\theta))$ obtained by augmenting the flow along $\mathcal{P}$ by amount θ. Then $(f(\theta), \pi)$ is an optimum pair of value $v + \theta$ for all $0 \leqq \theta \leqq \epsilon$. In this range the slope of $g(v)$ is $\pi_{\breve{t}} - \pi_{\breve{s}}$. To find optimum pairs for values beyond $v + \epsilon$, erase all the node labels and go back to Step 2 with the new pair $(f(\epsilon), \pi)$.

Step 6 Node price change Let (f, π) be the present pair, $\mathbf{X}$ the set of labeled nodes, and $\overline{\mathbf{X}}$ its complement. Define $\mathbf{A}^1 = \{(i,j) : (i,j)$ a forward α-arc in $(\mathbf{X}, \overline{\mathbf{X}})$ $\}$, $\mathbf{A}^2 = \{(i,j) : (i,j)$ a reverse γ-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ $\}$. If $\mathbf{A}^1 \cup \mathbf{A}^2 = \emptyset$ f is a maximum value flow in G, terminate. If $\mathbf{A}^1 \cup \mathbf{A}^2 \neq \emptyset$, let $\delta =$ min. $\{|\pi_j - \pi_i - c_{ij}| : (i,j) \in \mathbf{A}^1 \cup \mathbf{A}^2\}$. Find the new node price vector $\hat{\pi} = (\hat{\pi}_i)$ where $\hat{\pi}_i = \pi_i$ if $i \in \mathbf{X}$, $= \pi_i + \delta$ if $i \in \overline{\mathbf{X}}$. $(f, \hat{\pi})$ is the new optimum pair, find the new kilter status of each arc in the cut $[\mathbf{X}, \overline{\mathbf{X}}]$ wrt it. Make the list $= \mathbf{X}$ and return to Step 3.

Discussion

Since all the FAPs used in this algorithm are least cost FAPs at the time they are used, this is a shortest augmenting path method for finding optimum flows of increasing value, implemented using the OK method. Once initiated with an optimum pair, the method preserves the in-kilter status of every arc; hence all the pairs obtained in the method are optimal pairs for their value. To find optimum pairs of value $<$ the initial value v^1, begin with the original pair (f^1, π^1), and choose $\breve{t}, \breve{s}$ as the flow changing source, sink respectively, and repeat the same process. Here the FAPs identified will actually be flow reducing paths in the original network, and the flow value in G will decrease with each flow change. This process terminates when the minimum possible flow value in G is reached.

For all v between two consecutive node price change steps in this algorithm $g(v)$ is linear with slope $\pi_{\breve{t}} - \pi_{\breve{s}}$. By the node price updating formula we see that at each node price change step, this slope changes by δ in that step. Hence the slope of $g(v)$ is nondecreasing with v, establishing its convexity. It is piecewise linear convex. So, slope of $g(v)$ changes whenever node prices change in the algorithm, i.e., in Step 6. From LP theory we know that the number of distinct values of the slope of $g(v)$ is finite, so, Step 6 occurs only a finite number of times. In between two consecutive occurrences of Step 6, the kilter status of each arc remains unchanged, and hence the work carried out in this interval is the solution of a maximum value flow problem on the β-arc subnetwork at that stage; this takes only finite effort by the results in Chapter 2. So, this algorithm finds optimum pairs for all possible values after a finite amount of computation.

OK Algorithm Initiated with an Arbitrary Flow Vector

We consider the minimum cost flow problems (5.1), (5.2), (5.3) on the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$. Here we discuss the version of the OK algorithm that can be used to solve this problem beginning with a flow vector f that satisfies the conservation conditions at all the nodes (i.e., the equality conditions in the problem), but may violate the bounds on the nodes. Let (f, π) be such a pair where f is a flow

New Kilter Status for (i, j) wrt (f, π)

Status	Condition	Permissible Flow Change	KN(i,j)
a_1-arc	$f_{ij} < \ell_{ij}, \pi_j - \pi_i < c_{ij}$	Increase up to ℓ_{ij}	$\ell_{ij} - f_{ij}$
a_2-arc	$f_{ij} > k_{ij}, \pi_j - \pi_i < c_{ij}$	Decrease up to ℓ_{ij}	$f_{ij} - \ell_{ij}$
β_1-arc	$f_{ij} < \ell_{ij}, \pi_j - \pi_i = c_{ij}$	Increase up to k_{ij}	$\ell_{ij} - f_{ij}$
β_2-arc	$f_{ij} > k_{ij}, \pi_j - \pi_i = c_{ij}$	Decrease up to ℓ_{ij}	$f_{ij} - k_{ij}$
b_1-arc	$f_{ij} < \ell_{ij}, \pi_j - \pi_i > c_{ij}$	Increase up to k_{ij}	$k_{ij} - f_{ij}$
b_2-arc	$f_{ij} > k_{ij}, \pi_j - \pi_i > c_{ij}$	Decrease up to k_{ij}	$f_{ij} - k_{ij}$

vector of this type. We have the new kilter statuses for arcs wrt (f, π), in addition to the ones discussed earlier, see table above. All these new statuses are out-of-kilter statuses. For each new status we also specify the permissible flow changes on them, and the appropriate definition of the kilter number KN(i, j). There are no constraints on tension changes on arcs with these new statuses. For arcs with the kilter statuses defined earlier, permissible flow and tension changes remain exactly the same as before.

Let f^0 be the initial flow vector satisfying node conservation at all the nodes. Obtain an initial node price vector π^0 by the special procedure described earlier to satisfy $\pi_j^0 - \pi_i^0 \leqq c_{ij}$ for all uncapacitated arcs (i, j). (f^0, π^0) is the initial pair.

As before the algorithm selects an out-of-kilter arc as the distinguished arc, and tries to bring it into kilter by flow rerouting and node price changes alternately. In flow rerouting steps, the flow rerouting source, sink on the distinguished arc (p, q) are p, q respectively if (p, q) is a b_2-, β_2-, a_2-, or a-arc; or q, p respectively if it is an a_1-, b_1-, β_1-, or b-arc.

The labeling step is the same as before with the following exceptions. Node j can be forward labeled $(i, +)$ while scanning node i if $f_{ij} < k_{ij}$ and (i, j) is a b-, β-, a_1-, β_1-, or b_1-arc. Node j can be reverse labeled $(i, -)$ while scanning node i if $f_{ji} > \ell_{ji}$ and (j, i) is an a-, β-, a_2-, β_2-, or b_2-arc.

In the node price change step it may happen that $\mathbf{A}^1 \cup \mathbf{A}^2 = \emptyset$. If this occurs, it can be verified that the cut defined by the labeled and unlabeled nodes at that stage provides an instance where the conditions for the existence of a feasible flow vector (Chapter 2) are violated; hence terminate with the conclusion that no feasible flow vector exists in G.

It can be verified that all the finiteness proofs hold good for this version of the OK algorithm too.

Other Sensitivity Analysis

If changes take place in c_{ij}, ℓ_{ij}, or k_{ij} after an optimum pair is obtained, redefine the kilter status of arc (i, j) using the new values, and apply the OK algorithm again until this arc comes into kilter. Thus the OK algorithm is a very convenient tool for doing sensitivity analysis on minimum cost flow problems.

A Polynomially Bounded Implementation of the OK Algorithm Based on Scaling

We consider the minimum cost circulation problem (5.2) in the directed network $G = (\mathcal{N}, \mathcal{A}, \ell, k, c)$, with $|\mathcal{N}| = n$, $|\mathcal{A}| = m$. Since all the other minimum cost flow problems discussed in Section 5.1 can be transformed into this, it covers all those models. We assume that ℓ, k are integer vectors, and that $0 \leqq \ell \leqq k$. Let p be the smallest positive integer such that all the ℓ_{ij} and the finite capacities k_{ij} are all $\leqq 2^{\mathbf{p}}$. Here we present an implementation of the OK algorithm based on the **scaling technique** discussed earlier in Section 3.2 in the context of the transportation problem, that can solve this problem with a computational complexity of $O((p+1)mn^{\mathbf{2}})$.

The implementation applies the OK algorithm on a series of $(p+1)$ subproblems which provide successively closer approximations to the original problem. The subproblems differ in only the lower bound and capacity vectors. $\ell^r = (\ell^r_{ij}), k^r = (k^r_{ij})$, where $(\ell^r_{ij}) = \lfloor \ell_{ij}/2^{\mathbf{p}-r} \rfloor$, $(k^r_{ij}) = \lceil k_{ij}/2^{\mathbf{p}-r} \rceil$, are the lower bound and capacity vectors for the rth subproblem for $r = 0$ to p. Since $\ell^r_{ij} \leqq \ell_{ij}/2^{\mathbf{p}-r}$, and $k^r_{ij} \geqq k_{ij}/2^{\mathbf{p}-r}$, for all $r = 0$ to p, and for each $(i,j) \in \mathcal{A}$, if f is a feasible circulation for the original problem, $f/2^{\mathbf{p}-r}$ is a feasible circulation for the rth subproblem. Thus if the original problem is feasible, so is every subproblem.

If there are some uncapacitated arcs in G, determine the initial node price vector $\pi^0 = (\pi^0_i)$ so that $\pi^0_j - \pi^0_i \leqq c_{ij}$ for all (i,j) with $k_{ij} = \infty$. This takes $O(nm)$ effort using the efficient methods discussed in Chapter 4. This either finds an uncapacitated negative cost circuit, or finds the vector π^0. In the former case, we terminate, since the objective function is unbounded below in (5.2) if it is feasible.

The 0th subproblem is to find a minimum cost circulation problem in $G^0 = (\mathcal{N}, \mathcal{A}, \ell^0, k^0, c)$. $\ell^0 = 0$ and $k^0_{ij} = 0$ or 1 for all (i,j) with k_{ij} finite. So, $f^0 = 0$ is a feasible circulation in G^0. Use (f^0, π^0) as the initial pair to solve the 0th subproblem by the OK algorithm. The kilter number $\mathrm{KN}(i,j)$ wrt (f^0, π^0) is 0 or 1 for all arcs (i,j). So, solving the 0th problem by the OK algorithm takes at most $O(mn^{\mathbf{2}})$ effort.

In general, suppose we have an optimum pair (f^{r-1}, π^{r-1}) for the $(r-1)$th subproblem. The rth subproblem is to find a minimum cost circulation in $G^r = (\mathcal{N}, \mathcal{A}, \ell^r, k^r, c)$. Solve it by the OK algorithm using $(2f^{r-1}, \pi^{r-1})$ as the initial pair. Since (f^{r-1}, π^{r-1}) is an optimum pair for the $(r-1)$th subproblem, we have for all $(i,j) \in \mathcal{A}$

$$\pi_j^{r-1} - \pi_i^{r-1} > c_{ij} \quad \text{implies} \quad f_{ij}^{r-1} = k_{ij}^{r-1} \tag{5.17}$$

$$\pi_j^{r-1} - \pi_i^{r-1} < c_{ij} \quad \text{implies} \quad f_{ij}^{r-1} = \ell_{ij}^{r-1} \tag{5.18}$$

Arcs (i,j) satisfying (5.17), (5.18) are the γ-, α-arcs respectively at the end of the $(r-1)$th subproblem. Since k_{ij}^r is either $2k_{ij}^{r-1} - 1$ or $2k_{ij}^{r-1}$, if (i,j) satisfies (5.17), it will either be a γ-arc or a b-arc with KN(i,j) of 1 in the pair $(2f^{r-1}, \pi^{r-1})$ for the rth subproblem. By similar arguments it can be verified that every arc has kilter number of 0 or 1 wrt the initial pair in the rth subproblem. So, solving the rth subproblem by the OK algorithm takes a computational effort of at most O(mn^2).

We continue this way, using $(2f^{r-1}, \pi^{r-1})$ as the initial pair for the rth subproblem, where (f^{r-1}, π^{r-1}) is the optimal pair obtained for the $(r-1)$th subproblem, for $r = 1$ to p. Since the initial flow vector may not always be feasible, we may have to use the general version of the OK algorithm that can begin with an infeasible flow satisfying node conservation but not the bounds. If any of the subproblems in the sequence turns out to be infeasible, the original problem must be infeasible too; terminate. Otherwise, the pth subproblem is the original problem itself, and when we come to it and solve it, we will have an optimum pair for it.

The overall computational effort in this implementation is bounded above by O(pmn^2), so it is polynomially bounded. This implementation is basically of theoretical interest; it shows a way of solving minimum cost circulation problems in polynomial time using scaling and the OK algorithm.

The Parametric Maximum Profit Flow Problem

Consider the maximum profit flow problem (5.5) in the directed network G $= (\mathcal{N}, \mathcal{A}, \ell = 0, k, c, \breve{s}, \breve{t}, \lambda)$, where $k > 0, c \geqq 0$ and λ is the premium per unit material reaching $\breve{t}$ from $\breve{s}$. The problem is to be solved parametrically in λ in the range $\lambda \geqq 0$. The c.s. optimality conditions for the feasible flow vector, node price vector pair (f, π) are:

for each $(i, j) \in \mathcal{A}$

$$\begin{aligned} \pi_j - \pi_i > c_{ij} \quad &\text{implies} \quad k_{ij} \text{ is finite and } f_{ij} = k_{ij} \\ \pi_j - \pi_i < c_{ij} \quad &\text{implies} \quad f_{ij} = 0 \end{aligned} \tag{5.19}$$

$$\pi_{\breve{t}} - \pi_{\breve{s}} = \lambda \tag{5.20}$$

Therefore, a feasible flow vector, node price vector pair (f, π) is an optimum pair for a λ iff (5.20) holds, and every arc in the network has kilter status α-, β-, or γ- wrt (f, π) as defined earlier. This is also a consequence of the fact that if (f, π) is an optimum pair for (5.5) for λ, and the value of f is v, then f must minimize $\sum c_{ij} f_{ij}$ among all feasible flow vectors of value v. So, (5.5) can be solved by the OK algorithm for the parametric value minimum cost flow problem discussed earlier. That algorithm generates a series of optimal pairs (f, π) wrt which all the arcs are α-, β-, or γ-arcs, and hence f will be an optimum feasible flow vector for $\lambda = \pi_{\breve{t}} - \pi_{\breve{s}}$, the current value of λ. $\lambda = \pi_{\breve{t}} - \pi_{\breve{s}}$ changes only whenever a node price change step occurs in this algorithm. In between two consecutive node price change steps, flow changes may occur several times with the consequent increase in flow value. Each FAP obtained in that interval will be a least-cost FAP at that stage, of cost = the current λ, and all the flow vectors generated are therefore alternate optimum feasible flow vectors for the current value of λ.

The algorithm is initiated with $(f^0 = 0, \pi^0 = 0)$ which is an optimum pair for $\lambda = \lambda_0 = 0$.

In a general step let $(\overline{f}, \overline{\pi} = (\overline{\pi}_i))$ be the current optimum pair for $\lambda = \overline{\lambda} = \overline{\pi}_{\breve{t}} - \overline{\breve{s}}$. The parametric value minimum cost flow algorithm is continued. If an FAP, $\mathcal{P}$ say, from $\breve{s}$ to $\breve{t}$ wrt $\overline{f}$ is identified at this stage, its cost will be $\overline{\lambda}$, and let ϵ be its residual capacity. We get the new feasible flow vector $f(\theta)$ by carrying out flow augmentation by amount θ on this FAP. Each unit of this quantity arrives at $\breve{t}$ at an incremental cost of $\overline{\lambda}$ which is canceled by the premium earned by this unit when it reaches there; hence the net profit does not change. So, for $0 \leqq \theta \leqq \epsilon$

$f(\theta)$ is an alternate optimum feasible flow vector for $\lambda = \overline{\lambda}$.

If this $\epsilon = \infty$ (this can only happen if $\mathcal{P}$ has no reverse arcs, i.e., $\mathcal{P}$ is an FAC, and all arcs on it are uncapacitated), $f(\theta)$ is feasible for all $\theta \geqq 0$, and the profit associated with $f(\theta)$ diverges to ∞ as $\theta \to \infty$, for any $\lambda > \overline{\lambda}$, terminate.

If ϵ is finite, continue with the new optimum pair $(f(\epsilon), \overline{\pi})$.

Suppose a nonbreakthrough occurs when the current optimum pair is $(\hat{f}, \hat{\pi} = (\hat{\pi}_i)), \lambda = \hat{\lambda} = \hat{\pi}_{\breve{t}} - \hat{\pi}_{\breve{s}}$. Let $\mathbf{X}$ be the set of labeled nodes at this stage and $\overline{\mathbf{X}}$ its complement. Determine the sets $\mathbf{A}^1, \mathbf{A}^2$ as in the algorithm. If $\mathbf{A}^1 \cup \mathbf{A}^2 = \emptyset$, then we have $\hat{f}_{ij} = k_{ij}$ for all $(i, j) \in (\mathbf{X}, \overline{\mathbf{X}})$, and $\hat{f}_{ij} = 0$ for all $(i, j) \in (\overline{\mathbf{X}}, \mathbf{X})$, and so $[\mathbf{X}, \overline{\mathbf{X}}]$ is a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G, and $\hat{f}$ is a maximum value flow. Since all the arcs are in-kilter in the present pair, we know that $\hat{f}$ is a maximum value flow that minimizes $\sum c_{ij} f_{ij}$ among all maximum value flows. This implies that $\hat{f}$ remains a maximum profit flow for all $\lambda \geqq$ its present value $\hat{\lambda}$. Terminate.

If $\mathbf{A}^1 \cup \mathbf{A}^2 \neq \emptyset$, compute δ as in the algorithm and define the new node price vector $\pi(\Delta) = (\pi_i(\Delta))$ where $\pi_i(\Delta) = \hat{\pi}_i$ for $i \in \mathbf{X}$, or $=$ $\hat{\pi}_i + \Delta$ for $i \in \overline{\mathbf{X}}$. It can be verified that $(\hat{f}, \pi(\Delta))$ satisfies (5.19) for all $0 \leqq \Delta \leqq \delta$. So, $(\hat{f}, \pi(\Delta))$ is an optimum pair, i.e., $\hat{f}$ is an optimum flow for $\lambda = \hat{\lambda} + \Delta$, for all $0 \leqq \Delta \leqq \delta$. Continue the algorithm with the new pair $(\hat{f}, \pi(\delta))$ and $\lambda = \hat{\lambda} + \delta$. Verify that at least one arc which is an α-arc in $(\mathbf{X}, \overline{\mathbf{X}})$, or a γ-arc in $(\overline{\mathbf{X}}, \mathbf{X})$ in the pair $(\hat{f}, \hat{\pi})$, will become a β-arc in the new pair $(\hat{f}, \pi(\delta))$, and when tree growth is resumed, the node in $\overline{\mathbf{X}}$ on that arc will get labeled. Hence the total number of consecutive occurrences of a nonbreakthrough before a breakthrough occurs can never exceed n.

Let $Q(\lambda)$ denote the maximum profit, i.e., the optimum objective value in the maximum profit flow problem as a function of the premium λ. In this algorithm, there may be consecutive node price changes without any flow change in between. (f^0, π^0) is the initial pair. Let (f^r, π^r) denote the pair after the rth node price change step is completed and tree growth is about to be resumed again. Let v^r be the value of f^r. v^r is monotonic nondecreasing with r. Let $g_1 =$ smallest r such that $f^r \neq f^0$, and for $d \geqq 1$ let $g_{d+1} =$ smallest $r > g_d$ such that $f^r \neq f^{g_d}$. So, there is no change in the flow vector f^r for $g_d \leqq r \leqq g_{d+1}$ for all *d.* $\lambda_0 = 0$, and let $\lambda_r = \pi^r_{\bar{t}} - \pi^r_{\bar{s}}$. For $\lambda_0 = 0 \leqq \lambda \leqq \lambda_{g_1-1}$, $Q(\lambda) = 0$. For $\lambda_{g_d-1} \leqq \lambda \leqq \lambda_{g_{d+1}-1}$, $Q(\lambda)$ is linear with slope v_{g_d}. It is piecewise linear, and convex since its slope is nondecreasing with λ. See Figure 5.10.

$f^0 = 0$ is optimal for $\lambda_0 = 0 \leqq \lambda \leqq \lambda_{g_1-1}$. For $d \geqq 1$, f^{g_d} is optimal for $\lambda_{g_d-1} \leqq \lambda \leqq \lambda_{g_{d+1}-1}$.

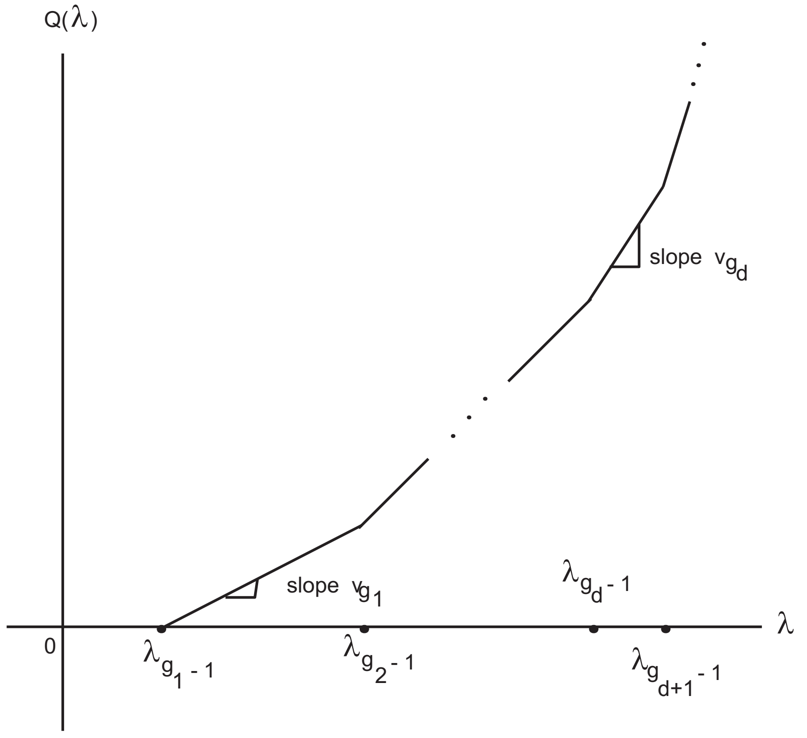


Figure 5.10: $Q(\lambda)$ is piecewise linear and convex.

5.4 The Primal Network Simplex Method

We consider the minimum cost flow problem (5.3) in the directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$. If there is an arc (p, q) for which $\ell_{pq} = k_{pq} = \alpha$, say, fix $f_{pq} = \alpha$, delete (p, q) from further consideration, and change V_p to $V_p - \alpha$ and V_q to $V_q + \alpha$. Repeat with other arcs of this type. After this we will have $\ell_{ij} < k_{ij}$ for each remaining arc (i, j). If the network is not connected, each connected component can be solved separately. So, in the sequel we assume that G is connected, that $\ell < k$, and that the necessary condition for feasibility, (5.4), holds. Let $|\mathcal{N}| = n, |\mathcal{A}| = m$.

A basic solution for this problem corresponds to a **partition** of the set of arcs $\mathcal{A}$ into ($\mathbb{T}$, **L, U**), where $\mathbb{T}$ is a spanning tree in G, and (**L, U**) is a partition of the out-of-tree arcs subject to the condition that all the arcs in **U** are capacitated. The *primal basic solution* corresponding to this partition is $f = (f_{ij})$ where $f_{ij} = \ell_{ij}$ for $(i, j) \in \mathbf{L}$, $= k_{ij}$ for $(i, j) \in \mathbf{U}$, and $(f_{ij} : (i, j) \in \mathbb{T})$ are determined so as to satisfy the equality constraints in (5.3) after fixing the flow amounts on all the arcs in **L**∪**U** as defined above. In the partition ($\mathbb{T}$, **L, U**), arcs in $\mathbb{T}$ are called **basic arcs**, and those in **L**∪**U** are called **nonbasic arcs**. The partition ($\mathbb{T}$, **L, U**) is said to be a **primal feasible partition** if the flow amounts in the corresponding primal basic solution on all the in-tree arcs satisfy the bound constraints in them, **primal infeasible partition** otherwise.

Because of (5.4), the system of equality constraints in (5.3) is redundant; any one of the constraints from them can be eliminated as a redundant constraint. We select the constraint corresponding to node n, say, as the one to eliminate, and fix n as the root node. This has the effect of setting the node price of n, π_n to 0 in the dual problem.

The **dual basic solution** corresponding to a partition ($\mathbb{T}$, **L, U**), is the node price vector $\pi = (\pi_i)$ where

$$\begin{aligned} \pi_n &= 0 \;\; (n \text{ is the root node in } \mathbb{T}) \\ \pi_j - \pi_i &= c_{ij} \text{ for all } (i, j) \in \mathbb{T} \end{aligned} \tag{5.21}$$

The node price π_i in this dual basic solution can be verified to be equal to the cost of the predecessor path of node i, treated as a path from root node to i (see Exercise 5.7).

The optimality conditions for a primal feasible partition ($\mathbb{T}$, **L, U**) to be optimal to the problem are

$$\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i) \begin{cases} \geqq 0 \text{ for all } (i,j) \in \mathbf{L} \\ \leqq 0 \text{ for all } (i,j) \in \mathbf{U} \end{cases} \tag{5.22}$$

$\overline{c}_{ij}$ is called the **reduced** (*or* **relative) cost coefficient** of arc (i,j) wrt the partition ($\mathbb{T}$, $\mathbf{L}$, $\mathbf{U}$). From the fact that $c_{pq} = \pi_q - \pi_p$ for all in-tree arcs (p,q) in $\mathbb{T}$, it can be verified that for any out-of-tree arc $(i,j), \overline{c}_{ij}$ is the cost of its fundamental cycle wrt $\mathbb{T}$, oriented such that (i,j) is a forward arc in it.

The **primal network simplex algorithm** is the specialization of the bounded variable primal simplex algorithm of LP to our problem. It needs an initial primal feasible partition, $(\mathbb{T}_0, \mathbf{L}_0, \mathbf{U}_0)$, say. So, to implement this algorithm we need an efficient scheme to do the following.

1. Find the primal and dual basic solutions associated with the initial partition $(\mathbb{T}_0, \mathbf{L}_0, \mathbf{U}_0)$.

The general step in this algorithm, called a **pivot step**, begins with the primal feasible partition, ($\mathbb{T}$, $\mathbf{L}$, $\mathbf{U}$), say, the associated feasible flow vector, node price vector pair (f, π), obtained at the end of the previous step, and carries out the following work.

2. Identify the set of arcs, $\mathbf{E}$ say, violating the optimality conditions (5.22) for the current partition ($\mathbb{T}$, $\mathbf{L}$, $\mathbf{U}$). $\mathbf{E}$ is called the **set of arcs which are eligible to enter the basic set** in this step. If $\mathbf{E} = \emptyset$, the present partition and the pair (f, π) are optimal; terminate. Otherwise, one of the arcs from $\mathbf{E}$ is selected as the **entering arc** by an **entering arc selection** (*or* **pivot choice) rule**.

There are several entering arc selection rules, and extensive computational studies have been carried out to determine which work best. One which performed very well is the **outward-node most negative evaluator rule**. This rule examines arcs leaving out of the nodes until it encounters the first node containing an arc violating the optimality criterion. It then selects the outward arc at this node which violates the optimality criterion by the largest amount, as the entering arc.

3. If the entering arc selected is (i, j), an effort is made to increase the flow amount on it if it is from **L**, or decrease the flow amount if it is from **U**, while leaving the flow amounts on all the other arcs in **L**$\cup$**U** unchanged. This involves making flow changes on the fundamental cycle, $\mathbb{C}$, of (i, j) wrt $\mathbb{T}$. $\mathbb{C}$ is called the **pivot cycle** in this pivot step. Orient $\mathbb{C}$ such that the entering arc (i, j) is a forward (reverse) arc if (i, j) is in **L** (**U**). With this orientation, it can be verified that the cost of $\mathbb{C}$ is $-|\bar{c}_{ij}|$, hence $\mathbb{C}$ is a negative cost cycle. Find θ, the residual capacity of $\mathbb{C}$; it is known as the **primal simplex minimum ratio** in this pivot step. The set of arcs, **D** on $\mathbb{C}$ which tie in the inequality for defining its residual capacity, is called the **set of arcs eligible to be dropping arcs** in this pivot step.

If $\theta = \infty$ (this can only happen if there are no reverse arcs on $\mathbb{C}$, i.e., it is a negative cost circuit, and all arcs on it are uncapacitated), the objective function is unbounded below in our problem; terminate.

If $\theta = 0$, $\mathbb{C}$ is not a residual cycle, and the pivot step is said to be a **degenerate pivot step**. In this case define $\hat{f} = f$.

If $\theta > 0$ and finite, $\mathbb{C}$ is a negative cost residual cycle, and the pivot step is said to be a **nondegenerate pivot step**. Cancel $\mathbb{C}$ in f and let the new flow vector be $\hat{f}$.

Select a dropping arc (p, q) say, from **D**. If $(p, q) = (i, j)$ (this can only happen if $\theta > 0$; this happens if the flow amount on the entering arc moves from its lower bound to its capacity or vice versa in the flow change just carried out) move (i, j) to the appropriate set in **L, U** based on the new flow amount on it. Let $(\mathbb{T}_1 = \mathbb{T}, \mathbf{L}_1, \mathbf{U}_1)$ be the new partition, and $(\hat{f}, \hat{\pi} = \pi)$ the new pair.

If $(p, q) \neq (i, j)$, let $\mathbb{T}_1$ be the tree obtained by replacing (p, q) in $\mathbb{T}$ by (i, j), delete (i, j) from **L** or **U** where it was before, and insert (p, q) in the appropriate set among these depending on the value of $\hat{f}_{pq}$, and let $(\mathbb{T}_1 = \mathbb{T}, \mathbf{L}_1, \mathbf{U}_1)$ be the new partition.

If $\mathbb{T}_1 \neq \mathbb{T}$, update the node labels and the node price vector.

Then go to the next step.

If a primal feasible partition is not known initially, the primal simplex algorithm described above cannot be applied directly. This is where the **primal simplex method** comes in; one should clearly distinguish it from the primal simplex algorithm. It selects an arbitrary spanning tree $\mathbb{T}$ in G (the algorithms discussed in Section 1.2.2 can be used for this), and by partitioning the out-of-tree arcs into **L, U** arbitrarily, generates an initial partition ($\mathbb{T}$, **L, U**). If ($\mathbb{T}$, **L, U**) is primal feasible, the original problem is solved by initiating the primal simplex algorithm with it. Otherwise, the primal simplex method begins a Phase I which modifies the lower bounds and capacities on arcs in $\mathbb{T}$ so that ($\mathbb{T}$, **L, U**) becomes primal feasible on the modified network. A **Phase *I* objective function**, which measures the extent of infeasibility of the current partition to the original problem, is constructed. The **Phase *I* problem** is that of minimizing the Phase I objective function on the modified network. Since the current partition is primal feasible on the modified network, the Phase I problem can be solved by the primal simplex algorithm initiated with it. When the Phase I problem is solved, we will either conclude that the original problem has no feasible solution, or obtain a primal feasible partition for it. In the later case, we go into a Phase II in which the original problem is solved by the primal simplex algorithm initiated with the primal feasible partition for it obtained at the end of Phase I.

We will now describe how the various operations in the primal simplex method can be carried out efficiently.

To Compute the Primal and Dual Basic Solutions Associated with a Given Partition

Let ($\mathbb{T}$, **L, U**) be a partition in G for (5.3), and let $\hat{f} = (\hat{f}_{ij})$ be the primal basic solution associated with it. We know that $\hat{f}_{ij} = \ell_{ij}$ for $(i, j) \in$ **L**, and $= k_{ij}$ for $(i, j) \in$ **U**. We substitute these values in the system of equality constraints in (5.3) and then obtain $\hat{f}_{ij}$ for $(i, j) \in \mathbb{T}$ by applying a procedure to solve the remaining system by back substitution. This procedure processes each non-root node exactly once. Once a node is processed, the flow amounts in $\hat{f}$ on all the arcs incident at it are known. At any stage, **Y** denotes the set of processed nodes at that stage. The procedure also maintains another set of nodes **X** satisfying the property that for each $i \in$**X**, flow amounts on all the arcs incident at i are already known at this time except on the arc joining it to its immediate predecessor.

The procedure is initiated with **X** = set of non-root terminal nodes in $\mathbb{T}$, $\mathbf{Y} = \emptyset$. In a general step, select a node from **X**, say i, and delete it from **X**. Find the flow amount on the arc joining i and its immediate predecessor from the equation $f(i, \mathcal{N}) - f(\mathcal{N}, i) = V_i$ and the known flow amounts on all the other arcs incident at i. Include i in **Y**. If all the brothers of i are already in **Y**, and the immediate predecessor of i is not the root, include it in **X**. If $\mathbf{X} = \emptyset$, the flow vector has been completely determined; terminate. Otherwise, go to the next step.

The node price vector $\hat{\pi}$ associated with the spanning tree $\mathbb{T}$ is determined from (5.21). These equations are solved by back substitution beginning at the root node, and going down in increasing order of level.

EXAMPLE 5.2

Consider the network in Figure 5.11. The exogenous flow at each node is entered by its side if it is nonzero. $\mathbb{T}$ is the spanning tree of thick arcs. c_{ij} is entered on arc (i, j). The flow amount on each out-of-tree arc, which is either the lower bound or the capacity depending on whether the arc is in **L** or **U**, is entered in a little box by the side of the arc. All the other data is omitted.

To find the primal basic solution corresponding to this partition, we begin with **X** = {1,2,3,4,8,10}, the set of non-root leaf nodes in $\mathbb{T}$, and $\mathbf{Y} = \emptyset$. We select 1 from **X** for processing. There is a demand of 3 units at node 1, 6 units arriving along arc (2, 1), and 5 units leaving through arc (1, 10). For conservation to hold at node 1, the flow amount on in-tree arc (5, 1) must be 2 units. Transfer 1 from **X** to **Y** and proceed. Continuing, we are lead to the following flow amounts on in-tree arcs: $(f_{5,1}, f_{2,5}, f_{3,6}, f_{4,6}, f_{6,7}, f_{12,7}, f_{12,13}, f_{9,5}, f_{9,11}, f_{11,10}, f_{11,8}, f_{13,11}) = (2,$ $16, 8, 5, 13, 8, -5, 2, 3, 7, 1, 6)$.

The node price vector associated with $\mathbb{T}$ is $\pi = (\pi_{13}$, to $\pi_1) = (0,$ $-6, 21, 26, -12, 38, 34, 19, -5, -11, -18, -21, 30)$.

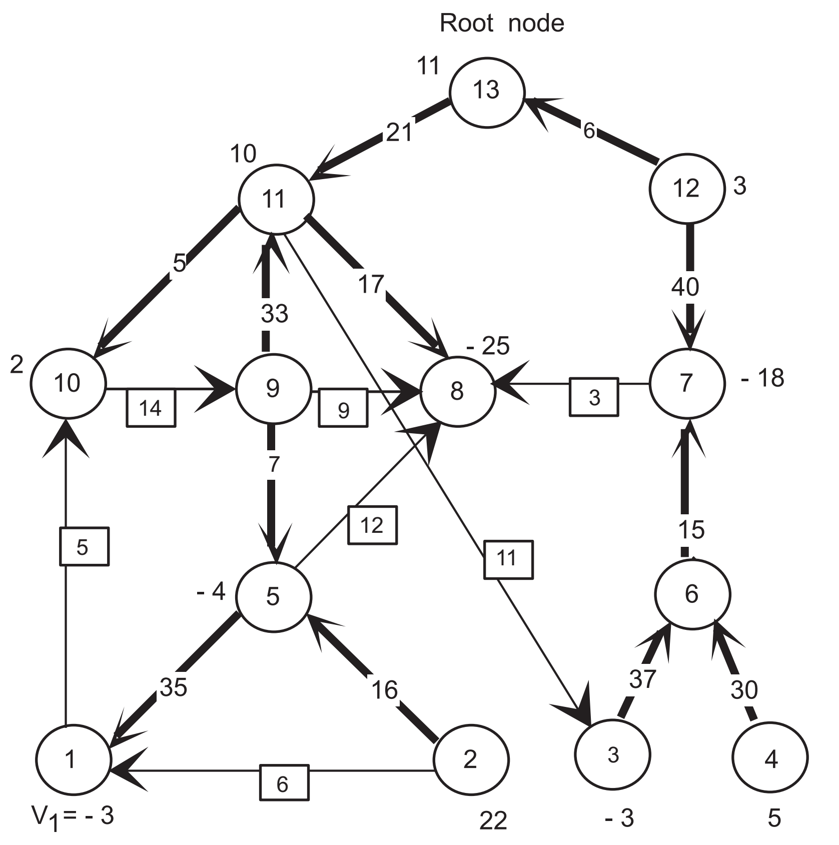


Figure 5.11:

Updating the Tree Labels and the Node Price Vector

Let $\mathbb{T}$ denote the current spanning tree in G associated with the tree labels P(i), S(i), YB(i), EB(i) for $i \in \mathcal{N}$. Let $\mathbb{T}'$ be the new spanning tree obtained after the entering arc replaces the dropping arc. The root node is never changed in this implementation. For $i \in \mathcal{N}$ let P$'(i)$, S$'(i)$, YB$'(i)$, EB$'(i)$ denote the tree labels corresponding to $\mathbb{T}'$. We now define some symbols used in the updating process. See Figure 5.12.

i_1, j_1	j_1 specifically denotes the node on entering arc whose predecessor path contains the dropping arc, and i_1 is the other node on the entering arc.
i_0	apex node, it is the first common node on the predecessor paths of i_1 and j_1.
i_2, j_*	being an in-tree arc, the dropping arc consists of a parent and a son node; these are called i_2, j_* respectively.So, $\mathrm{P}(j_*) = \pm i_2$.
$j_1, j_2, \ldots, j_{t+1} = j_*$	the sequence of nodes on the predecessor path of j_1 up to j_*.

The portion of the predecessor path of j_1 in $\mathbb{T}$ up to the node j_* is known as the **pivot stem** in this pivot step. The updating formulas can be viewed as those arising from a gravity model of the rooted tree. Think of the rooted tree $\mathbb{T}$ standing with the root at the top and successive levels going down. $\mathbb{T}'$ is obtained by introducing the entering arc into $\mathbb{T}$, and then snipping off the dropping arc. When this is done, the points along the path containing the entering arc and the pivot stem fall down by gravity, revolving around each node as the path falls down, giving the new tree $\mathbb{T}'$. In this process if some nodes acquire a new son, we assume that this son joins at the left of the existing sons of that point (i.e., as an elder brother of all the existing sons of that point). It is convenient to do the updating in the order indicated below.

Updating the predecessor indices The predecessor indices change only for nodes on the pivot stem. Set $\mathrm{P}'(j_1) = +i_1$ or $-i_1$ depending on whether the entering arc is (i_1, j_1) or (j_1, i_1). For $u = 2$ to $t + 1$ set $\mathrm{P}'(j_u) =$ (minus the sign of $\mathrm{P}(j_{u-1})$)j_{u-1}.

Updating the successor indices The successor indices change only for nodes on the entering arc, pivot stem and dropping arc. Set $\mathrm{S}'(i_1) = +j_1$ or $-j_1$ depending on whether the entering arc is (j_1, i_1) or (i_1, j_1). For $u = 1$ to t, set $\mathrm{S}'(j_u) =$ (sign of $\mathrm{P}(j_u)$)j_{u+1}. If $\mathrm{S}(i_2) \neq \pm j_*$,

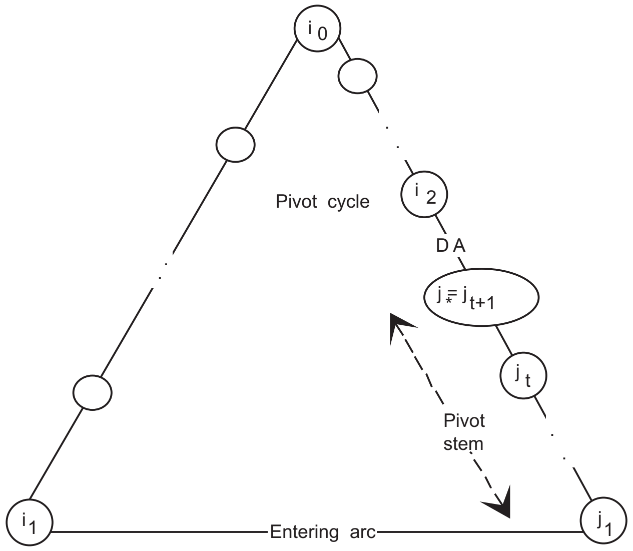


Figure 5.12: Fundamental cycle of the entering arc wrt $\mathbb{T}$. Arc orientations are not shown in the figure. DA is the dropping arc.

make $\mathrm{S}'(i_2) = \mathrm{S}(i_2)$. If $\mathrm{S}(i_2) = \pm j_*$, make $\mathbf{S}'(i_2) =$ (minus the sign of $\mathrm{P}(\mathrm{YB}(j_*)))\mathrm{YB}(j_*)$. If $\mathrm{S}(j_*) \neq \pm j_t$, make $\mathrm{S}'(j_*) = \mathrm{S}(j_*)$. If $\mathrm{S}(j_*) = \pm j_t$, make $\mathbf{S}'(j_*) =$ (minus the sign of $\mathrm{P}(\mathrm{YB}(j_t)))\mathrm{YB}(j_t)$.

Updating the brother indices Let $|\mathrm{S}(i)|$ denote the successor index of i without its sign. We assume that if a point is removed from the set of immediate successors of a node, then the elder-younger brotherly relationships among the remaining points in this set remain unchanged. The brother indices may change only for nodes in the set $\mathbf{H}(\mathbb{T}, j_*)$, the family of j_* in $\mathbb{T}$, and for the nodes $|\mathrm{S}(i_1)|$, $\mathrm{YB}(j_*)$, and $\mathrm{EB}(j_*)$ if these are not empty.

Set $\mathrm{YB}'(j_1) = |\mathrm{S}(i_1)|$. For each $u = 1$ to $t+1$, if $\mathrm{EB}(j_u) \neq \emptyset$, set $\mathrm{YB}'(\mathrm{EB}(j_u)) = \mathrm{YB}(j_u)$; and if $\mathrm{YB}(j_u) \neq \emptyset$, set $\mathrm{EB}'(\mathrm{YB}(j_u)) = \mathrm{EB}(j_u)$. If $\mathrm{S}(i_1) \neq \emptyset$, set $\mathrm{EB}'(|\mathrm{S}(i_1)|) = j_1$. For each $u = 1$ to $t+1$, set $\mathrm{EB}'(j_u) = \emptyset$, because these points join as the eldest to their new set of brothers. If $\mathrm{S}(j_1) \neq \emptyset$, set $\mathrm{EB}'(|\mathrm{S}(j_1)|) = j_2$. For each $u = 2$ to t if $|\mathrm{S}(j_u)| \neq j_{u-1}$, set $\mathrm{EB}'(|\mathrm{S}(j_u)|) = j_{u+1}$; otherwise, if $\mathrm{YB}(j_{u-1}) \neq \emptyset$, set $\mathrm{EB}'(\mathrm{YB}(j_{u-1})) = j_{u+1}$. $\mathrm{YB}'(j_2) = \mathrm{S}(j_1)$. For each $u = 3$ to $t+1$, if $|\mathrm{S}(j_{u-1})| \neq j_{u-2}$, set $\mathrm{YB}'(j_u) = |\mathrm{S}(j_{u-1})|$, and if $|\mathrm{S}(j_{u-1})| = j_{u-2}$, set $\mathrm{YB}'(j_u) = \mathrm{YB}(j_{u-2})$. If $\mathrm{YB}(j_*) \neq \emptyset$, $\mathrm{EB}'(\mathrm{YB}(j_*)) = \mathrm{EB}(j_*)$. If $\mathrm{EB}(j_*) \neq \emptyset$, $\mathrm{YB}'(\mathrm{EB}(j_*)) = \mathrm{YB}(j_*)$). Leave all other brother indices unchanged.

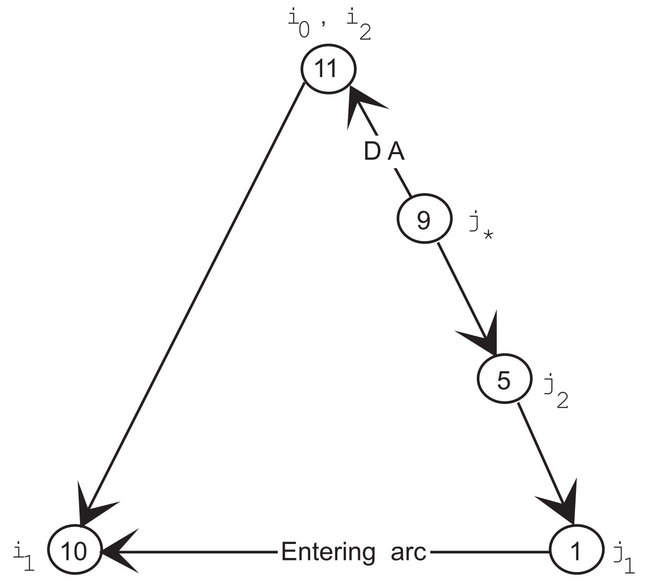


Figure 5.13: DA is the dropping arc.

As an example, consider the spanning tree $\mathbb{T}$ in Figure 1.14, Section 1.2.2. Let (1, 10) be the entering arc into $\mathbb{T}$, and let the in-tree arc (9, 11) be the dropping arc. The pivot cycle for this pivot step is in Figure 5.13, the updated spanning tree $\mathbb{T}'$ is in Figure 5.14, and the updated node labels are shown below.

Updated Tree Labels Corresponding to $\mathbb{T}'$

Node i	1	2	3	4	5	6	7	8	9	10	11	12	13
													Root
P(i)	-10	-5	-6	-6	-1	-7	+12	+11	-5	+11	+13	-13	$\emptyset$
S(i)	+5	$\emptyset$	$\emptyset$	$\emptyset$	+9	+3	+6	$\emptyset$	$\emptyset$	+1	-10	-7	-11
YB(i)	$\emptyset$	$\emptyset$	4	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	2	8	12	$\emptyset$	$\emptyset$
EB(i)	$\emptyset$	9	$\emptyset$	3	$\emptyset$	$\emptyset$	$\emptyset$	10	$\emptyset$	$\emptyset$	$\emptyset$	11	$\emptyset$

If other node labels such as the distance label, thread label, number of successors label, preorder distance label, last successor label, etc. are used in the implementation, they are updated in a manner similar to the above. The updating of the thread label is easy, as it changes only for the nodes on the pivot stem and their eldest and youngest children. See Glover and Klingman [1982], and Glover, Klingman, and Stutz [1974].

Updating the node price vector Let $\pi = (\pi_i), \pi' = (\pi'_i)$ be the node price vectors corresponding to the spanning tree $\mathbb{T}$, the updated spanning tree $\mathbb{T}'$ respectively. $\bar{c}_{ij} = c_{ij} - (\pi_j - \pi_i)$ are the relative cost coefficients wrt π. $\mathbf{H}(\mathbb{T}, j_*)$ is the family of j_* in $\mathbb{T}$. Then

$$\pi'_i = \begin{cases} \pi_i & \text{for } i \notin \mathbf{H}(\mathbb{T}, j_*) \\ \pi_i + \alpha\bar{c}_e & \text{for all } i \in \mathbf{H}(\mathbb{T}, j_*) \end{cases} \tag{5.23}$$

where $\bar{c}_e$ is the relative cost coefficient of the entering arc wrt π, and $\alpha = +1$ if the entering arc is (i_1, j_1), or -1 if it is (j_1, i_1). It can be verified that π' defined by (5.23) satisfies (5.21) for the tree $\mathbb{T}'$; hence it is the node price vector corresponding to $\mathbb{T}'$

As an example, consider the spanning tree $\mathbb{T}$ consisting of the thick arcs in Figure 5.11. The node price vector π corresponding to $\mathbb{T}$ in this network has been obtained in Example 5.2. Let $\mathbb{T}'$ obtained by replacing the dropping arc (9, 11) by the entering arc (1, 10). Suppose the cost coefficient $c_{1,10} = 5$. wrt π, the relative cost coefficient $\bar{c}_{1,10} = 5 - (26 - 30) = 9$. $j_* = 9$ in this example, and the family $\mathbf{H}(\mathbb{T}, 9) = \{ 9, 5, 1, 2 \}$. The new node price vector π' wrt $\mathbb{T}'$ is $(\pi'_{13}$ to $\pi'_1) = (0, -6, 21, 26, -21, 38, 34, 19, -14, -11, -18, -30, 21)$.

Updating the node price vector when one cost coefficient changes Let π be the node price vector corresponding to the spanning tree $\mathbb{T}$ in G. Suppose the cost coefficient c_{pq} of one in-tree arc

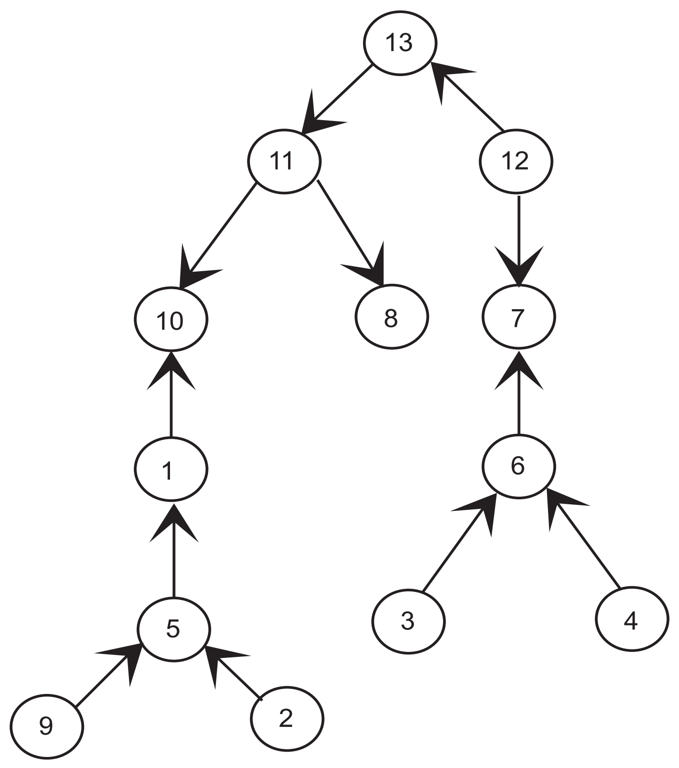


Figure 5.14: The updated spanning tree $\mathbb{T}'$.

(p, q) is changed to $c_{pq} + \delta$ while the tree itself and other data remains unchanged. This kind of change occurs during Phase I of the simplex method. Let $\mathbf{H}$ denote the family of son(p, q) in $\mathbb{T}$. $\hat{\pi} = (\hat{\pi}_i)$, the new node price vector is given by: $\hat{\pi}_i = \pi_i$ for all $i \notin \mathbf{H}$, and $= \pi_i + \alpha\delta$ for $i \in \mathbf{H}$; where $\alpha = +1$ if son$(p, q) = q$, and $= -1$ if son$(p, q) = p$.

Setting up the Phase I problem Let $(\mathbb{T}_0, \mathbf{L}_0, \mathbf{U}_0)$ be an initial partition selected in G, associated with the primal basic solution $f^0 = (f^0_{ij})$. If f^0 is primal infeasible, we set up a Phase I problem whose aim is to find a primal feasible partition in G. Define

$$\begin{aligned} \mathbf{K}_1 &= \{(i, j) : (i, j) \in \mathbb{T}_0, f^0_{ij} < \ell_{ij}\} \\ \mathbf{K}_2 &= \{(i, j) : (i, j) \in \mathbb{T}_0, f^0_{ij} > k_{ij}\} \end{aligned}$$

Arcs in $\mathbf{K}_1, \mathbf{K}_2$ are called **type 1, 2 arcs** respectively. The Phase I problem is defined on the same network $(\mathcal{N}, \mathcal{A})$ but has modified data: lower bound vector $\ell' = (\ell'_{ij})$, capacity vector $k' = (k'_{ij})$, cost vector $c^* = (c^*_{ij})$, where

$$\begin{array}{llll} \text{for } (i,j) \notin \mathbf{K}_1 \cup \mathbf{K}_2, & \ell'_{ij} = \ell_{ij}, & k'_{ij} = k_{ij}, & c^*_{ij} = 0 \\ \text{for } (i,j) \in \mathbf{K}_1, & \ell'_{ij} = f^0_{ij}, & k'_{ij} = \ell_{ij}, & c^*_{ij} = -1 \\ \text{for } (i,j) \in \mathbf{K}_2, & \ell'_{ij} = k_{ij}, & k'_{ij} = f^0_{ij}, & c^*_{ij} = +1 \end{array}$$

For the Phase I problem $(\mathcal{N}, \mathcal{A}, \ell', k', c^*, V)$, $(\mathbb{T}_0, \mathbf{L}_0, \mathbf{U}_0)$ is a feasible partition associated with the primal feasible basic solution f^0. Starting with this partition we minimize the Phase I objective function $\sum(c^*_{ij} f_{ij}$: over $(i,j) \in \mathcal{A})$ in Phase I, by the primal network simplex algorithm. From the definition of c^*, this has the effect of increasing the flow amounts on type 1 arcs, and decreasing the flow amounts on type 2 arcs, thereby bringing the flow vector closer to feasibility for the original problem. The type 1, 2 arcs play the role of artificial variables in Phase I problems for general LPs.

Whenever the flow vector changes in Phase I, the data and the sets $\mathbf{K}_1, \mathbf{K}_2$ are revised. Suppose $\hat{f} = (\hat{f}_{ij})$ is a new flow vector obtained during Phase I. For each arc (i,j) which was a type 1 arc before $\hat{f}$ was obtained, cancel its type 1 status and make it a regular arc and change $c^*_{ij}, \ell'_{ij}, k'_{ij}$ to 0, ℓ_{ij}, k_{ij} respectively if $\hat{f}_{ij} = \ell_{ij}$; otherwise just change ℓ'_{ij} to $\hat{f}_{ij}$. Make a corresponding alteration for each arc which was a type 2 arc before $\hat{f}$ was obtained. Hence, the data changes in every nondegenerate step in Phase I.

At some stage during Phase I let f be the present flow vector, and π^* the node price vector computed using the current c^* as the cost vector. If there are no type 1, 2 arcs, f is feasible to the original problem; move to Phase II with the present partition. Even if there are type 1, 2 arcs, Phase I terminates if for each $(i,j) \in \mathcal{A}$

$$\begin{array}{llll} f_{ij} = & \text{current } \ell'_{ij} & \text{if} & \overline{c}^*_{ij} > 0 \\ f_{ij} = & \text{current } k'_{ij} & \text{if} & \overline{c}^*_{ij} < 0 \end{array}$$

where $\overline{c}^*_{ij} = c^*_{ij} - (\pi^*_j - \pi^*_i)$. These are the optimality conditions for the Phase I problem. If these conditions hold, and $\mathbf{K}_1 \cup \mathbf{K}_2 \neq \emptyset$, the original problem is infeasible; terminate.

Resolution of Cycling in the Primal Network Simplex Algorithm

With respect to a feasible flow vector $\bar{f} = (\bar{f}_{ij})$ in G for (5.3), an arc $(i, j) \in \mathcal{A}$ is said to be an

$$\begin{array}{rll} \text{interior arc} & \text{if} & \ell_{ij} < \bar{f}_{ij} < k_{ij} \\ \text{lower boundary arc} & \text{if} & \ell_{ij} = \bar{f}_{ij} \\ \text{upper boundary or saturated arc} & \text{if} & \qquad \bar{f}_{ij} = k_{ij} \end{array}$$

A primal feasible partition $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ and the associated BFS f are said to be **primal nondegenerate** if all the arcs in $\mathbb{T}$ are interior arcs wrt f, **primal degenerate** if at least one arc in $\mathbb{T}$ is a lower boundary or saturated arc wrt f.

It can be verified that a feasible flow vector $\bar{f}$ for (5.3) is a BFS iff the set of interior arcs wrt $\bar{f}$ forms a forest, and a primal nondegenerate BFS iff this set forms a spanning tree in G.

The residual capacity, θ, of the pivot cycle in a pivot step of the primal simplex algorithm can be verified to be strictly > 0 if the flow vector, f at that stage is primal nondegenerate. If f is degenerate, θ could be 0, and in this case the pivot step becomes a **degenerate pivot step**. During a nondegenerate pivot step the BFS changes and the objective value strictly decreases. During a degenerate pivot step, the objective value and the BFS do not change, but the spanning tree and the node price vector change. Since the objective value is monotone nonincreasing in this algorithm, if a nondegenerate pivot step occurs in a partition, that partition can never reappear in the sequel. However, the algorithm may go through a sequence of consecutive degenerate pivot steps without any change in the objective value or the flow vector, and return again to the partition at the beginning of this sequence, thus creating a **cycle of degenerate pivot steps**. The algorithm can repeat this cycle indefinitely, and never terminate. This phenomenon is called **cycling under degeneracy in the primal network simplex**

algorithm. It is an indefinite repetition of the same finite cycle of degenerate pivot steps again and again, without ever satisfying a termination condition. An example of cycling in an assignment problem of order 4 has been constructed by L. Johnson and is reported in the paper Gassner [1964]. The cost matrix for the problem is

$$c = \begin{pmatrix} 3 & 5 & 5 & 11 \\ 9 & 7 & 9 & 15 \\ 7 & 7 & 11 & 13 \\ 13 & 13 & 13 & 17 \end{pmatrix}$$

It is a minimum cost flow problem on a complete bipartite network of order 4×4, in which all the arcs are uncapacitated. So, a partition for this problem is of the form ($\mathbb{T}$, **L**), basic and nonbasic arcs, with the flow amounts on all nonbasic arcs being 0 in the associated basic solution. We denote by the ordered pair (i, j) the arc joining row node i to column node j. A cycle of degenerate pivot steps for the primal network simplex algorithm in this problem is given below. It consists of 12 pivot steps at the end of which we get the initial partition back. In the following table, we give the basic set of arcs in each pivot step.

Pivot step	Basic set of arcs	Entering arc	Dropping arc
1	{(1,1), (2,2), (3,3), (4,4), (1,2), (2,3), (3,4)}	(1,3)	(2,3)
2	{(1,1), (2,2), (3,3), (4,4), (1,2), (1,3), (3,4)}	(4,2)	(1,2)
3	{(1,1), (2,2), (3,3), (4,4), (4,2), (1,3), (3,4) }	(3,2)	(3,4)
4	{(1,1), (2,2), (3,3), (4,4), (4,2), (1,3), (3,2) }	(4,1)	(4,2)
5	{(1,1), (2,2), (3,3), (4,4), (4,1), (1,3), (3,2) }	(4,3)	(1,3)
6	{(1,1), (2,2), (3,3), (4,4), (4,1), (4,3), (3,2) }	(2,1)	(4,1)
7	{(1,1), (2,2), (3,3), (4,4), (2,1), (4,3), (3,2) }	(3,1)	(3,2)
8	{(1,1), (2,2), (3,3), (4,4), (2,1), (4,3), (3,1) }	(2,4)	(2,1)
9	{(1,1), (2,2), (3,3), (4,4), (2,4), (4,3), (3,1) }	(2,3)	(4,3)
10	{(1,1), (2,2), (3,3), (4,4), (2,4), (2,3), (3,1) }	(1,4)	(2,4)
11	{(1,1), (2,2), (3,3), (4,4), (1,4), (2,3), (3,1) }	(3,4)	(3,1)
12	{(1,1), (2,2), (3,3), (4,4), (1,4), (2,3), (3,4) }	(1,2)	(1,4)
13	{(1,1), (2,2), (3,3), (4,4), (1,2), (2,3), (3,4) }		

It can be verified that all these pivot steps are degenerate pivot steps, and that every one of these basic sets is associated with the same BFS in which f_{ii} is 1 for all $i = 1$ to 4, and flows on all the other arcs are 0. The relative cost coefficient of the entering cell in each pivot step is - 2.

The occurrence of cycling is extremely rare in real-world applications. But, since the possibility of cycling exists, we cannot mathematically guarantee that the primal network simplex algorithm is finite unless methods for resolving it are used. One such method which is purely combinatorial, has been developed originally by Cunningham [1976, 1979]; we will show later that it is exactly the specialization of the lexicographic bounded variable primal simplex algorithm for general LP discussed in Section 11.4 of Murty [1983 of Chapter 1], to the minimum cost flow problem.

Let ($\mathbb{T}$, **L, U**) be a primal feasible partition in G associated with the BFS $f = (f_{ij})$. This partition is said to be a **strongly feasible partition** if for each $(i, j) \in \mathbb{T}$

$$\begin{aligned} f_{ij} = \ell_{ij} \quad &\text{implies} \quad (i,j) \text{ is directed away from root, i.e., } \mathrm{P}(j) = +i \\ f_{ij} = k_{ij} \quad &\text{implies} \quad (i,j) \text{ is directed towards the root,i.e., } \mathrm{P}(i) = -j \end{aligned} \tag{5.24}$$

i.e., a strongly feasible partition is one in which all lower boundary in-tree arcs are directed away from the root node, and all upper boundary in-tree arcs are directed towards the root node. The method for resolving cycling initiates the primal algorithm with a strongly feasible partition and maintains strong feasibility throughout by a special dropping arc selection rule in each pivot step. We will summarize the main features of this method. The details and proofs are mainly of theoretical interest; the interested reader should consult Cunningham [1976, 1979].

Cunningham [1976] has given an efficient procedure for obtaining a strongly feasible partition from an arbitrary feasible partition. Let β be the number of bad arcs, i.e.,boundary in-tree arcs which are wrongly oriented for strong feasibility, in the initial partition. Then the procedure goes through β stages. Each stage tries to replace a bad arc with a nonbasic arc. Each stage may require up to $(n-1)$ such pivot steps and leads to a reduction in the number of bad arcs by 1.

Once the primal algorithm is initiated with a strongly feasible partition, strong feasibility can be preserved by adopting the following dropping arc choice rule in each pivot step (there is no restriction on the entering arc choice).

Dropping Arc Choice Rule to Preserve Strong Feasibility

Let (p,q) be the entering arc in the pivot step in the strongly feasible partition ($\mathbb{T}$, **L, U**) associated with the pair (f,π), and $\mathbb{C}$ the pivot cycle in this pivot step with g as the apex node. Let **D** denote the set of eligible dropping arcs in this pivot step.

1. If $\bar{c}_{pq} = c_{pq} - (\pi_q - \pi_p) > 0$ and $f_{pq} = k_{pq}$, select the dropping arc in this pivot step to be the first arc encountered in **D** while traveling along $\mathbb{C}$ from g back to g in the direction opposite to the orientation of (p,q).

2. If $\bar{c}_{pq} < 0$ and $f_{pq} = \ell_{pq}$, select the dropping arc in this pivot step to be the first arc encountered in **D** while traveling along $\mathbb{C}$ from g back to g in the same direction as (p,q).

This rule identifies the dropping arc uniquely by unambiguous and simple combinatorial rules. If $(\mathbb{T}', \mathbf{L}', \mathbf{U}')$ is the partition obtained after this pivot step operated using this dropping arc choice rule, it will also be strongly feasible. Also, let (f', π') be the basic pair associated with this partition. If this pivot step is a degenerate pivot step (i.e., $f' = f$), then $\pi' \leqq \pi$ and $\sum_{i \in \mathcal{N}} \pi'_i < \sum_{i \in \mathcal{N}} \pi_i$ (these inequalities relating π and π' may not hold if the pivot step is nondegenerate). See Cunningham [1976] for proofs of these results.

The primal network simplex algorithm initiated with a strongly feasible partition, and operated using the above dropping arc choice rule, is called **the method of strongly feasible partitions**. In this method, the entering arc can be chosen arbitrarily among those eligible to enter in each pivot step. In each nondegenerate pivot step of this method, the primal objective value decreases strictly. In each degenerate pivot step, the primal objective value remains unchanged, but $\sum_{i \in \mathcal{N}} \pi_i$ strictly decreases. So, a partition can never reappear in this method, cycling cannot occur, and the method terminates finitely.

Like some of the other methods for resolving cycling in the primal simplex algorithm for general LP, the method of strongly feasible partitions also can be given a perturbation interpretation. Let ϵ be a small positive number, and consider the perturbed problem obtained by changing V_i in (5.3) into V_i^1 for all $i \in \mathcal{N}$, where $V_i^1 = V_i + \epsilon$ for $i \neq n$, and $= V_n - (n-1)\epsilon$ for $i = n$. Then it can be shown that when ϵ is positive but sufficiently small, a partition in G is feasible to the perturbed problem iff it is strongly feasible to the original problem. So, the method of strongly feasible partitions can be viewed as the usual primal network simplex algorithm applied to solve the perturbed problem, treating ϵ as a sufficiently small positive number without giving it a specific value.

The method of strongly feasible partitions can also be shown to be a specialization of the lexicographic bounded variable primal simplex algorithm for general LP (see Section 11.4 in Murty [1983 of Chapter 1]). Consider the bounded variable LP (1.7) discussed in Section 1.2.1. A feasible partition (x_B, x_L, x_U), where $x_B = (x_{i_1}, \dots, x_{i_m})$ say, associated with the basis B and the BFS $\overline{x} = (\overline{x}_j)$ for (1.7), is said to be a *strongly feasible partition* if for each r

$$\begin{aligned} \overline{x}_{i_r} = \ell_{i_r} \quad &\text{implies} \quad \text{the } r\text{th row of } B^{-1} \text{ is lexico positive} \\ \overline{x}_{i_r} = k_{i_r} \quad &\text{implies} \quad \text{the } r\text{th row of } B^{-1} \text{ is lexico negative} \end{aligned} \tag{5.25}$$

Our minimum cost flow problem (5.3) will be in the form (1.7) if we eliminate the flow conservation equation corresponding to the root node n. Let B be the basis associated with a spanning tree $\mathbb{T}$ for the resulting problem. From Theorem 1.8 we know that all nonzero entries in any row of B^{-1} always have the same sign. From this we conclude that the row corresponding to $(p, q) \in \mathbb{T}$ in B^{-1} is lexico positive iff (p, q) is directed away from the root node in $\mathbb{T}$, and lexico negative iff (p, q) is directed towards the root node. Hence for a feasible partition for (5.3), the definition of strong feasibility using (5.24) is identical to that using (5.25).

Also, the dropping arc choice rule in the method of strongly feasible partitions turns out to be exactly the same as the dropping variable choice rule in the lexicographic bounded variable primal simplex algorithm. Hence, the method of strongly feasible partitions is exactly the specialization of the lexicographic bounded variable primal simplex algorithm to our minimum cost flow problem. This specialization is made possible by the result in Theorem 1.8.

Two disadvantages of the method of strongly feasible partitions are that it needs an initial strongly feasible partition, and that the special dropping arc choice rule has to be used in every pivot step, even during sequences of nondegenerate pivot steps, since otherwise strong feasibility will be lost. Another method for resolving cycling in the primal network simplex algorithm developed by Gamble, Conn, and Pulleyblank [1988] is much simpler computationally. It can be initiated with any primal feasible partition, without the need to convert it into a strongly feasible one. The core of this method is also a special dropping arc choice rule. But the special rule needs to be invoked only when a degenerate pivot step occurs, and can be turned off when the next nondegenerate pivot step occurs. Because of this, finiteness is guaranteed in this method even though the anti-cycling mechanism is only maintained during degenerate pivot steps. See their paper for details.

Simultaneous Resolution of Cycling and Stalling in the Primal Network Simplex Algorithm

Cycling is the infinite repetition of the same cycle of degenerate pivot steps in the primal simplex algorithm. The method of strongly feasible partitions resolves cycling and converts the primal simplex algorithm into a finite algorithm. But the method of strongly feasible partitions could encounter another computationally expensive phenomenon called **stalling**, which is a finite but very long sequence of consecutive degenerate pivot steps, in which the number of steps is not bounded above by any polynomial in $|\mathcal{N}|$ or the size of the problem. An example of stalling has been constructed by Cunningham [1979]

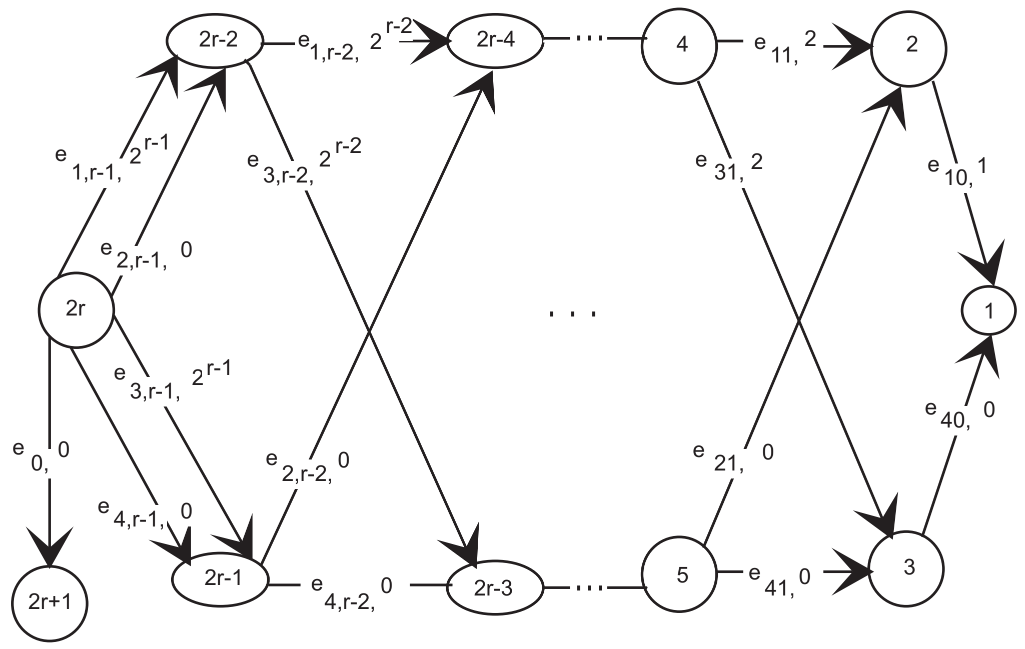


Figure 5.15: Network for the rth problem in the class for $r \geqq 2$. All lower bounds are 0, and capacities are $+\infty$. Source = node $2r$, with 1 unit available, sink = node $2r + 1$ with requirement 1 unit. Cost coefficients are entered on the arcs.

using a class of problems developed by J. Edmonds to show that the primal simplex algorithm for the shortest chain problem is an exponential growth algorithm in the worst case. This class has a minimum cost flow problem for each $r \geqq 2$, the rth problem being on a directed network with $2r + 1$ nodes and $4r - 1$ arcs. See Figure 5.15. Arcs are grouped into 5 groups, the bth arc in the ath group being denoted by $e_{a,b}$. For example, $a = 1$ corresponds to the group of arcs at the top of the network in Figure 5.15, etc. All lower bounds are 0, and all capacities are ∞. It is required to ship 1 unit of material from the source node, $2r$, to the sink node, $2r + 1$, at minimum cost. Since it is an uncapacitated minimum cost flow problem with a specified source and sink node, it is a shortest chain problem. Arc cost coefficients are entered on the arcs.

The only feasible flow vector for the rth problem is $\tilde{f}$ with a flow of 1 on the arc e_0, and 0 flow on all the other arcs. Hence $\tilde{f}$ is the optimum solution to the problem.

Fix the root at the source node $2r$. Since all the capacities are ∞, partitions for this problem are of the form $(\mathbb{T}, \mathbf{L})$, where $\mathbb{T}$ is a spanning tree, and every such partition corresponds to the flow vector $\tilde{f}$ and is therefore feasible. e_0 is contained in every spanning tree, and it is an interior arc in every partition since the flow on it is 1, and all the other arcs are lower boundary arcs.

Define the partition $(\mathbb{T}_0, \mathbf{L}_0)$ where $\mathbb{T}_0 = \{e_0, e_{1,0}$; and $e_{1,b}, e_{3,b}$ for $b = 1$ to $r-1$ $\}$, $\mathbf{L}_0 = \{e_{4,0}$; and $e_{2,b}, e_{4,b}$ for $b = 1$ to $r-1\}$. All arcs in $\mathbb{T}_0$ other than e_0 are lower boundary arcs directed away from the root node; hence $(\mathbb{T}_0, \mathbf{L}_0)$ is a strongly feasible partition. Consider the following order for the arcs in the network in the rth problem.

$$e_{1,0}, e_{4,0}, e_{1,b}, e_{2,b}, e_{3,b}, e_{4,b} \text{ for } b = 1 \text{ to } r-1, e_0 \qquad (5.26)$$

If the rth problem in the class is solved by the method of strongly feasible partitions initiated with the partition $(\mathbb{T}_0, \mathbf{L}_0)$ and executed using the entering arc choice rule: always choose as the entering arc the first arc in the order listed in (5.26) which is eligible to enter; then it goes through a sequence of $3(2^{\mathbf{r}} - 1) - 2r$ consecutive degenerate pivot steps before terminating. See Cunningham [1979] for a proof. This is stalling.

Cunningham [1979] has shown that a simple entering arc selection strategy will resolve stalling in the method of strongly feasible partitions for the class of minimum cost flow problems in pure networks. The requirement is that the entering arc selection strategy examine every arc periodically (say once in every γm pivot steps for some γ) and select it as the entering arc if it is eligible at that time. One such rule is **LRC (*least recently considered*)** entering arc choice rule. This rule arranges all the arcs in the network in a fixed order, say $e_1, \ldots, e_m$, at the beginning of the algorithm. The entering arc in a pivot step is the first eligible arc in the list $e_{t+1}, e_{t+2}, \ldots, e_m, e_1, \ldots, e_t$, where e_t was the entering arc in the previous step. So, this rule circles through the list of arcs beginning with the entering arc in the previous

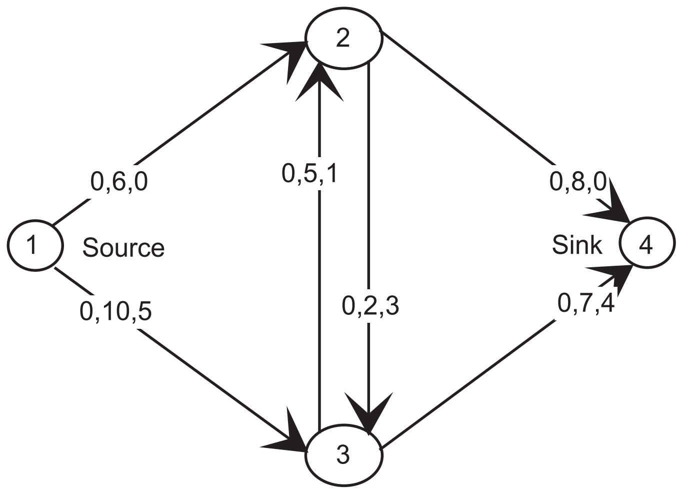


Figure 5.16:

step, looking for the entering arc. Hence each arc is examined once in every m pivot steps and given a chance to become the entering arc. Under this rule in the method of strongly feasible partitions, it has been shown that the total number of consecutive degenerate pivot steps can never exceed nm, and hence stalling cannot occur. Several other entering arc choice rules to prevent stalling in the method of strongly feasible partitions are discussed in Cunningham [1979].

The related problem in general LP: are there any entering and dropping variable choice rules for the primal simplex algorithm to solve an LP which resolve both cycling and stalling, still remains an open question at this time. It is the most important open question in LP theory. It can be shown that any such scheme consisting of entering and dropping variable choice rules which resolve both cycling and stalling in the primal simplex algorithm for LP at a degenerate BFS, would provide a polynomially bounded pivotal algorithm for checking the feasibility of a system of linear inequalities, and conversely. Hence the question of resolving both cycling and stalling in general LP is intimately related to the more fundamental question: is there a polynomially bounded primal simplex algorithm for solving an LP with rational data?

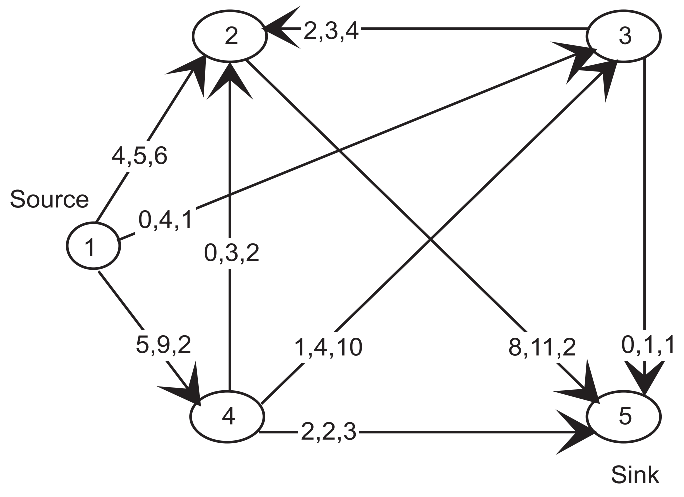


Figure 5.17:

Exercises

5.5 Let $\hat{\pi} = (\hat{\pi}_i)$ be the node price vector associated with the rooted spanning tree $\mathbb{T}$ rooted at node n in G, obtained by solving (5.21). For each $i \in \mathcal{N}$ prove that $-\hat{\pi}_i$ is the cost of the predecessor path of i in $\mathbb{T}$, as a path from i to n.

5.6 Find a minimum cost flow vector of value 5 in the network in Figure 5.16 by the OK algorithm. Draw the curve depicting the optimum objective value as a function of the value.

Find a minimum cost maximum value feasible flow vector in the network in Figure 5.17 by the OK algorithm. In both networks, the data on the arcs is lower bound, capacity, cost coefficient, in that order.

5.7 Let $\hat{f} = (\hat{f}_{ij})$ be the basic solution of (5.3) corresponding to the partition ($\mathbb{T}$, **L, U**). Let $(p, q) \in \mathbb{T}$ and $t = \text{son}(p, q)$. Define $\omega = +1$ if $t = q$, -1 if $t = p$. For any $i \in \mathcal{N}$ let $\overline{\mathbf{H}}(\mathbb{T}, i)$ denote the complement of $\mathbf{H}(\mathbb{T}, i)$. Prove that

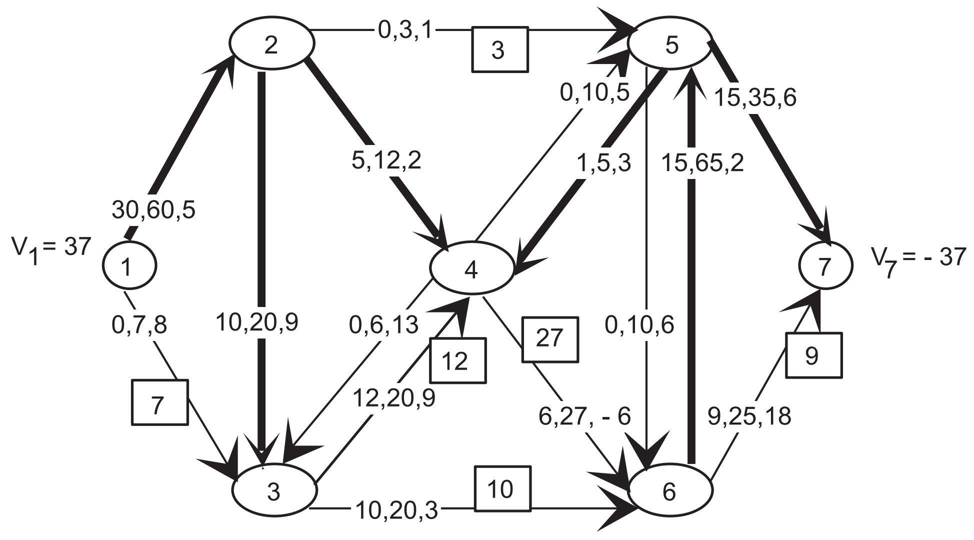


Figure 5.18:

$$\hat{f}_{pq} = \omega\Big(\sum_{j \in \overline{\mathbf{H}}(\mathbb{T},t)} V_j - \sum(\hat{f}_{ij} : \text{ over } (i,j) \in (\mathcal{A}\backslash\mathbb{T}) \cap (\overline{\mathbf{H}}(\mathbb{T},t), \mathbf{H}(\mathbb{T},t))) + \sum(\hat{f}_{ij} : \text{ over } (i,j) \in (\mathcal{A}\backslash\mathbb{T}) \cap (\mathbf{H}(\mathbb{T},t), \overline{\mathbf{H}}(\mathbb{T},t)))\Big)$$

5.8 Suppose (p,q) is the entering arc in a pivot step during Phase II of the primal network simplex method for solving (5.3). Prove that the relative cost coefficient $\overline{c}_{pq}$ at that time is the cost of the fundamental cycle of (p,q) wrt the spanning tree at that time.

5.9 In the network in Figure 5.18, $\ell_{ij}, k_{ij}, c_{ij}$ are marked on the arcs in that order. V_i is entered by the side of node i if it is nonzero. A spanning tree $\mathbb{T}$ is marked in thick arcs, and flow amounts chosen on out-of-tree arcs are entered in little squares on them if they are nonzero. Solve this problem by the primal network simplex method beginning with the partition provided by this information.

5.10 Consider the bipartite network representation of the assignment problem of order n on the bipartite network G $= (\mathcal{N}_1 = \{R_1, \ldots, R_n$

}, $\mathcal{N}_2 = \{ C_1, \ldots, C_n \}$, $\mathcal{A} = \{(R_i, C_j) : i, j = 1 \text{ to } n \})$. The arc (R_i, C_j) in $\mathcal{A}$ is directed from R_i to C_j for all i, j, and has lower bound 0 and capacity 1. Each node in $\mathcal{N}_1$ is a source with 1 unit of material available for shipment, and each node in $\mathcal{N}_2$ is a sink with 1 unit of material required. Let $\mathbb{T}$ be a spanning tree in G with column signature vector (2,…, 2,1) with only node C_n having degree 1. Treat $\mathbb{T}$ as a rooted tree with C_n as the root node. Define $\mathbf{L} = \mathcal{A} \backslash \mathbb{T}$ and $\mathbf{U} = \emptyset$.

In the basic solution associated with the partition ($\mathbb{T}$, **L**, **U**), prove that the flow on an in-tree arc is 1 if the arc is directed towards the root node, and 0 if it is directed away from the root node. Using this prove that ($\mathbb{T}$, **L**, **U**) is a highly degenerate strongly feasible partition for this problem. (Akgul and Ekin [1990 of Chapter 3]).

5.5 The Shortest Augmenting Path Method

This method is based on successive augmentations along cheapest FAPs. It solves the minimum cost flow problem by repeated application of a shortest chain algorithm in the residual network, each time saturating the FAP in the original network corresponding to the shortest chain obtained in the residual network at that stage. Thus the basic approach behind the algorithm is an **incremental approach**, since the flow value is incremented in each iteration by the least costly increment available at that stage. Hence, the algorithm has also been called a **buildup algorithm**, as it builds up the flow value at minimum cost thereby obtaining a sequence of flows of increasing value, each of minimum cost for its value.

This algorithm is essentially the same as the parametric value minimum cost flow algorithm discussed earlier. There it was based on the OK method; here we will discuss the version based on Dijkstra's shortest chain algorithm.

We consider the minimum cost flow problem in the form (5.1) in the directed network G = $(\mathcal{N}, \mathcal{A}, \ell = 0, k, c, \breve{s}, \breve{t}.\overline{v})$, where $|\mathcal{N}| = n$, $k > 0$, and $\overline{v}$ is the value required to be shipped from $\breve{s}$ to $\breve{t}$ at minimum cost. The method obtains a sequence of flow vector, node price vector pairs (f^r, π^r), $r = 0, 1, \ldots$, where π^r will always be dual feasible, and f^r always satisfies the lower bound and capacity conditions on all the arcs, and flow conservation at all the nodes, and its value v^r strictly increases in each step. When v^r reaches $\overline{v}$, primal feasibility is attained and the method terminates. Hence, the method is a dual method.

In each step the method solves a shortest chain problem in the residual network wrt the pair at that time. Since all the cost coefficients in it are $\geqq 0$, the shortest chain problem can be solved by Dijkstra's method with a computational effort of at most $O(n^{\mathbf{2}})$. To initiate the method, we need an optimum pair of zero value. For this reason we consider the case where either $c \geqq 0$, or a node price vector $\pi = (\pi_i)$ is known satisfying $\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i) \geqq 0$ for all $(i, j) \in \mathcal{A}$. In the latter case we replace c by $\overline{c} = (\overline{c}_{ij})$; this leads to an equivalent problem. Hence without any loss of generality we consider the case $c \geqq 0$.

Given an optimum pair (f, π) of value v, let $(\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i))$ be the reduced cost vector wrt π. By optimality, we know that the following c.s. conditions hold for each $(i, j) \in \mathcal{A}$

$$\begin{array}{rcl} 0 < f_{ij} < k_{ij} & \text{implies} & \overline{c}_{ij} = 0 \\ \overline{c}_{ij} < 0 & \text{implies} & f_{ij} = k_{ij} < \infty \\ \overline{c}_{ij} > 0 & \text{implies} & f_{ij} = 0 \end{array} \tag{5.27}$$

By these optimality conditions, the cost vector $\overline{c}'$ in the residual network $G(f, \pi) = (\mathcal{N}, \mathcal{A}(f), 0, \kappa, \overline{c}')$ is $\geqq 0$.

THE SHORTEST AUGMENTING PATH ALGORITHM

Initialization Begin with the optimum pair $(f^0 = 0, \pi^0 = 0)$ of value 0. This is optimal because $c \geqq 0$.

General step Let (f^r, π^r) be the present pair of value v^r. Find a shortest chain tree rooted at $\breve{s}$ in $G(f^r, \pi^r)$ using Dijkstra's algorithm. Terminate Dijkstra's algorithm as soon as $\breve{t}$ is permanently labeled.

If there exists no chain from $\breve{s}$ to $\breve{t}$ in $\mathrm{G}(f^r, \pi^r)$, there exists no FAP from $\breve{s}$ to $\breve{t}$ in G wrt f^r, and since $v^r < \overline{v}$, there is no feasible flow of value $\overline{v}$ in G; terminate.

If a shortest chain, $\mathcal{C}_r$, from $\breve{s}$ to $\breve{t}$ in $\mathrm{G}(f^r, \pi^r)$ has been found, define for $i \in \mathcal{N}$, μ_i^r = cost of shortest chain from $\breve{s}$ to i in $\mathrm{G}(f^r, \pi^r)$ if i is permanently labeled before Dijkstra's algorithm is terminated, or $= \mu_{\breve{t}}^r$ otherwise. Let $\mathcal{P}_r$ be the FAP from $\breve{s}$ to $\breve{t}$ wrt f^r in G corresponding to $\mathcal{C}_r$, and ϵ_r its residual capacity. Let $\gamma_r = \min. \{\epsilon_r, \overline{v} - v^r\}$. Carry out flow augmentation along $\mathcal{P}_r$ by γ_r units, and let f^{r+1} be the new flow vector obtained. Its value is $v^{r+1} = v^r + \gamma_r$. Define $\pi_i^{r+1} = \pi_i^r + \mu_i^r$ for all $i \in \mathcal{N}$ and $\pi^{r+1} = (\pi_i^{r+1})$. If $v^{r+1} = \overline{v}, (f^{r+1}, \pi^{r+1})$ is an optimum pair in G, terminate. Otherwise go to the next step with this new pair.

Discussion

1. For simplicity of notation, let (f, π) denote the pair satisfying (5.27) at the beginning of a step in this algorithm, and by $(\hat{f}, \hat{\pi})$ the pair obtained at the end of this step. Then $(\hat{f}, \hat{\pi})$ also satisfies (5.27).

   To prove this, let $\mathbf{X}$ be the set of permanently labeled nodes at the stage that Dijkstra's algorithm was terminated in this step. Let $\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i)$, $\hat{\overline{c}}_{ij} = c_{ij} - (\hat{\pi}_j - \hat{\pi}_i)$ be the reduced costs of arc (i, j) wrt the node price vectors $\pi, \hat{\pi}$ respectively. $\overline{c}', \hat{\overline{c}}'$ denote the cost vectors in the residual networks $\mathrm{G}(f, \pi)$, $\mathrm{G}(\hat{f}, \hat{\pi})$ respectively.

   For each $j \in \mathbf{X}$, let μ_j denote the cost of a shortest chain from $\breve{s}$ to j in $\mathrm{G}(f, \pi)$. For $j \notin \mathbf{X}$, let $\mu_j = \mu_{\breve{t}}$ as defined in the algorithm. By Theorem 4.2 and the result in Exercise 4.21, we have

$$\mu_j - \mu_i \leqq \overline{c}'_{ij} \text{ for all arcs } (i, j) \text{ in } \mathrm{G}(f, \pi) \qquad (5.28)$$

   Let $\mathcal{C}$ denote the shortest chain from $\breve{s}$ to $\breve{t}$ in $\mathrm{G}(f, \pi)$ identified in Dijkstra's algorithm in this step, and $\mathcal{P}$ the FAP in G wrt f corresponding to $\mathcal{C}$. (5.28) holds as a strict equation for arcs (i, j) on $\mathcal{C}$.

So, for arcs (i, j) on $\mathcal{C}$ we have $\mu_j - \mu_i = \overline{c}'_{ij}$. Therefore if (i, j) on $\mathcal{C}$ has a + label in $\mathrm{G}(f, \pi)$, then $(i, j) \in \mathcal{A}$ and $\mu_j - \mu_i = \overline{c}'_{ij} = c_{ij} - (\pi_j - \pi_i)$, and hence $c_{ij} = (\pi_j + \mu_j) - (\pi_i + \mu_i) = (\hat{\pi}_j - \hat{\pi}_i)$. If (i, j) has a $-$ label in $\mathrm{G}(f, \pi)$, then $(j, i) \in \mathcal{A}$ and $\mu_j - \mu_i = \overline{c}'_{ij} = -(c_{ji} - (\pi_i - \pi_j))$ and hence $c_{ji} = \hat{\pi}_i - \hat{\pi}_j$. Hence all arcs in G corresponding to arcs on $\mathcal{C}$ continue to satisfy (5.27) in the new pair $(\hat{f}, \hat{\pi})$. In fact, for all arcs (p, q) on $\mathcal{P}$ we have $c_{pq} = \hat{\pi}_q - \hat{\pi}_p$.

For any arc (i, j) in G not corresponding to an arc on $\mathcal{C}$, we have $\hat{f}_{ij} = f_{ij}$. From this and (5.28) it can be verified that all these arcs continue to satisfy (5.27) in the new pair $(\hat{f}, \hat{\pi})$.

2. Using the notation in 1., all the arc cost coefficients in $\mathrm{G}(\hat{f}, \hat{\pi})$ are $\geqq 0$.

   To see this, consider an arc (i, j) in $\mathrm{G}(\hat{f}, \hat{\pi})$. If it has a + label, then $(i, j) \in \mathcal{A}$ and $\hat{f}_{ij} < k_{ij}$ and $\hat{\overline{c}}'_{ij} = \hat{\overline{c}}_{ij} = c_{ij} - (\hat{\pi}_j - \hat{\pi}_i) \geqq 0$, by (5.27). If it has a - label, then $(j, i) \in \mathcal{A}$ and $\hat{f}_{ji} > \ell_{ji}$ and $\hat{\overline{c}}'_{ij} = -\hat{\overline{c}}_{ji} \geqq 0$ by (5.27). Hence all the arc cost coefficients in $\mathrm{G}(\hat{f}, \hat{\pi})$ are $\geqq 0$.

   From this, we see that in every step of this algorithm, the shortest chain problem to be solved is on a residual network in which the arc cost coefficients are $\geqq 0$. So, Dijkstra's algorithm can be applied to solve the shortest chain problem in every step.

3. Each FAP used to augment the flow in this algorithm is a least cost FAP among all the FAPs at that stage.

   To see this, let us use the notation in 1. From 1. we know that $c_{pq} = \hat{\pi}_q - \hat{\pi}_p$ for all arcs (p, q) on the FAP $\mathcal{P}$ used to generate $\hat{f}$ from f. Hence the cost of this FAP $\mathcal{P}$ is $\hat{\pi}_{\breve{t}} - \hat{\pi}_{\breve{s}}$.

   Suppose $\mathcal{P}'$ is an FAP from $\breve{s}$ to $\breve{t}$ in G wrt f. If (i, j) is a common arc on $\mathcal{P}$ and $\mathcal{P}'$ we have $c_{ij} = \hat{\pi}_j - \hat{\pi}_i$. If (i, j) is a forward arc on $\mathcal{P}'$ which is not on $\mathcal{P}$, then $\hat{f}_{ij} = f_{ij} < k_{ij}$ and by the result in (a), $\hat{\pi}_j - \hat{\pi}_i \leqq c_{ij}$. If (i, j) is a reverse arc on $\mathcal{P}'$ which is not on $\mathcal{P}$, then $\hat{f}_{ij} = f_{ij} > \ell_{ij}$ and by the result in 1., $\hat{\pi}_j - \hat{\pi}_i \geqq c_{ij}$. These facts imply that the cost of $\mathcal{P}'$ is $\geqq \hat{\pi}_{\breve{t}} - \hat{\pi}_{\breve{s}}$ = cost of $\mathcal{P}$.

So, the FAP $\mathcal{P}$ is the cheapest among all the FAPs available at the time it is used in the algorithm. Hence the name **shortest augmenting path method** for this algorithm.

4. The finiteness of this algorithm follows from the finiteness of the parametric value minimum cost flow algorithm discussed in Section 5.3.

5. When this method is specialized to solve the assignment problem (see Jonker and Volgenant [1987]), it leads to the shortest augmenting path algorithm for the assignment problem. Each augmenting path used in this method will be an alternating path. Since the assignment problem of order n needs at most n augmentations, the method terminates after at most n steps. The work in each step involves solving a shortest chain problem, which takes at most $O(n^2)$ effort by Dijkstra's method. So, the overall computational complexity of this method for an assignment problem of order n is also $O(n^3)$.

5.6 A Class of Primal-Dual Methods

We consider the minimum cost circulation problem (5.2) in the directed network $G = (\mathcal{N}, \mathcal{A}, \ell, k, c)$ in which ℓ, k, c are all finite integer vectors satisfying $\ell \leqq k$, and $|\mathcal{N}| = n, |\mathcal{A}| = m$. In this section we discuss a class of primal-dual algorithms for this problem. These methods maintain in every step; a flow vector, node price vector pair (f, π) called an **admissible pair** which always satisfies the following conditions

1. f is always an integer, bound feasible flow vector (i.e., $\ell \leqq f \leqq k$). π is always an integer vector.

2. all arcs in $\mathcal{A}$ are always α-, β-, or γ-arcs as defined in Section 5.3, in the pair (f, π).

By 2., we know that an admissible pair satisfies the c.s. optimality conditions. All but the final admissible pair obtained in these methods violate flow conservation at some nodes.

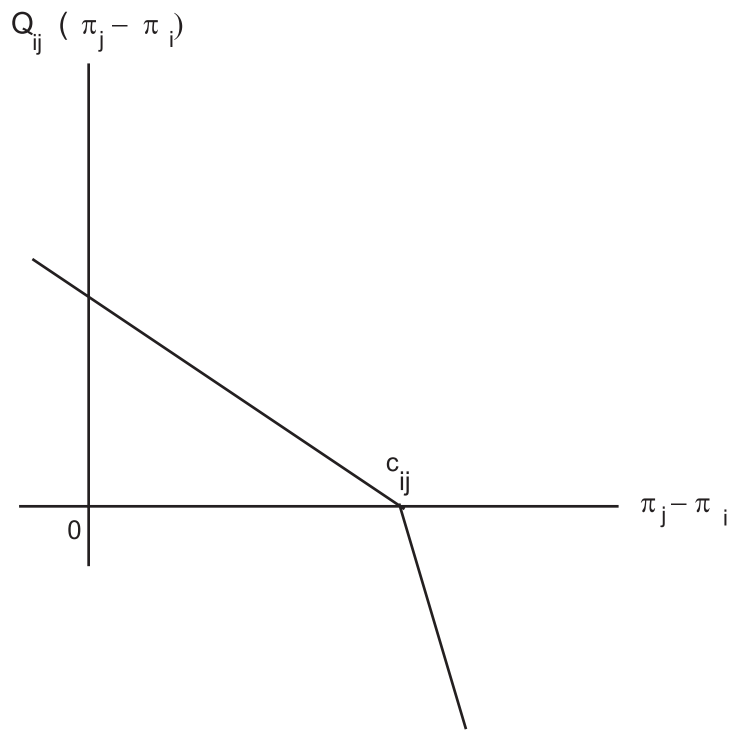


Figure 5.19: The slope of this piecewise linear function is $-\ell_{ij}$ when tension $< c_{ij}$, and $-k_{ij}$ when tension $> c_{ij}$.

Given a flow vector f in G, define *the deficit at node i in f* to be $d_i = f(\mathcal{N}, i) - f(i, \mathcal{N})$.

Given a node price vector $\pi = (\pi_i)$ in G, for each $(i, j) \in \mathcal{A}$ define $Q_{ij}(\pi_j - \pi_i) = \min.\{g_{ij}(c_{ij} - (\pi_j - \pi_i)) : \text{over } \ell_{ij} \leqq g_{ij} \leqq k_{ij}\ \}$. Define $Q(\pi) = \sum(Q_{ij}(\pi_j - \pi_i) : \text{over } (i,j) \in \mathcal{A})$. $Q_{ij}(\pi_j - \pi_i)$ is a piecewise linear concave function. A plot of it is shown in Figure 5.19.

It can be verified that $\sum(d_i : \text{over } i \in \mathcal{N}) = 0$. So, in a flow vector f if some node deficits are $\neq 0$, there must exist a node with a positive deficit, and another with a negative deficit. In an admissible pair (f, π) if deficits are 0 at all the nodes, f is a feasible flow vector, and the pair is an optimal pair. The methods discussed in this section always try to move to a pair in which all node deficits are 0, while maintaining admissibility, by various types of moves described below.

THEOREM 5.4 *Under the assumption that ℓ, k are finite vectors, the dual of the minimum cost circulation problem (5.2) in* G *is equivalent to the following unconstrained maximization problem*

$$\text{Maximize } \{Q(\pi): \text{ over } \pi \in \mathbb{R}^n\} \tag{5.29}$$

Proof Let π be the node price vector. Associate the dual variable vectors $\mu = (\mu_{ij}), \nu = (\nu_{ij})$ to the lower, upper bound constraints on f respectively. Then the dual of (5.2) is

$$\begin{array}{rll} \text{Maximize} & \mu\ell - \nu k & \\ \text{Subject to} & (\pi_j - \pi_i) + \mu_{ij} - \nu_{ij} & = c_{ij}, \text{ for } (i,j) \in \mathcal{A} \\ & \mu, \nu & \geqq 0 \end{array} \tag{5.30}$$

Since $\ell \leqq k$, in an optimum solution of (5.30), we will have $\mu_{ij} =$ max.$\{0,\ c_{ij} - (\pi_j - \pi_i)\}$ and $\nu_{ij} = -$ min.$\{0,\ c_{ij} - (\pi_j - \pi_i)\}$, for each $(i,j) \in \mathcal{A}$. Using these we can eliminate the variables μ_{ij}, ν_{ij} from the dual problem and express it purely in terms of the node prices π_is. Since at least one of the quantities among max. $\{0,\ c_{ij} - (\pi_j - \pi_i)\}$, min. $\{0,\ c_{ij} - (\pi_j - \pi_i)\}$ is always 0, it can be verified that the problem (5.30) expressed in terms of π is exactly equivalent to (5.29). ∎

In the algorithms discussed in this section, we use two criteria for measuring progress. One criterion is the total absolute deficit, $\sum(|d_i| :$ over $i \in \mathcal{N})$, which should be reduced to 0. When it is 0, the current pair is optimal. The second criterion is the dual functional $Q(\pi)$; it should be maximized. When it is maximum, the node price vector π is optimum.

The methods use three types of steps called **flow adjustment steps (FAS), price adjustment steps (PAS),** *and* **ascent steps** to solve the problem. All adjustment steps, FAS or PAS, move from one admissible pair (f, π) with a node deficit vector $d = (d_i)$, to another admissible pair $(\hat{f}, \hat{\pi})$ with node deficit vector $\hat{d} = (\hat{d}_i)$ subject to the following conditions

$$\begin{aligned} &\text{If } d_i > 0 \quad \text{then} \quad \hat{d}_i \leqq d_i \text{ and } \hat{\pi}_i \leqq \pi_i \\ &\text{If } d_i \leqq 0 \quad \text{then} \quad \hat{d}_i \leqq 0 \text{ and } \hat{\pi}_i \geqq \pi_i \end{aligned} \tag{5.31}$$

The adjustment step is called an FAS if $\hat{d}_i < d_i$ for at least one node i with $d_i > 0$, and a PAS if $\hat{\pi}_i > \pi_i$ for at least one node i with $d_i \leqq 0$.

An ascent step is a move from an admissible pair (f, π) to another admissible pair $(\hat{f}, \hat{\pi})$ satisfying $Q(\hat{\pi}) > Q(\pi)$.

An adjustment step, FAS or PAS, will never increase the total absolute deficit, but an FAS strictly decreases it. To see this let an adjustment step move from the admissible pair (f, π) associated with deficit vector $d = (d_i)$ to $(\hat{f}, \hat{\pi})$ associated with $\hat{d} = (\hat{d}_i)$. We have $\sum d_i = \sum \hat{d}_i = 0$. So,

$$\begin{aligned} \frac{1}{2}\sum |d_i| &= \sum(d_i : \text{ over } i \text{ satisfying } d_i > 0) \\ &\geqq \sum(d_i : \text{ over } i \text{ satisfying } \hat{d}_i > 0) \text{ by (5.31)} \\ &\geqq \sum(\hat{d}_i : \text{ over } i \text{ satisfying } \hat{d}_i > 0) \\ &= \frac{1}{2}\sum |\hat{d}_i| \end{aligned}$$

Also, we see that $\sum |d_i| = \sum |\hat{d}_i|$ iff $\hat{d}_i = d_i$ for all i satisfying $d_i > 0$, and hence this cannot happen in an FAS. Hence an FAS strictly decreases the total absolute deficit, and a PAS never increases it.

An FAS or PAS may decrease the value of $Q(\pi)$. We will call an adjustment step (FAS or PAS) *harmless* if it does not decrease the value of $Q(\pi)$.

An ascent step may not always qualify as an adjustment step. Indeed in an ascent step, the total absolute deficit may increase.

There are several means by which we can look for ways of making adjustment steps. One is to search for the so-called **flow altering paths** or FAℓPs. Given an admissible pair (f, π), an FAℓP wrt it is a path from a node with a positive deficit to one with a negative deficit, all of whose arcs are β-arcs, and all forward arcs have residual capacity, and all reverse arcs have flow $>$ its lower bound. Given an FAℓP wrt the pair (f, π), define its capacity to be $\epsilon = \min.$ $\{$ deficit of start node, $-$deficit of terminal node, residual capacities of forward arcs, $f_{ij} - \ell_{ij}$ for reverse arcs (i, j) $\}$.

Given an FAℓP $\mathcal{P}$ of capacity ϵ wrt the admissible pair (f, π), the *flow change operation using it* leads to the admissible pair $(\overline{f}, \pi)$, in which $\overline{f}$ is obtained from f by increasing (decreasing) the flow amount on forward (reverse) arcs of $\mathcal{P}$ by ϵ. As a result of this flow change, the deficits of both the start and terminal nodes remain of the same sign (could be 0), but each of their absolute values decrease by ϵ, and the deficits of all other nodes remain unaltered. Since an FAℓP always consists of β-arcs only, the c.s. conditions continue to hold in the new pair. Also, if ℓ, k, f are all integer vectors, ϵ is an integer, and hence the new flow vector $\overline{f}$ will also be an integer vector.

We will now describe a procedure, Procedure 1, which carries out adjustment steps by searching for FAℓPs using a tree growth scheme. To avoid confusion with steps in the main methods, we will call the steps in the procedure *items*.

PROCEDURE 1 : TO CARRY OUT AN ADJUSTMENT STEP BY SEARCHING FOR AN FAℓP

Initialization Let (f, π) be the current admissible pair with $d = (d_i)$ as the deficit vector. If $d = 0$, (f, π) is an optimum pair, terminate the method. Otherwise select a negative deficit node i_0 and label it with $\emptyset$. List $= \{i_0\}$. Go to Item 1.

Item 1 Select a node for scanning If list $= \emptyset$, go to Item 3. Otherwise select a node from the list, say p, for scanning. Delete p from the list, and go to Item 2.

Item 2 Scanning Let p be the node to be scanned. Find all unlabeled nodes j such that (p, j) is a β-arc with $f_{pj} > \ell_{pj}$, label them with $(p, -)$ and include them in the list. Find all unlabeled nodes i such that (i, p) is a β-arc with $f_{ip} < k_{ip}$, label them with $(p, +)$ and include them in the list.

Terminate this procedure if a node with positive deficit is labeled. If that node is q, the predecessor path of q is an FAℓP wrt (f, π). Carrying out the flow change operation with that FAℓP is an FAS. Since there is no change in the node price vector this is a harmless FAS.

If no node with a positive deficit is labeled go back to Step 1.

Item 3 Node price change We come to this item if no node with positive deficit is labeled, and list $= \emptyset$. Let $\mathbf{X}$ = set of labeled nodes, and $\overline{\mathbf{X}}$ its complement. Let $\mathbf{A}^1$ be the set of γ-arcs in $(\mathbf{X}, \overline{\mathbf{X}})$, and $\mathbf{A}^2$ the set of α-arcs in $(\overline{\mathbf{X}}, \mathbf{X})$.

If $\mathbf{A}^1 \cup \mathbf{A}^2 = \emptyset$, then $k(\overline{\mathbf{X}}, \mathbf{X}) - \ell(\mathbf{X}, \overline{\mathbf{X}}) = f(\overline{\mathbf{X}}, \mathbf{X}) - f(\mathbf{X}, \overline{\mathbf{X}}) = f(\mathcal{N}, \mathbf{X}) - f(\mathbf{X}, \mathcal{N}) = d(\mathbf{X}) < 0$, by Theorem 2.11 this implies that there is no feasible circulation in G; terminate the method.

If $\mathbf{A}^1 \cup \mathbf{A}^2 \neq \emptyset$, let $\delta = \min. \{|c_{ij} - (\pi_j - \pi_i)| : (i, j) \in \mathbf{A}^1 \cup \mathbf{A}^2\}$. δ is finite and > 0. Define $\hat{\pi} = (\hat{\pi}_i)$ where $\hat{\pi}_i = \pi_i + \delta$ for $i \in \mathbf{X}$, or $= \pi_i$ for $i \in \overline{\mathbf{X}}$. Since $d_i \leqq 0$ for all $i \in \mathbf{X}$, changing (f, π) to $(f, \hat{\pi})$ satisfies all the conditions for being a PAS. Terminate the procedure.

Now we will describe another procedure that tries to make adjustment steps that involve a single node together with its immediate neighbors, For each $i \in \mathcal{N}$ define the following wrt an admissible pair (f, π) in G.

$$\begin{aligned} \mathbf{B}_i^+(f, \pi) &= \{j : (j, i) \text{ is a } \beta\text{-arc and } f_{ji} < k_{ji}\} \\ \mathbf{A}_i^-(f, \pi) &= \{j : (i, j) \text{ is a } \beta\text{-arc and } f_{ij} > \ell_{ij}\} \end{aligned}$$

PROCEDURE 2: SEQUENCE OF SINGLE NODE FAS AND PASs

Single node FASs Let (f, π) be the present admissible pair with $d = (d_i)$ as the node deficit vector. Select a node q with $d_q < 0$.

Look for a node $j \in \mathbf{B}_i^+(f, \pi)$ for which $d_j > 0$. If such a node j exists, let $\epsilon = \min. \{-d_q, d_j, k_{jq} - f_{jq}\}$. Add ϵ to f_{jq} and d_q; subtract ϵ from d_j. This is a single node FAS.

If there is no node j like the above, look for a node $p \in \mathbf{A}_i^-(f, \pi)$ for which $d_p > 0$. If such a node p exists, let $\epsilon = \text{min.}\ \{-d_q, d_p, f_{qp} - \ell_{qp}\}$. Subtract ϵ from f_{qp}, d_p and add ϵ to d_q. This is a single node FAS too.

Repeat with the new pair and node q if its deficit is still < 0. If its deficit became 0, select another node with negative deficit and repeat.

Single node PASs Let (f, π) be the present admissible pair with $d = (d_i)$ as the node deficit vector. Select a node q with $d_q < 0$. If there are no β-arcs of form (j, q) with $f_{jq} < k_{jq}$, or (q, j) with $f_{qj} > \ell_{qj}$ go to Item 1; otherwise go to Item 2.

Item 1 Define $\mathbf{E} = \{(j, q) : f_{jq} < k_{jq}\} \cup \{(q, j) : f_{qj} > \ell_{qj}\}$. If $\mathbf{E} = \emptyset$, all arcs incident into q are saturated, and all those incident out of q are lower boundary arcs, and yet $d_q < 0$, so there is no feasible circulation in G; terminate the method.

If $\mathbf{E} \neq \emptyset$, let $\delta = \text{min.}\ \{|c_{uw} - (\pi_w - \pi_u)| : (u, w) \in \mathbf{E}\}$. $\delta > 0$. Add δ to π_q; this operation is a single node PAS.

Item 2 If

$$\epsilon = \sum_{j \in \mathbf{B}_q^+(f,\pi)} (k_{jq} - f_{jq}) + \sum_{j \in \mathbf{A}_q^-(f,\pi)} (f_{qj} - \ell_{qj}) \leqq -d_q \qquad (5.32)$$

change f_{jq} to k_{jq} for each $j \in \mathbf{B}_q^+(f, \pi)$, and f_{qj} to ℓ_{qj} for each $j \in \mathbf{A}_q^-(f, \pi)$. Both these sets become $\emptyset$ wrt the new flow vector. With the new pair go back to Item 1.

Discussion

The flow change in Item 2 decreases the total absolute deficit iff some node in the original $\mathbf{B}_q^+(f, \pi) \cup \mathbf{A}_q^-(f, \pi)$ had positive deficit; otherwise, the total absolute deficit remains unchanged by this flow change.

It can be shown that the directional derivative of $Q(\pi)$ along the direction of increasing π_q and leaving other node prices unchanged is $\sum(k_{qj}$: over (q,j) is a γ-arc) + $\sum(\ell_{qj}$: over (q,j) an α- or β-arc) $-\sum(k_{jq}$: over (j,q) a γ-, or β-arc) $-\sum(\ell_{jq}$: over (j,q) an α-arc) $= -d_q$ $-\sum(k_{jq} - f_{jq}$: over $j \in \mathbf{B}_q^+(f,\pi)) - \sum(f_{qj} - \ell_{qj}$: over $j \in \mathbf{A}_q^-(f,\pi))$. From this it follows that the single node PAS above increases $Q(\pi)$ iff strict inequality holds in (5.32). If (5.32) holds as an equation, the single node PAS at q leaves $Q(\pi)$ unchanged, but the flow change carried out reduces the deficit at node q to 0.

When we are in Item 2 for making a single node PAS at node q, if (5.32) is violated, i.e., if (5.33) holds

$$\sum_{j\in\mathbf{B}_q^+(f,\pi)} (k_{jq} - f_{jq} + \sum_{j\in\mathbf{A}_q^-(f,\pi)} (f_{qj} - \ell_{qj}) > -d_q \tag{5.33}$$

then no further progress is possible at node q by single node PASs. We can now attempt a single node FAS, but if

$$d_q < 0 \text{ and } d_j \leqq 0 \text{ for all } j \in \mathbf{B}_q^+(f,\pi) \cup \mathbf{A}_q^-(f,\pi) \tag{5.34}$$

no further progress can be made at node q with either a single node PAS or FAS. When (5.33), (5.34) both hold, we need to perform FAS or PAS involving multiple nodes. We will now describe a procedure that leads to a multiple node adjustment step that helps also to achieve large price increases consistent with the philosophy of a single node PAS.

PROCEDURE 3: MULTIPLE NODE ADJUSTMENT STEPS

Item 1 Let (f,π) be the current admissible pair with $d = (d_i)$ as the deficit vector. Select a node q with $d_q < 0$ satisfying (5.33), (5.34).

Item 2 Let $\mathbf{D}$ be either $\emptyset$ or a subset of $\mathbf{B}_q^+(f,\pi) \cup \mathbf{A}_q^-(f,\pi)$ satisfying

$$\sum_{j \in \mathbf{B}_q^+(f,\pi) \cap \mathbf{D}} (k_{jq} - f_{jq}) + \sum_{j \in \mathbf{A}_q^-(f,\pi) \cap \mathbf{D}} (f_{qj} - \ell_{qj}) \overset{<}{=} -d_q \qquad (5.35)$$

Let $\overline{\mathbf{D}} = (\mathbf{B}_q^+(f,\pi) \cup \mathbf{A}_Q^-(f,\pi)) \setminus \mathbf{D}$. Because of (5.33) and (5.35) $\overline{\mathbf{D}} \neq \emptyset$. Proceed similar to Procedure 1, with the only difference being that the initial labels will be given only to nodes in the subset $\overline{\mathbf{D}}$ rather than the entire set $\mathbf{B}_q^+(f,\pi) \cup \mathbf{A}_Q^-(f,\pi)$. Label q with $\emptyset$. Label each of the nodes in $\mathbf{B}_q^+(f,\pi) \cap \overline{\mathbf{D}}$ with $(q,+)$, and each of the nodes in $\mathbf{A}_q^-(f,\pi) \cap \overline{\mathbf{D}}$ with $(q,-)$. Put all nodes in $\overline{\mathbf{D}}$ in the list of labeled and unscanned nodes (in this procedure, the root node q is not put in the list unlike Procedure 1)

Continuation now consists of deleting a node p from the list and scanning it as in Procedure 1. If a node with positive deficit is labeled, an FAℓP has been identified; stop tree growth and carry out an FAS using it as under Procedure 1. This procedure terminates after this FAS.

If tree growth stops without a node having positive deficit getting labeled, carry out a PAS using the cut of labeled and unlabeled nodes as under Procedure 1. Then change f_{jq} to k_{jq} for all $j \in \mathbf{B}_q^+(f,\pi) \cap \mathbf{D}$, and f_{qj} to ℓ_{qj} for all $j \in \mathbf{A}_q^-(f,\pi) \cap \mathbf{D}$.

Discussion

If no FAℓP is identified in Procedure 3, it can be verified that the work carried out at the end is a PAS. The flow changes in this PAS increase the deficit of each node in $\mathbf{D}$, while the deficit of q still remains nonpositive because of (5.35). The adjustment step in Procedure 3 is harmless. This procedure coincides with Procedure 1 if $\mathbf{D} = \emptyset$. However, taking $\mathbf{D} \neq \emptyset$ is advantageous if strict inequality holds in (5.35) as this makes the multiple node PAS in Procedure 3 an ascent step.

We will now discuss a procedure for carrying out an ascent step. Let (f,π) be the present admissible pair. Let $\mathbf{S}$ be a proper subset of nodes, and let its incidence vector be $x(\mathbf{S}) = (x_i(\mathbf{S}))$ where $x_i(\mathbf{S})$

$= 1$ if $i \in \mathbf{S}$, 0 otherwise. Let $\overline{\mathbf{S}}$ be the complement of $\mathbf{S}$. Define $a(\mathbf{S}, \pi) = \sum_{(i,j)\in\mathcal{A}} a_{ij}(\mathbf{S}, \pi)$, where

$$a_{ij}(\mathbf{S}, \pi) = \begin{cases} \ell_{ij} & \text{if} \quad (i,j) \in (\mathbf{S}, \overline{\mathbf{S}}) \text{ is an } \alpha- \text{ or } \beta-\text{arc} \\ k_{ij} & \text{if} \quad (i,j) \in (\mathbf{S}, \overline{\mathbf{S}}) \text{ is a } \gamma-\text{arc} \\ -\ell_{ij} & \text{if} \quad (i,j) \in (\overline{\mathbf{S}}, \mathbf{S}) \text{ is an } \alpha-\text{arc} \\ -k_{ij} & \text{if} \quad (i,j) \in (\overline{\mathbf{S}}, \mathbf{S}) \text{ is a } \beta- \text{ or } \gamma-\text{arc} \\ 0 & \quad\quad \text{otherwise} \end{cases} \tag{5.36}$$

It can be verified that $a(\mathbf{S}, \pi)$ is the directional derivative of the dual functional $Q(\pi)$ along the direction $x(\mathbf{S})$. So, if $a(\mathbf{S}, \pi) > 0$, to increase $Q(\pi)$ we can move in the direction $x(\mathbf{S})$, i.e., increase node prices of nodes in $\mathbf{S}$ by equal amounts while keeping node prices outside of $\mathbf{S}$ unchanged. The ascent step uses this fact.

The main feature of this procedure is that the choice of ascent directions is very simple. A node i_0 with nonzero deficit is chosen, and an ascent is attempted along the coordinate direction of π_{i_0}. If such an ascent is not possible and a reduction in the absolute deficit cannot be effected through flow change, an adjacent node of i_0, say i_1, is chosen and an ascent attempted along the sum of the coordinate vectors corresponding to π_{i_0} and π_{i_1}. If such an ascent is not possible, and flow change is not possible either, an adjacent node of either i_0 or i_1 is chosen and the process is continued.

PROCEDURE 4 : ASCENT STEP

Item 1 Let (f, π) be the present admissible pair with nonzero deficit vector $d = (d_i)$. Select a node, say i_0 with $d_{i_0} < 0$. Root a tree at i_0 by labeling it with $\emptyset$. List $= \{i_0\}$, $\mathbf{S} = \emptyset$.

General item Delete a node, say p, from the list, insert it into $\mathbf{S}$, and scan it. Scanning p involves labeling all unlabeled nodes in

$\mathbf{B}_p^+(f,\pi)$ with the label $(p,+)$, and unlabeled nodes in $\mathbf{A}_p^-(f,\pi)$ with the label $(p,-)$ and inserting all these newly labeled nodes in the list.

If a node with positive deficit is labeled, but $a(\mathbf{S},\pi) \leqq 0$, an FA$\ell$P has been identified. Carry out an FAS using this FAℓP as under Procedure 1, and terminate this procedure.

On the other hand, if we have $\overline{\mathbf{S}} = \mathcal{N}\backslash\mathbf{S} \neq \emptyset$ and $a(\mathbf{S},\pi) > 0$, carry out the following ascent step. Let $\mathbf{E} = \{(i,j) : (i,j)$ is a γ-arc in $(\mathbf{S}, \overline{\mathbf{S}})$, or an α-arc in $(\overline{\mathbf{S}}, \mathbf{S})\}$. If $\mathbf{E} = \emptyset$, there is no feasible circulation in G; terminate the method. Otherwise, define

$$\delta \quad = \quad \text{min. } \{|c_{ij} - (\pi_j - \pi_i)| : (i,j) \in \mathbf{E}\} \tag{5.37}$$

and the new flow vector, node price vector pair $(\hat{f} = (\hat{f}_{ij}), \hat{\pi} = (\hat{\pi}_i))$, where

$$\hat{f}_{ij} \quad = \quad \begin{cases} k_{ij} \text{ if } (i,j) \text{ is a } \beta\text{-arc in } (\overline{\mathbf{S}}, \mathbf{S}) \text{ and } i \text{ labeled} \\ \\ \ell_{ij} \text{ if } (i,j) \text{ is a } \beta\text{-arc in } (\mathbf{S}, \overline{\mathbf{S}}) \text{ and } j \text{ labeled} \\ \\ f_{ij} \text{ otherwise} \end{cases}$$

$$\hat{\pi}_i \quad = \quad \begin{cases} \pi_i + \delta & \text{if } i \in \mathbf{S} \\ \\ \pi_i & \text{if } i \in \overline{\mathbf{S}} \end{cases}$$

The procedure terminates with the new pair $(\hat{f}, \hat{\pi})$.

If the conditions for neither FAS nor ascent step are satisfied, go to the next item in tree growth.

Discussion

If $\mathbf{E} = \emptyset$ at the occurrence of an ascent step, from (5.36) we have $0 < a(\mathbf{S},\pi) = -k(\overline{\mathbf{S}},\mathbf{S}) + \ell(\mathbf{S},\overline{\mathbf{S}})$, i.e., $k(\overline{\mathbf{S}},\mathbf{S}) - \ell(\mathbf{S},\overline{\mathbf{S}}) < 0$, which implies that there is no feasible circulation in G by Theorem 2.11.

If tree growth stops without the conditions for an FAS being satisfied, let $\mathbf{X}$ be the set of labeled nodes at that time. Since all nodes in $\mathbf{X}$ are scanned, $\mathbf{S} = \mathbf{X}$ at this stage. Since the conditions for an FAS were not satisfied so far, $d_i \leqq 0$ for all $i \in \mathbf{X}$. So, $\overline{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X} = \overline{\mathbf{S}} = \mathcal{N} \backslash \mathbf{S} \neq \emptyset$, since all nodes with positive deficit are in it. We also have $f_{ij} = k_{ij}$ for all β-arcs $(i, j) \in (\overline{\mathbf{S}}, \mathbf{S})$, and $= \ell_{ij}$ for all β-arcs $(i, j) \in (\mathbf{S}, \overline{\mathbf{S}})$. By (5.36) this implies that $a(\mathbf{S}, \pi) > 0$ at this stage and an ascent step can therefore be carried out now.

Hence, Procedure 4 will either terminate by carrying out an FAS (which is harmless since there is no change in the node price vector) or by carrying out an ascent step. However, in the ascent step, the total absolute deficit may strictly increase. In this procedure, often the condition for an ascent step is likely to be satisfied well before tree growth stops; hence this procedure tends to terminate earlier than Procedure 3.

ALGORITHMS FOR MINIMUM COST CIRCULATION PROBLEMS

These algorithms can be initiated with any node price vector π, and a bound feasible flow vector selected so that (f, π) is an admissible pair.

The **classical Primal-Dual algorithm** is based solely on Procedure 1. It carries out FAS or PAS by applying Procedure 1 repeatedly until all the node deficits are converted to 0, at which stage we have an optimum pair (f, π). This algorithm is guaranteed to terminate after a finite number of iterations, either with an optimum pair, or with the conclusion that there is no feasible circulation in G.

Other Primal-Dual algorithms consist of a sequence in any order, of FAS, PAS, and ascent steps through Procedures 2, 3, 4. These procedures can be combined in different ways to yield a variety of algorithms. This flexibility allows the construction of algorithms that can be tailored to the problem at hand for maximum effectiveness. In these algorithms, most of the ascent directions tend to be single coordinate directions, leading to the interpretation of these algorithms as coordinate ascent or relaxation methods. This is an important characteristic and a key factor in the practical efficiency of these algorithms. We will now show that any algorithm in this class terminates with an optimum solution in a finite number of steps if the problem is feasible.

THEOREM 5.5 *Suppose the minimum cost circulation problem in G is solved by an algorithm consisting of repeated applications of Procedures 2, 3, 4 in any order beginning with an initial admissible pair (f^0, π^0). The algorithm terminates with an optimum pair after a finite number of steps, if the problem is feasible, under the following assumptions.*

1. *ℓ, k, c, f^0, π^0 are all finite integer vectors*
2. *either all steps in the algorithm are adjustment steps (FAS or PAS) or all adjustment steps used are harmless.*

Proof By assumption (a), all the quantities ϵ and δ in every step of the algorithm will always be positive integers, and all the pairs (f, π) obtained in the algorithm are always integer vector pairs.

$Q(\pi)$ is piecewise linear in arc tensions, $\pi_j - \pi_i$, and $a(\mathbf{S}, \pi)$ is the rate of change of $Q(\pi)$ along $x(\mathbf{S})$. The actual change in $Q(\pi + \lambda x(\mathbf{S}))$ as a function of the step size λ is linear up to the point where its slope changes. The value of $\lambda > 0$ closest to 0 where the slope of $Q(\pi + \lambda x(\mathbf{S}))$ changes, is the one for which a new arc incident to $\mathbf{S}$ becomes a β-arc; i.e., it is δ given by (5.37), and hence $\geqq 1$. For all $0 \leqq \lambda \leqq \delta$, we therefore have $Q(\pi + \lambda x(\mathbf{S})) = Q(\pi) + \lambda a(\mathbf{S}, \pi)$. Hence whenever an ascent step is carried out, $Q(\pi)$ increases by a positive integer amount. Therefore, if the problem is feasible, under assumption (b), it is not possible to carry out an infinite number of ascent steps in the algorithm, so after some iteration all steps will be adjustment steps, or else the algorithm will terminate finitely. Hence it is sufficient to prove this theorem under the assumption that all steps are adjustment steps. We will do this now.

Each time an FAS is carried out the total absolute deficit strictly decreases, while each time a PAS is carried out the total absolute deficit does not increase. So, if the algorithm does not terminate finitely, after a finite number of iterations it must execute PAS exclusively. Let (f^r, π^r), with deficit vector $d^r = (d_i^r), r = 1, 2, \ldots$ refer to the sequence of admissible pairs in the algorithm after it began to execute PAS exclusively. In this sequence, the deficits of nodes with positive deficits will be constant, and the deficits of nodes with nonpositive deficits will remain nonpositive. The node price of each node with nonpositive deficit will not decrease, and for at least one of these nodes it strictly increases. Hence $\mathbf{N} = \{ \ i : \pi_i^r \to \infty \text{ as } r \to \infty \ \} \neq \emptyset$. Let $\overline{\mathbf{N}}$ be the complement of $\mathbf{N}$.

$d_i^r \leqq 0$ for all $i \in \mathbf{N}$, and this inequality is strict for at least one i, hence $\sum(d_i^r : \text{over } i \in \mathbf{N}) < 0$. By the definition of $\mathbf{N}$ all arcs in $(\overline{\mathbf{N}}, \mathbf{N})$ are γ-arcs, and those in $(\mathbf{N}, \overline{\mathbf{N}})$ are α-arcs. Using this in $\sum(d_i^r : \text{over } i \in \mathbf{N}) < 0$, we have $k(\overline{\mathbf{N}}, \mathbf{N}) - \ell(\mathbf{N}, \overline{\mathbf{N}}) < 0$, implying that there is no feasible circulation in G. ∎

In our description of the procedures, scanning operations started at nodes with negative deficit only. It is possible to start scanning at nodes with positive deficit as well. In practice, it was found that this modification improves the running time by at least a factor of two. The finite termination property for the method, in feasible networks, is preserved, provided that all PASs originating from positive deficit nodes *strictly* improve the dual functional value.

If the primal problem is infeasible, then it is possible for the methods based solely on Procedures 2, 3, 4, as described, to cycle forever. An easy way to circumvent this difficulty is to switch to Procedure 1 after some fixed number of iterations. This guarantees that the method will either conclude that the problem is infeasible, or find an optimum pair, after a finite number of iterations.

Comment 5.1 The new Primal-Dual methods in this section are from Bertsekas [1985], Bertsekas, Hosein, and Tseng [1987], and Bertsekas and Tseng [1988]. These papers provide results from computational experiments comparing the performance of these methods with other Primal-Dual methods, OK methods, primal network simplex and dual network simplex methods, on networks with number of nodes up to 8000, and number of arcs up to 35,000, and randomly generated data. The typical time to solve a minimum cost circulation problem on a network with 100 nodes and 5000 arcs by these methods is 12 seconds on a VAX 11/750 mainframe computer; this increases to 171 seconds when the number of nodes and arcs increases to 3000 and 35,000 respectively.

5.7 The Dual Network Simplex Method for Minimum Cost Flow Problems

We consider the minimum cost flow problem (5.3) in the connected directed network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, c, V)$ where ℓ is finite and $\ell \leqq k$. Let $\mathcal{A}^\infty = \{(i,j) : k_{ij} = \infty\}$. Let $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ be a partition in G associated with the primal and dual basic solution pair $(f = (f_{ij}), \pi = (\pi_i))$. Let $\bar{c} = (\bar{c}_{ij} = c_{ij} - (\pi_j - \pi_i))$ be the reduced cost vector wrt π. $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ is said to be a **dual feasible partition** if

$$\bar{c}_{ij} \begin{cases} \geqq 0 \text{ for all } (i,j) \in \mathcal{A}^\infty \\ \geqq 0 \text{ for all } (i,j) \in \mathbf{L} \\ \leqq 0 \text{ for all } (i,j) \in \mathbf{U} \end{cases} \tag{5.38}$$

If $\mathcal{A}^\infty = \emptyset$, we can generate a dual feasible partition in G by selecting an arbitrary spanning tree $\mathbb{T}$ in G, computing the associated $\pi, \bar{c}$ vectors, and then classifying the out-of-tree arcs into **L, U** based on the sign of their reduced cost coefficients.

If $\mathcal{A}^\infty \neq \emptyset$, we can apply the initialization procedure discussed in Section 5.3 to generate shortest chain trees in each connected component of $(\mathcal{N}, \mathcal{A}^\infty)$, so that the node price vector corresponding to each of these trees satisfies $\pi_j - \pi_i \leqq c_{ij}$ for all arcs (i,j) in that component. The procedure may terminate either by finding a negative cost circuit in the subnetwork $(\mathcal{N}, \mathcal{A}^\infty)$, or by finding such trees in each connected component of this subnetwork. In the former case, the dual of (5.3) is infeasible, and the objective value is unbounded below in (5.3) if it is feasible, so, we terminate. In the latter case we augment the shortest chain trees in the connected components of $(\mathcal{N}, \mathcal{A}^\infty)$ with arcs from $\mathcal{A} \backslash \mathcal{A}^\infty$ selected arbitrarily, to get a spanning tree $\mathbb{T}$ in G, and then by partitioning the out-of-tree arcs as mentioned above, generate a dual feasible partition.

The dual feasible partition $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ associated with the reduced cost vector $\bar{c}$ is said to be **dual nondegenerate** if $\bar{c}_{ij} > 0$ for all $(i,j) \in \mathbf{L}$, and $\bar{c}_{ij} < 0$ for all $(i,j) \in \mathbf{U}$; **dual degenerate** otherwise.

The procedure outlined above for finding an initial dual feasible partition is the Phase I of the dual network simplex method. The dual network simplex algorithm needs an initial dual feasible partition to solve (5.3).

THE DUAL NETWORK SIMPLEX ALGORITHM

Initialization Begin with a dual feasible partition.

General step Let $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ be the present dual feasible partition associated with the basic solution pair $(f = (f_{ij}), \pi = (\pi_i))$ and reduced cost vector $\bar{c} = (\bar{c}_{ij})$.

Check primal feasibility of the partition If $\ell_{ij} \leqq f_{ij} \leqq k_{ij}$ for all $(i,j) \in \mathbb{T}$, $(\mathbb{T}, \mathbf{L}, \mathbf{U})$ is primal feasible and hence optimal, terminate.

Select dropping arc Let $\mathbf{D}$ = set of all in-tree arcs (i,j) satisfying either $f_{ij} < \ell_{ij}$ or $f_{ij} > k_{ij}$. $\mathbf{D}$ is the set of arcs **eligible to be dropping arcs** in this step. Select one arc from $\mathbf{D}$, (r,s) say, to be the **dropping** *or* **leaving arc**.

Select entering arc Let $[\mathbf{X}, \overline{\mathbf{X}}]$ be the fundamental cutset of (r,s) wrt $\mathbb{T}$, with $r \in \mathbf{X}$. Define

$$\begin{array}{lll} \mathbf{S}^1 = \mathbf{L} \cap (\overline{\mathbf{X}}, \mathbf{X}) & , & \mathbf{S}^2 = \mathbf{U} \cap (\mathbf{X}, \overline{\mathbf{X}}), \text{ if } f_{rs} < \ell_{rs} \\ \mathbf{S}^1 = \mathbf{L} \cap (\mathbf{X}, \overline{\mathbf{X}}) & , & \mathbf{S}^2 = \mathbf{U} \cap (\overline{\mathbf{X}}, \mathbf{X}), \text{ if } f_{rs} > k_{rs} \end{array}$$

If $\mathbf{S}^1 \cup \mathbf{S}^2 = \emptyset$, (5.3) is infeasible, terminate.

If $\mathbf{S}^1 \cup \mathbf{S}^2 \neq \emptyset$, define $\delta = \min.\ \{\ |\bar{c}_{ij}|\text{: over } (i,j) \in \mathbf{S}^1 \cup \mathbf{S}^2\}$. δ is called the **dual simplex minimum ratio** in this pivot step. Let $\mathbf{E}$ be the set of arcs in $\mathbf{S}^1 \cup \mathbf{S}^2$ which tie for the minimum in the definition of δ. $\mathbf{E}$ is the set of arcs **eligible to enter** in this pivot step. Select one arc from $\mathbf{E}$, (p,q) say, as the **entering arc**. Let $\mathbb{C}$ be the fundamental cycle of (p, q) wrt $\mathbb{T}$. $\mathbb{C}$ will contain (r, s). Orient $\mathbb{C}$ so that (r, s) is a forward arc on it.

Delete (p, q) from **L** or **U** where it was. Replace (r, s) in $\mathbb{T}$ by (p, q) and let $\mathbb{T}'$ be the resulting spanning tree. Include (r, s) in **L** if $f_{rs} < \ell_{rs}$, and in **U** if $f_{rs} > k_{rs}$. This gives the new partition. Let $\epsilon = \ell_{rs} - f_{rs}$ if $f_{rs} < \ell_{rs}$, or $k_{rs} - f_{rs}$ if $f_{rs} > k_{rs}$. The basic flow vector corresponding to the new partition is obtained from f by adding ϵ to the flow amounts on the forward arcs of $\mathbb{C}$, and subtracting ϵ from the flow amounts on reverse arcs on $\mathbb{C}$. Go to the next step.

The entering arc choice guarantees that dual feasibility is maintained throughout the algorithm. It can be verified that if the entering arc (p, q) replaces the dropping arc (r, s) in a pivot step, then in the flow vector obtained immediately after this step, the flows on both the arcs $(p, q), (r, s)$ satisfy the respective bounds on them.

The flow vector changes in every step of this algorithm. The node price vector changes in a pivot step if the quantity $\delta > 0$, remains unchanged if $\delta = 0$. Hence, a pivot step in this algorithm is said to be **degenerate** if $\delta = 0$ in that step, **nondegenerate** if $\delta > 0$.

In large networks, in particular if $|\mathcal{A}|$ is much larger than $|\mathcal{N}|$, the selection of the entering arc by the dual simplex minimum ratio criterion is a time consuming process, and is inherently wasteful. In all LPs in which the number of variables is much greater than the number of constraints, this problem exists for applying the dual simplex algorithm.

Exercises

5.11 Prove that the dual objective function strictly increases in every nondegenerate pivot step in the dual network simplex method.

5.12 Research Problem It is not known whether cycling can occur under dual degeneracy in the dual network simplex algorithm. Either construct an example of its occurrence, or prove that it cannot occur.

5.13 Develop combinatorial methods for resolving cycling under degeneracy in the dual network simplex algorithm (Partovi[1984]).

5.14 Develop dropping and entering arc choice rules in the dual network simplex algorithm which simultaneously resolve both cycling and stalling in it.

5.8 A Strongly Polynomial Algorithm for Minimum Cost Flow Problems

Consider a single commodity minimum cost flow problem on a directed network G $= (\mathcal{N}, \mathcal{A}, \ell, k, c)$ with $|\mathcal{N}| = n, |\mathcal{A}| = m$. If all the data is rational, the *size* of this problem is defined to be the total number of binary bits of storage needed to store all the data in the problem. The **dimension** of this problem is defined to be the total number of data elements in the statement of the problem, irrespective of whether the data is rational or not. So, the dimension of our minimum cost flow problem is proportional to n and m; it is $5m + n + 2$ (number of nodes and arcs; lower bound, capacity, cost coefficient, head node, and tail node on each arc; and the exogenous flow amount at each node).

The concept of **polynomial boundedness** of an algorithm is defined in the context of problem instances with rational data of finite size. An algorithm for this problem is said to be a **polynomially bounded algorithm** if the total number of arithmetic operations (additions, multiplications and divisions, comparisons, etc.) in the algorithm, when applied on an instance with rational data, is bounded above by a polynomial in the size of the instance; and the size of each of the numbers occurring during the algorithm is always bounded above by a polynomial in the size of the original instance.

The first polynomially bounded algorithms for minimum cost flow problems were developed by Edmonds and Karp [1972 of Chapter 2]; they are based on the scaling technique (see Sections 3.2 and 5.3).

An algorithm for this problem is said to be a **strongly polynomial algorithm** if the total number of arithmetic operations in the algorithm is bounded above by a polynomial in the dimension of the problem even when the algorithm is applied on instances with real (i.e., non-rational) data, as long as the required operations are carried out exactly; and when the algorithm is applied on instances with rational data, the size of each of the numbers occurring during the algorithm is always bounded above by a polynomial in the size of the original instance.

The scaling based polynomially bounded algorithms for minimum cost flow problems discussed in Sections 3.2 and 5.3 are not strongly polynomial since the total number of arithmetic operations in those algorithms grows with the size of the instance, even when the dimension remains fixed. For the maximum value flow problem, strongly polynomial algorithms have been known since the early 1970's. The shortest augmenting path method, Dinic's method, Dinic-MKM method, and the preflow-push method discussed in Chapter 2 are all strongly polynomial algorithms for the maximum value flow problem. However, the problem of developing a strongly polynomial algorithm for the minimum cost flow problem remained open until 1985. In fact in their 1972 paper Edmonds and Karp stated the following challenge: "A challenging open problem is to give a method for the minimum cost flow problem having a bound of computation which is a polynomial in the number of nodes, and is independent of both costs and capacities." The first strongly polynomial algorithm for minimum cost flow problems is due to Tardos [1985]. Since then several strongly polynomial algorithms have been developed for minimum cost flow problems. In this section we discuss a strongly polynomial algorithm for minimum cost flow problems, perhaps the simplest conceptually among all those known, based on canceling minimum mean residual cycles.

We consider the minimum cost circulation problem (5.2) in G. From Theorem 5.1 we know that a feasible circulation f is a minimum cost circulation iff there exist no negative cost residual cycles wrt it. One of the earliest algorithms for finding minimum cost circulations, due to Klein [1967], called the **cycle canceling algorithm**, begins with a feasible circulation and repeats the step of finding a negative cost residual cycle and canceling it. If the cycles to cancel are selected arbitrarily, the computational requirements of this algorithm may grow exponentially with the size, and the algorithm may not even terminate finitely under irrational data (an example of this can be constructed from Ford and Fulkerson's maximum value flow problem discussed in Example 2.1 posed as a minimum cost circulation problem). A natural question to ask is whether the cycle canceling algorithm can be made efficient, in fact strongly polynomial, by a judicious choice of the cycles to cancel. This question has been answered in the affirmative by Goldberg and Tarjan [1989], and we present their results here.

Define the *average* or *mean cost* of a simple cycle to be its cost divided by the number of arcs in it. A simple cycle whose average cost is as small as possible is called a **minimum mean cycle**. A minimum mean cycle can be found by an algorithm of Karp [1978] in at most O(nm) time. Goldberg and Tarjan showed that if the cycle to cancel in the cycle canceling algorithm is selected to be a minimum mean residual cycle in each step, then it terminates after at most O($nm^{\mathbf{2}}$log n) cycles have been canceled. Hence, with this selection rule, the cycle canceling algorithm solves the minimum cost circulation problem in at most O($n^{\mathbf{2}}m^{\mathbf{3}}$ log n) time, which makes it an elegant strongly polynomial algorithm.

An Algorithm to Find a Minimum Mean Simple Circuit in a Directed Network

Let H = (**V, E**, d) be a directed network with d as the vector of arc weights (or lengths, or cost coefficients). The vector d is not required to be $\geqq 0$; it can be arbitrary. Let $e_1, \dots, e_p$ be the arcs in a simple circuit $\vec{\mathbb{C}}$ in H, with weights $d_1, \dots, d_p$ respectively. Then the *mean cost (*or *weight)* of $\vec{\mathbb{C}}$ is defined to be $a(\vec{\mathbb{C}}) = \frac{1}{p}\sum(d_t : \text{over } t = 1 \text{ to } p)$. A *minimum mean simple circuit* in H is one which has the least mean cost among all the simple circuits in H. Here we discuss an algorithm for finding a minimum mean simple circuit due to Karp [1978].

If H is not strongly connected, its strong components can be found in O(|**E**|) effort. The minimum mean simple circuit can be computed separately in each strongly connected component of H, and the best of these is the minimum mean simple circuit in H. So, henceforth we assume that H is strongly connected.

The cost of a minimum mean simple circuit is called the *minimum circuit mean.* Let λ^* denote the minimum circuit mean in H. Let $|\mathbf{V}| = n, |\mathbf{E}| = m$. Select any node from $\mathbf{V}$, say s, as the origin node. For each node $i \in \mathbf{V}$, and $r \geqq 0$ integer define $F_r(i)$ to be the minimum cost of a chain (not necessarily simple) from s to i in H containing exactly r arcs, ∞ if no such chain exists. These quantities $F_r(i)$, for all $i \in \mathbf{V}$ and $r = 0$ to n can be computed from the following recursive equations.

$$\begin{aligned} F_0(i) &= \infty \text{ for all } i \neq s, \text{ and } 0 \text{ for } i = s \\ F_r(i) &= \text{min. } \{F_{r-1}(j) + d_{ji} : \text{ over } j \text{ such that } (j,i) \in \mathbf{E}\} \end{aligned} \quad (5.39)$$

where d_{ji} is the weight or cost of arc (j,i) in $\mathbf{E}$. Computation of $F_r(i)$ for all $i \in \mathbf{V}$, $r = 0$ to n by this recursive method takes O(nm) effort.

ALGORITHM FOR MINIMUM MEAN SIMPLE CIRCUIT
IN A STRONGLY CONNECTED NETWORK

Step 1 Compute $F_r(i)$ for all $i \in \mathbf{V}$, $r = 0$ to n.

Step 2 Compute λ^* from

$$\lambda^* = \text{min.}_{\; i \in \mathbf{V}} \left\{ \text{max.}_{\; 0 \leqq r \leqq n-1} \left\{ \frac{F_n(i) - F_r(i)}{n - r} \right\} \right\} \quad (5.40)$$

To get a simple circuit yielding the minimum circuit mean, find a minimizing i and r in (5. 40), find a minimum cost chain consisting of n arcs from s to i, and extract a simple circuit of $n - r$ arcs within that chain.

Validity of the Minimum Mean Simple Circuit Algorithm

1. First we will prove that (5.40) holds when $\lambda^* = 0$. Since $\lambda^* = 0$, there are no negative cost circuits in H, and there is a 0-cost circuit. Also, H is strongly connected. So, there is a shortest simple chain from

s to i consisting of $< n$ arcs for every $i \in \mathbf{V}$; let its cost be π_i. Then $F_n(i) \geqq \pi_i$, and $\pi_i = \min.\ \{ F_r(i) : r = 0 \text{ to } n-1 \}$. So, $F_n(i) - \pi_i = \max.\ \{ F_n(i) - F_r(i) : r = 0 \text{ to } n-1 \} \geqq 0$. Hence

$$\max.\ \left\{ \frac{F_n(i) - F_r(i)}{n-r} : r = 0, \ldots, n-1 \right\} \geqq 0 \qquad (5.41)$$

Equality holds in (5.41) iff i is such that $F_n(i) = \pi_i$. We will now show that there exists a node i satisfying this condition. Let $\overrightarrow{\mathbb{C}}$ be a simple circuit in H of cost 0, and w a node on it. Let $\mathcal{C}(w)$ be a shortest simple chain from s to w. Then any chain consisting of $\mathcal{C}(w)$ followed by any nonnegative integer number of repetitions of $\overrightarrow{\mathbb{C}}$ is also a shortest chain from s to w in H. But the initial part of any shortest chain must also be a shortest chain from the origin to that point. After some repetitions of $\overrightarrow{\mathbb{C}}$ at the end of $\mathcal{C}(w)$ an initial part consisting of exactly n arcs will appear; suppose its end point is w'. This implies that this part has cost $F_n(w') = \pi_{w'}$. So, (5.41) holds as an equation when $i = w'$. This establishes (5.40) in this case.

2. We now prove that (5.40) holds in general. Let us study the effect of subtracting a constant α from the weight of every arc in H. This subtracts α from λ^*, and subtracts $r\alpha$ from $F_r(i)$ for all i and r. Hence it subtracts α from the right hand side of (5.40). Hence both sides of (5.40) are affected by the same quantity when the cost vector d is translated by a constant. Choosing α to be the unknown λ^* and using the result in 1., establishes (5.40) in general.

3. The computation of $F_r(i)$ using the recursive equations (5.39) can be accomplished with $O(nm)$ effort. Using the $F_r(i)$, λ^* can be computed from (5.40) and a minimum mean simple circuit identified as discussed earlier, with an additional effort of $O(n^2)$. Thus the overall effort in this algorithm for the minimum mean simple circuit is $O(nm)$.

The Minimum Mean Cycle Canceling Algorithm

Let f be a feasible circulation in G. Every residual cycle wrt f in G becomes a simple circuit in the residual network G(f) with the same cost, and vice versa. So, by finding a minimum mean simple circuit in G(f) by the algorithm described above, we can find a minimum mean residual cycle wrt f in G.

THE MINIMUM MEAN CYCLE CANCELING ALGORITHM FOR MINIMUM COST CIRCULATION PROBLEMS

Initialization Find a feasible circulation in G using the algorithms discussed in Chapter 2.

General step Let f be the present feasible circulation in G. Find a minimum mean residual cycle wrt f in G, $\mathbb{C}$ say. If the cost of $\mathbb{C}$ is $\geqq 0$, f is a minimum cost circulation in G; terminate. Otherwise cancel $\mathbb{C}$ in f and go to the next step with the new feasible circulation obtained.

Proof of Strong Polynomiality of the Minimum Mean Cycle Canceling Algorithm

(i) ϵ-optimality For $\epsilon \geqq 0$, a feasible circulation $f = (f_{ij})$ in G is said to be an *ϵ-optimal circulation* if there exists a node price vector $\pi = (\pi_i)$ such that for all $(i,j) \in \mathcal{A}$

$$\text{if} \quad \overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i) \begin{cases} > \epsilon & \text{then } f_{ij} = \ell_{ij} \\ \\ < -\epsilon & \text{then } f_{ij} = k_{ij} \end{cases} \tag{5.42}$$

By the c.s. optimality conditions, a 0-optimal feasible circulation is a minimum cost circulation in G.

(ii) Let $\alpha > 0$, and (f', π') a feasible circulation, node price vector pair in G satisfying

$$\text{if } \overline{c}'_{ij} = c_{ij} - (\pi'_j - \pi'_i) \begin{cases} \geqq \alpha & \text{then } f'_{ij} = \ell_{ij} \\ \\ \leqq -\alpha & \text{then } f'_{ij} = k_{ij} \end{cases} \tag{5.43}$$

for every $(i,j) \in \mathcal{A}$. Then for any arc $(p,q) \in \mathcal{A}$ if $|\overline{c}'_{pq}| \geqq n\alpha$, then $\overline{f}_{pq} = f'_{pq}$ in every minimum cost feasible circulation $\overline{f} = (\overline{f}_{ij})$ in G.

To prove this, let $\overline{f}$ be a minimum cost feasible circulation in G such that $\overline{f}_{pq} \neq f'_{pq}$ for an arc $(p, q) \in \mathcal{A}$ satisfying $|\overline{c}'_{pq}| \geqq n\alpha$. Suppose $\overline{f}_{pq} > f'_{pq}$ (a proof similar to the following can be constructed if $\overline{f}_{pq} < f'_{pq}$). If $\overline{c}'_{pq} \leqq -n\alpha$, then $f'_{pq} = k_{pq}$ and $\overline{f}_{pq} > k_{pq}$ cannot hold, so we must have $\overline{c}'_{pq} \geqq n\alpha$. $\overline{f} - f'$ is a nonzero circulation in G; by Lemma 5.2 there exists an oriented simple cycle $\mathbb{C}$ containing the arc (p, q) such that on all forward arcs (i, j) on $\mathbb{C}$ we have $\overline{f}_{ij} - f'_{ij} > 0$, and on all the reverse arcs (i, j) on $\mathbb{C}$ we have $\overline{f}_{ij} - f'_{ij} < 0$. Let $\epsilon =$ min. $\{ |\overline{f}_{ij} - f'_{ij}| : (i, j)$ on $\mathbb{C} \}$. So, $\epsilon > 0$. Obtain the circulation f'' from $\overline{f}$ by decreasing the flow amount on all the forward arcs of $\mathbb{C}$ by ϵ and by increasing the flow amount on all the reverse arcs of $\mathbb{C}$ by ϵ. f'' is easily verified to be a feasible circulation in G.

On the forward arc (p, q) on $\mathbb{C}$, $\overline{c}'_{pq} \geqq n\alpha$. On all the other forward arcs (i, j) on $\mathbb{C}$ other than (p, q) we have $k_{ij} \geqq \overline{f}_{ij} > f'_{ij}$, i.e., $f'_{ij} < k_{ij}$ which by (5.43) implies that $\overline{c}'_{pq} \geqq -\alpha$. On all the reverse arcs (i, j) on $\mathbb{C}$ we have $\ell_{ij} \leqq \overline{f}_{ij} < f'_{ij}$, i.e., $f'_{ij} > \ell_{ij}$, which by (5.43) implies $\overline{c}'_{pq} \leqq \alpha$. Let r be the number of arcs on $\mathbb{C}$, since $\mathbb{C}$ is simple $r \leqq n$. Cost of $\mathbb{C} = \overline{c}'_{pq} + \sum(\overline{c}'_{ij}$: over forward arcs $(i, j) \neq (p, q)$ on $\mathbb{C}) - \sum(\overline{c}'_{ij}$: over reverse arcs (i, j) on $\mathbb{C}) \geqq n\alpha - (r - 1)\alpha = (n + 1 - r)\alpha > \alpha > 0$. So cost of $f'' = cf'' = cf' - \epsilon($cost of $\mathbb{C}) < cf'$ since the cost of $\mathbb{C} > 0$ by the above. Since f'' is a feasible circulation, we have a contradiction to the hypothesis that f' is a minimum cost feasible circulation in G. Hence the statement given above must be true.

(iii) Let $|\mathcal{N}| = n$. If c is an integer vector, for every $0 \leqq \epsilon < 1/n$, every ϵ-optimal feasible circulation is a minimum cost feasible circulation.

To prove this, let f be an ϵ-optimal feasible circulation in G for some $0 \leqq \epsilon < 1/n$. Let $\mathbb{C}$ be a residual cycle wrt f. So, by definition, there exists a node price vector $\pi = (\pi_i)$ such that for any arc (i, j) on $\mathbb{C}$

$$\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i) \begin{cases} \geqq -\epsilon & \text{if } (i, j) \text{ is a forward arc on } \mathbb{C} \\ \leqq \ \ \epsilon & \text{if } (i, j) \text{ is a reverse arc on } \mathbb{C} \end{cases} \tag{5.44}$$

Since $\mathbb{C}$ is a simple cycle, it has at most n arcs. Hence by (5.44) the cost of $\mathbb{C} = \sum(\bar{c}_{ij}$: over forward arcs (i, j) on $\mathbb{C}) - \sum(\bar{c}_{ij}$: over reverse arcs (i, j) on $\mathbb{C}) \geqq -n\epsilon > -1$. However, as c is an integer vector, the cost of $\mathbb{C}$ is an integer, and since it is > -1, it must be $\geqq 0$. Hence every residual cycle wrt f has a nonnegative cost; by Theorem 5.1, f is a minimum cost feasible circulation.

(iv) Definitions Let f be a feasible circulation in G. Define $\epsilon(f)$ to be the minimum $\epsilon \geqq 0$ for which f is ϵ-optimal. Define $\mu(f)$ to be the mean cost of a minimum mean residual cycle wrt f.

(v) If f is an ϵ-optimal feasible circulation wrt a node price vector π (i.e., f, π together satisfy (5.42)) then on every arc (i, j) in the residual network $\text{G}(f, \pi) = (\mathcal{N}, \mathcal{A}(f), \bar{c}')$, the cost coefficient $\bar{c}'_{ij} \geqq -\epsilon$. This follows because if (i, j) is an arc in $\text{G}(f, \pi)$ with a + (−) label, then it corresponds to: (i, j) in G satisfying $f_{ij} < k_{ij}$ ((j, i) in G satisfying $f_{ji} > \ell_{ji}$), and by ϵ-optimality of f wrt π we must have $\bar{c}'_{ij} \geqq -\epsilon$.

(vi) Relationship Between $\epsilon(f), \mu(f)$ Let f be a feasible circulation in G. Then $\epsilon(f) = \max.\ \{0, -\mu(f)\}$.

To prove this, let π be a node price vector in G satisfying (5.42) with f for $\epsilon = \epsilon(f)$. Let $\mathbb{C}$ be any residual cycle wrt f, $c(\mathbb{C})$ its cost, and l the number of arcs in it. By the definition of $\epsilon(f)$, (5.42) holds for f, π and $\epsilon = \epsilon(f)$, and the cost of $\mathbb{C}$ wrt $c, \bar{c}$ as cost vectors is the same. So, $c(\mathbb{C}) \geqq -l\epsilon(f)$, or $c(\mathbb{C})/l \geqq -\epsilon(f)$, i.e., the mean cost of the residual cycle $\mathbb{C}$ is $\geqq -\epsilon(f)$. Since this holds for all residual cycles wrt f, we have $\mu(f) \geqq -\epsilon(f)$, or $\epsilon(f) \geqq -\mu(f)$.

If $\mu(f) \geqq 0$, by Theorem 5.1 f is a minimum cost circulation, hence $\epsilon(f) = 0 = \max.\ \{0, -\mu(f)\}$ in this case.

Now assume $\mu(f) < 0$. Every residual cycle in G wrt f corresponds to a simple circuit in $\text{G}(f)$ with the same cost, and vice versa, and $\mu(f)$ is the minimum circuit mean in $\text{G}(f) = (\mathcal{N}, \mathcal{A}(f), c')$. Let $\delta = -\mu(f) > 0$. Adding δ to the cost of every arc in $\text{G}(f)$ has the effect of adding δ to the mean cost of every circuit in it; hence after this change there will be no negative cost circuits in $\text{G}(f)$. Obtain a directed network H $= (\mathcal{N}^1, \mathcal{A}^1, c^1)$ where $\mathcal{N}^1 = \mathcal{N} \cup \{s\}$, s being an artificial origin node.

$\mathcal{A}^1 = \mathcal{A}(f) \cup \{(s,i) : i \in \mathcal{N}\}$. c^1, the cost vector on $\mathcal{A}^1$ is defined by making the cost of all arcs of the form (s,i) to be 0, and the cost of all arcs $(i,j) \in \mathcal{A}(f)$ to be $c'_{ij} + \delta$. So, H is a directed network with no negative cost circuits, and all nodes in H can be reached from s by a chain, since (s,i) is an arc in H for all $i \in \mathcal{N}$. Find a shortest chain tree rooted at s in H with c^1 as the arc cost vector, and let ν_i be the cost of the shortest chain from s to i in H for each $i \in \mathcal{N}$. From the optimality properties for shortest chains, we know that $\nu_j - \nu_i \leqq c'_{ij} + \delta$ for all $(i,j) \in \mathcal{A}(f)$.

If (i,j) is a + labeled arc in $\mathcal{A}(f)$ but there is no arc (j,i) in $\mathcal{A}(f)$, we must have $(i,j) \in \mathcal{A}$ and $f_{ij} = \ell_{ij} < k_{ij}$, and $\nu_j - \nu_i \leqq c'_{ij} + \delta = c_{ij} + \delta$, or $c_{ij} - (\nu_j - \nu_i) \geqq -\delta$.

If (i,j) is a - labeled arc in $\mathcal{A}(f)$ but there is no arc (j,i) in $\mathcal{A}(f)$, we must have $(j,i) \in \mathcal{A}$ and $f_{ji} = k_{ji} > \ell_{ji}$, and $\nu_j - \nu_i \leqq c'_{ij} + \delta = -c_{ji} + \delta$, so $c_{ji} - (\nu_i - \nu_j) \leqq \delta$.

If (i,j) with a + label, and (j,i) with a - label are both in $\mathcal{A}(f)$, we must have $(i,j) \in \mathcal{A}$, $\ell_{ij} < f_{ij} < k_{ij}$, and $\nu_j - \nu_i \leqq c'_{ij} + \delta$, and $\nu_j - \nu_i \leqq c'_{ji} + \delta$, which together imply $-\delta \leqq c_{ij} - (\nu_j - \nu_i) \leqq \delta$.

Hence on arcs $(i,j) \in \mathcal{A}$ satisfying $c_{ij} - (\nu_j - \nu_i) \geqq \delta$, we must have $f_{ij} = \ell_{ij}$; and on arcs $(i,j) \in \mathcal{A}$ satisfying $c_{ij} - (\nu_j - \nu_i) \leqq -\delta$, we must have $f_{ij} = k_{ij}$. This means that with ν as the node price vector f satisfies (5.42) for $\epsilon = \delta$, i.e., f is δ-optimal, or $(-\mu(f))$-optimal. So, $\epsilon(f) \leqq -\mu(f)$. We have already proved above that $\epsilon(f) \geqq -\mu(f)$. So, in this case $\epsilon(f) = -\mu(f) = \max.\ \{0, -\mu(f)\}$.

Hence, in general, $\epsilon(f) = \max.\ \{\ 0, -\mu(f)\ \}$.

(vii) Definition Admissible Network Let (f,π) be a feasible circulation, node price vector pair, and $\mathrm{G}(f,\pi) = (\mathcal{N}, \mathcal{A}(f), \overline{c}')$ the residual network wrt it. The *admissible network* wrt (f,π) denoted by $\mathrm{G}'(f,\pi)$ is the subnetwork of $\mathrm{G}(f,\pi)$ consisting of all the arcs (i,j) in it for which $\overline{c}'_{ij} < 0$.

(viii) Let (f,π) be an ϵ-optimal feasible circulation, node price vector pair in G satisfying (5.42). If the admissible network $\mathrm{G}'(f,\pi)$ is acyclic, then f is actually $(1 - 1/n)\epsilon$-optimal.

To prove this, let $\vec{\mathbb{C}}$ be a simple circuit in $G(f, \pi)$ consisting of l arcs, say. By (v), the cost coefficients of all the arcs in $G(f, \pi)$ are $\geqq -\epsilon$. However, since $G'(f, \pi)$ is acyclic by hypothesis, $\vec{\mathbb{C}}$ cannot completely lie in $G'(f, \pi)$, and hence at least one arc on $\vec{\mathbb{C}}$ has cost in $G(f, \pi) \geqq 0$. So the mean cost of $\vec{\mathbb{C}}$ is $\geqq (l-1)(-\epsilon)/l = -\epsilon + (\epsilon/n) \geqq -(1-1/n)\epsilon$. Thus the mean cost of every simple circuit in $G(f, \pi)$ is $\geqq -(1-1/n)\epsilon$. Thus, $\mu(f) \geqq -(1-1/n)\epsilon$. By (vi), $\epsilon(f) \leqq (1-1/n)\epsilon$, so f is $(1-1/n)\epsilon$-optimal.

(ix) Let f be a nonoptimal feasible circulation in G. Canceling a minimum mean cycle in f leads to a feasible circulation $\tilde{f}$ satisfying $\epsilon(\tilde{f}) \leqq \epsilon(f)$.

To prove this, let π be a node price vector wrt which f is ϵ-optimal for $\epsilon = \epsilon(f)$. Let $\mathbb{C}$ be the minimum mean cycle canceled, and $\vec{\mathbb{C}}$ the simple circuit in $G(f, \pi)$ corresponding to it. By (v), the cost coefficient of arc (i, j) on $G(f, \pi)$, $\bar{c}'_{ij} \geqq -\epsilon(f)$ for all arcs (i, j) on $\vec{\mathbb{C}}$. But $\vec{\mathbb{C}}$ is a minimum mean circuit in $G(f, \pi)$, whose cost is $-\epsilon(f)$ by (vi), which implies that $\bar{c}'_{ij} = -\epsilon(f)$ for all (i, j) on $\vec{\mathbb{C}}$. So, every new residual arc created after the cancellation (such an arc must be the reversal of an arc on $\vec{\mathbb{C}}$) has cost coefficient of $\epsilon(f)$ in $G(\tilde{f}, \pi)$. So, every arc (i, j) in $G(\tilde{f}, \pi)$ has cost coefficient $\geqq -\epsilon(f)$ too. Hence the minimum circuit mean in $G(\tilde{f}, \pi)$ is $\geqq -\epsilon(f)$. This implies that $\mu(\tilde{f}) \geqq -\epsilon(f)$. Hence $\epsilon(\tilde{f}) \leqq \epsilon(f)$ by (vi).

(x) Let f be a feasible circulation in G, and $\tilde{f}$ the feasible circulation obtained after m minimum mean cycle cancellations beginning with f. Then $\epsilon(\tilde{f}) \leqq (1-1/n)\epsilon(f)$.

The proof of this goes as follows. Let π be a node price vector wrt which f is ϵ-optimal for $\epsilon = \epsilon(f)$. The admissible network $G'(f, \pi)$ changes with every change in the flow vector. By (v) the cost coefficient of every arc in $G(f, \pi)$ is $\geqq -\epsilon(f)$. Canceling a cycle with all its arcs in the admissible network only adds arcs of positive reduced cost to the residual network and deletes at least one arc from the admissible network, as established in (ix). Consider two cases.

Case 1 Suppose none of the cycles canceled contains an arc (i,j) of nonnegative reduced cost. Then each cancellation deletes at least one arc from the admissible network. So, after m cancellations, the admissible network has no arcs, which implies that the flow vector at that time is optimal; hence $\epsilon(\tilde{f}) = 0$, establishing the result in this case.

Case 2 Suppose some cycle canceled contains an arc of nonnegative reduced cost. Let the first such cycle correspond to the circuit $\vec{\mathbb{C}}$ in the residual network, and let the flow vector be f^1 just before that cancellation. Using the same arguments as in the proof of (viii) we verify that the mean cost of $\vec{\mathbb{C}}$ is $\geqq -(1-1/n)\epsilon(f)$. Hence $\epsilon(f^1) \leqq (1-1/n)\epsilon(f)$ by (vi). By the argument in (ix) applied repeatedly, we have $\epsilon(\tilde{f}) \leqq \epsilon(f^1)$, and hence $\epsilon(\tilde{f}) \leqq (1-1/n)\epsilon(f)$ in this case too.

(xi) If the cost vector $c = (c_{ij})$ is an integer vector, let $\gamma =$ max. $\{\ |c_{ij}| : (i,j) \in \mathcal{A}\ \}$. Then this algorithm terminates after at most O($nm\log(n\gamma)$) cancellations.

To prove this, let f^0 denote the initial feasible circulation. Taking $\pi^0 = 0$ we verify that $\epsilon(f^0) \leqq \gamma$. Let f^r denote the feasible circulation obtained in the algorithm after r cancellations. By (iii) we know that if r is such that $\epsilon(f^r) < 1/n$, then f^r is a minimum cost circulation, and the algorithm can terminate. From (x) we know that

$$\epsilon(f^r) \leqq (1-1/n)^{\lfloor\frac{r-1}{m}\rfloor}\epsilon(f^0) \leqq (1-1/n)^{\lfloor\frac{r-1}{m}\rfloor}\gamma$$

and from the above argument it is sufficient to make this $< 1/n$. Hence the maximum number of cancellations required in the algorithm is r where r satisfies

$$(1-1/n)^{\lfloor\frac{r-1}{m}\rfloor} < 1/n\gamma \text{ i.e., } \lfloor\frac{r-1}{m}\rfloor \geqq \frac{-\log(n\gamma)}{\log(1-1/n)} \geqq n\log(n\gamma)$$

since $\log(1 - 1/n) \leqq -1/n$ for $n > 1$. Hence it is sufficient to take $r = \mathrm{O}(nm\log(n\gamma))$. This shows that the algorithm is polynomially bounded when the cost vector is integral.

(xii) Definition For given $\epsilon > 0$, define an arc $(p, q) \in \mathcal{A}$ to be ϵ-fixed iff the flow f_{pq} is the same for all ϵ-optimal feasible circulations $f = (f_{ij})$.

(xiii) Let $\epsilon > 0$ and (f, π) an ϵ-optimal feasible circulation, node price vector pair satisfying (5.42) together. $\overline{c}_{ij} = c_{ij} - (\pi_j - \pi_i)$ is the reduced cost coefficient of $(i, j) \in \mathcal{A}$ wrt π. If $(p, q) \in \mathcal{A}$ satisfies $|\overline{c}_{pq}| \geqq 2n\epsilon$, then (p, q) is ϵ-fixed.

This follows from (ii).

(xiv) From (ix) we know that $\epsilon(f)$ is nonincreasing as the algorithm progresses. So, at some stage of the algorithm, if an arc (p, q) becomes fixed, then the flow f_{pq} remains the same in the sequel. Therefore, when all arcs are fixed, the current circulation must be optimal (since an optimal circulation is ϵ-optimal for any $\epsilon \geqq 0$, and therefore must agree with the current circulation on all the fixed arcs).

(xv) For arbitrary real valued cost vector c, the algorithm terminates after at most $O(nm^{\mathbf{2}}\log\, n)$ cancellations.

To prove this, let $a = mn\lceil(1+logn)\rceil$. We divide the cancellations in the algorithm into groups of a consecutive cancellations each. We will now show that each group of cancellations fixes the flow on a distinct arc, i.e., the flow on that arc will not change in subsequent iterations.

Consider a group of cancellations. Let f^0 denote the circulation at the beginning of the first iteration in this group, and f^a the circulation at the end of the last iteration in this group. Let $\epsilon_0 = \epsilon(f^0), \epsilon_a = \epsilon(f^a)$, and let π^a be a node price vector wrt which f^a satisfies the ϵ_a-optimality criterion. Let $\mathbb{C}_0$ be the cycle canceled in the first iteration of the group. By the result in (x), and the choice of a, we have

$$\epsilon_a \leqq \epsilon_0(1 - 1/n)^{n\lceil(1+\log\, n)\rceil} \leqq \frac{\epsilon_0}{2n}$$

By (vi) the mean cost of $\mathbb{C}_0$ is $-\epsilon_0$. The mean cost of $\mathbb{C}_0$ remains the same even when the arc cost coefficients are the reduced cost coefficients $c_{ij} - (\pi_j^a - \pi_i^a)$. Since the mean cost of $\mathbb{C}_0$ is - ϵ_0, at least one arc (i, j) on $\mathbb{C}_0$ must have reduced cost $c_{ij} - (\pi_j^a - \pi_i^a) \leqq -\epsilon_0 \leqq -2n\epsilon_a$, and by the arguments in (ix), (xiii), (xiv), the flow on this arc will not change after this group of iterations is completed.

Thus each group of cancellations fixes at least one distinct arc. Hence the algorithm terminates after at most m groups, or $nm^{\mathbf{2}}\lceil(1+\log n)\rceil$ cancellations in all. Hence the total number of cancellations needed in the algorithm is at most $\mathrm{O}(nm^{\mathbf{2}}\log n)$.

Each cancellation involves $\mathrm{O}(nm)$ effort to find the cycle to cancel by Karp's algorithm discussed earlier, and $\mathrm{O}(m)$ for the cancellation itself, leading to a total effort of $\mathrm{O}(nm)$ per iteration. So, the overall computational effort in this algorithm is at most $\mathrm{O}(n^{\mathbf{2}}m^{\mathbf{3}}\log n)$, establishing the strong polynomiality of the algorithm.

Based on a more flexible selection of cycles to cancel, versions of this algorithm in which the effort per iteration is $\mathrm{O}(\log n)$ instead of $\mathrm{O}(nm)$ have been developed in Goldberg and Tarjan [1989].

There are several other strongly polynomial algorithms developed for minimum cost flow problems; see Fujishige [1986], Galil and Tardos [1986], Orlin [1984, 1988], and Tardos [1985]. Some of these algorithms are based on making modifications to a non-strongly-polynomial algorithm to convert it into a strongly polynomial algorithm. However, the kind of tricks needed to make an algorithm amenable to a strongly polynomial proof do not normally work well in practical computation. Hence the major impact of these algorithms is mathematical and theoretical.

The question of whether strongly polynomial algorithms exist for the general LP remains an open question. Again, it is one of the most mathematically challenging open questions in LP theory. Outside of the special class of pure minimum cost flow problems, results on this question are practically unknown. The class of generalized network flow problems discussed in Chapter 8 is slightly more general than the class of pure network flow problems. Even for that class no strongly polynomial algorithms are known at the moment.

5.9 Minimum Separable Piecewise Linear Convex Cost Flow Problems

Interval	Slope of $c_{ij}(f_{ij})$ in the interval
ℓ_{ij} to x^1_{ij}	c^1_{ij}
x^1_{ij} to x^2_{ij}	c^2_{ij}
$\vdots$	$\vdots$
x^{r-1}_{ij} to k_{ij}	c^r_{ij}

Consider a directed single commodity flow network G $= (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t}, \overline{v})$ in which the cost of flow on each arc $(i, j) \in \mathcal{A}$ is a piecewise linear convex function $c_{ij}(f_{ij})$, i.e., the interval ℓ_{ij} to k_{ij} is divided into a finite number, say r, of intervals, and in each interval $c_{ij}(f_{ij})$ is linear with slopes given above.

The break points and slopes satisfy the conditions $\ell_{ij} < x^1_{ij} < x^2_{ij} < \ldots < x^{r-1}_{ij} < k_{ij}$ and $c^1_{ij} < c^2_{ij} < \ldots < c^r_{ij}$; these are the conditions for the continuous piecewise linear function $c_{ij}(f_{ij})$ to be convex. The problem is to find a feasible flow vector which minimizes $\sum(c_{ij}(f_{ij})$: over $(i, j) \in \mathcal{A})$, given the break points and slopes associated with each arc.

One way of interpreting the objective function $c_{ij}(f_{ij})$ is the following. Each unit of flow on arc (i, j) between ℓ_{ij} and x^1_{ij} flows at a cost of c^1_{ij} per unit. Once the flow amount on arc (i, j) reaches x^1_{ij}, every additional unit flow on it pays a cost of c^2_{ij} until the flow amount reaches the value x^2_{ij}. This can be handled by treating this flow as if it were flowing through another parallel directed arc from i to j on which the cost coefficient is c^2_{ij} per unit flow. And so on. So, for the cost function given in the above tableau, we replace the original arc (i, j) with r parallel arcs with data as in Figure 5.20. These r parallel arcs correspond to the arc (i, j) in the original network. The flow f_{ij} in the original problem corresponds to the sum of the flow amounts on the parallel arcs corresponding to it in the transformed problem.

Each arc in the original network is transformed separately in the same way. The linear minimum cost flow problem in the transformed

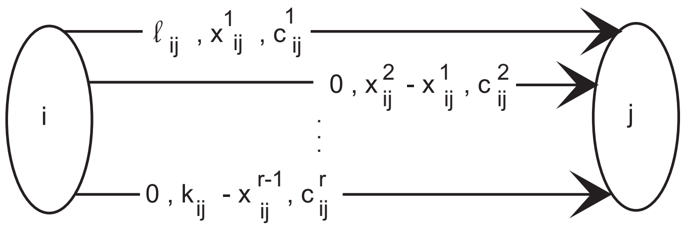


Figure 5.20: Data on the arcs is lower bound, capacity, and unit cost coefficient, in that order.

network is solved by any of the methods discussed earlier. If that problem is infeasible, or unbounded, the same holds for the original piecewise-linear minimum cost flow problem. Otherwise, the flow vector in the original network, corresponding to an optimum solution in the transformed network, is optimal for the original problem.

Suppose the parallel arcs in the transformed network corresponding to the arc (i, j) in the original network, are $(i, j)_1, \ldots, (i, j)_r$ in increasing order of the interval of the original flow variable f_{ij} corresponding to them. Then, because of the monotone increasing property of the slopes, the following property will hold in any optimum flow for the transformed problem

$$\text{flow on } (i, j)_t \text{ is } 0 \text{ unless } (i, j)_1, \ldots, (i, j)_{t-1} \text{ are all saturated} \quad (5.45)$$

It is this property which makes the original piecewise linear cost minimization problem equivalent to the linear cost minimization problem in the transformed network.

For any arc (i, j) in the original problem, if $c_{ij}(f_{ij})$ is piecewise linear and continuous, but not convex, then the slopes will not be monotone increasing. In this case, in a minimum cost flow in the transformed network (5.45) may not hold. The original problem is equivalent to the linear minimum cost flow problem in the transformed network, subject to conditions of the form (5.45) for each arc (i, j) ((5.45) holds automatically if the original objective function is convex because of the monotonicity of the slopes). The constraints (5.45) are not linear constraints; finding a minimum cost flow problem subject to them in the transformed network is not an LP. This can of course be transformed into an integer programming problem, or solved directly by enumerative methods. Thus the transformation of the original piecewise linear minimum cost flow problem into the linear minimum cost flow problem on the transformed network is only valid if the piecewise linear function is convex.

5.10 Dynamic Network Flow Problems

In all the flow problems discussed so far, we ignored any consideration of traversal time, and there was at most one flow amount entering each arc; that's why flow vectors in those problems are referred to as **static flows**. In this section we consider flow models called **dynamic flow models**; here the traversal time for each arc in the network is provided as input, and a separate flow amount can enter each arc at its tail node at every integer point of time, even while there may be other units of material that entered this arc earlier still in transit on it. One may either impose a requirement that all the units of the commodity must make the trip from the source to the sink within a given amount of time, or want to determine the maximum amount of the commodity that can reach the sink from the source in a specified amount of time. In dynamic flow models, the transit of each unit of flow has to be completely organized along the time axis. We have to specify when it would enter each arc along the chain that it travels, whether it will be temporarily held over or stored at certain nodes for one or more time periods before it can resume its journey across the next arc, and so forth. Thus, time delays are associated with both arcs (because of transit time across the arc) and nodes (because of any holdover or storage) in this model.

Static problems are those in which a single optimum solution is desired. In dynamic problems we need the optimum solution for each point of time over a time horizon.

Dynamic flow problems occur in many areas. Examples are communication systems and traffic and railway systems. In these systems, the networks used are called *store and forward networks*. Other areas in which these problems arise are production systems, material handling systems, military logistics systems etc. In modern automatic production systems, material flows from one process to the next through conveyor belts or is transshipped automatically by robots. In these systems there may be buffer zones where material may be held over or stored before it is transshipped to the next process, and different processes may have different cycle times. To analyze the flow of material through such a dynamic system one discretizes time by dividing the time horizon into equal intervals called **time slices**; and approximates the processing time of each process, and the transit time between processes by integer multiples of the common discrete time unit (the length of a time slice). The material flow can then be modeled using a dynamic network flow model.

Let G $= (\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$ be the network on which the problem is defined, with $|\mathcal{N}| = n$, $|\mathcal{A}| = m$. Here $k = (k_{ij})$ where k_{ij} is the maximum number of units of the commodity that can enter arc (i, j) at its tail node i at the beginning of each time period. We are also given the vector $a = (a_{ij} : (i, j) \in \mathcal{A})$, where a_{ij} is the traversal time across arc (i, j). We assume $k > 0, a \geqq 0$ and that they are both integer vectors. We assume that all the action begins at time point 0, and that material can enter any arc at its tail only at integer points of time in amounts limited by the arc's capacity, and since all transit times are assumed integer, the material will always finish its travel across an arc reach the head node of that arc at an integer point of time.

For each $i \in \mathcal{N}$, we define $a_{ii} = 1$; this represents the fact that material stored or held over at a node is available for change of status (either continue to be stored again, or begin journey across an arc incident out of that node) at the beginning of the next time period.

For each $i \in \mathcal{N}$, k_{ii} denotes the maximum amount of material that can be held over or stored at node i. Let T, a positive integer, denote the time horizon in the problem. It is the maximum number of time periods for which the program has to be worked out. The decision variables in the problem are clearly: for each $(i, j) \in \mathcal{A}$ and each $\tau =$ 0, 1, $\ldots, T$

$f(i,j,\tau)$ = amount of material entering (i,j) at time point τ

$f(i,i,\tau)$ = amount of material held over at i from time point τ to $\tau+1$

The vector $(f(i,j,\tau))$ is known as a **dynamic flow** on the network G. Flow units in it travel across G over time, obeying node and arc capacity constraints and conservation at each node and at each point of time. The node-arc flow vectors discussed earlier, which do not have any concept of time, will be referred to as **static flows** for the sake of distinction.

The constraints that the dynamic flows have to satisfy for feasibility appear more complex than those for static flows. However, in actuality, the T-period dynamic flow on G can be viewed as a static flow on a **time-expanded** version G(T) of G. The procedure for constructing G(T) from G is the following:

1. For each node i in G, there are $T+1$ nodes denoted by $i(\tau)$, $\tau = 0, 1, \ldots, T$ in G(T), and arcs $(i(\tau), i(\tau+1))$, $0 \leqq \tau \leqq T-1$ with capacity k_{ii} to represent holdover at node i.

2. For each arc (i,j) in G, there are arcs $(i(\tau), j(\tau+a_{ij})), \tau = 0, \ldots, T-a_{ij}$ with capacity k_{ij} in G(T).

Associate the flow variable $f(i,j,\tau)$ in the dynamic flow, with the arc $(i(\tau), j(\tau+a_{ij}))$, and the flow variable $f(i(\tau), i(\tau+1), \tau)$ with the arc $(i(\tau), i(\tau+1))$, in G(T). Treat the nodes $\breve{s}(0), \ldots, \breve{s}(T)$ as the source nodes, and the nodes $\breve{t}(0), \ldots, \breve{t}(T)$ as the sink nodes in G(T) (in fact, since there are hold-over arcs at $\breve{s}$ and $\breve{t}$, we could equally well treat $\breve{s}(0)$ as the only source node, and $\breve{t}(T)$ as the only sink node in G(T).

With these associations, it can be verified that the constraints for the feasibility of the dynamic flow vector in G become the usual flow conservation and capacity constraints for a static node-arc flow vector in the time expanded network G(T). So, feasible static node-arc flow vectors in G(T) become feasible dynamic flows in G with a complete specification of how much material enters each arc or passes through each node at each point of time, and vice versa.

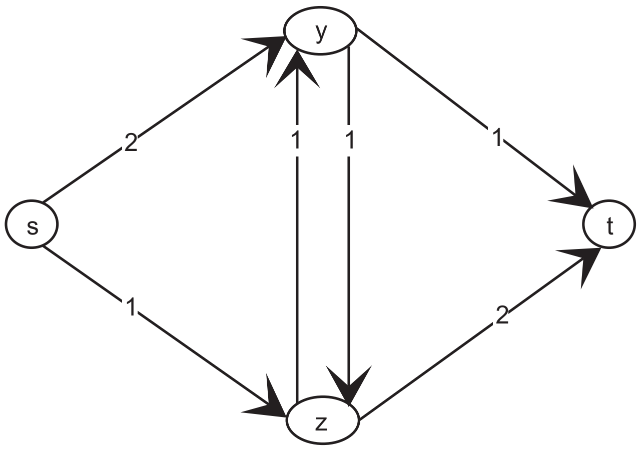


Figure 5.21: The network G, with arc traversal time data.

As an example in Figure 5.21 we show a network G consisting of nodes s, t, y, z with traversal time data entered on the arcs. Other data such as capacities are not shown. In Figure 5.22 we show G(5), the 5-period time-expanded version of G. In drawing the time-expanded version in Figure 5.22, we represented time intervals horizontally, and the nodes of the original network vertically at each time point.

Since an item released from a node at a specific time does not return to that location at the same or an earlier time, $\mathrm{G}(T)$ cannot contain any circuits, and is therefore acyclic always. The number of nodes in $\mathrm{G}(T)$ is $n(T+1)$, and the number of arcs is $(n+m)T + m - \sum(a_{ij}$: over $(i,j) \in \mathcal{A})$. In the time-expanded network there are no parallel arcs and it is usually sparse, with its density decreasing as T increases.

A dynamic flow problem is said to be **stationary** if the network parameters such as capacities, arc traversal times, and so on, are constant over time.

Ford and Fulkerson [1962 of Chapter 1] introduced a remarkably simple and effective algorithm for the stationary maximum value dynamic flow problem, i.e., one in which the objective is to maximize the amount of material reaching the sink node. It does not require the construction of the time-expanded network $\mathrm{G}(T)$ for solving this

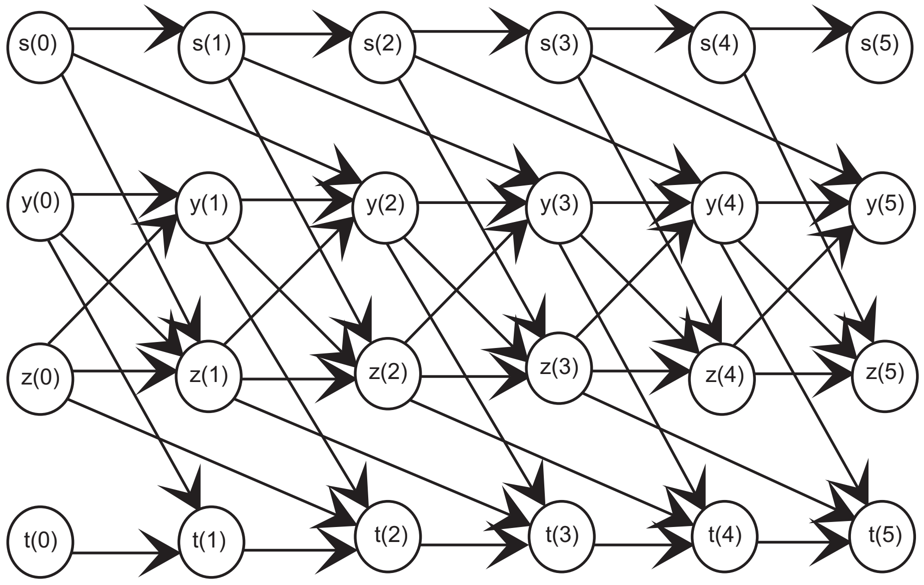


Figure 5.22: The time-expanded network G(5).

problem for any T!. They show that a maximum value dynamic flow in the stationary case can be generated from a static flow f in the smaller network G that maximizes the linear function $w(f, v) = (T + 1)v - \sum(a_{ij}f_{ij} : \text{over } (i, j) \in \mathcal{A})$, where v is the value of f in G. Decompose f into a set of arc-chain flows from $\breve{s}$ to $\check{t}$ in G. Start each chain flow in this set at time 0 and repeat it after each time period so long as there is enough time left in the horizon for the flow along the chain to arrive at the sink. This leads to a maximum value dynamic flow in this case. We do not discuss the proof of validity of this algorithm here since it only handles a special problem; the interested reader should consult Ford and Fulkerson's book.

In the most general dynamic model, the arc capacities k_{ij} and traversal time a_{ij} may not be constant over time, but may vary. Surprisingly, the technique of time expansion provides a general way of reducing the complicated dynamic situation into the familiar static one in the space-time framework by replicating the physical network over the time domain. But we do pay a price for this transformation, as the size of the time-expanded network blows up linearly with the length of the time horizon studied. If T is large, even a relatively modest network G expanded through T periods can lead to an enormous time-expanded network G(T) which may be computationally very expensive to deal with. Thus, even though the technique of time-expansion is conceptually very nice and simple, the blowup in size makes it unappealing, particularly when the time horizon is large. Unfortunately, efficient techniques such as the one mentioned earlier for the stationary dynamic maximum value flow problem are not known at the moment for other dynamic network flow problems. But research continues with the aim of devising efficient techniques for finding at least approximate solutions.

5.11 Multicommodity Flow Problems

In this section we discuss static multicommodity flow problems, i.e., those involving p ($\geqq 2$) commodities on a directed connected pure network G $= (\mathcal{N}, \mathcal{A}, 0, k)$ with $0 < k < \infty$. To model these problems, we should measure quantities of all commodities in common units, for example, truck loads. The capacities of the arcs are also stated in the same units. Under this, it is possible to add the flows of different commodities on an arc to yield the total flow of all commodities put together on that arc, and we assume that the capacity of the arc applies to this total flow.

By introducing a supersource and supersink for each commodity if necessary, as discussed in Chapter 2, it is possible to formulate the multicommodity flow problem in such a way that each commodity originates at some specified source, and is required to be transported to a specified sink node. Let $\breve{s}^r, \breve{t}^r$ be the source and sink respectively for the rth commodity, $r = 1$ to p. In the node-arc flow model flow conservation must hold at each node for each commodity separately.

We denote the node-arc flow amount of the rth commodity on arc (i, j) by f^r_{ij}, and $f^r = (f^r_{ij} : (i, j) \in \mathcal{A})$ is the flow vector of this commodity, $r = 1$ to p. Let $c^r = (c^r_{ij} : (i, j) \in \mathcal{A})$ be the original cost vector for the rth commodity, where c^r_{ij} is the cost of shipping one unit of this commodity across $(i, j) \in \mathcal{A}$, $r = 1$ to p. Let E be the node-arc incidence matrix of G. Suppose a specified flow value of $\overline{v}^r$ units of the rth commodity is required to be transported from $\breve{s}^r$ to $\breve{t}^r$, $r = 1$ to p. This leads to the following **multicommodity minimum cost flow problem**.

$$\begin{array}{cccc|l}
f^1 & f^2 & \dots & f^p & \\ \hline
I & I & \dots & I & \leqq k \\
\multicolumn{4}{c|}{\dots\dots\dots\dots\dots} & \dots\dots \\
E & 0 & \dots & 0 & = q^1\overline{v}_1 \\
0 & E & \dots & 0 & = q^2\overline{v}_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \dots & E & = q^p\overline{v}_p \\ \hline
\multicolumn{4}{c}{f^r \geqq 0 \text{ for all } r} &
\end{array}$$

where q^r is a column vector of the hypothetical arc $(\breve{s}_r, \breve{t}_r)$. The constraint at the top representing the arc capacities for the sum of the flows of all the commodities is called the **bundle constraint**. Without the bundle constraint the problem breaks down into several single commodity flow problems, one for each commodity, and can be solved efficiently by the algorithms discussed earlier. The bundle constraint makes the problem difficult, the main difficulty being that of optimally splitting the capacity of each arc among the various commodities. If $|\mathcal{N}| = n, |\mathcal{A}| = m$, this problem is an LP in pm variables, subject to $m + p(n - 1)$ constraints. In applications, the networks tend to be large, $n = 1000$, $m = 10{,}000$ is very common even in small scale applications. Even if there are only 3 commodities to be considered on such a network, the problem is an LP in 30,000 variables subject to 12,997 constraints, a large scale LP.

Applications for multicommodity flow models are many; most of these tend to be large-scale problems. Its wide applicability has made the multicommodity flow problem a focus of intense research activity by groups working in large scale LP.

The coefficient matrix in this problem is in general not totally unimodular, and hence it may not have an optimum integer solution even if all the data are integral.

One of the earliest approaches proposed for solving multicommodity flow problems based on network methodology is by Ford and Fulkerson [1958] using an arc-chain formulation of the problem. Inspired by it, Dantzig and Wolfe [1960] later extended it into the decomposition principle for general LP. The arc-chain formulation leads to an LP involving only $m+p$ constraints but an enormously large number (growing exponentially with n) of variables. The remarkable thing about this approach is that it is able to process this problem without having the data for all the variables on hand at any point of time. Using the revised simplex format, the method solves the problem by maintaining the data corresponding to only $m+p$ variables in the problem at any time, and in each step it generates the data corresponding to exactly one additional variable as needed. Hence the approach is called a **column generation approach**. The column generation approach is of fundamental importance in large scale optimization and combinatorial optimization.

$$\begin{array}{lll}
\text{Minimize} & \displaystyle\sum_{r=1}^{p}\sum_{h=1}^{d_r} g_h^r x_h^r & \\
\text{subject to} & \displaystyle\sum_{r=1}^{p}(\sum(x_h^r & : \text{ over } h \text{ s. t. } \mathcal{C}_h^r \text{ contains } e_u)) + \\
 & & y_u = k_u, u = 1 \text{ to } m \qquad\qquad (5.46)\\
 & \displaystyle\sum_{h=1}^{d_r} x_h^r & = \overline{v}_r, r = 1 \text{ to } p \\
 & \text{all } x_h^r & , y_u \geqq 0
\end{array}$$

In the parlance of the decomposition principle of LP, we decompose along the dotted line in the original problem. This leads to the arc-chain formulation of the problem. Denote the arcs in $\mathcal{A}$ by $e_1, \ldots, e_m$; and their capacities by $k_1, \ldots, k_m$. For $r = 1$ to p, let d_r be the total number of distinct simple chains from $\breve{s}_r$ to $\breve{t}_r$ in G (typically this is likely to be a very large number); let these chains themselves be $\mathcal{C}_1^r, \ldots, \mathcal{C}_{d_r}^r$. Let x_h^r be the amount of the rth commodity shipped along $\mathcal{C}_h^r$, $h = 1$ to d_r, $r = 1$ to p; these are the variables in the arc-chain formulation. Let $g_h^r = \sum(c_u^r$: over u s.t. e_u is on $\mathcal{C}_h^r) =$ cost of the chain $\mathcal{C}_h^r$ for the rth commodity. Then the arc-chain formulation of this minimum cost flow problem is (5.46) given above.

The number of constraints in (5.46) is $m+p$, and hence basic vectors for it consist of $m+p$ variables. To get in initial feasible basic vector, we introduce the artificial variables $y_{m+1}, \ldots, y_{m+p}$ and construct the following Phase I problem.

$$\begin{array}{rll} \text{Minimize } w = \sum\limits_{r=1}^{p} y_{m+r} & & \\ \text{subject to } \sum\limits_{r=1}^{p}(\sum(x_h^r & : \text{ over } h \text{ s. t. } \mathcal{C}_h^r \text{ contains } e_u)) + & \\ & y_u = k_u, u = 1 \text{ to } m & (5.47) \\ \sum\limits_{h=1}^{d_r} x_h^r & +y_{m+r} = \overline{v}_r, r = 1 \text{ to } p & \\ \text{all } x_h^r & , y_u \geqq 0 & \end{array}$$

$y = (y_1, \ldots, y_{m+p})$ is an initial feasible basic vector for (5.47) with which Phase I is initiated. The original column vector of y_t in (5.47) is the tth column vector of I, the unit matrix of order $m+p$, for $t = 1$ to $m+p$. The original column vector of the arc-chain flow variable x_h^r associated with the simple chain $\mathcal{C}_h^r$ in (5.47) is $(a_{1h}, \ldots, a_{m+p,h})^T$, where $a_{ih} = 1$ if the arc e_i is on $\mathcal{C}_h^r$, 0 otherwise, for $i = 1$ to m; and $a_{m+t,h} = 1$ if $t = r$, 0 otherwise, for $t = 1$ to p.

If $(\sigma_1, \ldots, \sigma_{m+p})$ is the Phase I dual basic solution corresponding to a feasible basic vector for (5.47), the Phase I relative cost coefficient wrt this basic vector, of y_u is $-\sigma_u$, and that of x_h^r is $-\sigma_{m+r} - \sum(\sigma_u :$ over u s. t. e_u is on $\mathcal{C}_h^r)$.

Thus to check whether the present basic vector satisfies the Phase I termination criterion in the simplex method, we need to check whether for each $r = 1$ to p

$$-\sigma_{m+r} - \sum(\sigma_u : \text{ over } u \text{ s. t. } e_u \text{ is on } \mathcal{C}_h^r\ ) \geqq 0 \qquad (5.48)$$

i.e., for all $h = 1$ to d_r, the length of every simple chain from $\breve{s}_r$ to $\breve{t}_r$ in G with $(-\sigma_1, \ldots, -\sigma_m)$ as the vector of arc lengths, is $\geqq \sigma_{m+r}$. Even though d_r is very large, this can be carried out efficiently by finding the shortest chain from $\breve{s}_r$ to $\breve{t}_r$. Hence, even though there are a lot of variables in (5.47), we can check whether any feasible basic vector for it satisfies the primal simplex termination criterion, or otherwise select an entering variable to carry out a primal simplex pivot step, using the following procedure. First check whether $-\sigma_u < 0$ for some u between 1 to m; if so, select y_u as the entering variable into the present basic vector. If $-\sigma_u \geqq 0$ for all $u = 1$ to m, find the shortest chains from $\breve{s}_r$ to $\breve{t}_r$ in G with $(-\sigma_1, \dots, -\sigma_m)$ as the vector of arc lengths, for $r = 1$ to p in some order. Since $(-\sigma_1, \dots, -\sigma_m) \geqq 0$ at this stage, Dijkstra's algorithm can be used to solve these shortest chain problems. If the length of the shortest chain from $\breve{s}_r$ to $\breve{t}_r$ is $\geqq \sigma_{m+r}$ for all $r = 1$ to p, then Phase I termination criterion is satisfied by the present basic vector; terminate Phase I. If the present Phase I objective value $w > 0$, the original problem is infeasible. If $w = 0$, go to Phase II with the present basic vector. If, for some r, the length of the shortest chain from $\breve{s}_r$ to $\breve{t}_r$, $\mathcal{C}^r_{h0}$ say, is $< \sigma_{m+r}$, associate the variable x_{h_0} with $\mathcal{C}^r_{h0}$ and choose $x^r_{h_0}$ as the entering variable into the present basic vector for a Phase I primal simplex pivot step, and continue.

In Phase II the procedure is very similar. Let $(\gamma_1, \dots, \gamma_{m+p})$ be the Phase II dual basic solution corresponding to a feasible basic vector for (5.47). The Phase II relative cost coefficient wrt the present basic vector, of y_u is $-\gamma_u$, for $u = 1$ to m, and of x^r_h is $g^r_h - \gamma_{m+r} - \sum(\gamma_u :$ over u s. t. e_u is on $\mathcal{C}^r_h)$. Hence, if $-\gamma_u < 0$ for some u between 1 to m, select y_u as the entering variable into the present basic vector for a primal simplex pivot step. If $-\gamma_u \geqq 0$ for all $u = 1$ to m; for $r = 1$ to p compute a shortest chain from $\breve{s}_r$ to $\breve{t}_r$ in G with $(-\gamma_1, \dots, -\gamma_m)$ as the vector of arc lengths using Dijkstra's algorithm. If for some r, $\mathcal{C}^r_{h_1}$ is the shortest chain obtained in this process and its length is $< \gamma_{m+r} - g^r_{h_1}$, select the arc-chain flow variable $x^r_{h_1}$ associated with it as the entering variable into the present basic vector for a primal simplex pivot step. Otherwise, if the length of the shortest chain $\mathcal{C}^r_{h_1}$ from $\breve{s}_r$ to $\breve{t}_r$ obtained in this process is $\geqq \gamma_{m+r} - g^r_{h_1}$ for all $r = 1$ to p, then the Phase II optimality criterion is satisfied and the present BFS is an optimum solution to the problem.

In each step we need to solve at most p shortest chain problems, which takes $O(pn^2)$ effort by Dijkstra's method. Once an entering variable is selected, if its original column is d, we need to find its updated column $\overline{d}$, by solving the system of equations $B\overline{d} = d$, where B is the present basis consisting of the original columns of the present basic variables, for carrying out the primal simplex pivot step. B is of order $(m+p) \times (m+p)$. If m is large (say, $\geqq 10,000$) unless techniques that can take advantage of its special structure (it is a 0-1 matrix, and is usually very sparse) are used, finding the updated column of the entering variable itself could be a big computational burden.

There are several other approaches proposed for solving multicommodity flow problems. At the moment, the most practically useful algorithms for these problems seem to be interior point methods for LP implemented on parallel processing supercomputers taking advantage of the special nature of the coefficient matrix (all entries being 0, ± 1) and its sparsity. See Adler, Resende, Veiga and Karmarkar [1989], and Kapoor and Vaidya [1985, 1988].

5.12 Exercises

5.15 $G = (\mathcal{N}, \mathcal{A})$ is a connected network. If $\mathbb{T}_1$ and $\mathbb{T}_2$ are any two spanning trees in G, and e_1 is a line in $\mathbb{T}_1$ not in $\mathbb{T}_2$, then show that there exists a line e_2 in $\mathbb{T}_2$ not in $\mathbb{T}_1$ such that replacing e_1 by e_2 in $\mathbb{T}_1$ leads to another spanning tree.

5.16 Consider the assignment problem, (3.1), of order n. Let $\bar{x}$ be the assignment
$\{(1, a_1), \dots, (n, a_n)\}$ where $(a_1, \dots, a_n)$ is a permutation of $(1, \dots, n)$. Let $G = (\mathcal{N}_1, \mathcal{N}_2; \mathcal{A})$ be the bipartite network where $\mathcal{N}_1 = \{1, \dots, n\}, \mathcal{N}_2 = \{1, \dots, n\}, \mathcal{A} = \mathcal{N}_1 \times \mathcal{N}_2$. Let $(\mathbb{T}, L)$ be a partition in G corresponding to the assignment $\bar{x}$. $\mathbb{T}$ has $2n - 1$ arcs of G, and L has $n^2 - 2n + 1 = (n-1)^2$ nonbasic arcs. $\bar{x}$ is a degenerate BFS of (3.1), $\mathbb{T}$ consists of the arcs $(i, a_i), i = 1$ to n, and $n - 1$ other arcs corresponding to basic arcs with zero flow. Given the partition $(\mathbb{T}, L)$, we are interested in counting the number of possible pivots in this partition which are nondegenerate, while attempting to enter each of the $(n-1)^2$ nonbasic arcs

in L one at a time. In particular, we are interested in determining the maximum and minimum number of potential nondegenerate pivots.

(i) Of all the alternate partitions representing $\bar{x}$, prove that the maximum number of nondegenerate pivots admitted by any such partition is $\frac{1}{2}n(n-1)$. Show that this maximum number is achieved by the partition $(\mathbb{T}, L)$ only if $\mathbb{T}$ is a path. One such partition is obtained when $\mathbb{T}$ consists of the arcs $\{(i, a_i) : i = 1$ to $n\} \cup \{(i, a_{i+1}) : i =1$ to $n-1\}$.

(ii) Of all the alternate partitions representing $\bar{x}$, prove that the minimum number of nondegenerate pivots admitted by any such partition is $(n-1)$. Show that this minimum number is achieved by the partition $(\mathbb{T}, L)$ where $\mathbb{T}$ consists of the arcs $\{(i, a_i) : i = 1$ to $n\} \cup \{(i, a_1) : i = 2$ to $n\}$.

(Bazaraa and Sherali [1982])

5.17 Determining single commodity equilibrium trade flow. This problem deals with a simple equilibrium model of interregional trade in a single commodity. We have the following data: N = number of regions, $a_i > 0$ is the equilibrium price in the ith region in the absence of imports and exports, $b_i > 0$ is the elasticity of supply and demand in the ith region, c_{ij} is the cost per unit shipped from region i to region j. The c_{ij} obey triangle inequality, that is, $c_{ij} \leq c_{ih} + c_{hj}$ for all i, j, h. The decision variables in the problem are: p_i = equilibrium price in the ith region, y_i = net imports into the ith region, x_{ij} = actual exports from region i to region j.

If $p_i > a_i$ then supply locally exceeds demand in the ith region, the difference being available for export. y_i is not restricted in sign; negative values of y_i are interpreted as exports. The following linear relation holds.

$$p_i = a_i - b_i y_i \tag{5.49}$$

Interregional trade equilibrium conditions are

$$\begin{aligned} p_i + c_{ij} &\geqq p_j, \quad \text{for all } i, j && (5.50)\\ (p_i + c_{ij} - p_j)x_{ij} &= 0, \quad \text{for all } i, j && (5.51) \end{aligned}$$

If (5.50) does not hold, exports from i to j will increase until the elasticity effects in markets i and j rise, and prices will adjust so that additional profit for export no longer exists. If $x_{ij} > 0$, we must have $p_i + c_{ij} = p_j$; hence the complementary slackness conditions (5.51) hold. The y_i and x_{ij} are linked through the flow conservation equations.

$$y_i - \sum_{j=1}^{N} x_{ji} + \sum_{j=1}^{N} x_{ij} = 0, i = 1 \text{ to } N. \qquad (5.52)$$

The problem is to develop a procedure for computing the equilibrium prices p_i and flows x_{ij}, given the data. Consider the quadratic program (QP):

$$\begin{aligned} \text{maximize } z = \sum_{i=1}^{N}(\quad a_i y_i \quad & -\frac{1}{2} b_i y_i^2 - \sum_{j=1}^{N} c_{ij} x_{ij}) && (5.53)\\ \text{subject to} \quad x_{ij} \quad & \geqq 0, \text{ and } (5.52) \end{aligned}$$

The objective function z in (5.53) can be interpreted as a net social payoff function. Show that the KKT optimality conditions for (5.53) are the equilibrium conditions (5.49) to (5.52). The dual of the QP (5.53) is the following problem, and an optimum λ for it is an equilibrium price vector p.

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{j=1}^{N} \frac{(a_i - \lambda_i)^2}{2b_i} && (5.54)\\ \text{subject to } \lambda_j - \quad \lambda_i \quad & \leqq c_{ij}, \text{ for all } i, j \end{aligned}$$

(i) Prove that there exists an equilibrium solution in which the trade routes of positive flows from a forest.

(ii) Using (i) show that there exists an equilibrium solution in which the set of regions is partitioned into trading coalitions. The members of each coalition trade only with each other and the set of trade routes with positive flows within each coalition forms a spanning tree for the coalition. Formally, a coalition is a set $\mathbf{C}$ of r nodes and $r-1$ trade routes with positive flows among them having the following properties

(a) Internal equilibrium: (5.49) to (5.52) are satisfied for all nodes and routes within $\mathbf{C}$, with no flows between the set $\mathbf{C}$ and regions outside of $\mathbf{C}$.

(b) Tree structure: The set of trade routes with positive flows inside $\mathbf{C}$ form a spanning tree for $\mathbf{C}$.

Prove that any coalition satisfying (a) and (b) has an equilibrium solution in which the following property holds

(c) Alternating arc orientation: A coalition $\mathbf{C}$ with $|\mathbf{C}| \geqq 2$ satisfying (a) and (b) has an equilibrium solution in which each node is either an exporter or an importer, i.e., no transshipment occurs.

(iii) Develop a tree growing algorithm for finding an equilibrium solution, which builds up a set of coalitions that are, at termination, in equilibrium with each other as well as being in internal equilibrium.

(Glassey [1978]).

5.18 The Flow Circulation Sharing Problem Consider a region in which there are p power plants dependent on coal supplies. Over a horizon of w weeks, let d_{ij} = normal amount of coal used by ith plant in jth week. Suppose there is a prolonged strike at some coal mines in the region during the horizon. Let: y_{ij} = amount of coal assigned to ith plant in jth week, $x_{ij} = \sum_{t=1}^{j} y_{it}$ = cumulative amount of coal shipped to plant i through week j, $D_{ij} = \sum_{t=1}^{j} d_{ij}$ = normal demand from plant i through week j. An equitable distribution scheme for coal deliveries in this period of shortage should try to

$$\text{maximize } \{\text{minimum } \{\frac{x_{ij}}{D_{ij}} : \ i = 1 \text{ to } p, j = 1 \text{ to } w\}\} \qquad (5.55)$$

Let k_j = total amount of coal available for distribution in week j for all the plants, $j = 1$ to w. Now the problem of finding equitable sharing of coal among the plants can be posed as that of finding a circulation in a directed network, to minimize an objective function of the form in (5.55). For example, the network for such a circulation problem when $p = 2, w = 2$ is given in Figure 5.23.

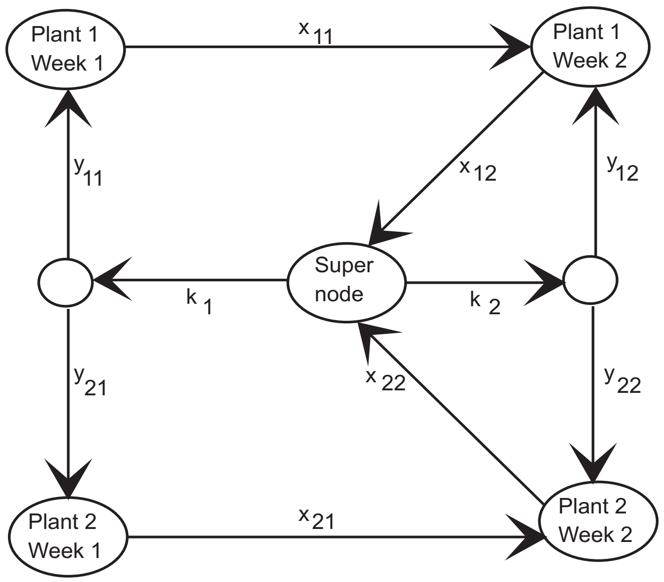


Figure 5.23:

Formulate the general problem in the following form: given a single commodity directed network G = $(\mathcal{N}, \mathcal{A}, \ell, k)$ and a subset of arcs $\mathbf{A} \subset \mathcal{A}$ with a positive constant D_{ij} for each arc $(i, j) \in \mathbf{A}$, among feasible circulations $f = (f_{ij})$ in G find one that maximizes minimum $\{f_{ij}/D_{ij} : \ (i, j) \in \mathbf{A}\}$. Develop a network flow algorithm for this problem (Brown [1983]).

5.19 The "More-for-Less" or "More-for-Nothing" Paradoxes in the Distribution Model Consider the $m \times n$ balanced transportation problem (5.56) with supply vector $a = (a_1, \dots, a_m) > 0$, demand vector $b = (b_1, \dots, b_n) > 0$, and unit cost matrix $c = (c_{ij}) \geqq 0$.

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \quad x_{ij} \\ \text{subject to } & \sum_{j=1}^{n} \quad x_{ij} = a_i, \ i = 1 \text{ to } m \\ & \sum_{i=1}^{m} \quad x_{ij} = b_j, \ j = 1 \text{ to } n \\ & \quad x_{ij} \geqq 0, \ \text{for all } i, j \end{aligned} \tag{5.56}$$

As usual, we assume that $\sum a_i = \sum b_j$, and denote the optimum objective value in this problem by $g(a,b)$, as a function of the supply-demand vectors a, b, while c is fixed. It is sometimes possible to find vectors $a' = (a'_1, \dots, a'_m) \geq a$, $b' = (b'_1, \dots, b'_n) \geq b$ satisfying $\sum a'_i = \sum b'_j > \sum a_i$, such that $g(a', b') \leq g(a, b)$. If $g(a', b') < g(a, b)$ we are able to ship more total goods for less total cost (even though all $c_{ij} \geqq 0$, and we ship the same amount or more from each source, and to each destination, than in the original problem (5.56)). Hence this is known as the **more for less paradox**. If $g(a', b') = g(a, b)$ under the same conditions, it is called the **more for nothing paradox**. As an example, consider the problem with the following data.

		cost of shipping ($/unit)				supply
to destination $j =$		1	2	3	4	a_i
from source $i =$	1	1	6	3	5	20
	2	7	3	1	6	10
	3	9	4	5	4	25
demand	b_j	11	13	17	14	

(a) In this numerical example, verify that the optimum objective value strictly decreases if a_2 and b_1 are increased by the same positive quantity δ up to 9.

(b) Assuming that a primal optimum BFS for this problem is nondegenerate, give an explanation for these paradoxes based on marginal analysis, by deriving the marginal value associated with such a change from the optimum dual solution.

(c) Show that an optimum solution for (5.56) may not be optimal to (5.57) even if $c > 0$

$$\begin{array}{rlll} \text{minimize} & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \quad x_{ij} & & \\ \text{subject to} & \sum_{j=1}^{n} \quad x_{ij} & \geqq a_i, \ i = 1 \text{ to } m & \qquad (5.57) \\ & \sum_{i=1}^{m} \quad x_{ij} & \geqq b_j, \ j = 1 \text{ to } n & \\ & x_{ij} & \geqq 0, \ \text{for all } i, j & \end{array}$$

(d) Let $\bar{x} = (\bar{x}_{ij})$ be a nondegenerate optimal BFS for (5.56), and $(\bar{u}, \bar{v})$ the corresponding dual optimum solution. If i, j are such that $\bar{u}_i + \bar{v}_j \leq 0$, both a_i and b_j can be increased by the same positive quantity δ while decreasing (if $\bar{u}_i + \bar{v}_j < 0$) or not changing (if $\bar{u}_i + \bar{v}_j = 0$) the optimum objective value. Give δ the maximum possible value which keeps the basis associated with the BFS $\bar{x}$ primal feasible. Repeat with other pairs i, j satisfying the same condition. So we are increasing the supplies and demands maximally to where the "more for less (nothing) paradox" just stops. Let a', b' be the resulting supply, demand vectors obtained at the end of this operation. Prove that every optimum BFS for the balanced transportation problem (5.56) with a', b' replacing a, b, must be degenerate.

(Charnes and Klingman [1971], Szwarc [1971], Charnes, Duffuaa and Ryan [1980])

5.20 Consider the single commodity flow network G $= (\mathcal{N}, \mathcal{A}, 0, k, c, \breve{s}, \breve{t})$, with $k > 0$ and finite. Let the "parametric value problem" refer to the problem of finding a minimum cost flow of value v in G, for each possible value v. Let $\bar{v}$ be the maximum flow value in G from $\breve{s}$ to $\breve{t}$. For $0 \leqq v \leqq \bar{v}$, let $g(v)$ be the minimum cost for a flow value of v in G. Then $g(v)$ is a piecewise linear convex function defined over the interval $0 \leqq v \leqq \bar{v}$. A breakpoint for $g(v)$ is a v where the slope of $g(v)$ changes.

Let x, y be real variables. For any convex function $p(x) : \mathbb{R}^1 \to \mathbb{R}^1 \cup \{\infty\}$ the conjugate of $p(x)$ is defined to be

$$p^*(y) = \text{ minimum } \{p(x) - yx\} \text{ over } x \in \mathbb{R}^1.$$

The function $p^*(y)$ is called the conjugate of $p(x)$. It can be shown that $p^*(y)$ is a well-defined convex function and that $p^{**} = p$. Also, if $p(x)$ is piecewise linear and $\{x : \ p(x) < \infty\}$ is bounded, then $p^*(y)$ is also piecewise linear and the number of breakpoints of $p^*(y)$ is one less than the number of breakpoints of $p(x)$. In fact, each breakpoint of $p^*(y)$ corresponds to an interval of constant slope for $p(x)$ and vice versa. See Rockafellar [1970].

Let $[\mathbf{X}, \bar{\mathbf{X}}]$ be any cut separating $\breve{s}$ and $\breve{t}$ in G with $\breve{s} \in \mathbf{X}, \breve{t} \in \bar{\mathbf{X}}$. Introduce one new arc $(\breve{s}, \breve{t})$ with 0 lower bound, 0 cost, and infinite capacity and let G$'$ denote the resulting network. Define parametric cost-coefficients $c_{ij}(\lambda)$ on G$'$ by the following

$$c_{ij}(\lambda) = \begin{cases} c_{ij} - \lambda & \text{if } (i,j) \in (\mathbf{X}, \bar{\mathbf{X}}). \\ c_{ij} + \lambda & \text{if } (i,j) \in (\bar{\mathbf{X}}, \mathbf{X}). \\ c_{ij} & \text{otherwise.} \end{cases}$$

Let parametric cost problem refer to the problem of finding a feasible flow vector $f = (f_{ij})$ of value $\bar{v}$ in G$'$ that minimizes $\sum(c_{ij}(\lambda) f_{ij}$: over $(i,j) \in \mathcal{A}')$. Let $h(\lambda)$ denote the minimum objective value in this parametric cost problem as a function of the parameter λ.

Prove that $h(\lambda) = -g^*(\lambda)$, and therefore that $h(\lambda)$ is a piecewise-linear concave function in which the number of breakpoints is one less than that for $g(v)$ (Carstensen [1983], Zadeh [1973]).

5.21 Minimum Cost Flow Problems in Series- Parallel Networks A two-terminal series-parallel network is a directed network with exactly one source and one sink, which is generated recursively as follows:

i) a single arc together with its tail and head as the source and sink nodes is a series parallel network.

ii) If S_1 and S_2 are series-parallel networks, so is the network obtained by either of the following operations.

  a) parallel composition: identify (i.e., make into a single node) the source of S_1 with the source of S_2, and the sink of S_1 with the sink of S_2.

  b) Series composition: identify the sink of S_1 with the source of S_2.

Let G $= (\mathcal{N}, \mathcal{A}, 0, k, c, \breve{s}, \breve{t})$ be a directed acyclic single commodity flow network. Let $\bar{v}$ denote the maximum value of flow from $\breve{s}$ to $\breve{t}$ in G. Prove that G is a two terminal series-parallel network iff for every arbitrary nonnegative capacity vector k and arbitrary cost vector c and $0 \leqq v \leqq \bar{v}$, a minimum cost flow of value v in G can be obtained by the scheme discussed in Section 5.2 beginning with $f = 0$ for value 0, and augmenting flow along a shortest chain from $\breve{s}$ to $\breve{t}$ consisting of arcs in G with positive residual capacity at that stage until the required value is reached, reverse arcs are never used and flow is never reduced on any arc (Bein, Brucker and Tamir [1985]).

5.22 Let G $= (\mathcal{N}, \mathcal{A})$ be a connected directed single commodity flow network with $V = (V_i)$ as the vector of exogenous flow amounts at the nodes satisfying $\sum_{i \in \mathcal{N}} V_i = 0$. $\mathbb{T}$ is a spanning tree in G with a node, say 1, selected as the root node. Fix the flow amounts on all the out-of-tree arcs at 0, and let $\bar{f}$ denote the resulting basic flow vector corresponding to $\mathbb{T}$. For each $i \in \mathcal{N}, i \neq 1$, let P$(i)$ denote the immediate predecessor of i in $\mathbb{T}$ and let $\mathbf{H}(\mathbb{T}, i)$ be the family of node i in $\mathbb{T}$. For each $i \neq 1$, prove that the flow amount in $\bar{f}$ on the in-tree arc joining i and P(i) is $\sum(\alpha_j V_j :$ over $j \in \mathbf{H}(\mathbb{T}, i))$ where $\alpha_j = +1$ or

-1 for all $j \in \mathbf{H}(\mathbb{T}, i)$. Discuss how to determine whether $\alpha_j = +1$ or -1 for each j in $\mathbf{H}(\mathbb{T}, i)$.

5.23 Consider the ordinary uncapacitated transportation problem

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \quad x_{ij} & & \\ \text{subject to } & \sum_{j=1}^{n} \quad x_{ij} & = a_i, \ i = 1 \text{ to } m \\ & \sum_{i=1}^{m} \quad x_{ij} & = b_j, \ j = 1 \text{ to } n \\ & \quad x_{ij} & \geqq 0, \ \text{for all } i, j \end{aligned} \tag{5.58}$$

where all the cost coefficients satisfy $L \leq c_{ij} \leq 2L$ for some positive L and $a_i > 0, b_j > 0$ for all i, j and $\sum a_i = \sum b_j$. Now consider the following optimization problem, with the same data, in which the variables are unrestricted in sign.

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \quad |x_{ij}| & & \\ \text{subject to } & \sum_{j=1}^{n} \quad x_{ij} & = a_i, \ i = 1 \text{ to } m \\ & \sum_{i=1}^{m} \quad x_{ij} & = b_j, \ j = 1 \text{ to } n \end{aligned} \tag{5.59}$$

(i) Prove that both these problems have the same optimal solutions.

(ii) Prove that if $\bar{x} = (\bar{x}_{ij})$ is feasible to (5.59), it is optimal iff there exists $u = (u_i), v = (v_j)$ satisfying the following for all $i = 1$ to $m, \ j = 1$ to n.

$$\begin{aligned} |u_i + v_j| &\leq c_{ij} \\ \bar{x}_{ij} > 0 \text{ implies } c_{ij} - u_i - v_j &= 0 \\ \bar{x}_{ij} < 0 \text{ implies } c_{ij} + u_i + v_j &= 0 \end{aligned} \tag{5.60}$$

(iii) Consider the following algorithm for solving (5.59). It moves along a path of basic solutions of (5.59); the solutions are not necessarily positive. It starts with any basic solution of (5.59) (which may not be $\geqq 0$) and generates a sequence of basic solutions using operations (a) and (b) given below, until the termination criterion is satisfied.

(a) Let β be the set of basic cells associated with the current solution $\bar{x}$ of (5.59). Define for $(p, q) \in \beta$

$$c^*_{p,q} = \begin{cases} c_{pq}, & \text{if } \bar{x}_{pq} \geq 0 \\ -c_{pq}, & \text{if } \bar{x}_{pq} < 0. \end{cases}$$

Determine $u = (u_i), v = (v_j)$ to satisfy $u_p + v_q = c^*_{pq}$ for $(p, q) \in \beta$, and the additional condition $v_n = 0$, say, to take care of the redundancy in the constraints in (5.59).

Termination condition: If $|u_i + v_j| \leq c_{ij}$ for all i, j, then the present solution $\bar{x}$ is optimal for (5.59); terminate.

Go to (b) if termination condition is not satisfied.

(b) Select a cell (h, g) such that $|u_h + v_g| > c_{hg}$ as the entering cell into the basic set β. To determine the dropping basic cell, find the unique cycle in $\beta \cup \{(h, g)\}$. Make x_{hg} a small positive quantity and determine the modified values of x_{ij} for (i, j) on the cycle. Let Δ^+ be the set of cells in the cycle that decrease their absolute value in this operation, and Δ^- the remaining cells in the cycle.

1. Prove that $\Delta^+ \neq \emptyset$ if $u_h + v_g > c_{hg}$, and $\Delta^- \neq \emptyset$ if $u_h + v_g < -c_{hg}$. If $u_h + v_g > c_{hg}$, select the dropping cell from β to be (p, q) corresponding to the smallest $|\bar{x}_{ij}|$ among cells in Δ^+. If $u_h + v_g < -c_{hg}$, select the dropping cell from β to be (p, q) corresponding to the smallest $|\bar{x}_{ij}|$ among cells in Δ^-.
2. Find the basic solution of (5.59) corresponding to the new basic set, and repeat the whole process with it.

Prove that the change in the objective value of the solution for (5.59) in this step is $|\bar{x}_{pq}|(c_{hg} - u_h - v_g)$, if $u_h + v_g > c_{hg}$, or $|\bar{x}_{pq}|(c_{hg} + u_h + v_g)$, if $u_h + v_g < -c_{hg}$.

(iv) Discuss a method for solving (5.58), through (5.59) using this method.

(Finke and Ahrens [1978])

5.24 The Bottleneck Transportation Problem Bottleneck transportation problems usually arise in shipping perishable commodities which have to be distributed quickly. Suppose there are m sources and n destinations. Let $a_i > 0$

$$\begin{aligned} \text{minimize } (\text{maximum } \{t_{ij} : \quad & x_{ij} > 0\}) \\ \text{subject to } \sum_{j=1}^{n} & x_{ij} = a_i, \; i = 1 \text{ to } m \qquad (5.61) \\ \sum_{i=1}^{m} & x_{ij} = b_j, \; j = 1 \text{ to } n \\ 0 \leqq & x_{ij} \leq k_{ij}, \text{ for all } i, j \end{aligned}$$

be the units available at the ith source, and $b_j > 0$ the units required at the jth destination, satisfying $\sum a_i = \sum b_j$. Let $k_{ij} > 0$ be the capacity of the arc (i, j) joining source i and destination j, and $t_{ij} \geq 0$ be the time necessary to carry out the shipment on this arc. Then the balanced capacitated bottleneck transportation problem with this data is (5.61). Given a feasible solution $\bar{x} = (\bar{x}_{ij})$, develop necessary and sufficient optimality conditions for it to be optimal for (5.61). Also develop an algorithm for solving (5.61). Apply the algorithm to solve the problem with data, $m = n = 5$, $k_{ij} = 24$ for all (i, j), and

$j =$		t_{ij} in days 1	2	3	4	5	a_i
$i =$	1	13	16	3	12	28	15
	2	22	28	15	12	20	15
	3	21	29	6	18	14	12
	4	17	29	3	25	29	32
	5	4	14	4	15	2	26
b_j		3	39	27	7	24	

The transportation paradox which occurs in the linear transportation problem can also occur in the bottleneck problems. For example, consider the following version of (5.61) with overshipments allowed at both the sources and destinations.

$$\begin{aligned} &\text{minimize (maximum } \{t_{ij} : \ x_{ij} > 0\}) \\ &\text{subject to } \sum_{j=1}^{n} x_{ij} \geq a_i, \ i = 1 \text{ to } m \qquad (5.62) \\ &\sum_{i=1}^{m} x_{ij} \geq b_j, \ j = 1 \text{ to } n \\ &0 \leq x_{ij} \leq k_{ij} \end{aligned}$$

Develop conditions for the occurrence of the paradox (i.e., the optimum objective value in (5.62) being strictly less than that in (5.61)), and give an explanation for the paradox in terms of dual solutions and marginal analysis.

Show that the paradox occurs in the numerical problem with the data given above (the optimum objective value strictly decreases as a_1 and b_1 are both increased by the same positive quantity, keeping all other data unchanged).

Let BT(OS) denote the bottleneck time (optimum objective value in (5.62)) as a function of overshipment amount $\sum_i \sum_j x_{ij} - \sum_i a_i$. Develop an algorithm to draw the complete curve of BT(OS). Apply this algorithm on the numerical example discussed above, and draw the BT(OS) curve for it (Finke [1983]).

5.25 Applications in Portfolio Management Consider the problem of selecting a dynamic portfolio of securities in order to maximize total return over a fixed planning horizon. This problem can be modeled as a network flow problem in which each arc represents a security and each node (other than a terminal sink node representing the end of the planning horizon) signifies the beginning of an investment period.

For example, if the investor has the option of purchasing one-year bonds that return 4% per annum, two-year bonds at 6% per annum, or four-year bonds at 5% per annum, the network corresponding to a four year planning horizon is given in Figure 5.24.

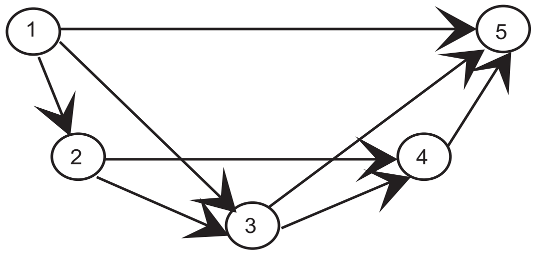


Figure 5.24:

(i) Show that the network for this model is acyclic. Show that the problem of finding the best investment policy over an n period planning horizon can be formulated as that of finding a longest chain from node 1 to node $n+1$ in this network. Using this, find the best investment policy in the example cited above for a 4 year planning horizon.

(ii) Assume that the accumulated interest is considered as profit and not reinvested. Consider the problem of maximizing this total profit over the entire planning horizon, where the original amount can be reinvested as many times as necessary until the end of the planning horizon. Also, suppose we are given lower and upper bounds on the amount to be invested in each possible security at each time point (these bounds may vary with time for each security). Formulate this problem as a maximum profit network flow problem.

Construct this model for the three bond examples discussed above over a six year planning horizon. Assume that the money available to be invested is \$100, and that the bounds are as in the table given below. Find an optimum investment policy in this numerical example.

	First two years		Next three years		Final year
Security	Lower	Upper	Lower	Lower	
1-year bonds	15	35	15	40	0-100
2-year bonds	30	60	40	70	x
4-year bonds	20	40	0	25	x

(Golden and Keating [1982])

5.26 Consider the ordinary transportation problem

$$\begin{aligned} \text{minimize} \ \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \quad & x_{ij} \\ \text{subject to} \ \sum_{j=1}^{n} \quad & x_{ij} \ \leq a_i, \ i = 1 \text{ to } m \\ \sum_{i=1}^{m} \quad & x_{ij} \ \geq b_j, \ j = 1 \text{ to } n \\ & x_{ij} \ \geqq 0, \ \text{for all } i, j \end{aligned}$$

with the usual assumptions that $a_i > 0, b_j > 0$ for all i, j and $\sum a_i \geq \sum b_j$. Suppose there is the additional condition that each demand center j must be entirely supplied from a single source. This requirement appears in many practical applications. It appears frequently in the supplying of supermarket orders from a network of central warehouses. It is commonly required in military applications as troops going in the same mission usually leave from the same staging area, etc. Discuss approaches for solving the transportation problem with this additional condition (Nagelhout and Thompson [1980]).

5.27 Consider a stationary dynamic maximum value flow problem on the network G = $(\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$ with $a = (a_{ij})$ as the vector of arc traversal times, over a T-period horizon. Let $v_1, \dots, v_T$ represent the flow value reaching the sink in periods $1, \dots, T$. Prove that any feasible flow of value $v = v_1 + \dots + v_T$ over the horizon which satisfies either of conditions $(a), (b)$ below, also satisfies the other two.

(a) Maximize $\sum_{\tau=1}^{p} v_\tau$ for $p = 1, \dots, T$ (i.e., maximize the output for the first p periods for all p).

(b) Minimize $\sum_{\tau=1}^{T} c_\tau v_\tau$, where $c_1 < c_2 < \dots < c_\tau$ (i.e., minimize the weighted sum of flow values in the various time periods, where the weights are increasing with time).

(c) Minimize p such that $v_{p+\tau} = 0$ for $\tau = 1, \dots, T - p$ (i.e., minimize the number of time periods required to send a flow of v from $\breve{s}$ to $\breve{t}$).

(Jarvis and Ratliff [1982]).

5.28 Warehousing Problem In this problem, we are required to plan the operations of a warehouse over an n-period planning horizon. The warehouse purchases, stores and sells in each period, a single commodity that is subject to known fluctuations in selling prices and purchasing costs. The warehouse has a fixed capacity (γ units) and all new purchases and hold-overs from previous periods are stored there before selling. In each period any amount can be purchased, stored and sold at the known specified prices, subject to the warehouse capacity. We have the additional data and variables listed below.

b_i = warehousing cost per unit in period i (this applies to z_i defined below)
c_i = purchase cost per unit in period i, $i = 1$ to n
d_i = selling price per unit in period i, $i = 1$ to n
s_o = initial stock in warehouse, given
s_n = specified final stock desired in warehouse after selling in period n
x_i = amount purchased in period i, $i = 1$ to n
s_i = amount held in warehouse after selling in period i, $i = 1$ to n
(s_n is specified in the data given above)
$z_i = s_{i-1} + x_i$ = amount in warehouse after purchasing (but before selling) in period i, this has to be $\leqq$ capacity γ for $i = 1$ to n
y_i = amount sold in period i (this is $\leqq z_i$), $i = 1$ to n

It is required to determine a purchasing and selling plan which will maximize the total net profit. Formulate this as a network flow problem. Prove that the maximum total net profit is a multiple of the warehouse capacity γ.

5.29 Consider an m-source, n-sink, p-commodity transportation problem denoted by MCTP(m, n, p): it is to find $(x^r_{ij} : i = 1$ to m, $j = 1$ to n, $r = 1$ to $p)$ to

$$\begin{array}{rlll} \text{Minimize} & \sum_i \sum_j \sum_r c^r_{ij} x^r_{ij} & & \\ \text{subject to} & \sum_j x^r_{ij} & = a^r_i, & \text{for all } i, r \\ & \sum_i x^r_{ij} & = b^r_j, & \text{for all } j, r \\ & \sum_r x^r_{ij} & + s_{ij} = k_{ij}, & \text{for all } i, j \\ & x^r_{ij}, s_{ij} & \geqq 0, & \text{for all } i, j, r \end{array} \tag{5.63}$$

where the data satisfy $\sum_i a^r_i = \sum_j b^r_j$, for all r. Prove that the constraint coefficient matrix of MCTP$(m, n, 2)$ is totally unimodular if all capacitated arcs are incident to a common node.

Prove that the necessary and sufficient condition for the constraint coefficient matrix of a capacitated MCTP(m, n, p) with $p \geqq 2$, and k finite, to be unimodular is that either m or n to be $\leqq 2$.

Show that every MCTP$(m, 2, p)$, or MCTP$(2, n, p)$ can be transformed into a one commodity capacitated transportation problem (Rebman [1974]; Evans, Jarvis, and Duke [1977]; Evans [1976]).

Comment 5.2 The minimum cost flow problem is a classical problem. Historically it is among the first linear programming problems to be modeled and studied. The problem was posed and a rudimentary algorithm discussed for it in Kantorovitch [1939]. A computational method for the transportation model that would now be called "primal simplex" has been proposed by Hitchcock [1941] where he shows that an optimum solution will be at an extreme point, and develops methods to iteratively construct better extreme point solutions. Later Koopmans [1949] developed node potentials and the optimality criterion, and showed that an extreme point is based on a spanning tree. Dantzig [1951] developed the special variant of the primal simplex algorithm for the transportation problem. He observed the spanning tree property of the basis, the integrality properties of the optimum solutions, and the Dantzig property. Orden [1956] showed that these results extend to the transshipment problem.

The first combinatorial algorithms based on the primal-dual approach were developed by Ford and Fulkerson [1957]. This work was motivated by the Hungarian method of Kuhn [1955 of Chapter 3] for the assignment problem.

The shortest augmenting path (or the build-up) method based on successive shortest chain computations was developed independently in Jewell [1958], Iri [1960], and Busacker and Gowan [1961]. Originally, all these approaches were based on solving the minimum cost flow problem through a sequence of shortest chain computations on residual networks which may have arbitrary arc lengths. Later, Tomizawa [1971] pointed out that the approach can be implemented using node potentials in such a way that all shortest chain computations are on networks with nonnegative arc lengths.

Minty [1960] and Fulkerson [1961] independently developed the out-of-kilter method for minimum cost flow problems. Edmonds and Karp [1972 of Chapter 2] showed that the out-of-kilter method can be converted into a polynomially bounded algorithm by implementing it using a special capacity scaling technique developed by them. This technique applies the out-of-kilter method on a series of problems which provide successively closer approximations to the original problem. It is the first polynomial time algorithm for the minimum cost flow problem, and the first polynomial time algorithm for a significant special class of linear programs. The practical significance of scaling techniques is not very clear, but it has great theoretical value.

The original negative cycle canceling algorithm is due to Klein [1967]. This simple, classical algorithm may not even terminate if the capacities are irrational. On problems with integer data it is a finite algorithm, but the number of iterations can grow exponentially with the size of the data in the worst case. Recently Goldberg and Tarjan [1989] have shown that this algorithm can be converted into a strongly polynomial algorithm by an appropriate choice of the cycle to cancel in each iteration (a minimum mean residual cycle).

Zadeh [1973a, b] has presented examples of network flow problems on which the classical methods, the primal simplex algorithm with Dantzig's pivot rule, the dual simplex algorithm, the original negative cycle canceling algorithm, the shortest augmenting path algorithm, and the out-of-kilter algorithm, all require an amount of effort growing exponentially with the size of the problem.

The fact that these algorithms displayed exponential growth on one specially constructed pathological class of problems does not mean that they won't work well on network models arising in applications. Computational experience with these methods has always turned out to be much better than their bleak performance on these pathological examples. In the 1950s and 1960s, primal-dual and then the out-of-kilter were popular methods for solving network models. In those days data structures for generating efficient implementations of the primal simplex method for network flow problems were not available. The first tree label data structures for manipulating trees were suggested by Scions [1964] and Johnson [1966]. Early implementations based on them for the primal simplex method for minimum cost flow problems, by Srinivasan and Thompson [1972]; Glover, Karney and Klingman [1974]; and Glover, Karney, Klingman, and Napier [1974] performed much better than implementations of the primal-dual and out-of-kilter methods available at that time. They not only showed that the primal implementations are faster, but also that they require less storage and are most suitable when using secondary storage devices, and are more compatible as embedded parts of more general optimization systems. Improved data structures were developed in Glover, Karney, and Klingman [1972]; Glover, Klingman, and Stutz [1974]; Bradley, Brown, and Graves [1977]; and Barr, Glover, and Klingman [1979]; and numerous others. With these breakthroughs in coding, the primal simplex method has replaced the other methods as the computational workhorse of commercial network software in the 1970s.

The primal simplex method and efficient ways of implementing it to solve minimum cost flow problems have been the subject of very rigorous investigations by many groups in the 1970s and 1980s. There is a gulf of difference between the mathematical statement of an algorithm and its implementation using appropriate data structures, and methods for updating them from one step to the next. Empirical computational tests comparing different algorithms or different implementations of the same algorithm on representative classes of problems of varying sizes, sparsity and other characteristics, are an important research activity. Substantial reductions in computational effort can typically be achieved through appropriate implementation of an algorithm. Data structure design must go hand in hand with pivot selection strategy in implementing the primal simplex method. Any change in pivot strategy requires a commensurate change in data structures to be effective. Mulvey [1978] describes the results of an organized experimental design and a detailed series of empirical tests to compare various pivot strategies, methods for attaining feasibility (big-M, Phase I, etc.) and a variety of other features. He found that the internal tactics of the code are very important. His most successful implementation uses a candidate list of nonbasic arcs (about 40) to select the entering arc from, and re-forms it periodically (every 20 pivots or so) to improve performance. Some of the other papers dealing with computational studies and implementation issues are Aashtiani and Magnanti [1976]; Ali, Helgason, Kennington, and Lall [1978]; Clausen [1968]; Gibby, Glover, Klingman, and Mead [1983]; Glover and Klingman [1982]; Grigoriadis [1986]; Grigoriadis and Hsu [1980]; Hatch [1975]; and Helgason and Kennington [1977].

The supremacy of the primal simplex method for solving minimum cost flow problems has continued into the 1980s. Recently, however, a new class of relaxation algorithms (Section 5.6) were proposed by Bertsekas and his associates, and investigations reported in Bertsekas [1985]; Bertsekas, Hosein, and Tseng [1987]; and Bertsekas and Tseng [1988] indicate that these algorithms exhibit nice empirical behavior, and are competitive or better than the primal simplex method.

Public domain computer codes for the minimum cost flow problems are available from Bertsekas and Tseng [1988]; Grigoriadis and Hsu [1980]; Kennington and Helgason [1980 of Chapter 1]; and Simeone, Toth, Gallo, Maffioli, and Pallotino [1988].

Armacost and Mehrotra [1991] compared an implementation of the network primal simplex method with an implementation of the recently developed interior point method (the dual affine scaling method) on sparse minimum cost flow problems. They found that the network simplex method outperforms the affine scaling method by a substantial margin on these problems; also its advantage seems to grow with the number of nodes and the density of the network.

The first strongly polynomial time minimum cost flow algorithm is due to Tardos [1985]. This method is based on the repeated use of a concept of approximate optimality. Theoretically, the simplest strongly polynomial time algorithm for minimum cost flow problems is perhaps that of Goldberg and Tarjan [1989]. Many other strongly polynomial time algorithms are now available for minimum cost flow problems, among them are Fujishige [1986]; Galil and Tardos [1988]; Orlin [1984, 1988]; and Tardos [1986]. The corresponding question in general LP (is there a strongly polynomial algorithm for it?) still remains an open question.

Many different approaches for obtaining exact or approximate optimum solutions for multicommodity network flow problems have been suggested over the years. In this chapter we discussed an approach suggested by Ford and Fulkerson [1958] based on column generation for the problem in arc-chain formulation. This approach subsequently inspired Dantzig and Wolfe [1960] to develop the decomposition principle for LP. For other approaches for multicommodity flow problems based on partitioning or decomposition, see the survey papers of Ali, Helgason, Kennington and Lall [1980]; Assad [1978]; Geoffrion and Graves [1974]; Grigoriadis and White [1972]; Hartman and Lasdon [1972]; Kennington [1978]; Kennington and Shalaby [1977]; Kleitman [1971]; Swoveland [1973]; Tomlin [1966]; and Kennington and Helgason [1980 of Chapter 1]. Since 1984, the highly publicized success of interior point methods based on barrier functions, or method of centers, or other approaches of nonlinear programming specialized to LP, has resulted in many researchers shifting their attention to these new methods for solving large scale LPs and multicommodity flow problems in particular. See Adler, Resende, Veiga, and Karmarkar [1989], and Kapoor and Vaidya [1985].

5.13 References

H. A. AASHTIANI and T. L. MAGNANTI, June 1976, “Implementing Primal-Dual Network Flow Algorithms,” OR Center report 055-76, MIT, Cambridge, Mass.

I. ADLER, M. G. C. RESENDE, G. VEIGA and N. KARMARKAR, 1989, “An Implementation of Karmarkar’s Algorithm for Linear Programming,” *MP A*, 44, no. 3(297-335).

R. K. AHUJA, J. BATRA, and S. K. GUPTA, May 1984, “A Parametric Algorithm for Convex Cost Network Flow and Related Problems,” *EJOR*, 16, no. 2(222-235).

A. ALI, R. HELGASON, J. KENNINGTON, and H. LALL, 1978, “Primal Simplex Network Codes: State of the Art Implementation Technology,” *Networks* 8(315-339).

A. ALI, R. HELGASON, J. KENNINGTON, and H. LALL, 1980, “Computational Comparison Among Three Multicommodity Network Flow Algorithms,” *OR*, 28, no. (995-1000).

A. ARMACOST and S. MEHROTRA, 1991, “A Computational Comparison of the Network Simplex Method with the Dual Affine Scaling Method,” *Opsearch*, to appear.

A. A. ASSAD, 1978, “Multicommodity Network Flows, A Survey,” *Networks*, 8(37-91).

F. BARAHONA and E. TARDOS, 1989, “Note on Weintraub’s Minimum Cost Circulation Algorithm,” *SIAM Journal on Computing* , 18(579-583).

R. BARR, F. GLOVER, and D. KLINGMAN, Feb. 1979, “Enhancements of Spanning Tree Labeling Procedures for Network Optimization,” *INFOR* 17, no. 1(16-34).

M. S. BAZARAA and H. D. SHERALI, 1982, “A Property Regarding Degenerate Pivots for Linear Assignment Networks,” *Networks*, 12, no. 4(469-474).

W. W. BEIN, P. BRUCKER, and A. TAMIR, 1985, “Minimum Cost Flow Algorithms for Series-Parallel Networks,” *DAM* 10,no. 2(117-124).

D. P. BERTSEKAS, 1985, “A Unified Framework for Primal-Dual Methods in Minimum Cost Network Flow Problems,” *MP*, 32(125-145).

D. P. BERTSEKAS, P. HOSEIN, and P. TSENG, 1987, “Relaxation Methods for Network Flow Problems With Convex Arc Costs,” *SIAM J. on Control and Optimization*, 25(1219-1243).

D. P. BERTSEKAS and P. TSENG, Jan.-Feb. 1988, “Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems,” *OR* 36, no. 1(93-114).

R. G. BLAND and D. L. JENSEN, 1985, “On the Computational Behavior of a Polynomial Time Network Flow Algorithm,” Tech. report 661, SORIE, Cornell University, Ithaca, N.Y.

G. H. BRADLEY, G. G. BROWN and G. W. GRAVES, 1977, “Design and Implementation of Large Scale Transshipment Algorithms,” *MS*, 24(1-34).

J. R. BROWN, 1983, “The Flow Circulation Sharing Problem,” *MP*, 25, no. 2(199-227).

R. G. BUSACKER and P. J. GOWAN, 1961, “A Procedure for Determining a Family of Minimum-Cost Network Flow Patterns," ORO Tech. report 15, John Hopkins University.

P. J. CARSTENSEN, 1983, "Complexity of Some Parametric Integer and Network Programming Problems," *MP*, 26, no. 1(64-75).

A. CHARNES and D. KLINGMAN, 1971, "The More-for-Less Paradox in the Distribution Model," *Cahiers de Centre d'Etudes Recherche Operationelle*, 13, no. 1(11-22).

A. CHARNES, S. DUFFUAA, and M. RYAN, 1980, "Degeneracy and the More-for-Less Paradox," *Journal of Information and Optimization Sciences*, 1, no. 1(52-56).

R. J. CLAUSEN, March 1968, "The Numerical Solution of Network Problems Using the Out-of-Kilter Algorithm," RAND Corp. memo RM-5456-PR, Santa Monica, Calif.

W. H. CUNNINGHAM, 1976, "A Network Simplex Method," *MP*, 11(105-116).

W. H. CUNNINGHAM, 1979, "Theoretical Properties of the Network Simplex Method," *MOR*, 4(196-208).

W. H. CUNNINGHAM and J. G. KLINCEWICZ, 1983, "On Cycling in the Network Simplex Method," *MP*, 26(182-189).

G. B. DANTZIG, 1951, "Application of the Simplex Method to a Transportation Problem," in T. C. Koopmans(ed.) *Activity Analysis of Production and Allocation*, Wiley, N.Y.

G. B. DANTZIG and P. WOLFE, 1960, "Decomposition Principle for Linear Programs," *OR*, 8(101-111).

J. R. EVANS, 1976, "A Combinatorial Equivalence Between a Class of Multicommodity Flow Problems and the Capacitated Transportation Problem," *MP*, 10(401-404).

J. R. EVANS, J. J. JARVIS, and R. A. DUKE, 1977, "Graphic Matroids and the Multicommodity Transportation Problem," *MP*, 13(323-328).

S. EVEN, A. ITAI, and A. SHAMIR, Dec. 1976, "On the Complexity of Timetable and Multicommodity Flow Problems," *SIAM J. on Computing*, 5, no. 4(691-703).

G. FINKE, May 1983, "Minimizing Overshipments in Bottleneck Transportation Problems," *INFOR*, 21, no. 2(121-135).

G. FINKE and J. H. AHRENS, Feb. 1978, "A Variant of the Primal Transportation Algorithm," *INFOR*,16, no. 1(35-46).

C. O. FONG and V. SRINIVASAN, 1977, "Determining All Nondegenerate Shadow Prices for the Transportation Problem," *TS*, 11, no. 3(199-222).

L. R. FORD and D. R. FULKERSON, 1958, "A Suggested Computation for Maximal Multicommodity Network Flows," *MS*, 5(97-101).

H. FRIESDORF and H. HAMACHER, 1982, "Weighted Minimum Cost Flows," *EJOR*, 11(181-192).

S. FUJISHIGE, 1986, "A Capacity Rounding Algorithm for Minimum Cost Circulation Problem: A Dual Framework of the Tardos Algorithm," *MP*, 35(298-308).

D. R. FULKERSON, 1961, "An Out-of-Kilter Method for Minimum Cost Flow Problems," *SIAM J. on Applied Mathematics*, 9(18-27).

Z. GALIL and E. TARDOS, 1988, "An O($n^2(m + n \log n) \log n$) Min. Cost Flow

Algorithm," *JACM*, 35(374-386).

A. R. GAMBLE, A. R. CONN, and W. R. PULLEYBLANK, 1988, "A Network Penalty Method," Tech. report, Dept. of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada.

B. J. GASSNER, 1964, "Cycling in the Transportation Problem," *NRLQ*, 11(43-58).

B. GAVISH, P. SCHWEITZER, and E. SHLIFER, 1977, "The Zero Pivot Phenomenon in Transportation Problems and its Computational Implications," *MP*, 12(226-240).

A. M. GEOFFRION and G. W. GRAVES, 1974, "Multicommodity Distribution System Design by Benders Decomposition," *MS*, 20(822-844).

D. GIBBY, F. GLOVER, D. KLINGMAN, and M. MEAD, 1983, "A Comparison of Pivot Selection Rules for Primal Simplex Based Network Codes," *OR Letters*, 2(199-202).

C. R. GLASSEY, 1978, "A Quadratic Network Optimization Model for Equilibrium Single Commodity Trade Flows," *MP*, 14(98-107).

S. GLICKSMAN, L. JOHNSON, and L. ESELSON, June 1960, "Coding the Transportation Problem," *NRLQ*, 2(169-184).

F. GLOVER, K. KARNEY, D. KLINGMAN, 1972, "The Augmented Predecessor Index Method for Locating Stepping Stone Paths and Assigning Dual Prices in Distribution Problems," *TS*, 6, no. 1(171-180).

F. GLOVER, D. KARNEY, and D. KLINGMAN, 1974, "Implementation and Computational Comparisons of Primal, Dual and Primal-Dual Computer Codes for Minimum Cost Network Flow Problems," *Networks*, 4(191-212).

F. GLOVER, D. KARNEY, D. KLINGMAN, and A. NAPIER, 1974, "A Computational Study on Start Procedures, Basis Change Criteria and Solution Algorithms for Transportation Problems," *MS*, 20, no. 5(793-813).

F. GLOVER and D. KLINGMAN, 1977, "Network Applications in Industry and Government," *AIIE Transactions*, 9, no. 4(363-376).

F. GLOVER and D, KLINGMAN, Nov. 1982, "Recent Developments in Computer Implementation Technology for Network Flow Algorithms," *INFOR*, 20, no. 4(433-452).

F. GLOVER, D. KLINGMAN, and A. NAPIER, 1972, "An Efficient Dual Approach to Network Problems," *Opsearch*, 9, no. 1(1-18).

F. GLOVER, D. KLINGMAN, and J. STUTZ, 1974, "Augmented Threaded Index Method for Network Optimization," *INFOR*, 12, no. 3(293-298).

A. V. GOLDBERG and R. E. TARJAN, Oct. 1989, "Finding Minimum-Cost Circulations by Canceling Negative Cycles," *JACM*, 36, no. 4(873-886).

A. V. GOLDBERG and R. E. TARJAN, Aug. 1990, "Finding Minimum-Cost Circulations by Successive Approximation," *MOR*, 15, no. 3(430-466).

B. L. GOLDEN and K. D. KEATING, 1982, "Network Techniques for Solving Asset Diversification Problems in Finance," *COR*, 9, no. 3(173-195).

M. D. GRIGORIADIS, July 1978, "Algorithms for Minimum Cost Single and Multi-Commodity Network Flow Problems," Notes for Summer School in Combinatorial

Optimization, SOGESTA, Urbino, Italy.

M. D. GRIGORIADIS, 1986,"An Efficient Implementation of the Network Simplex Method," *MPS*, 26(83-111),

M. D. GRIGORIADIS and T. HSU, 1979, "The Rutgers Minimum Cost Network Flow Subroutines," *SIGMAP Bulletin of the ACM*, 26(17-18).

M. D. GRIGORIADIS and T. HSU, Nov. 1980, "The Rutgers Minimum Cost Network Flow Subroutines," RNET Documentation, Dept. of Computer Science, Rutgers University, New Brunswick, N.J.

M. D. GRIGORIADIS and W. W. WHITE, 1972, "A Partitioning Algorithm for the Multicommodity Network Flow Problem," *MP*, 3(157-177).

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

J. K. HARTMAN and L. S. LASDON, 1972, "A Generalized Upper Bounding Algorithm for Multicommodity Network Flow Problems," *Networks*, 1(333-354).

R. HASSIN, 1983, "The Minimum Cost Flow Problems, a Unifying Approach to Dual Algorithms and a New Tree Search Algorithm," *MP*, 25(228-239).

R. S. HATCH, 1975, "Bench Marks Comparing Transportation Codes Based on Primal Simplex and Primal-Dual Algorithms," *OR*, 23(1167-1172).

R. V. HELGASON and J. L. KENNINGTON, March 1977, "An Efficient Procedure for Implementing a Dual Simplex Network Flow Algorithm," *AIIE Transactions*, 9(63-68).

F. L. HITCHCOCK, April 1941, "The Distribution of a Product From Several Sources to Numerous Localities," *Journal of Mathematics and Physics* 20, no. 2(224-230).

M. IRI, 1960, "A New Method for Solving Transportation-Network Problems," *J. OR Society of Japan*, 3(27-87).

J. J. JARVIS and H. D. RATLIFF, Jan. 1982, "Some Equivalent Objectives for Dynamic Network Flow Problems," *MS*, 28, no. 1(106-109).

W. S. JEWELL, 1958, "Optimal Flow Through Networks," Tech. report 8, OR Center, MIT, Cambridge, Mass.

E. JOHNSON, 1966, "Networks and Basic Solutions," *OR*, 14(619-623).

L. V. KANTOROVITCH, 1939, "Mathematical Methods in the Organization and Planning of Production," Publication House of the Leningrad University, translated in *MS*, 6, 1960(366-422).

S. KAPOOR and P. M. VAIDYA, 1985, "Fast Algorithms for Convex Quadratic Programming and Multicommodity Flows," *Proc. 18th annual ACM Symp. on Theory of Computing*, (147-159).

S. KAPOOR and P. M. VAIDYA, 1988, "Speeding Up Karmarkar's Algorithm for Multicommodity Flows," Tech. report, Dept. of Computer Science, University of Illinois at Urbana-Champaign, Il.

R. M. KARP, 1978, "A Characterization of the Minimum Cycle Mean in a Digraph," *Discrete Mathematics*, 23(309-311).

J. L. KENNINGTON, Mar. -Apr. 1978, "A Survey of Linear Cost Multicommodity Network Flow," *OR*, 26, no. 2(209-236).

J. L. KENNINGTON and M. SHALABY, 1977, “An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems,” *MS*, 23(994-1004).

M. KLEIN, Nov. 1967, “A Primal Method for Minimum Cost Flows With Applications to Assignment and Transportation Problems,” *MS*, 14, no. 3(205-220).

D. L. KLEITMAN, 1971, “An Algorithm for Certain Multicommodity Flow Problems,” *Networks*, 1(75-90).

T. C. KOOPMANS, 1949, “Optimum Utilization of the Transportation System,” *Proceedings of the International Statistical Conference*, Vol. 5.

E. MINIEKA, 1974, “Dynamic Network Flows with Arc Changes,” *Networks*, 4(255-265).

G. J. MINTY, 1960, “Monotone Networks,” *Proceedings of the Royal Society of London*, 257A(196-212).

J. M. MULVEY, 1978, “Pivot Strategies for Primal Simplex Network Codes,” *JACM*, 25(266-270).

J. M. MULVEY, 1978,“Testing of a Large Scale Network Optimization Program,” *MP*, 15(291-314).

R. V. NAGELHOUT and G. L. THOMPSON, 1980, “A Single Source Transportation Algorithm,” *COR*, 7, no. 3(185-198).

M. NANIWADA, 1969, “Multicommodity Flows in a Communication Network,” *Electronics Commun.* , Japan, 52-A(34-41).

A. ORDEN, April 1956, “The Transshipment Problem,” *MS*, 2, no. 3(276-285).

J. B. ORLIN, 1984, “Genuinely Polynomial Simplex and Non-Simplex Algorithms for the Minimum Cost Flow Problem,” OR 1615-84, Sloan School of Mgt. MIT, Cambridge, Mass.

J. B. ORLIN, 1988, “A Faster Strongly Polynomial Minimum Cost Flow Algorithm,” *Proc. 20th ACM Symp. Theory of Computing*, (377-387).

M. H. PARTOVI, 1984, “A Study of Degeneracy in the Simplex Algorithm for Linear Programming and Network Flow Problems,” Ph. D. dissertation, IOE Dept. University of Michigan, Ann Arbor, Mich.

K. R. REBMAN, 1974, “Total Unimodularity and the Transportation Problem: A Generalization,” *Linear Algebra and its Applications*, 8(11-24).

H. ROCK, 1980, “Scaling Techniques for Minimal Cost Network Flows,” in U. Page (ed.) *Discrete Structures and Algorithms*, Carl Hanser, Munchen,(181-191).

R. T. ROCKAFELLAR, 1970, *Convex Analysis*, Princeton University Press, Princeton, N.J.

R. SAIGAL, 1967, “Multicommodity Flows in Directed Networks,” ORC 67-38, University of California, Berkeley, Calif.

S. R. SCHMIDT, P. A. JENSEN, and J. W. BARNES, 1982, “An Advanced Dual Incremental Network Algorithm,” *Networks*, 12, no. 4(475-492).

H. I. SCOINS, 1964, “The Compact Representation of a Rooted Tree and the Transportation Problem,” International Symposium on Mathematical Programming, London.

B. SIMEONE, P. TOTH, G. GALLO, F. MAFFIOLI, and S. PALLOTINO (eds.), *Fortran Codes for Network Optimization*, *AOR*, 13.

S. SINGH, July 1986, "Improved Methods for Storing and Updating Information in the Out-of-Kilter Algorithm," *JACM*, 33, no. 3(551-567).

V. SRINIVASAN and G. L. THOMPSON, 1972, "Accelerated Algorithms for Labeling and Relabeling of Trees With Applications to Distribution Problems," *JACM*, 19, no. 4(712-726).

V. SRINIVASAN and G. L. THOMPSON, 1973, "Benefit Cost Analysis of Coding Techniques for the Primal Transportation Algorithm," *JACM*, 20(194-213).

C. SWOVELAND, 1973, "A Two-Stage Decomposition Algorithm for a Generalized Multicommodity Flow Problem," *INFOR*, 11(232-244).

W. SZWARC, 1971, "The Transportation Paradox," *NRLQ*, 18(185-202).

E. TARDOS, 1985, "A Strongly Polynomial Minimum Cost Circulation Algorithm," *Combinatorica*, 5(247-256).

N. TOMIZAWA, 1971, "On Some Techniques Useful for Solution of Transportation Network Problems," *Networks*, 1(173-194).

J. A. TOMLIN, 1966, "Minimum Cost Multicommodity Network Flows," *OR*, 14(45-51).

R. R. VEMUGANTI and M. BELLMORE, 1973, "On Multicommodity Maximal Dynamic Flows," *OR*, 21, no. 1(10-21).

A. WEINTRAUB, 1974, "A Primal Algorithm to Solve Network Flow Problems with Convex Costs," *MS*, 21(87-97).

W. W. WHITE, 1972, "Dynamic Transshipment Networks: An Algorithm and its Application to the Distribution of Empty Containers," *Networks*, 2(211-236).

N. ZADEH, 1973a, "A Bad Network Flow Problem for the Simplex Method and Other Minimum Cost Flow Algorithms," *MP*, 5(255-266).

N. ZADEH, 1973b, "More Pathological Examples for Network Flow Problems," *MP*, 5(217-224).

Index

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