Contents

Chapter 2

Single Commodity Maximum Value Flow Problems in Pure Networks

The problem of finding a maximum value flow in the directed single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ can be solved by several methods, one of which is the bounded variable simplex method. In this chapter we discuss efficient network algorithms for this problem, the **augmenting path methods**, and the recently developed preflow-push algorithms.

If $\ell = 0$, one may be tempted to believe that this problem can be solved by the following simple scheme. Since $\ell = 0$, one can begin with the feasible flow vector $f^0 = 0$. Find a chain from $\breve{s}$ to $\breve{t}$ consisting of unsaturated arcs only, and increase the flow on each arc on it by the residual capacity of this chain; repeat this process until a stage is reached where there is no chain from $\breve{s}$ to $\breve{t}$ consisting only of unsaturated arcs. At that stage, terminate the process. We now have a feasible flow vector $\tilde{f}$ wrt which there exists no FAC from $\breve{s}$ to $\breve{t}$. This $\tilde{f}$ is a maximal or blocking feasible flow vector, but unfortunately, it may not be of maximum value, as illustrated in Figure 1.26 in Example 1.3. In this scheme, the flow on each arc either stayed the same or increased, but never decreased. To reach higher value flows, we may have to decrease the flow on some of the arcs. This leads to the possibil-

ity of flow augmentation using FAPs rather than FACs. For example, there is no FAC from 1 to 4 in the network on the right of Figure 1.30. However the path 1, (1, 3), 3, (2, 3), 2, (2, 4), 4 containing the reverse arc (2, 3) is an FAP, and augmentation using it leads to an increase in flow value by 1 unit. The first class of methods that we will discuss are based on flow augmentation using FAPs. These methods are also called **labeling algorithms** or **label tree methods**, since they grow a tree rooted at $\breve{s}$ in each step to look for an FAP, and the tree itself is stored using predecessor labels on the nodes.

2.1 Simple Transformations

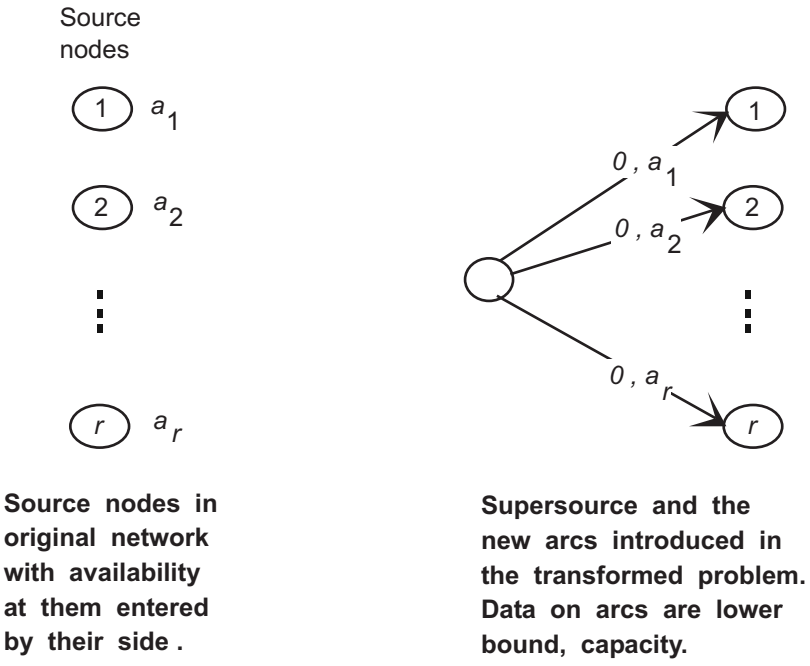
Supersource (Supersink) to Replace Several Source (Sink) Nodes

If there are $r(> 1)$ source nodes in the problem, i with a_i units available, for $i = 1$ to r, then introduce a new supersource $\breve{s}$ with unlimited availability and new arcs $(\breve{s}, i)$ with lower bound 0 and capacity $a_i, for i = 1$ to r. See Figure 2.1.

If there are $p(> 1)$ sink nodes, with sink node $n - j$ having a requirement of b_{n-j} units, for $j = p - 1$ to 0, introduce a new supersink $\breve{t}$ with unlimited requirement and new arcs $(n - j, \breve{t})$ for $j = p - 1$ to 0. If b_{n-j} is a minimal requirement at node $n - j$, the lower bound and capacity on the arc $(n - j, \breve{t})$ can be set at b_{n-j} and ∞ respectively. If it is required to supply exactly b_{n-j} units to node $n - j$, one of the following strategies can be used: (i) Set both lower bound and capacity on the arc $(n - j, \breve{t})$ equal to b_{n-j}, or (ii) set lower bound = 0, capacity $= b_{n-j}$ on arc $(n - j, \breve{t})$, and look for a flow vector in which this arc is saturated.

Transformation of Node Capacities

If node i has a node transit capacity of c_i in the original network, replace it by an arc (i_1, i_2) with an arc flow capacity of c_i. Nodes i_1, i_2 represent the *receiving and departing ends* of i. See Figure 2.2.



Source nodes in original network with availability at them entered by their side .

Supersource and the new arcs introduced in the transformed problem. Data on arcs are lower bound, capacity.

Figure 2.1:

Combining Parallel Arcs Into A Single Arc

If there are r parallel arcs joining i to j with lower bounds $\ell_1, \ldots, \ell_r$ and capacities $k_1, \ldots, k_r$ respectively, replace them by a single arc (i, j) with lower bound $\ell_1 + \ldots + \ell_r$ and capacity $k_1 + \ldots + k_r$. In the sequel we will assume that there is at most one arc from a node to any other node.

2.2 Some Results

The conditions for the feasibility of a node-arc flow vector $f = (f_{ij})$ in the single commodity flow network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ are:

$$f(i, \mathcal{N}) - f(\mathcal{N}, i) = \begin{cases} 0 & \text{if } i \neq \breve{s}, \breve{t} \\ v & \text{if } i = \breve{s} \\ -v & \text{if } i = \breve{t} \end{cases} \tag{2.1}$$

$$\ell \leqq f \leqq k \tag{2.2}$$

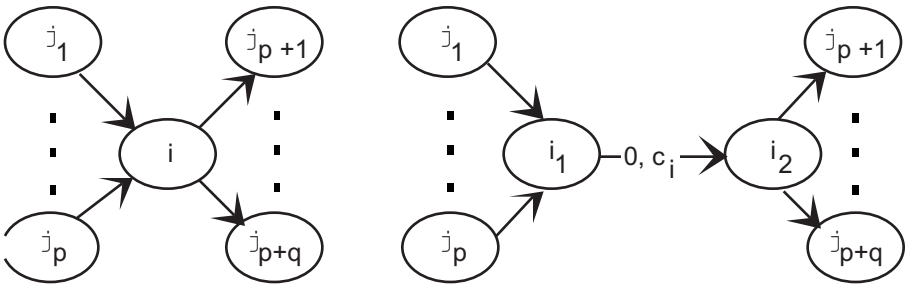


Figure 2.2: On the left is node i with transit capacity c_i in the original network. On right is the corresponding portion in the transformed network in which i is replaced by arc (i_1, i_2) with lower bound, capacity entered on it.

THEOREM 2.1 *Let [*$\mathbf{X}$, $\bar{\mathbf{X}}$*] be a cut separating* $\check{s}$ *and* $\check{t}$*, and let* f *be a feasible flow vector in* G*. The quantity* $f(\mathbf{X}, \bar{\mathbf{X}}) - f(\bar{\mathbf{X}}, \mathbf{X})$*, called the net flow across the cut [*$\mathbf{X}$, $\bar{\mathbf{X}}$*] in* f*, is equal to* v*.*

Proof The result follows by adding the conservation equations in (2.1) over $i \in \mathbf{X}$. ∎

THEOREM 2.2 *If a feasible flow vector exists in* G*, the maximum flow value is* $\leqq$ *the capacity of any cut separating* $\check{s}$ *and* $\check{t}$*. Also, if* $\hat{f}$ *is a feasible flow vector of value* $\hat{v}$*, [*$\mathbf{Y}$, $\bar{\mathbf{Y}}$*] is a cut separating* $\check{s}$ *and* $\check{t}$ *in* G*, and* $\hat{v}$ *= the capacity of the cut [*$\mathbf{Y}$, $\bar{\mathbf{Y}}$*], then* $\hat{f}$ *is a maximum value feasible flow vector and [*$\mathbf{Y}$, $\bar{\mathbf{Y}}$*] is a minimum capacity cut separating* $\check{s}$ *and* $\check{t}$ *in* G*.*

Proof Let f be any feasible flow vector of value v, and let $[\mathbf{X}, \bar{\mathbf{X}}]$ be any cut separating $\check{s}$ and $\check{t}$ in G. By Theorem 2.1 we have $v = f(\mathbf{X}, \bar{\mathbf{X}}) - f(\bar{\mathbf{X}}, \mathbf{X}) \leqq f(\mathbf{X}, \bar{\mathbf{X}}) - \ell(\bar{\mathbf{X}}, \mathbf{X})$ (since $f \geqq \ell$) $\leqq k(\mathbf{X}, \bar{\mathbf{X}}) - \ell(\bar{\mathbf{X}}, \mathbf{X})$ (since $f \leqq k$) = capacity of the cut $[\mathbf{X}, \bar{\mathbf{X}}]$. The second result now follows directly. ∎

THEOREM 2.3 *A feasible flow vector in* G *has maximum value iff there exists no FAP from* $\breve{s}$ *to* $\breve{t}$ wrt *it.*

Proof Let f be a feasible flow vector in G. If there exists an FAP from $\breve{s}$ to $\breve{t}$ wrt f, we can get a flow vector of higher value by augmentation, therefore f does not have maximum value.

Suppose there exists no FAP from $\breve{s}$ to $\breve{t}$ wrt f. We will now show that f has maximum value. Define $\mathbf{X} = \{i : i \in \mathcal{N}$, either $i = \breve{s}$, or there exists an FAP from $\breve{s}$ to i wrt $f\}$, $\bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X}$.

If $\mathcal{P}$ is an FAP from $\breve{s}$ to i wrt f (define $\mathcal{P}$ to be the empty path if $i = \breve{s}$), and $j \in \mathcal{N}$ is such that

$$\text{either } \ (\mathbf{i}) \ \ (i,j) \in \mathcal{A} \text{ and } \ f_{ij} < k_{ij}, \quad \text{or } \ (\mathbf{ii}) \ \ (j,i) \in \mathcal{A} \text{ and } \ f_{ji} > \ell_{ji}$$

then by including (i,j) under case (i) or (j,i) under case (ii) at the end of $\mathcal{P}$, we extend it to j. This implies that if $i \in \mathbf{X}$ and j satisfies (i) or (ii), then $j \in \mathbf{X}$ also. Hence, for $(i,j) \in \mathcal{A}$, we have $f_{ij} = k_{ij}$ if $i \in \mathbf{X}$ and $j \in \bar{\mathbf{X}}$, or $f_{ij} = \ell_{ij}$ if $i \in \bar{\mathbf{X}}$ and $j \in \mathbf{X}$. So, $f(\mathbf{X}, \bar{\mathbf{X}}) - f(\bar{\mathbf{X}}, \mathbf{X}) = k(\mathbf{X}, \bar{\mathbf{X}}) - \ell(\bar{\mathbf{X}}, \mathbf{X})$. And since there is no FAP from $\breve{s}$ to $\breve{t}$ wrt f, $\breve{t} \in \bar{\mathbf{X}}$. Therefore, $[\mathbf{X}, \bar{\mathbf{X}}]$ is a cut separating $\breve{s}$ and $\breve{t}$. By Theorems 2.1 and 2.2, these facts imply that f is a maximum value flow vector, and $[\mathbf{X}, \bar{\mathbf{X}}]$ is a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G. ∎

Given a feasible flow vector f in G, the argument in the proof of Theorem 2.3 suggests the following **labeling** or **tree growth scheme** to determine the set $\mathbf{X}$ of nodes defined there.

TREE GROWTH SUBROUTINE TO FIND $\mathbf{X}$

Step 1 Plant a tree with root at $\breve{s}$ Label $\breve{s}$ with $\emptyset$.

Step 2 Tree growth step Look for a labeled node i and an unlabeled node j satisfying (i) or (ii) stated above. If (i) holds, label j with $(i,+)$; if (ii) holds label j with $(i,-)$. In either case, i is the immediate predecessor of j and the arc (i,j) in case (i) or the arc (j,i) in case (ii) is known as the **arc used in labeling node** j. It becomes an in-tree arc in this step.

Repeat Step 2 as often as possible. Terminate when no further tree growth is possible.

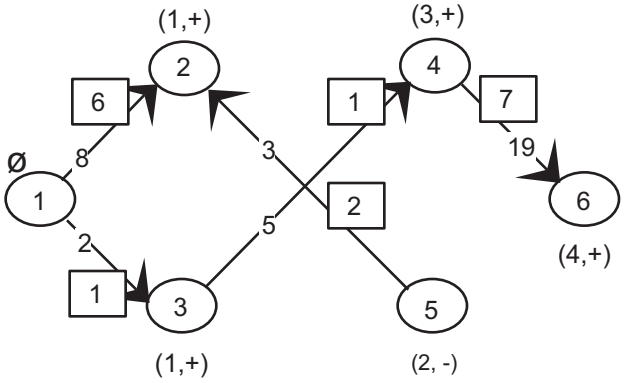


Figure 2.3: Only in-tree arcs are shown here. Node labels are entered by the side of the nodes. All lower bounds are 0. Arc capacities are entered on the arcs. If flow on an arc is nonzero, it is entered in a box by its side.

The set $\mathbf{X}$ defined in the proof of Theorem 2.3 is the set of all labeled (i.e., in-tree) nodes at termination. For each $i \in \mathbf{X}$, its predecessor path written in reverse order beginning with $\breve{s}$ is an FAP from $\breve{s}$ to i wrt f.

As an example consider the flow vector in the network in Figure 1.24. The rooted tree obtained when this scheme is applied on it is given in Figure 2.3. Nodes labeled in successive executions of Step 2 are 2, 3, 5, 4, 6, in that order. So, in this example $\mathbf{X}$ is $\mathcal{N}$.

The rooted tree grown is usually called the **label tree**. At some stage of the tree growth process, if $\breve{t}$ gets labeled, it is an indication that an FAP from $\breve{s}$ to $\breve{t}$ has been identified. This event is called a **breakthrough**, and the label tree is said to have become an **augmenting tree** when it occurs. It can be interpreted as the tree bearing fruit. On the other hand if the tree growth stops without $\breve{t}$ ever getting labeled, it is an indication that there exists no FAP from $\breve{s}$ to $\breve{t}$ wrt the present flow vector f; this event is known as a **nonbreakthrough**. By Theorem 2.3, this implies that f has maximum value in G.

THEOREM 2.4 *THE MAXIMUM FLOW MINIMUM CUT THEOREM If a feasible flow vector exists in the single commodity flow network* $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$*, the maximum value among feasible flow vectors is equal to the minimum capacity of cuts separating* $\breve{s}$ *and* $\breve{t}$*.*

Proof The maximum flow value in G is infinite iff there exists a chain from $\breve{s}$ to $\breve{t}$ consisting only of arcs of infinite capacity. If such a chain exists, by the result in Exercise 1.12, every cut separating $\breve{s}$ and $\breve{t}$ must contain an arc of this chain as a forward arc; hence the capacity of every cut separating $\breve{s}$ and $\breve{t}$ is also infinite, and hence the theorem holds.

If the maximum flow value in G is finite, this theorem follows from Theorems 2.2 and 2.3. ∎

Theorem 2.4 points out the connection between the maximum value flow problem and the minimum capacity cut problem. Since a cut separating $\breve{s}$ and $\breve{t}$ has the property of blocking all the paths between $\breve{s}$ and $\breve{t}$, the minimum capacity cut problem arises in disconnecting the network for interrupting the communication between $\breve{s}$ and $\breve{t}$ (i.e., for the interdiction of the physical transportation of supplies at minimum expense). A minimum capacity cut can be viewed as a minimum cost subset of arcs that intersects every path from $\breve{s}$ to $\breve{t}$. In fact, it is a project to evaluate the capacity of the Eastern European rail network to support a large scale conventional war, and the effort required for interdiction, formulated in 1956 by General F. S. Ross and T. E. Harris, that motivated L. R. Ford and D. R. Fulkerson to study the maximum value flow problem and led to their discovery of the maximum flow minimum cut theorem. See Picard and Queyranne [1982], Billera and Lucas [1978], and Hoffman [1978].

Unfortunately, the corresponding result does not hold for multicommodity flow problems (i.e., those dealing with the simultaneous shipping of several commodities). Consider a p ($\geqq 2$) commodity flow problem on the directed network $\bar{\mathrm{G}} = (\mathcal{N}, \mathcal{A}, \ell = 0, k)$. Assume that each arc can be used for the flow of any combination of commodities, that all commodities are measured in a common unit (e.g., a truckload) and that the capacity on each arc applies to the sum of the flows of all the commodities. Clearly, this problem can be transformed into one in which there is a specified source and sink pair for each commodity. Let s_r, t_r be the source and sink for the rth commodity, $r = 1$ to p. Here a **disconnecting set of arcs** can be defined to be a subset of arcs whose removal disconnects all the chains from s_r to t_r for each $r = 1$ to p. We

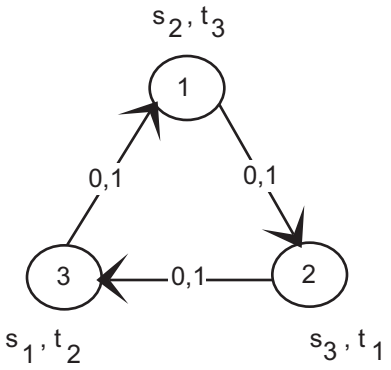


Figure 2.4: A three commodity flow network. Lower bound, capacity for total arc flow entered on the arcs; s_r, t_r are the source, sink nodes for the rth commodity, $r = 1$ to 3.

define the capacity of such a disconnecting set to be the sum of their capacities. In contrast to single commodity flows, the maximum value flow which maximizes the sum of the flow values of all the commodities, may be strictly less than the minimum disconnecting set capacity. We now present an example from Ford and Fulkerson [1962 of Chapter 1] to illustrate this point.

In the network in Figure 2.4, the maximum flow value is $\frac{3}{2}$ obtained by shipping $\frac{1}{2}$ unit of the rth commodity across the unique chain from s_r to t_r, $r = 1$ to 3. Any pair of arcs in this network, e.g., {(1, 2), (1, 3)}, is a disconnecting set, since removal of this pair disconnects the chains from s_r to t_r, for all $r = 1, 2, 3$. So, the minimum disconnecting set capacity is 2, strictly greater than the maximum flow value of $\frac{3}{2}$.

The Duality Interpretation of the Maximum Flow Minimum Cut Theorem

Consider the maximum value flow problem in the directed connected single commodity flow network $\bar{G} = (\mathcal{N}, \mathcal{A}, \ell = 0, k, \check{s}, \check{t})$. This is the problem of maximizing v subject to (2.1), (2.2) with $\ell = 0$. Associate the dual variable π_i to the conservation equation corresponding to node i, and the dual variable u_{ij} to the capacity constraint on arc (i, j). From Chapter 1 we know that any one of the equality constraints

in (2.1) can be eliminated because of redundancy. We eliminate the equation corresponding to $\breve{t}$. This has the effect of setting $\pi_{\breve{t}} = 0$ in the dual problem. Hence the dual problem is equivalent to

$$\begin{aligned} \text{Minimize } z(\pi, u) = \sum(k_{ij}u_{ij} \quad &: \quad \text{over } (i,j) \in \mathcal{A}) \\ \text{Subject to } \pi_j - \pi_i + u_{ij} \quad &\geqq \quad 0, \text{ for each } (i,j) \in \mathcal{A} \quad (2.3) \\ \pi_{\breve{s}} - \pi_{\breve{t}} \quad &= \quad 1 \\ \pi_{\breve{t}} = 0, \text{ and } u_{ij} \quad &\geqq \quad 0, \text{ for each } (i,j) \in \mathcal{A} \end{aligned}$$

Let $\mathbf{\Gamma} = \{(\pi, u) : (\pi, u)$ is feasible to (2.3) $\}$. From the structure of the constraints in (2.3), it is clear that the values of any of the u_{ij} variables can be increased arbitrarily in any feasible solution without affecting its feasibility. Thus $\mathbf{\Gamma}$ is an unbounded set in the π, u-space. Let $(\tilde{\pi}, \tilde{u}) \in \mathbf{\Gamma}$ be an extreme point of it. Then the results in Exercise 2.1 state that all $\tilde{\pi}$ and $\tilde{u}_{ij}$ are 0 or 1, that $[\mathbf{X}, \bar{\mathbf{X}}]$ (where $\mathbf{X} = \{i : i \in \mathcal{N}$ is s. t. $\tilde{\pi}_i = 1\}$, and $\bar{\mathbf{X}}$ is its complement) is a cut separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$, and that the set $\{(i,j) : (i,j) \in \mathcal{A}$ is s. t. $\tilde{u_{ij}} = 1\}$ is $(\mathbf{X}, \bar{\mathbf{X}})$. Thus, $z(\tilde{\pi}, \tilde{u})$ is the capacity of the cut $[\mathbf{X}, \bar{\mathbf{X}}]$. Thus, every BFS (π, u) of (2.3) corresponds to a cut separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$, such that $z(\pi, u) =$ the capacity of this cut. Hence, by the duality theorem of LP, when feasible flows exist, the maximum flow value is equal to the minimum capacity of cuts separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$; this is the result in the maximum flow minimum cut theorem (Theorem 2.4). Also, the weak duality theorem of LP implies that the value of any feasible flow vector in $\bar{\mathrm{G}}$ is $\leqq$ the capacity of any cut separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$. This is the first result in Theorem 2.2.

Assume that k is a finite positive vector. Consider the problem of finding a maximum capacity cut separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$. Since there are only a finite number of cuts separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$, and each has finite capacity, this is a combinatorial optimization problem with a finite optimum objective value. The minimum capacity among cuts separating $\breve{s}$ and $\breve{t}$ is the optimum objective value in the LP (2.3). By analogy, one is tempted to look at the LP: maximize $\{z(\pi, u)$: over $(\pi, u) \in \mathbf{\Gamma}\}$, for the maximum capacity cut problem. However, since $\mathbf{\Gamma}$ is unbounded, and $k > 0$, $z(\pi, u)$ is unbounded above on $\mathbf{\Gamma}$. So, the

maximum capacity cut cannot be found directly from this LP; $z(\pi, u)$ has a finite minimum over $\mathbf{\Gamma}$, but is unbounded above. See Figure 2.5.

Since only extreme points of $\mathbf{\Gamma}$ correspond to cuts, the maximum capacity cut problem is exactly the discrete optimization problem: maximize $\{z(\pi, u) :$ over the finite set of extreme points of $\mathbf{\Gamma}\}$. And the minimum capacity cut problem is to minimize $z(\pi, u)$ over the finite set of extreme points of $\mathbf{\Gamma}$. Both are discrete optimization problems requiring the optimization of a linear function over the finite set of extreme points of $\mathbf{\Gamma}$. So, on the surface both problems seem very comparable. The main difference between them is that there exists a solution to the minimization problem which solves the LP (2.3) without any extreme point condition. Better still, given an extreme point of $\mathbf{\Gamma}$, there are necessary and sufficient optimality conditions to check efficiently whether that point solves the minimization problem. In contrast, given an extreme point of $\mathbf{\Gamma}$, no nontrivial optimality conditions (short of total enumeration, i.e., comparing this point with every other extreme point of $\mathbf{\Gamma}$) are known to check whether it solves the maximization problem.

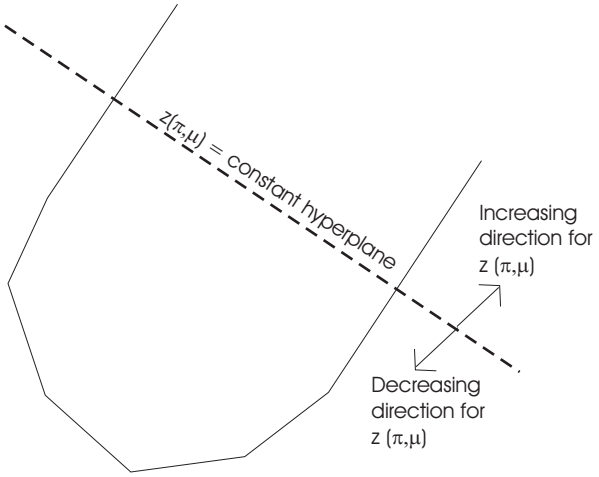


Figure 2.5:

The maximum capacity cut problem is a special case of the following:

GENERAL PROBLEM Given a finite system of linear constraints, for which the set of feasible solutions is an unbounded convex polyhedron $\mathbf{K}$, and a linear function $z(x)$ unbounded above over $\mathbf{K}$, find an extreme point of $\mathbf{K}$ which maximizes $z(x)$ over the finite set of extreme points of $\mathbf{K}$.

It is a difficult discrete optimization problem for which no efficient algorithms are known. When all the data is rational, this problem is NP-hard. Several other hard combinatorial optimization problems such as the traveling salesman problem are special cases of this problem. At the moment, the only methods known for this problem are enumerative methods with exponential growth in computational effort in the worst case.

Exercises

2.1 Prove the following. If $(\tilde{\pi} = (\tilde{\pi}_i), \tilde{u} = (\tilde{u_{ij}}))$ is a BFS of (2.3), then (i) $\tilde{\pi}_i, \tilde{u_{ij}}$ are all equal to either 0 or 1 for all i, j, (ii) for each $(i, j) \in \mathcal{A}, \tilde{u_{ij}} = 1$ iff $\tilde{\pi}_i = 1$ and $\tilde{\pi}_j = 0$; $\tilde{u_{ij}} = 0$ otherwise. Conversely, if $(\tilde{\pi}, \tilde{u})$ is a feasible solution of (2.3) satisfying (i) and (ii), then it is a BFS. Hence show that a feasible solution $(\tilde{\pi}, \tilde{u})$ for (2.3) is an extreme point of $\mathbf{\Gamma}$ iff it satisfies (i) and (ii). Then show that every BFS (π, u) of (2.3) corresponds to a cut separating $\breve{s}$ and $\breve{t}$ in $\bar{\mathrm{G}}$, such that $z(\pi, u)$ = the capacity of that cut, and vice versa.

2.2 Let $A = (a_{ij})$ be a given positive square matrix of order n. For any $\mathbf{X} \subset \{1, \ldots, n\}$, define $\bar{\mathbf{X}} = \{1, \ldots, n\} \setminus \mathbf{X}$. Consider the four problems of minimizing or maximizing $a(\mathbf{X}, \bar{\mathbf{X}})$, or $a(\mathbf{X}, \bar{\mathbf{X}}) + a(\bar{\mathbf{X}}, \mathbf{X})$, over the class of subsets of $\{1, \ldots, n\}$. Which of these problems can be solved by network flow methods? Why?

2.3 Research Problem : Develop a system of linear constraints in π, u, which, when combined with those in (2.3), describes the convex hull of the set of extreme points of $\mathbf{\Gamma}$.

2.3 Single Path Labeling Methods Beginning With A Feasible Flow Vector

We consider the problem of finding a maximum value feasible flow vector in the directed single commodity flow network G = $(\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$. It involves two phases. They are:

PHASE 1 This phase finds an initial feasible flow vector in G if one exists. If $\ell = 0$, $f = 0$ is a feasible flow vector. Hence this phase is not needed, and we go directly to Phase 2 with $f = 0$. Methods for carrying out Phase 1 when $\ell \neq 0$ are discussed in Section 2.6.

PHASE 2 This phase requires an initial feasible flow vector, say f^0 in G, as an input. This phase constructs a sequence of feasible flow vectors of strictly increasing values, and terminates only when a maximum value feasible flow vector is obtained.

Here we discuss a class of algorithms for Phase 2. In each stage, they try to find an FAP from $\breve{s}$ to $\breve{t}$ wrt the present feasible flow vector $\bar{f}$, say. If none are found, $\bar{f}$ is a maximum value flow vector, and the method terminates. Otherwise an FAP is found, flow augmentation is carried out using it, and the whole process is repeated with the new flow vector. Since the methods deal with one FAP at a time, they are called **single path methods**. The methods differ in the manner in which the search for an FAP is carried out in each step.

2.3.1 An Initial Version of the Labeling Method

This version uses the tree growth scheme discussed earlier to search for FAPs. Tree growth occurs one arc at a time.

INITIAL VERSION

Step 1 Plant a tree with root at $\breve{s}$ Let $\bar{f} = (\bar{f}_{ij})$ be the present feasible flow vector. Label $\breve{s}$ with $\emptyset$.

Step 2 Tree growth Look for a pair of nodes i, j satisfying one of the following.

(i) Forward labeling rule Node i is labeled; j is unlabeled; $(i,j) \in \mathcal{A}$ and $\bar{f}_{ij} < k_{ij}$

(ii) Reverse labeling rule Node i is labeled; j is unlabeled; $(j,i) \in \mathcal{A}$ and $\bar{f}_{ji} > \ell_{ji}$

If such a pair does not exist, there is a nonbreakthrough. Go to Step 4. If a pair of nodes i, j satisfying one of the above rules is found, label j with $(i,+)$ under rule (i), or with $(i,-)$ under rule (ii). Node i is the immediate predecessor of j. If $j = \breve{t}$, there is a breakthrough. Go to Step 3. Otherwise, repeat this Step 2.

Step 3 Flow augmentation Since $\breve{t}$ is labeled, the predecessor path $\mathcal{P}$ of $\breve{t}$, written in reverse order beginning with $\breve{s}$ is an FAP. Compute ϵ, the residual capacity of $\mathcal{P}$, and $\hat{f} = (\hat{f}_{ij})$ where

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} + \epsilon & \text{if } (i,j) \text{ is a forward arc on } \mathcal{P} \\ \bar{f}_{ij} - \epsilon & \text{if } (i,j) \text{ is a reverse arc on } \mathcal{P} \\ \bar{f}_{ij} & \text{if } (i,j) \text{ is not on } \mathcal{P} \end{cases}$$

Erase the labels on all the nodes (this operation is called **chopping down the present tree**) and go back to Step 1 with $\hat{f}$ as the new flow vector.

Step 4 Termination The present flow vector $\bar{f}$ is a maximum value flow vector in G. Let $\mathbf{X}$ = set of in-tree (i.e., labeled) nodes, and $\bar{\mathbf{X}}$ its complement. ($\mathbf{X}$, $\bar{\mathbf{X}}$) is a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G. Terminate.

Discussion

In executing the algorithm, each time check whether the sink can be labeled. If so, label it and move over to Step 3. If the sink cannot be labeled, label some other unlabeled node if possible and continue. Each time Step 2 is carried out, one new node and arc join the tree. So, Step 2 can be carried out consecutively at most $n-1$ times ($n =$

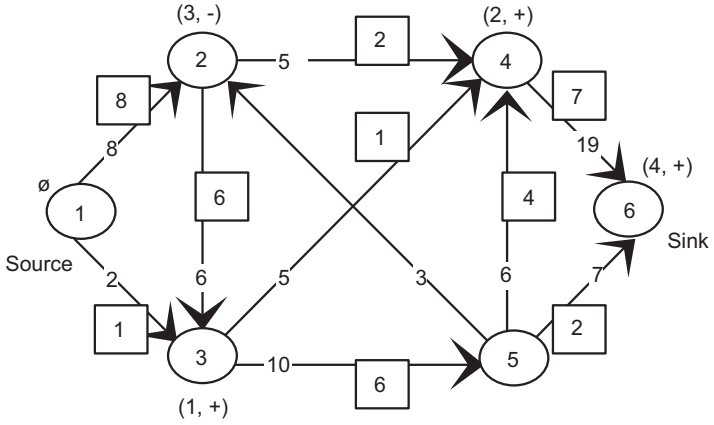


Figure 2.6:

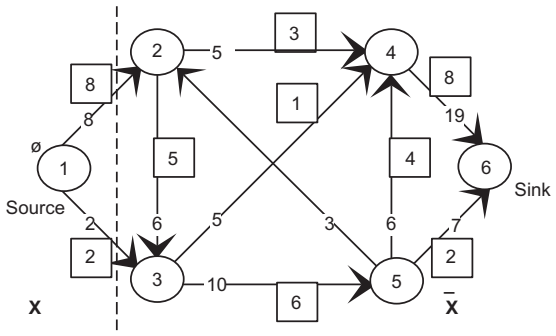


Figure 2.7:

$|\mathcal{N}|$) before going to Step 3 or 4. The work between two consecutive occurrences of Steps 3 and 4 represents a **tree growth routine**, also called a **labeling routine**. It terminates either with a breakthrough or a nonbreakthrough. The total number of trees grown in this algorithm is 1 + the number of flow augmentations carried out.

As an example, consider the network in Figure 1.24 with a feasible flow vector of value 7 entered there. All lower bounds are zero, and capacities are entered on the arcs. Nonzero flow amounts are entered in little squares by the side of the arcs. We label the source, node 1 with $\emptyset$, then node 2 with $(1, +)$, then node 5 with $(2, -)$, and then the sink, node 6 with $(5, +)$, in that order. So, there is a breakthrough.

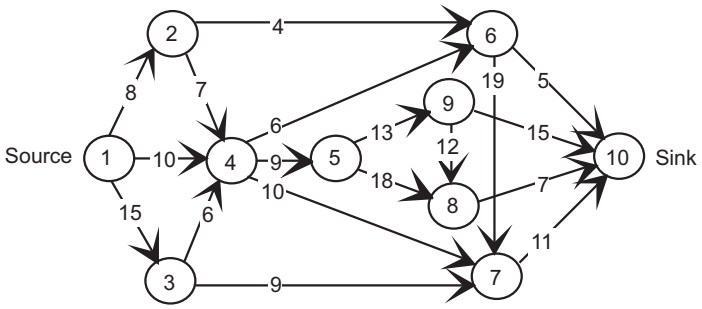


Figure 2.8: All lower bounds are 0, and capacities are entered on the arcs.

The FAP has forward arcs (5, 6), (1, 2) and reverse arc (5, 2); $\epsilon_1 = \min\{7-0, 8-6\} = 2$, $\epsilon_2 = \min.\{2\} = 2$, the residual capacity $\epsilon = \min\{\epsilon_1, \epsilon_2\} = 2$. The new flow vector is shown in Figure 2.6.

We chop down the old tree, root another tree in Figure 2.6, and grow it. There is a breakthrough again. The new FAP has forward arcs (4, 6), (2, 4), and (1, 3) and reverse arc (2, 3). Flow augmentation is carried out and the new flow vector is shown in Figure 2.7. A new tree grown in Figure 2.7 ended in nonbreakthrough. So, the flow vector in Figure 2.7, of value 10, is a maximum value flow vector in this network. A minimum capacity cut $[\mathbf{X}, \bar{\mathbf{X}}]$ separating the source and the sink is marked in Figure 2.7 with a dashed line.

Exercises

2.4 Discuss how to find a feasible flow vector of value equal to a specified number v^* in G.

2.5 Find a maximum value flow vector from source to sink in the networks in Figures 2.8 to 2.10.

2.6 Prove that there exists r arc disjoint chains from $\breve{s}$ to $\breve{t}$ in the directed network $G = (\mathcal{N}, \mathcal{A}, \breve{s}, \breve{t})$ iff there is no cut separating $\breve{s}$ and $\breve{t}$ with less than r forward arcs.

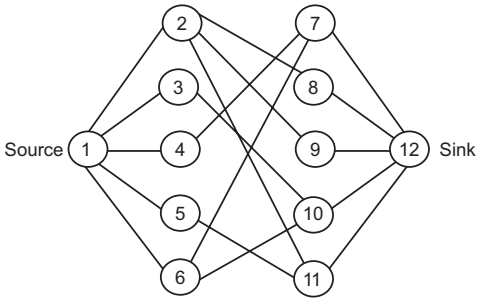


Figure 2.9: All the arcs are directed from left to right, all lower bounds are 0, and capacities are 1.

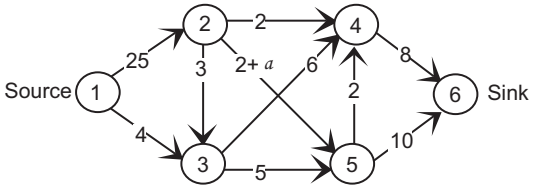


Figure 2.10: Here λ is a parameter. All lower bounds are 0, and capacities are entered on the arcs. Obtain the solution to the problem as a function of λ, for $\lambda \geqq 0$.

2.7 In a club of m men and n women, **compatibility** is a mutual relationship in a man-woman pair. The subset of women compatible with the ith man, $i = 1$ to m is given. Formulate the problem of forming the maximum number of compatible couples as a maximum value flow problem.

2.8 Maximum Value Flows in Undirected Networks Consider the maximum value flow problem in a connected undirected network $G = (\mathcal{N}, \mathcal{A}, \ell = 0, k, \check{s}, \check{t})$. One way of handling this problem is to replace each edge $(i; j)$ by the pair of arcs $(i, j), (j, i)$ both with lower bound 0 and capacity $k_{i;j}$, but this is undesirable because it doubles the number of lines. Another way is to orient each edge arbitrarily. For example, make edge $(i; j)$ into arc (i, j) say, with capacity $k_{i;j}$ and lower bound $-k_{i;j}$. Adopt the convention that if the flow amount on

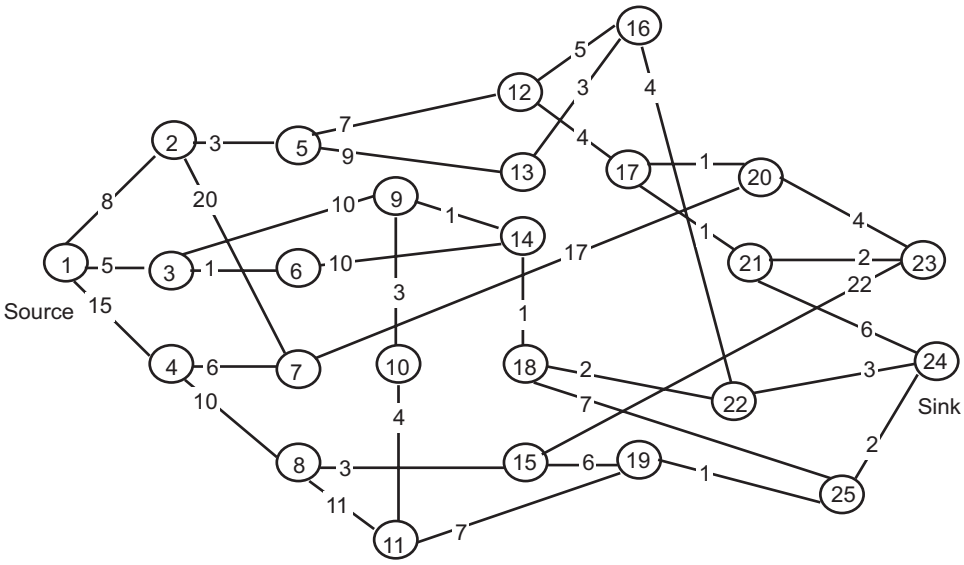


Figure 2.11:

this arc is a negative number, say $-\alpha$, it implies that the actual flow on the corresponding edge is $+\alpha$ in the direction opposite to the selected orientation for this edge. Use this approach to solve the maximum value flow problem on the network in Figure 2.11. All lower bounds are zero, and capacities are entered on the edges.

If there exists an FAP from $\breve{s}$ to $\breve{t}$ wrt a feasible flow vector $\bar{f}$, all the nodes along this path can be labeled and hence $\breve{t}$ can be labeled. Hence the sink will be labeled after at most $(n-1)$ tree growth steps, where $n = |\mathcal{N}|$, and a breakthrough will occur.

If ℓ, k, and the initial feasible flow vector f^0 are all integer vectors, the residual capacity of all the FAPs identified will always be a positive integer. Hence, in this case, either this version terminates with a maximum value feasible flow vector after a finite number of trees are grown, or it constructs an infinite sequence of flow vectors whose values diverge to $+\infty$.

Given an FAP which is a simple path, by labeling the nodes in

the order in which they appear on it, we can guarantee that the tree growth routine will terminate by discovering that particular FAP. So, if $\{f^0, f^1, \ldots\}$ is a sequence of feasible flow vectors of increasing value, satisfying the following Property 1, then we are guaranteed that this version can be executed, beginning with f^0, so that it produces exactly this sequence of flow vectors. Hence, in order to establish the worst case computational complexity of this version, it is enough to study the following question: "What is the maximum number of feasible flow vectors in a sequence $\{f^0, f^1, \ldots\}$ satisfying Property 1 ?" Edmonds and Karp [1972] have constructed the network in Figure 2.12 to answer this question.

PROPERTY 1 For each r, f^{r+1} is obtained by augmenting the flow vector f^r using some FAP (which is a simple path) from $\breve{s}$ to $\breve{t}$ wrt f^r.

Let f_t denote the flow amount on arc e_t, $t = 1$ to 5 in the network in Figure 2.12. $\bar{f} = (\bar{f}_t : t = 1\text{to } 5)^T = (M, M, 0, M, M)^T$ of value $2M$ is a maximum value feasible flow vector. Define the sequence of feasible flow vectors $\{f^s : s = 0, 1, \ldots, 2M\}$ in this network, with $f^0 = 0$, $f^{2r-1} = (r, r-1, 1, r-1, r)^T$, $f^{2r} = (r, r, 0, r, r)^T$, for $r = 1$ to M. In this sequence f^s has value s for each $s = 0$ to $2M$. For $r = 1$ to M, f^{2r-1} is obtained by augmenting f^{2r-2} using the FAP 1, (1, 2), 2, (2, 3), 3, (3, 4), 4 wrt it; and f^{2r} is obtained by augmenting f^{2r-1} using the FAP 1, (1, 3), 3, (2, 3), 2, (2, 4), 4 wrt it. So this sequence of flow vectors satisfies Property 1, and it has $2M + 1$ vectors in it. Assuming that M is a positive integer, the size of the maximum value flow problem on the network in Figure 2.12 (i.e., the number of binary digits needed to store all the data in it) is $\log(1+M)$ plus a constant, and the computational effort required by this version to solve this problem, if it follows this sequence, is that for $2M$ tree growths, which grows exponentially with the size. Thus even though the initial version is a finite algorithm for problems with finite integer data, it is not a polynomially bounded algorithm. Even on networks with few nodes and arcs, it may require an unduly large amount of computational effort depending on the order in which the nodes are selected for labeling.

EXAMPLE 2.1

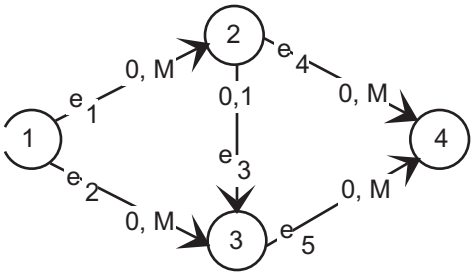


Figure 2.12: Network displaying the worst case behavior of the initial version of the labeling method. Lower bounds, capacities entered on the arcs. M is a positive integer.

Ford and Fulkerson [1962 of Chapter 1], have provided the following example to illustrate the fact that if the arc capacities are irrational numbers, and the starting feasible flow vector is an integer vector, the initial version of the labeling method may not terminate and may even converge in the limit to a flow vector whose value is strictly less than the true maximum flow value. Let $\alpha = \frac{-1+\sqrt{5}}{2}$, an irrational number satisfying $\alpha^{\mathbf{r}+2} = \alpha^{\mathbf{r}} - \alpha^{\mathbf{r}+1}$, for integer $r \geq 0$. Since $0 < \alpha < 1$, the infinite sum $\sum_{p=0}^{\infty} \alpha^{\mathbf{p}}$ converges to a quantity which we denote by β. The network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ is given by the following.

$$\mathcal{N} = \{\check{s}, \check{t}, x_i, y_i, i = 1 \text{ to } 4\}, |\mathcal{N}| = 10$$

$$\mathcal{A} = \{(\check{s}, x_i), (y_i, \check{t}) : i = 1 \text{ to } 4\} \cup \{(y_i, y_j), (x_i, y_j), (y_i, x_j) : \text{ for } i \neq j = 1 \text{ to } 4\} \cup \{e_i = (x_i, y_i) : i = 1 \text{ to } 4\}$$

with $\ell = 0$, capacities of e_1, e_2, e_3, e_4 to be 1, $\alpha, \alpha^{\mathbf{2}}, \alpha^{\mathbf{2}}$ respectively, and the capacities of all other arcs to be β. In the initial feasible flow vector, f^0, there is one unit of flow on arcs $(\check{s}, x_1), (x_1, y_1)$, and $(y_1, \check{t})$, and zero flow on all the other arcs. Hence, $v(f^0)$, the value of f^0, is 1. In f^0, the residual capacities of the arcs e_1, e_2, e_3, e_4 are 0, $\alpha, \alpha^{\mathbf{2}}, \alpha^{\mathbf{2}}$ respectively. The following construction yields two feasible flow vectors per step and generates a sequence of feasible flow vectors in this network satisfying Property 1.

GENERAL STEP At the beginning of this step, we have a flow vector, f^p, say, with value v_p, such that there exists a permutation of the arcs e_1, e_2, e_3, e_4, which we will denote by e'_1, e'_2, e'_3, e'_4 with residual capacities $0, \alpha^{\mathbf{r}}, \alpha^{\mathbf{r}+1}, \alpha^{\mathbf{r}+1}$ respectively, in f^p, for some r. Denote the tail and head nodes on e'_i by $x'_i, y'_i, i = 1$ to 4. The chain $\check{s}, (\check{s}, x'_2), x'_2, (x'_2, y'_2), y'_2, (y'_2, x'_3), x'_3, (x'_3, y'_3), y'_3, (y'_3, \check{t}), \check{t}$, is an FAC wrt f^p whose residual capacity is $\alpha^{\mathbf{r}+1}$. Denote the flow vector obtained after augmenting f^p using this FAC by f^{p+1}. The arcs e'_1, e'_2, e'_3, e'_4 have residual capacities $0, \alpha^{\mathbf{r}} - \alpha^{\mathbf{r}+1} = \alpha^{\mathbf{r}+2}, 0, \alpha^{\mathbf{r}+1}$ respectively in f^{p+1}. The path consisting of arcs $(\check{s}, x'_2), (x'_2, y'_2), (y'_2, y'_1), (x'_1, y'_1), (x'_1, y'_3), (x'_3, y'_3), (x'_3, y'_4), (y'_4, \check{t})$, in that order is an FAP wrt f^{p+1} with residual capacity $\alpha^{\mathbf{r}+2}$. Let f^{p+2} be the feasible flow vector obtained after augmenting f^{p+1} using this FAP, and v_{p+2} its value. $v_{p+2} - v_p = \alpha^{\mathbf{r}}$. In f^{p+2}, the arcs e'_1, e'_2, e'_3, e'_4 have residual capacities $\alpha^{\mathbf{r}+2}, 0, \alpha^{\mathbf{r}+2}, \alpha^{\mathbf{r}+1}$ respectively. So, rearrange them in the order e'_2, e'_4, e'_1, e'_3, and go to the next step.

This procedure leads to an infinite sequence of feasible flow vectors satisfying Property 1, whose value is strictly increasing and converges to $\sum_{r=0}^{\infty} \alpha^{\mathbf{r}} = \beta$, whereas the maximum flow value in this network can be verified to be 4β. So, even though the sequence $\{f^p : p = 0, 1, \ldots\}$ converges to a limit, its limit is not a maximum value flow vector in this example.

2.3.2 The Scanning Version of the Labeling Method

In the initial version, when $\bar{f}$ is the present feasible flow vector, we search for a labeled node i and an unlabeled node j satisfying either (i) or (ii) of Step 2. If such a labeled node i is found, we can label not only this node j, but all other unlabeled nodes j that satisfy this condition with i. This operation of labeling all the unlabeled nodes j satisfying this condition is called **scanning the labeled node** i. The use of scanning leads to an improved version. This version is often called the **Ford-Fulkerson Labeling Method**. Here, nodes may be in three possible states, **unlabeled, labeled and unscanned, labeled and scanned**. **List** always refers to the present set of labeled and unscanned nodes.

SCANNING VERSION

Step 1 Plant a tree with root at $\breve{s}$ Let $\bar{f} = (\bar{f}_{ij})$ be the present feasible flow vector. Label $\breve{s}$ with $\emptyset$; $\breve{s}$ is now labeled and unscanned. List $= \{\breve{s}\}$.

Step 2 Select a node from list for scanning If list $= \emptyset$, go to Step 5. Otherwise select a node from it to scan.

Step 3 Scanning Let i be the node to be scanned.

(i) Forward labeling Identify all unlabeled nodes j satisfying $(i, j) \in \mathcal{A}$ and $\bar{f}_{ij} < k_{ij}$, and label all of them with $(i, +)$.

(ii) Reverse labeling Identify all unlabeled nodes j satisfying $(j, i) \in \mathcal{A}$ and $\bar{f}_{ji} > \ell_{ji}$, and label all of them with $(i, -)$.

Each newly labeled node in this step is now labeled and unscanned. Include all of them in the list. Node i is now labeled and scanned. Delete it from the list. If $\breve{t}$ is now labeled, there is a breakthrough, go to Step 4. If $\breve{t}$ is not yet labeled. Go to Step 2.

Step 4 Flow augmentation Same as Step 3 in the initial version.

Step 5 Termination Same as Step 4 in the initial version.

Discussion

As an example consider the network in Figure 2.13 with an initial feasible flow vector of value 1 marked there. When the source node 1 is scanned, both nodes 2 and 3 get labeled. At this stage the list is {2, 3}, and we select 2 from it for scanning next. This leads to a breakthrough. Node labels are shown in Figure 2.13. Notice that even though 1, (1, 3), 3, (2, 3), 2, (2, 4), 4 is an FAP, which is a simple path, we cannot obtain this FAP under the scanning version, because, when node 1 is scanned, both nodes 3 and 2 get labeled with (1, +).

Flow augmentation can be carried out using the FAP identified in Figure 2.13, and the method continued. It can be verified that the

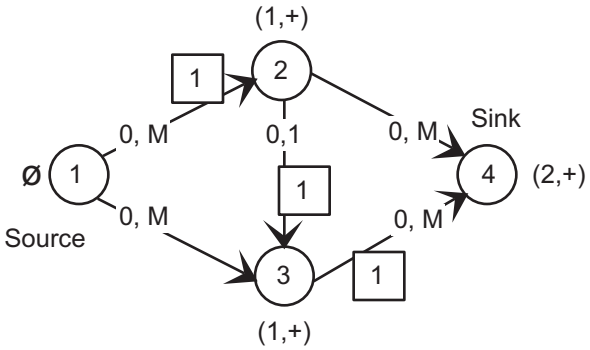


Figure 2.13: Data on the arcs is lower bound, capacity. Here $M > 2$. Nonzero flow amounts are entered in boxes by the side of the arcs.

method terminates with the maximum value flow vector after growing three trees. This compares with $2M$ trees that the initial version may require.

Since it may not be possible to obtain some FAPs under the scanning version, we cannot use the technique of constructing a sequence of flow vectors satisfying Property 1 of Section 2.3.1 to study its computational complexity, at least not in all the networks. However, we will now show that given any network G, we can obtain a modified network $\hat{\text{G}}$ such that the behavior of the initial version, when applied on G, is exactly duplicated by the scanning version, when applied on $\hat{\text{G}}$, by choosing an appropriate order for scanning the nodes from the list. The modification consists of adding an artificial node in the middle of each arc in G (i.e., replacing each arc (i, j) in G by the pair of arcs $(i, p), (p, j)$ both with the same data and the same flow amount as the original arc (i, j) in G) where p is the artificial node introduced on (i, j).

Let $\text{G} = (\mathcal{N}, \mathcal{A})$ be the original network with $|\mathcal{N}| = n$, $|\mathcal{A}| = m$, and $\hat{\text{G}} = (\hat{\mathcal{N}}, \hat{\mathcal{A}})$ the corresponding modified network. So, $|\hat{\mathcal{N}}| = n + m, |\hat{\mathcal{A}}| = 2m$. Each arc in G corresponds to a unique pair of arcs in $\hat{\text{G}}$. Under this correspondence every path in G corresponds to a unique path in $\hat{\text{G}}$ with twice as many arcs. As an example, the modified network corresponding to the one given in Figure 2.13 is shown in Figure 2.14.

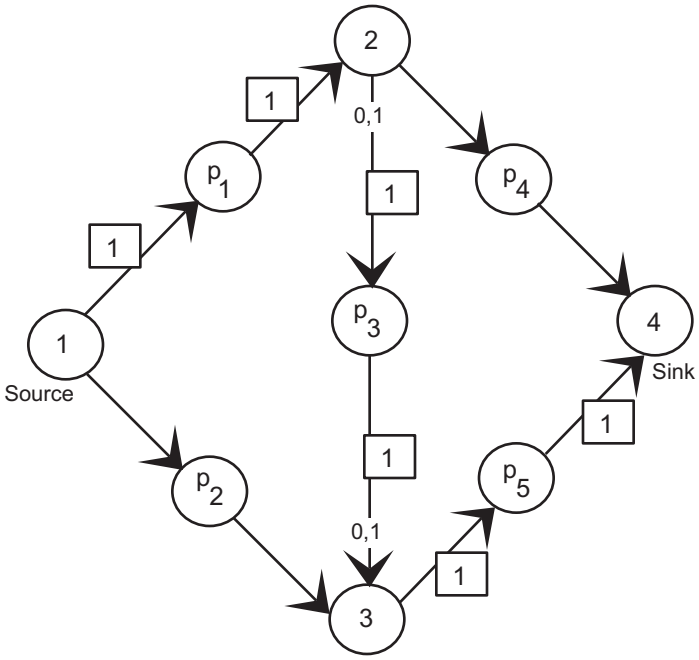


Figure 2.14: Modified network corresponding to the one in Figure 2.13. Lower bound, capacity on nonvertical arcs are 0, M; p_1 to p_5 are the artificial nodes inserted.

Let $e_1, \ldots, e_m$ be the arcs in G, and p_r the artificial node introduced in the middle of $e_r, r = 1$ to m. Let e'_r, e''_r be the pair of arcs into which e_r is split when p_r is introduced in its middle. Let f_r be the flow variable associated with e_r in G, and f'_r, f''_r the flow variables associated with e'_r, e''_r in $\hat{\mathrm{G}}$, $r = 1$ to m. In applying the scanning version on $\hat{\mathrm{G}}$, the following facts can be verified to hold.

In every feasible flow vector $(f'_r, f''_r : r = 1 \text{ to } m)$ in $\hat{\mathrm{G}}$, $f'_r = f''_r$ for all r, and by making f_r equal to this common quantity, $r = 1$ to m, we get a feasible flow vector in G, and vice versa. In $\hat{\mathrm{G}}$ an artificial node p_r gets labeled only when tail(e'_r) or head(e''_r) is scanned. When an original node $i \in \mathcal{N}$ is scanned in $\hat{\mathrm{G}}$, the only nodes that get labeled are artificial nodes. When an artificial node p_r is scanned in $\hat{\mathrm{G}}$, at most one node gets labeled; it is either tail(e'_r), or head(e''_r).

From these facts we conclude that if $\hat{\mathcal{P}}$ is an FAP from $\check{s}$ to $\check{t}$ in $\hat{\mathrm{G}}$

wrt a feasible flow vector $\hat{f}$, when the scanning version is applied on $\hat{\mathrm{G}}$ with $\hat{f}$ as the flow vector, it is possible to scan the nodes on $\hat{\mathcal{P}}$ in the order in which they appear on it, and if this is done, the tree growth routine will terminate by discovering the FAP $\hat{\mathcal{P}}$.

As an example, let $\mathcal{P}$ be the FAP 1, (1, 3), 3, (2, 3), 2, (2, 4), 4 in the network G in Figure 2.13. We have seen earlier that this FAP cannot be obtained under the scanning version. The corresponding FAP, $\hat{\mathcal{P}}$, in the modified network $\hat{\mathrm{G}}$ drawn in Figure 2.14 is 1, (1, p_2), p_2, (p_2, 3), 3, (p_3, 3), p_3, (2, p_3), 2, (2, p_4), p_4, (p_4, 4), 4. It can be verified that the nodes 1, p_2, 3, p_3, 2, p_4 can be scanned in this order when the scanning version is applied in $\hat{\mathrm{G}}$, leading to the FAP $\hat{\mathcal{P}}$ at termination.

From these facts and the earlier results on the initial version, we conclude that if all the data and the initial feasible flow vector are all integral, then the scanning version will either terminate after a finite number of steps with a maximum value flow vector or obtain an infinite sequence of flow vectors whose value diverges monotonically to $+\infty$. And if some capacities are irrational and the initial feasible flow vector is an integer, the scanning version may produce an infinite sequence of flow vectors whose values converge to a quantity strictly less than the maximum flow value (i.e., it may not work). However, surprisingly, a simple selection rule fixes this problem.

A general selection rule, known as a **consistent labeling procedure**, is one in which the choice for the node to be scanned from the list is determined uniquely by the set of nodes currently in the list (i.e., whenever the list is a particular subset of nodes, it always selects the same node from that subset to scan). We have the following theorem by A. Tucker [1977] on the finite convergence of the scanning version operated with a consistent labeling procedure.

THEOREM 2.5 *When the scanning version is applied on the directed single commodity flow network* $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, \check{s}, \check{t})$ *beginning with an initial feasible flow vector* f^0*, using a consistent labeling procedure, the scanning version finds a maximum value flow vector after growing at most a finite number of trees.*

Proof Let $|\mathcal{N}| = n$. We do not assume that ℓ, k, f^0 are integer

vectors; the proof is valid in general. The proof is based on induction on n. Number the nodes serially, beginning with 1 for the source and ending with n for the sink. The statement of the theorem is obviously true for $n = 2$.

Induction Hypothesis The statement of the theorem is true in networks with number of nodes $\leq n-1$.

Now we will prove that the theorem must also hold for $G = (\mathcal{N}, \mathcal{A})$ with $|\mathcal{N}| = n$, under the induction hypothesis. Proof is by contradiction. Suppose the theorem does not hold in G. So, there exists a consistent labeling procedure such that when the scanning version is applied on G with it, the algorithm continues indefinitely. 1 is the source node and the flow on arcs of the form $(1, i)$ is never decreased during the algorithm. Let $\mathbf{S} = \{i : i \in \mathbf{A}(1)$, and $(1, i)$ is never saturated in the algorithm$\}$. $\mathbf{S} \neq \emptyset$ by the hypothesis. Suppose the consistent labeling procedure requires the selection of the node $p \in \mathbf{S}$ for scanning whenever the list is the set $\mathbf{S}$.

Let $\{f^0, f^1, \ldots\}$ be the sequence of flow vectors generated by the algorithm. From the definition of $\mathbf{S}$, there exists a finite positive integer r such that for all $i \in \mathbf{A}(1) \backslash \mathbf{S}$, $f_{1i}^u = k_{1i}$ for all $u \geq r$. Hence, for all $u \geq r$, during the uth tree growth in the algorithm, node p is scanned immediately after 1. Also since $(1, p)$ is never saturated in the algorithm, the results obtained will remain unchanged if k_{1p} is changed to $+\infty$. Make this change.

Form a new network $\bar{G} = (\bar{\mathcal{N}}, \bar{\mathcal{A}})$ by just coalescing node p into node 1 in G, leaving all the data unchanged. For $u \geq r$, let $\bar{f}^u$ be the flow vector obtained by deleting f_{1p}^u from f^u. Now apply the same algorithm on $\bar{G}$ beginning with the initial feasible flow vector $\bar{f}^r$ in it. Coalescing node p into 1 contracts the arc $(1, p)$ into node 1; this has no effect on the results of labeling in G, since k_{1p} was changed to ∞. The consistency of the labeling procedure for all nodes other than p is kept unaffected while applying the algorithm on $\bar{G}$. The sequence of scanning remains the same, since node p was being scanned immediately after node 1 in G. Thus applying the algorithm on $\bar{G}$ with the initial flow $\bar{f}^r$ is equivalent to applying it in G beginning with f^r. However, since $|\bar{\mathcal{N}}| = n - 1$, by the induction hypothesis, the algorithm finds a

maximum value flow vector in $\bar{G}$ after a finite number of iterations and terminates; a contradiction. Hence, the theorem must also hold for the network G with $|\mathcal{N}| = n$, under the induction hypothesis. Since the theorem is true when the network has only two nodes, by induction it is true in general. ∎

Denote the arcs in G by $e_1, \ldots, e_m$. Correspondingly, let $\ell = (\ell_1, \ldots, \ell_m), k = (k_1, \ldots, k_m)$, and $f = (f_1, \ldots, f_m)$ denote the lower bound, capacity, and flow vectors. Given a feasible flow vector $f = (f_t)$ in G define the **partition of arcs corresponding to** f to be $(\mathbf{L}_f, \mathbf{I}_f, \mathbf{U}_f)$, where $\mathbf{L}_f = \{t : f_t = \ell_t\}, \mathbf{I}_f = \{t : \ell_t < f_t < k_t\}, \mathbf{U}_f = \{t : f_t = k_t\}$. Clearly the total number of such distinct partitions is $\leq 3^{\mathbf{m}}$. Another proof of Theorem 2.5 comes from the result in the following exercise.

Exercise

2.9 Prove that the partitions of arcs corresponding to flow vectors in the sequence of feasible flow vectors generated by the scanning version using a consistent labeling rule applied on G are all distinct. And prove that the number of flow vectors in this sequence is $\leq n!$, where $n = |\mathcal{N}|$ (Megiddo and Galil [1979]).

The result in Exercise 2.9 shows that the number of flow augmentations carried out in the scanning version with a consistent labeling procedure is $\leq$ min. $\{n!, 3^{\mathbf{m}}\}$. Even though this number is finite, it grows very rapidly with n, m. In fact, Megiddo and Galil [1979] have constructed an infinite class of maximum value problems with integer capacities and zero lower bounds to demonstrate that the number of flow augmentations in the scanning version under a general consistent labeling procedure may grow exponentially with the size of the problem. Their construction involves the basic network structure of the type in Figure 2.14; however, the arc capacities are different. The network for the rth problem in their class is a combination of r similar structures placed side by side with the leftmost node of each structure

coinciding with the rightmost node of the structure to its left, with capacities of all the nonvertical arcs = $2^{\mathbf{r}}$, and those of vertical arcs = $2^{\mathbf{r}-1}$. This rth problem has $1+8r$ nodes and $10r$ arcs. They show that the scanning version with a consistent labeling procedure specified by them and initiated with the zero flow vector requires $2^{\mathbf{r}}$ flow augmentations before reaching a maximum value flow vector in the rth problem. This is exponential growth with size.

2.3.3 Shortest Augmenting Path Method

Define the length of a simple path to be the number of arcs in it. In this version, due to Edmonds and Karp [1972], the node to be scanned is selected from the list in order to guarantee that the FAP obtained is the shortest among all the FAPs from the source to the sink at that stage. This is the reason for its name. It is also known as the **Edmonds-Karp version of the labeling method**.

In this method, nodes are selected from the list for scanning on a **first labeled first scanned** basis (i.e., nodes are scanned in the order they are labeled). This makes the tree grow in a breadth-first manner; hence the method uses a **breadth-first search strategy** to find an FAP. It turns out that the FAPs obtained by a breadth-first search strategy are the shortest.

The method is the same as the scanning version of Section 2.3.2 with one minor difference. Here the list is maintained as a queue, from top to bottom. When new nodes are entered into the list in Step 3, they are always entered at the bottom. When a node is to be selected for scanning in Step 2, it is always the topmost node in the list.

As an example, we apply this method on the network in Figure 1.24 with the feasible flow vector entered there. Here the list has to be maintained as an ordered set. We order the nodes in it from left to right (left corresponds to top, and right to bottom). Source node 1 gets labeled with $\emptyset$, list = {1}. Node 1 is scanned leading to the label of (1, +) for both nodes 2 and 3. Node 1 leaves the list and nodes 2, 3 are added to the list at the right in some order, say in natural order, so the list is {2, 3} now. Node 2 is selected for scanning, node 4 gets labeled with (2, +), and node 5 with (2, −). The list is {3, 4,

5} now. Scanning of node 3 leads to no new labels. Then node 4 is scanned, leading to the label of (4, +) to the sink node 6, and hence a breakthrough. The FAP with arcs (1, 2), (2, 4), (4, 6), of length 3, can be verified to be a shortest FAP wrt the present flow vector. Flow augmentation can be carried out, and the method can be continued in the same manner.

For the sake of discussion let d_i denote the depth label for labeled node i at any stage of this algorithm. These d_i are not used or maintained in the algorithm. Clearly, d_i is the length of the FAP traced by the current labels, from the source to i at that stage. Also, since a node gets labeled only after its parent, the "first labeled, first scanned" policy used in this algorithm guarantees that the node selected for scanning is always a node with the smallest depth label among those in the list.

Define the **distance** of node i at any stage of this algorithm to be the length of a shortest FAP from source to i wrt the flow vector at that stage, or $+\infty$ if no FAP exists to i.

THEOREM 2.6 *In this version, the FAP from the source to any labeled node i traced by the labels is a shortest FAP from the source to i* wrt *the feasible flow vector at that stage.*

Proof Let $\bar{f}$ be the feasible flow vector at this stage, and let d_i denote the depth label for the labeled node i in the tree being grown. Let d be a positive integer.

Induction Hypothesis The following statements are true for any $r \leq d-1$.

1. During the labeling routine, if the depth label of the node being scanned is r, then the distance of all the unlabeled nodes at this time is $\geq r+1$.

2. For all labeled nodes i with $d_i = r$, the FAP from source to i traced by the current labels is a shortest FAP from source to i at that stage.

Statements 1 and 2 are obviously true for $r = 0$. We will now prove that under the induction hypothesis, these statements must also be true for $r = d$.

Let j be a labeled node with $d_j = d$. If the parent of j is p, then $d_p = d - 1$. Applying Statement 1 to the time when p was being scanned, we conclude that the distance of j at this stage is $\geqq d$. But the FAP from source to j traced by the current labels has length d, so it is a shortest FAP from source to j at this stage. This proves that Statement 2 must hold for $r = d$.

Let x be an unlabeled node at the time that j was being scanned; x must also be unlabeled earlier when p was being scanned, which implies by Statement 1 under the induction hypothesis that the distance of x must be $\geqq d$. However, since x remained unlabeled when all the labeled nodes of depth $\leqq d - 1$ were scanned, the distance of x cannot be d. So, the distance of x must be $\geqq d + 1$. This proves that Statement 1 also holds when $r = d$.

Hence, by induction, the statements in the induction hypothesis are true for all r. This proves the theorem. ∎

Now we will study the worst case computational complexity of this version. Let f^0 denote the initial feasible flow vector and f^u the flow vector obtained after u trees are grown and the growth of the $(u+1)$th tree is about to begin, for $u \geqq 1$. So, f^u is the current feasible flow vector during the growth of the $(u+1)$th tree. Let θ_i^{u+1} be the length of a shortest FAP from the source $\breve{s}$ to i, and ϑ_i^{u+1} the length of a shortest FAP from i to the sink $\breve{t}$ wrt f^u. These quantities are defined to be $+\infty$ if an FAP of the type in its definition does not exist. So, from Theorem 2.6, if i is a labeled node in the $(u+1)$th tree, its depth in this tree is θ_i^{u+1}.

THEOREM 2.7 *For all nodes i and $u \geqq 1$ we have $\theta_i^{u+1} \geqq \theta_i^u$, and $\vartheta_i^{u+1} \geqq \vartheta_i^u$.*

Proof We will first prove $\theta_i^{u+1} \geqq \theta_i^u$. If this is not true, there must exist a p and some nodes i, for which $\theta_i^{p+1} < \theta_i^p$. Obviously $p \geqq 1$. Let r be a node satisfying $\theta_r^p > \theta_r^{p+1} = \min. \{\theta_i^{p+1} : i \text{ such that } \theta_i^{p+1} < \theta_i^p\}$.

The source node $\breve{s}$ satisfies $\theta_{\breve{s}}^u = 0$ for all $u \geqq 1$. Also, for any $u \geqq 1$, if $\theta_i^u = 0, i$ must be $\breve{s}$. Since $\theta_r^{p+1} < \theta_r^p$, we have $r \neq \breve{s}$, and

hence $\theta_r^{p+1} \geqq 1$. Let the label on node r in the $(p+1)$th tree growth routine be $(j,+)$. (A proof similar to the following holds when the second symbol in the label is $-$.) Then (j,r) is an in-tree arc in the $(p+1)$th tree, so $f_{jr}^p < k_{jr}$, and from Theorem 2.6 we have

$$\theta_r^{p+1} = \theta_j^{p+1} + 1 \tag{2.4}$$

From (2.4) and the choice of r, we have $\theta_j^{p+1} \geqq \theta_j^p$. So

$$\theta_r^{p+1} \geqq \theta_j^p + 1 \tag{2.5}$$

Now, suppose $f_{jr}^{p-1} < k_{jr}$. By the manner in which scanning is done, it is clear that $\theta_r^p \leqq \theta_j^p + 1$. This and (2.5) together imply that $\theta_r^p \leqq \theta_j^p + 1 \leqq \theta_r^{p+1}$; this is a contradiction. Hence $f_{jr}^{p-1} = k_{jr}$. This, and the fact that (j,r) is a forward in-tree arc in the $(p+1)$th tree growth routine, together imply that (j,r) must have been a reverse arc in the FAP identified during the pth tree growth routine. By Theorem 2.6 and the manner in which labeling is done, this implies $\theta_j^p = \theta_r^p + 1$. This together with (2.5) implies that $\theta_r^{p+1} \geqq \theta_r^p + 2$, a contradiction to the choice of r. Hence we must have $\theta_i^{u+1} \geqq \theta_i^u$ for all i and $u \geqq 1$. A similar proof holds for $\vartheta_i^{u+1} \geqq \vartheta_i^u$ for all i and $u \geqq 1$. ∎

THEOREM 2.8 *Let m, n be the number of arcs, nodes respectively in the network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, \check{s}, \check{t})$. Beginning with an initial feasible flow vector in G, the shortest augmenting path method terminates with a maximum value flow vector after at most $mn/2$ trees are grown.*

Proof Let (i,j) be an arc on an FAP from $\check{s}$ to $\check{t}$ wrt a flow vector $\tilde{f}$ in G, and $\hat{f}$ the flow vector obtained after augmenting $\tilde{f}$ using this FAP. Then (i,j) is said to be a **critical forward arc** if $\hat{f}_{ij} = k_{ij}$, a **critical reverse arc** if $\hat{f}_{ij} = \ell_{ij}$, and a **critical arc** if it is either a critical forward or reverse arc.

Suppose (i,j) is a critical arc in the FAP obtained during the ath tree growth routine in the shortest augmenting path method applied on G beginning with an initial feasible flow vector, and again for the next time in the bth tree growth routine, $b > a$. Assume that (i,j)

was a critical forward arc in the FAP of the ath tree growth routine (a proof similar to the following holds if it is a critical reverse arc on this FAP). So,

$$f_{ij}^a = k_{ij} \tag{2.6}$$

There are two cases to consider. Either $f_{ij}^{b-1} = k_{ij}$, or $< k_{ij}$. Consider the case $f_{ij}^{b-1} = k_{ij}$ first. In this case (i, j) must be a critical reverse arc in the FAP of the bth tree growth routine. So, from the manner in which labeling is done we have $\theta_j^a = \theta_i^a + 1$, and $\theta_i^b = \theta_j^b + 1$. Also, from Theorem 2.7 we have $\theta_j^b \geqq \theta_j^a$. From all these we have $\theta_i^b = \theta_j^b + 1 \geqq \theta_j^a + 1 = \theta_i^a + 2$. Also, from Theorem 2.7 we have $\vartheta_i^b \geqq \vartheta_i^a$. So, in this case we have

$$\theta_i^b + \vartheta_i^b \geqq \theta_i^a + \vartheta_i^a + 2 \tag{2.7}$$

Now consider the case where $f_{ij}^{b-1} < k_{ij}$. From (2.6), we conclude that this can only happen if there exists a w such that $a + 1 \leqq w \leqq b - 1$, and in the FAP obtained during the wth tree growth routine (i, j) is a reverse arc. Using the same arguments as above, in this case we have

$$\theta_i^w + \vartheta_i^w \geqq \theta_i^a + \vartheta_i^a + 2 \tag{2.8}$$

However, since $b > w$, by Theorem 2.7, $\theta_i^b + \vartheta_i^b \geqq \theta_i^w + \vartheta_i^w$. This, together with (2.8) implies that (2.7) holds in this case also. Thus (2.7) holds always.

If node i is on the FAP obtained in the uth tree growth routine, the length of that FAP is $\theta_i^u + \vartheta_i^u$. By (2.7) we therefore conclude that each succeeding FAP obtained during the algorithm in which the arc (i, j) is critical, is longer than the preceding one by at least two arcs. Also, in this algorithm, an FAP can have at most $n - 1$ arcs. So, no arc can appear as a critical arc in an FAP obtained during this version more than $n/2$ times. So, the total number of FAPs obtained during this version cannot exceed $mn/2$. ∎

The result in Theorem 2.8 holds irrespective of whether the data (lower bounds, capacities, and the initial feasible flow vector) is integral, rational, or irrational. Each application of the tree growth routine requires at most $O(m)$ effort in terms of comparisons, additions, or lookups. So, the overall computational effort required by the shortest

augmenting path method is at most $O(nm^2)$. Since $m \leqq n(n-1)$, this is at most $O(n^5)$. Zadeh [1972] has constructed networks with n nodes and m arcs on which this method requires $O(nm)$ flow augmentations to find a maximum value flow.

2.4 Multipath Labeling Methods Begi -nning with a Feasible Flow Vector

For the maximum value flow problem in the directed single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$, we discuss here a new class of augmentation methods that use all shortest FAPs simultaneously. In each FAP used in a step, the orientation of the reverse arcs is changed so that the FAP gets transformed into a chain from $\breve{s}$ to $\breve{t}$. When these chains corresponding to all the FAPs used in a step are put together, we get an acyclic network known as an **auxiliary network** wrt the present flow vector in G; it will be a partial subnetwork of the residual network at this stage, containing all the nodes and arcs that lie on at least one shortest chain from $\breve{s}$ to $\breve{t}$ in it.

The auxiliary network wrt the flow vector $\bar{f}$ is denoted by $AN(\bar{f}) = (N(\bar{f}), A(\bar{f}))$. Since it is a partial subnetwork of the residual network $G(\bar{f})$, arcs in it carry + or − labels. So, $A(\bar{f})$ is partitioned into $A^+(\bar{f}) \cup A^-(\bar{f})$. All lower bounds in $AN(\bar{f})$ are 0. If $(i,j) \in A^+(\bar{f})$, it has a + label and capacity $\kappa_{ij} = k_{ij} - \bar{f}_{ij}$ and it corresponds to $(i,j) \in \mathcal{A}$ which satisfies $\bar{f}_{ij} < k_{ij}$. If $(i,j) \in A^-(\bar{f})$, it has a − label and capacity $\kappa_{ij} = \bar{f}_{ji} - \ell_{ji}$ and it corresponds to $(j,i) \in \mathcal{A}$ which satisfies $\bar{f}_{ji} > \ell_{ji}$. Every chain in $AN(\bar{f})$ from $\breve{s}$ to $\breve{t}$ becomes an FAP wrt $\bar{f}$ in G when the orientations of the − labeled arcs in it are reversed.

All these methods find a maximal or blocking flow vector in $AN(\bar{f})$ using efficient special procedures. To avoid confusion with flow vectors in G, we will use the symbol $g = (g_{ij})$ to denote flow vectors in auxiliary networks. The maximal flow vector in the auxiliary network is then used to revise the flow vector in G, and the whole process is repeated. Here are the steps in these methods.

THE GENERAL MULTIPATH METHOD

Step 1 Initialization Initiate the algorithm with a feasible flow vector in G.

Step 2 Auxiliary network construction Let $\bar{f}$ be the present flow vector. Construct AN($\bar{f}$). This may lead to two possible outcomes: (1) The conclusion that $\bar{f}$ is of maximum value. In this case terminate the algorithm. (2) AN($\bar{f}$) itself.

Step 3 Find a maximal feasible flow vector in the auxiliary network Use a procedure to find a maximal feasible flow vector $\hat{g}$ in AN($\bar{f}$).

Step 4 Augmentation Compute the new flow vector $\hat{f} = (\hat{f}_{ij})$ in G by

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} & \text{if } (i,j) \text{ is not an arc in AN}(\bar{f}) \\ \bar{f}_{ij} + \hat{g}_{ij} & \text{if } (i,j) \text{ is a } + \text{ arc in AN}(\bar{f}) \\ \bar{f}_{ij} - \hat{g}_{ij} & \text{if } (j,i) \text{ is a } - \text{ arc in AN}(\bar{f}) \end{cases}$$

$\hat{f}$ is a feasible flow vector in G of value $\bar{v} + \hat{w}$, where $\hat{w}$ is the value of $\hat{g}$ in AN($\bar{f}$). Go to Step 2 with $\hat{f}$.

So, for each method we need only to describe the method to be used for constructing the auxiliary network, and the procedure to be used for finding a maximal feasible flow vector in it.

2.4.1 Dinic's Method

In this method, the auxiliary network is called the **layered network**. Nodes in it are partitioned into nonempty subsets $\mathcal{N}_0, \mathcal{N}_1, \ldots$ called **layers**, and every arc in it joins a node in some layer to a node in the next. The procedure for constructing it builds a layer at a time and terminates when (i) $\breve{t}$ lies in a layer (in this case this will be the last layer), or (ii) $\breve{t}$ is not in any layer so far, and there are no nodes that can be included in the next layer (this implies that the present flow vector in G has maximum value). So, if the procedure completes

the construction of the layered network, the last layer contains $\breve{t}$. The **length** of a layered network is the number of layers in it.

PROCEDURE FOR CONSTRUCTING THE LAYERED NETWORK

Let $\bar{f}$ be the present feasible flow vector in G of value $\bar{v}$.

Defining the initial layer Define $\mathcal{N}_0 = \{\breve{s}\}$.

Constructing the next layer Let $\mathcal{N}_r$ be the last layer constructed so far. Define $\mathbf{X}_r = \bigcup_{h=0}^{r} \mathcal{N}_h$, $\bar{\mathbf{X}}_r = \mathcal{N} \backslash \mathbf{X}_r$, and $\mathrm{A}_{r+1}(\bar{f}) = \mathrm{A}^{+}_{r+1}(\bar{f}) \cup \mathrm{A}^{-}_{r+1}(\bar{f})$, where

$$\mathrm{A}^{+}_{r+1}(\bar{f}) = \{(i,j) : i \in \mathcal{N}_r, j \in \bar{\mathbf{X}}_r, (i,j) \in \mathcal{A}, \text{ and } \bar{f}_{ij} < k_{ij}\}$$

$$\mathrm{A}^{-}_{r+1}(\bar{f}) = \{(i,j) : i \in \mathcal{N}_r, j \in \bar{\mathbf{X}}_r, (j,i) \in \mathcal{A}, \text{ and } \bar{f}_{ji} > \ell_{ji}\}$$

If $\mathrm{A}_{r+1}(\bar{f}) = \emptyset$, $\bar{f}$ has maximum value in G and $[\mathbf{X}_r, \bar{\mathbf{X}}_r]$ is a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G. Terminate the whole method.

If $\mathrm{A}_{r+1}(\bar{f}) \neq \emptyset$, define the next layer to be $\mathcal{N}_{r+1} = \{j : j$ is the head of some arc in $\mathrm{A}_{r+1}(\bar{f})\}$, and include all the arcs in $\mathrm{A}_{r+1}(\bar{f})$ in the layered network. Each arc $(i,j) \in \mathrm{A}^{+}_{r+1}(\bar{f})$ gets a $+$ label, and has capacity $\kappa_{ij} = k_{ij} - \bar{f}_{ij}$. Each arc $(i,j) \in \mathrm{A}^{-}_{r+1}(\bar{f})$ gets a $-$ label, and has capacity $\kappa_{ij} = \bar{f}_{ji} - \ell_{ji}$. The lower bounds for all the arcs in the layered network are always zero.

If $\breve{t} \in \mathcal{N}_{r+1}$, define $\mathrm{N}(\bar{f}) = \mathbf{X}_r \cup \mathcal{N}_{r+1}$, and $\mathrm{A}(\bar{f})$ to be the set of all arcs included so far. ($\mathrm{N}(\bar{f})$, $\mathrm{A}(\bar{f})$) is the layered network; terminate the construction procedure.

If $\breve{t} \notin \mathcal{N}_{r+1}$, repeat this step for constructing the next layer.

If $\mathrm{A}_{r+1}(\bar{f}) = \emptyset$, we have $\breve{s} \in \mathbf{X}_r$ and $\breve{t} \in \bar{\mathbf{X}}_r$, so $[\mathbf{X}_r, \bar{\mathbf{X}}_r]$ is a cut separating $\breve{s}$ and $\breve{t}$. Besides, $\mathrm{A}_{r+1}(\bar{f}) = \emptyset$ implies that $\bar{f}_{ij} = k_{ij}$ for all $(i,j) \in (\mathbf{X}_r, \bar{\mathbf{X}}_r)$ and that $\bar{f}_{ij} = \ell_{ij}$ for all $(i,j) \in (\bar{\mathbf{X}}_r, \mathbf{X}_r)$, i.e., $\bar{f}(\mathbf{X}_r, \bar{\mathbf{X}}_r) - \bar{f}(\bar{\mathbf{X}}_r, \mathbf{X}_r) = k(\mathbf{X}_r, \bar{\mathbf{X}}_r) - \ell(\bar{\mathbf{X}}_r, \mathbf{X}_r)$. These facts together

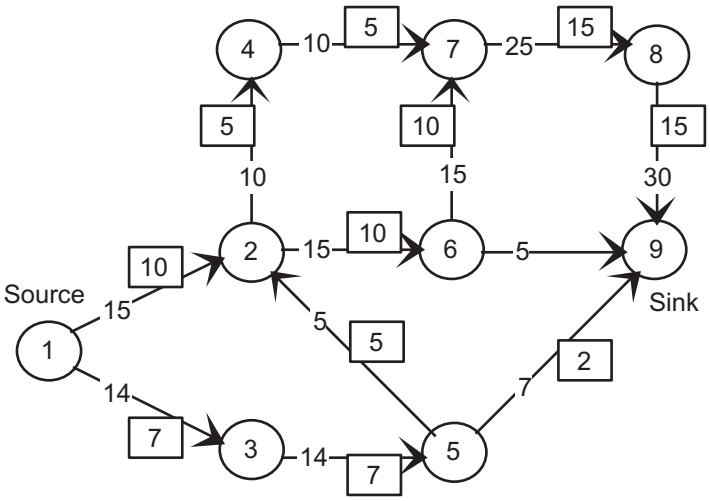


Figure 2.15: All lower bounds are 0, and capacities are entered on the arcs. Nonzero flow amounts are entered in boxes by the side of the arcs.

with the results in Theorems 2.1 and 2.2 imply that in this case $\bar{f}$ is a maximum value flow and that $[\mathbf{X}_r, \bar{\mathbf{X}}_r]$ is a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G.

Since all the arcs in a layered network are forward arcs joining a node in a layer to the next layer, it is an acyclic network.

L, the length of the layered network wrt $\bar{f}$, is clearly the length (in terms of the number of arcs) of a shortest FAP in G from $\breve{s}$ to $\breve{t}$ wrt $\bar{f}$. $\mathrm{L} \leqq n-1$. Every chain from $\breve{s}$ to $\breve{t}$ in the layered network has length L. If $i \in \mathcal{N}_h$, any chain from $\breve{s}$ to i in the layered network has length h arcs, and any chain from i to $\breve{t}$ has $\mathrm{L}-h$ arcs. And every chain from $\breve{s}$ to $\breve{t}$ in the layered network corresponds to a shortest FAP in G wrt the present flow vector.

In the procedure for constructing the layered network, we may have to examine each arc in G at most twice, once from each of its nodes. So, the computational effort needed to construct the layered network is at most $\mathrm{O}(m)$, where $m = |\mathcal{A}|$.

As an example consider the network in Figure 2.15 with an initial

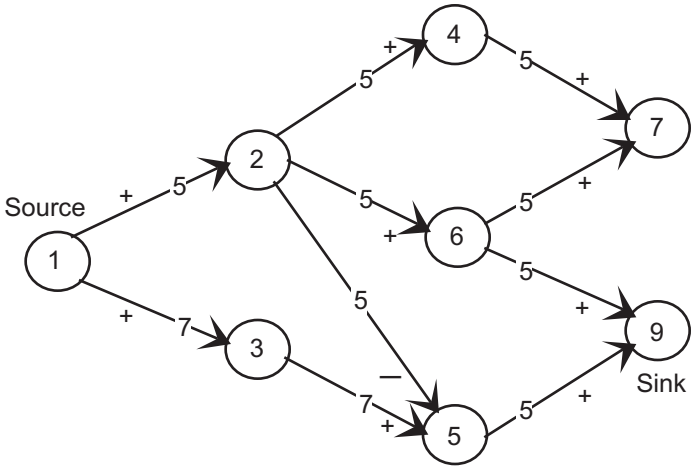


Figure 2.16: The layered network. Capacities are marked on the arcs. Arc labels, $+, -$ are also entered. Length of this layered network is 3.

feasible flow vector $\bar{f}$ of value 17 marked on it. The layered network wrt $\bar{f}$ in this example is given in Figure 2.16, with the nodes in each layer aligned vertically.

DINIC'S PROCEDURE FOR FINDING A MAXIMAL FLOW IN THE LAYERED NETWORK

Let $\mathcal{L} = (\mathrm{N}, \mathrm{A}, 0, \kappa, \breve{s}, \breve{t})$ denote the layered network. The procedure begins with the flow vector $g^0 = 0$ in $\mathcal{L}$. It augments the flow using FACs detected by **depth first search**. It maintains a subset $\mathbf{F}$ of unsaturated arcs satisfying the property that if there is an FAC from $\breve{s}$ to $\breve{t}$ in $\mathcal{L}$ wrt the present flow vector g, all the arcs on it must lie in $\mathbf{F}$. Arcs are deleted from $\mathbf{F}$ whenever they become saturated or whenever it becomes clear that there exists no FAC containing them. Termination occurs when $\mathbf{F}$ becomes $\emptyset$. Let $\mathcal{N}_0, \ldots, \mathcal{N}_{\mathrm{L}}$ be the layers in $\mathcal{L}$.

Step 1 Initialization Start with the flow vector g^0 in $\mathcal{L}$. Let $\mathbf{F} = \mathrm{A}$.

Step 2 Begin depth first search Label $\breve{s}$ with $\emptyset$ and make it the *current node.*

Step 3 Search for an arc incident out of the current node Let $i \in \mathcal{N}_r$ be the current node. Look for an arc incident out of i in $\mathbf{F}$. If none, go to Step 6 if $i = \breve{s}$, or to Step 5 otherwise. If there is such an arc, suppose (i, j) is the one selected. Clearly $j \in \mathcal{N}_{r+1}$. Label j with predecessor index i. If $j = \breve{t}$, an FAC has been found; go to Step 4. Otherwise make j the new current node replacing i from this status, and repeat this step.

Step 4 Flow augmentation Find the FAC by a backwards trace of node labels beginning with $\breve{t}$ and add its residual capacity to the flow amounts on all its arcs. Delete all the saturated arcs in the new flow vector from $\mathbf{F}$. Erase the labels on all the nodes and go back to Step 2.

Step 5 Arc deletion from F Since i is the current node, and there exists no arc incident out of i in $\mathbf{F}$, there exists no FAC from $\breve{s}$ to $\breve{t}$ through i wrt the present flow vector. Let p be the predecessor index of i. Delete all arcs incident into i from $\mathbf{F}$; make p the current node replacing i from this status, and go back to Step 3

Step 6 Termination The present flow vector $\bar{g}$ is a maximal flow vector in $\mathcal{L}$. Terminate.

Discussion

When the procedure reaches Step 6, $\breve{s}$ is the current node and there exists no arc incident out of it in $\mathbf{F}$. This implies that there exists no FAC from $\breve{s}$ to $\breve{t}$ in $\mathcal{L}$ wrt the present flow vector $\bar{g}$ (i.e., it is a maximal flow vector).

Whenever Steps 4 or 5 occur, at least one arc is deleted from $\mathbf{F}$. So, the total number of times that either Step 4 or 5 can occur is $m = |\mathrm{A}|$. The amount of work in between two consecutive occurrences of either Step 4 or 5 is at most that of L consecutive node labelings, which is at most $\mathrm{O}(n)$, where n is the number of nodes in $\mathcal{L}$. Thus the overall computational effort in this procedure is at most $\mathrm{O}(mn)$.

As an example consider the layered network $\mathcal{L}$ in Figure 2.16. To find a maximal flow vector in this by Dinic's procedure, we begin with $g^0 = 0$, and $\mathbf{F}$ = set of all arcs in $\mathcal{L}$. We label the nodes in the order 1,

2, 4, 7, and then Step 5 occurs. Arcs (4, 7), (6, 7) get deleted from **F** and 4 becomes the current node. Again Step 5 occurs, and the arc (2, 4) gets deleted from **F**. Now 2 becomes the current node, and nodes 5, 9 are labeled in this order next. We have an FAC consisting of arcs (1, 2), (2, 5), (5, 9), with residual capacity 5. The new flow vector is given in Figure 2.17.

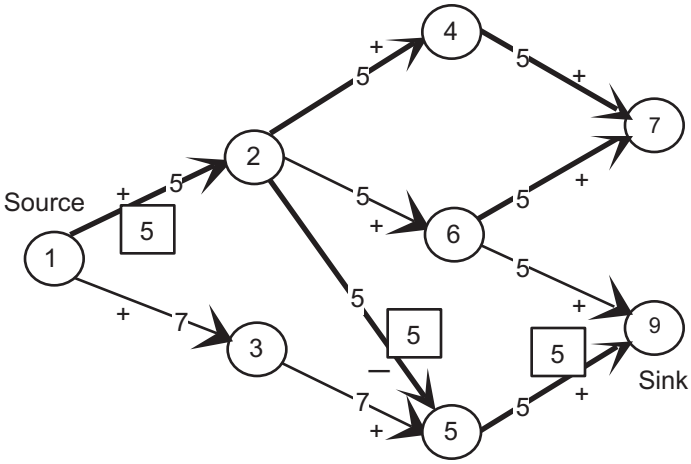


Figure 2.17: Nonzero flow amounts are entered in boxes by the side of the arcs. Thick arcs are those deleted from **F** at this stage.

At this stage the set **F** consists of only the thin arcs in Figure 2.17. When the procedure continues, after some labeling, arcs (3, 5), (1, 3) get deleted from **F** in that order, and the procedure terminates. The flow vector in Figure 2.17 is maximal.

THEOREM 2.9 *The layered network construction step has to be carried out at most* $(n-1)$ *times in this method before a maximum value flow vector is found in* G*, where* $n = |\mathcal{N}|$.

Proof We will prove that each successive layered network is strictly longer than the previous one in this method. Let $\mathcal{L}_u$ denote the uth layered network constructed in this method, and L_u its length, for $u = 1, 2, \ldots$. Let $f^{u-1} = (f_{ij}^{u-1})$ be the feasible flow vector in G at the beginning of construction of $\mathcal{L}_u, u = 1, 2, \ldots$. Let $\mathcal{N}_0^u, \mathcal{N}_1^u, \ldots, \mathcal{N}_{L_u}^u$ be the layers in $\mathcal{L}_u$. Fix u. We will now prove that $L_{u+1} > L_u$.

There exists a chain of length L_{u+1} in $\mathcal{L}_{u+1}$ from $\breve{s}$ to $\breve{t}$. Let $\mathcal{C}$ be such a chain and suppose the sequence of nodes on this chain is $\breve{s} = i_0, i_1, \ldots, i_{-1+L_{u+1}}, i_{L_{u+1}} = \breve{t}$. Then $i_r \in \mathcal{N}_r^{u+1}$ for all $r = 0$ to L_{u+1}. We consider two cases.

Case 1 All the nodes $i_r, r = 0$ to L_{u+1} appear in $\mathcal{L}_u$: In this case suppose $i_r \in \mathcal{N}_p^u$. We will now show that $r \geqq p$ by induction on r. For $r = 0$, $i_0 = \breve{s} \in \mathcal{N}_0^u$, so the statement is true.

Induction Hypothesis For all $a \leqq r$, if $i_a \in \mathcal{N}_b^u$, then $a \geqq b$.

We will now show that under the induction hypothesis, the statement there must also hold for $a = r + 1$. $i_r \in \mathcal{N}_p^u$ and suppose $i_{r+1} \in \mathcal{N}_d^u$. By the induction hypothesis, $r \geqq p$. We have to show that these facts imply that $r + 1 \geqq d$. If $d \leqq p + 1$, we are done. Suppose $d > p + 1$. This implies that (i_r, i_{r+1}) is not an arc in $\mathcal{L}_u$. So, the flow on the arc in G corresponding to (i_r, i_{r+1}) in $\mathcal{L}_{u+1}$ must have remained unchanged as we move from f^{u-1} to f^u, and since (i_r, i_{r+1}) is an arc in $\mathcal{L}_{u+1}$, but not $\mathcal{L}_u$, we have a contradiction. So d cannot be $> p+1$. Hence the statement in the induction hypothesis must also be true for $a = r + 1$. By induction it is true for all a.

Since $i_{L_{u+1}} = \breve{t}$, and $\breve{t} \in \mathcal{N}_{L_u}^u$, we have $L_{u+1} \geqq L_u$. If $L_{u+1} = L_u$, by the above statement the entire chain $\mathcal{C}$ must be in $\mathcal{L}_u$. So, $\mathcal{C}$ is in both $\mathcal{L}_u$ and $\mathcal{L}_{u+1}$. If g^u is the maximal flow vector obtained in $\mathcal{L}_u$ in Dinic's method, at least one arc in $\mathcal{C}$ must be saturated in g^u, and by the augmentation step in this method, this arc cannot be in $\mathcal{L}_{u+1}$, which is a contradiction. So, $L_{u+1} > L_u$ in this case.

Case 2 Not all the nodes $i_r, r = 0$ to L_{u+1} appear in $\mathcal{L}_u$: i_0 and $i_{L_{u+1}}$ ($\breve{s}$ and $\breve{t}$ respectively) are in both $\mathcal{L}_u$ and $\mathcal{L}_{u+1}$. Let $r + 1$ be the smallest value of d such that i_d does not appear in $\mathcal{L}_u$. So, $0 < r + 1 < L_{u+1}$; i_r appears in $\mathcal{L}_u$, in the layer $\mathcal{N}_p^u$, say; and (i_r, i_{r+1}) is an arc in $\mathcal{L}_{u+1}$ but not in $\mathcal{L}_u$. Let (i, j) be the arc in G corresponding to (i_r, i_{r+1}) in $\mathcal{L}_{u+1}$. Since (i_r, i_{r+1}) is not an arc in $\mathcal{L}_u$, we must have $f_{ij}^u = f_{ij}^{u-1}$. This and the fact that (i_r, i_{r+1}) is an arc in $\mathcal{L}_{u+1}$ imply that the only possible reason for (i_r, i_{r+1}) not being an arc in $\mathcal{L}_u$ must be that $p + 1 = L_u$. By the inductive argument of Case 1, $r \geqq p$. Thus $r + 1 \geqq L_u$, and therefore $L_{u+1} > L_u$ in this case too.

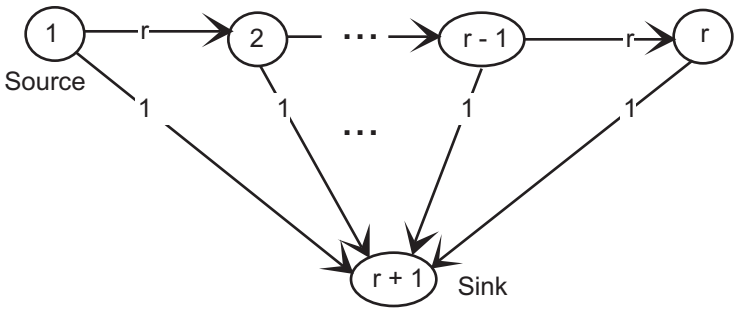


Figure 2.18: Network for the rth problem in the class. All lower bounds are 0. The horizontal arcs have capacity r, and others have capacity 1.

The length of any layered network is at most $n-1$, where $n = |\mathcal{N}|$. Since each successive layered network obtained in the method is strictly longer than the previous, the maximum number of layered networks constructed in Dinic's method before termination is at most $(n-1)$. ∎

We will now provide a class of examples due to Waissi [1985] in which Dinic's method constructs $(n-1)$ layered networks in an n node network. The network for the rth problem in the class has $r+1$ nodes and is given in Figure 2.18. The arc flow capacity is r for all arcs $(i, i+1), i = 1$ to $r-1$, and 1 for all arcs $(i, r+1), i = 1$ to r. To find a maximum value flow in it, beginning with the 0-flow vector, Dinic's method generates r layered networks, shown in Figure 2.19. There is a unique simple chain from source to sink in each layered network, and the unique maximal flow in this layered network consists of a flow of 1 unit on each arc of this chain and zero flow on the other arcs. The value of the flow vector in the original network goes up by 1 after augmentation using the maximal flow vector in each successive layered network, finally reaching the maximum value of r.

The effort required to construct a layered network and to find a maximal flow vector in it have already been shown to be $O(mn)$, where $m = |\mathcal{A}|, n = |\mathcal{N}|$. By Theorem 2.9, at most $n-1$ layered networks are constructed in the method before termination. Thus the overall effort in Dinic's method is at most $O(mn^{\mathbf{2}}) \leq O(n^{\mathbf{4}})$.

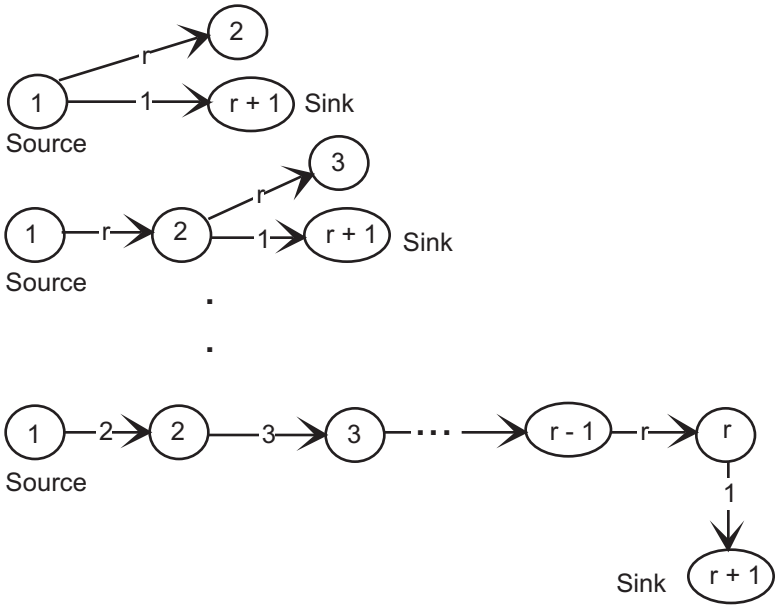


Figure 2.19: The various layered networks generated. All arcs are + arcs. Capacities are entered on the arcs.

2.4.2 Dinic-MKM Method

The auxiliary network used in this method is also Dinic's layered network. But to find a maximal flow in each layered network, it uses an algorithm of V. M. Malhotra, M. P. Kumar, and S. N. Maheswari [1978]. Let $\mathcal{L}$ = (N, A, 0, $\kappa, \check{s}, \check{t}$) be the layered network. This algorithm does not use FACs; it changes the flow vector by operations called **flow pushing** and **pulling**. It maintains a set of nodes **Y** and a set of arcs **F** in $\mathcal{L}$. **Y** will always be the set of nodes on arcs in **F**. As the algorithm progresses, saturated arcs are deleted from **F**. Nodes are also deleted from **Y**, and whenever a node is deleted from **Y**, all the arcs incident at it are deleted from **F**. The following property will always hold: If there is an FAC from $\check{s}$ to $\check{t}$ in $\mathcal{L}$ wrt the present flow vector g, all the nodes on it must lie in **Y**, and all the arcs on it must lie in **F**.

If $\bar{g}$ is the present flow vector in $\mathcal{L}$, and **Y,F** are the present sets, for each $i \in$ **Y**, define wrt $\bar{g}$, **Y, F**

$$\alpha(i) = \text{ in-potential of } i = \sum(\kappa_{ji} - \bar{g}_{ji} : \text{ over } j \text{ s. t. } (j,i) \in \mathbf{F})$$

$$\beta(i) = \text{ out-potential of } i = \sum(\kappa_{ij} - \bar{g}_{ij} : \text{ over } j \text{ s. t. } (i,j) \in \mathbf{F})$$

$$\rho(i) = \text{flow-potential of } i = \begin{cases} \text{min. } \{\alpha(i), \beta(i)\} & \text{for } i \neq \breve{s}, \breve{t} \\ \beta(i) & \text{for } i = \breve{s} \\ \alpha(i) & \text{for } i = \breve{t} \end{cases}$$

Whenever the flow vector g or the sets $\mathbf{Y}$, $\mathbf{F}$ change, these flow potentials have to be updated.

THE MKM ALGORITHM FOR FINDING A MAXIMAL FLOW IN A LAYERED NETWORK

Step 1 Initialization Start with $g^0 = 0$ in $\mathcal{L}$. Let $\mathbf{Y} = \text{N}$, $\mathbf{F} = \text{A}$. Compute $\rho(i)$ for all $i \in \mathbf{Y}$. If $\rho(i) > 0$ for all $i \in \mathbf{Y}$, go to Step 2. If $\rho(i) = 0$ for at least one $i \in \mathbf{Y}$, go to Step 6.

Step 2 Reference node selection Find $\rho = \text{min. } \{\rho(i) : i \in \mathbf{Y}\}$. Let $p \in \mathbf{Y}$ be a node which attains this minimum, break ties arbitrarily. Node p is the present **reference node**, and ρ is the present **reference potential**.

Step 3 Flow pushing and flow pulling Let p be the reference node and ρ the reference potential.

FLOW PUSHING Begin at p, and push an excess flow of ρ out of p. This requires increasing the flow on the arcs in $\mathbf{F}$ which are in the forward star of p, one by one, saturating them one after the other, until the total increase reaches ρ. In this process, at most one outgoing arc has flow increased on it but remains unsaturated.

Now the excess flow has been pushed from node p to its adjacent nodes in $\mathbf{Y}$ in the next layer. If i is one of these nodes, and the flow increase on the arc (p,i) was γ, push the excess flow of γ out of node i in exactly the same way. Repeat with all

adjacent nodes of p which received excess flow. Then repeat this process for nodes in the next layer that received excess flow, and continue this way until all the excess flow of ρ units reaches $\breve{t}$. In this process we can never get stuck with excess supply that cannot be pushed out of a node, because of the definition of ρ, which implies that the potential of every node in $\mathbf{Y}$ is $\geq \rho$.

FLOW PULLING Pull an excess flow of ρ units into p. This requires increasing the flow on arcs in $\mathbf{F}$ incident into p, one by one, saturating them one after the other, until the total increase reaches ρ, making sure that at most one incoming arc has flow added to it but remains unsaturated. If the flow increase on an arc (j, p) was δ, pull an excess flow of δ into node j in exactly the same way. Repeat with all adjacent nodes of p from which excess flow was pulled into p. Then repeat this process for nodes in the preceding layer from which excess flow was pulled, and continue the same way until all the excess flow of ρ units is pulled out of $\breve{s}$. Again we can never get stuck in this pulling process.

After the flow pushing and pulling is completed, we again have a feasible flow vector in $\mathcal{L}$.

Step 4 Updating the sets Y, F after Step 3 Delete all the saturated arcs from $\mathbf{F}$. If all the arcs into or out of a node i are deleted from $\mathbf{F}$, delete that node i from $\mathbf{Y}$ and all arcs incident at node i from $\mathbf{F}$.

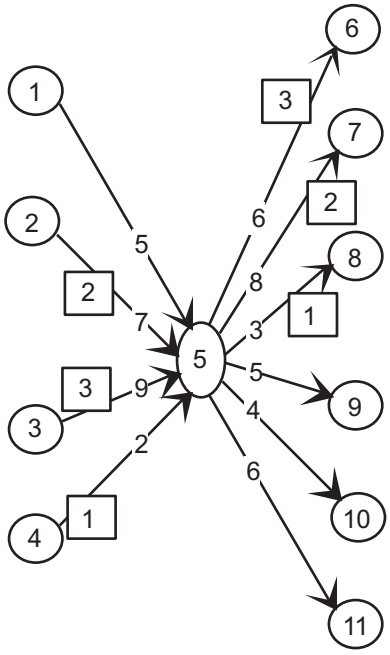
Step 5 Updating the potentials Update the in and out-potentials and the flow potential of all the nodes in $\mathbf{Y}$ wrt the present flow vector in $\mathcal{L}$, and the present sets $\mathbf{Y}$, $\mathbf{F}$. If $\rho(i) > 0$ for all $i \in \mathbf{Y}$, go to Step 2; otherwise go to Step 6.

Step 6 Updating Y, F after Step 5 If there are iı $\mathbf{Y}$ with $\rho(i) = 0$, go to Step 7 if either $\rho(\breve{s})$ or $\rho(\breve{t})$ is zero. If both $\rho(\breve{s}), \rho(\breve{t})$ are > 0, delete all nodes i for which $\rho(i) = 0$ from $\mathbf{Y}$ and all the arcs incident at such nodes from $\mathbf{F}$. Go to Step 5.

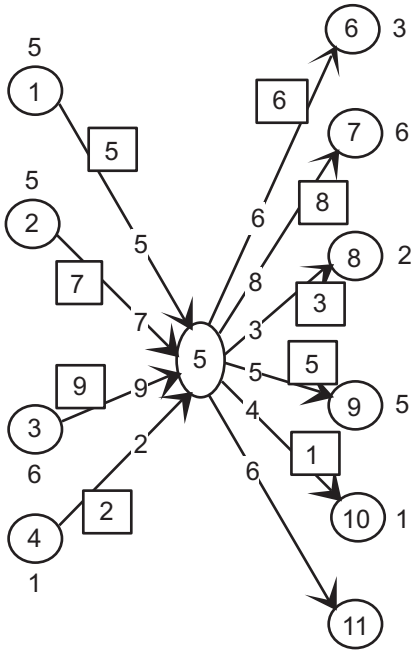
Step 7 Termination Now, $\rho(\breve{s})$, or $\rho(\breve{t})$ has become 0. So, the present flow vector is a maximal flow vector in $\mathcal{L}$. Terminate.

Discussion

An example of flow pushing and pulling is given in Figures 2.20 (a) and (b). Node 5 is the reference node with reference potential of 17 units. Only nodes adjacent to node 5 in the present set **Y** are shown in Figure 2.20. In pushing the flow out of node 5, we begin saturating the arcs incident out of node 5 in **F** one by one in some order, say, from top to bottom, until a total of 17 units is pushed. In pulling we do the same thing with arcs incident into node 5. The situation after the pushing and pulling at node 5 is indicated in Figure 2.20 (b).



(a) Arcs incident at reference node 5 are shown with their capacities and nonzero flow amounts in boxes.



(b) Flows on arcs incident at 5 after pushing and pulling. Amount by the side of each node is the amount to be pulled into or pushed out of it.

Figure 2.20:

When we reach Step 7 in this algorithm, the flow potential of either $\breve{s}$ or $\breve{t}$ is 0. Then there exists no FAC in $\mathcal{L}$ from $\breve{s}$ to $\breve{t}$ wrt the present flow vector. Hence the flow vector in $\mathcal{L}$ at that time is a maximal flow vector.

Whenever Step 3 is completed (i.e., after pushing reaches all the way to $\breve{t}$ and pulling reaches all the way to $\breve{s}$) flow conservation holds at all the nodes and the flow vector becomes feasible. The net result of this step is to increase the flow value by the reference potential. Then either all the incoming arcs or all the outgoing arcs at the reference node have become saturated. These get deleted from $\mathbf{F}$, and the reference node gets deleted from $\mathbf{Y}$ when we move over to Step 4. Likewise, each time an $i \in \mathbf{Y}$ with $\rho(i) = 0$ is noticed in Step 6, all the arcs at that node are deleted from $\mathbf{F}$ and that node is deleted from $\mathbf{Y}$. Thus Steps 3 and 4, or Step 6 with an $i \in \mathbf{Y}$ satisfying $\rho(i) = 0$, can occur at most n times before termination, where n is the number of nodes in $\mathcal{L}$.

Let b_r be the number of arcs deleted from $\mathbf{F}$ during the rth time that a node is deleted from $\mathbf{Y}$. If it has occurred in Step 6, the effort involved in executing that step is $O(b_r)$. If it has occurred in Step 4, the effort in flow pushing and pulling in the preceding Step 3 is at most $O(n + b_r)$, because in Step 3, at each node, at most one outgoing or incoming arc has flow added to it, but remains unsaturated in this step. Thus the effort needed during the rth execution of Steps 3 and 4, or Step 6 with an $i \in \mathbf{Y}$ satisfying $\rho(i) = 0$, is at most $O(n + b_r)$. Summing over all these executions, we find that the overall effort is at most $O(\sum_{r=1}^{n}(n + b_r))$. Since an arc deleted from $\mathbf{F}$ is never again considered for flow change during the algorithm, $\sum_r b_r \leq m$, where m is the number of arcs in $\mathcal{L}$. Thus the overall effort needed by this algorithm is at most $O(n^{\mathbf{2}} + m) = O(n^{\mathbf{2}})$.

The practical efficiency of the MKM algorithm for finding a maximal flow in a layered network $\mathcal{L} = (\mathrm{N}, \mathrm{A})$ can be improved considerably by applying a routine known as the **backward pass routine** to find a partial subnetwork of $\mathcal{L}$ known as the **referent** before applying the algorithm. Let $\mathcal{N}_0, \ldots, \mathcal{N}_{\mathrm{L}}$ be the layers in $\mathcal{L}$. This routine takes L+2 steps, and it prunes all the nodes and arcs in $\mathcal{L}$ which are not contained on any chain from $\breve{s}$ to $\breve{t}$ in $\mathcal{L}$ very efficiently. The remaining portion of the layered network after this pruning is completed is called

the *referent*.

BACKWARD PASS ROUTINE

Step 0 Define $\hat{\mathcal{N}}_{\rm L} = \{\breve{t}\}$.

General step r, $r = 1$ **to L** When we reach this step, the sets of nodes $\hat{\mathcal{N}}_{\rm L}, \ldots, \hat{\mathcal{N}}_{{\rm L}-r+1}$ would have been obtained. Define $\hat{\mathcal{N}}_{{\rm L}-r} = \{i : i \in \mathcal{N}_{{\rm L}-r}$, and there is a $j \in \hat{\mathcal{N}}_{{\rm L}-r+1}$ s.t. $(i,j) \in {\rm A}\}$, $\hat{\rm A}_{{\rm L}-r+1} = \{(i,j) : i \in \hat{\mathcal{N}}_{{\rm L}-r}, j \in \hat{\mathcal{N}}_{{\rm L}-r+1}, (i,j) \in {\rm A}\}$.

$\hat{\mathcal{N}}_{{\rm L}-r} \neq \emptyset$, as otherwise the referent is empty (i.e., there is no chain from $\breve{s}$ to $\breve{t}$ in $\mathcal{L}$). This cannot happen from the manner in which the layered network is constructed. Go to the next step.

Step L+1 Let $\hat{\rm N} = \bigcup_{r=0}^{\rm L} \hat{\mathcal{N}}_r$, $\hat{\rm A} = \bigcup_{r=1}^{\rm L} \hat{\rm A}_r$. $\hat{\mathcal{L}} = (\hat{\rm N}, \hat{\rm A})$ is the referent. Terminate.

Discussion

As an example, consider the layered network in Figure 2.16. When the backward pass is applied on it, nodes 7, 4 and arcs (4, 7), (6, 7), (2, 4) get pruned as they do not lie on any chain from node 1 (source) to node 9 (sink) in this network.

In the MKM algorithm for finding a maximal flow in $\mathcal{L}$, the pruning performed in the backward pass gets carried out at the beginning during the sequence of steps consisting of Step 1, followed by consecutive pairs of Steps 6 and 5, and finally another Step 6, before the method goes to Step 2 for the first time. But this pruning is carried out much more efficiently in the backward pass. Thus the practical efficiency of the MKM algorithm for maximal flow in $\mathcal{L}$ improves considerably if the backward pass is carried out immediately after Step 1 in the MKM algorithm if there is at least one $i \in \mathbf{Y}$ with $\rho(i) = 0$ at that time, instead of going to Step 6 from Step 1. Then, when the referent is obtained, begin MKM algorithm again in Step 1 with the referent.

By Theorem 2.9, the layered network construction step has to be carried out at most $(n-1)$ times in the Dinic-MKM method before a maximum value flow vector is found in G beginning with a feasible

flow vector. The construction of each layered network and finding a maximal flow vector in it by MKM algorithm needs an effort of at most $O(m) + O(n^2) = O(n^2)$, where n, m are the number of nodes and arcs in G. Thus the overall effort for finding a maximum value flow vector in G beginning with a feasible flow vector by Dinic-MKM method is at most $O(n^3)$.

2.5 The Preflow Push Algorithm

In this section we consider the problem of finding a maximum value flow in the directed single commodity flow network G=$(\mathcal{N}, \mathcal{A}, 0, k, \check{s}, \check{t})$ where $k > 0, |\mathcal{N}| = n, |\mathcal{A}| = m$. All the algorithms discussed so far for this problem maintain a feasible flow vector throughout, and augment the flow value along augmenting paths, either one path at a time or all shortest augmenting paths at once using the layered network. An alternative method for this problem based on the concept of preflows has been initiated by Karzanov [1974]. A **preflow** in G is a vector $g = (g_{ij}; (i,j) \in \mathcal{A})$ which is bound feasible (i.e., $0 \leqq g \leqq k$) but in which, for each $i \in \mathcal{N}$, the amount of material flowing into node i is only required to be greater than or equal to the amount flowing out of node i. That is, a preflow g in G may not satisfy the flow conservation equations; instead it satisfies

$$g(\mathcal{N}, i) - g(i, \mathcal{N}) \geqq 0, \quad \text{for all } i \in \mathcal{N} \backslash \{\check{s}, \check{t}\}$$

Here we will describe an algorithm for the maximum value flow problem in G by A.V. Goldberg and R.E. Tarjan [1986, 1988] that maintains a preflow, and pushes local flow excess towards the sink. Only when the algorithm terminates does the preflow become a flow, and then it is a maximum value flow. In this section the symbol g denotes a preflow in G.

In Section 2.6, we will describe how the maximum value flow in a network with nonzero lower bounds on arc flows, can be found using the preflow push algorithm twice in two phases.

Given a preflow g in G, for each $i \in \mathcal{N} \backslash \{\check{s}, \check{t}\}$, the excess in g at node i is defined to be $e(i) = g(\mathcal{N}, i) - g(i, \mathcal{N})$, it is the net flow into

node i in the preflow g. A node $i \in \mathcal{N}\backslash\{\check{s}, \check{t}\}$ is said to be an **active node** wrt g if $e(i) > 0$. The algorithm works by pushing the excess from active nodes to $\check{t}$ or to nodes estimated to be closer to $\check{t}$. However, if $\check{t}$ is not reachable from an active node, the algorithm pushes the excess there to nodes estimated to be closer to $\check{s}$. The algorithm terminates when there are no active nodes; at that time the preflow is a feasible flow of maximum value.

In the residual network $\mathrm{G}(g) = (\mathcal{N}, \mathcal{A}(g), 0, \kappa)$ wrt g, an estimate of the distance of a node from $\check{t}$ in terms of the number of arcs, is kept by maintaining a node label, called the **distance label**, denoted by $d(i)$, which is always a nonnegative integer or $+\infty$. Since it estimates the distance from the node to $\check{t}$, we define $d(\check{t}) = 0, d(\check{s}) = n$ always. In the algorithm, the preflow, distance label vector pair, g, d, are required to always satisfy

$$d(\check{t}) = 0, d(\check{s}) = n$$

$$d(i) \overset{\leq}{=} d(j) + 1, \quad \text{for every arc } (i, j) \text{ in } \quad \mathrm{G}(g) \tag{2.9}$$

The purpose of this definition is to make sure that if $d(i) < n$ for any i, then $d(i)$ is a lower bound on the actual distance from i to $\check{t}$ in $\mathrm{G}(g)$; and if $d(i) \overset{\geq}{=} n$, then $d(i) - n$ is a lower bound on the actual distance from $\check{s}$ to i in $\mathrm{G}(g)$. Conditions (2.9) are called the **validity conditions** for the distance label vector d.

Given the preflow g and distance label vector d, an arc (i, j) in $\mathrm{G}(g)$ is said to be **admissible** wrt g, d, if i is an active node and $d(j) = d(i) - 1$.

PREFLOW PUSH ALGORITHM

Initialization Define the initial preflow g^0 in G by $g^0_{\check{s}j} = k_{\check{s}j}$ for all $j \in \mathbf{A}(\check{s})$, and $g^0_{pq} = 0$ for all other arcs $(p, q) \in \mathcal{A}$. The simplest choice for the initial distance node labels is $d^0 = (d^0(i))$ where $d^0(\check{s}) = n, d^0(i) = 0$ for all $i \in \mathcal{N}\backslash\{\check{s}\}$. A better choice which improves the practical efficiency of the algorithm is to determine $d^0(i)$ to be the distance labels obtained in a backward breadth-first-search of $\mathrm{G}(g^0)$ starting at node $\check{t}$.

General step Let g, d be the present preflow, distance label pair. If there are no active nodes, terminate; g is a maximum value feasible flow vector in G. Otherwise perform one of the two following operations in any order.

PUSH Select an active node i and an admissible arc (i, j) in $G(g)$ incident out of i. If (i, j) has a + label in $G(g)$, increase g_{ij} by $\varepsilon = \min\{e(i), \kappa_{ij}\}$; if (i, j) has a − label in $G(g)$, decrease g_{ji} by ε. This push is said to be **saturating** if $\varepsilon = \kappa_{ij}$; otherwise it is **nonsaturating**. This push has the effect of decreasing $e(i)$ by ε and increasing $e(j)$ by ε if $j \neq \breve{t}$.

RELABEL Select an active node i which has no admissible arc incident out of it in $G(g)$. So, we have $d(i) \leqq d(j)$ for every j such that $(i, j) \in \mathcal{A}(g)$. It can be shown that we will have $0 < d(i) < n$. Replace $d(i)$ by $\min.\{d(j) + 1 : j$ such that $(i, j) \in \mathcal{A}(g)\}$. This relabel operation resets $d(i)$ to the largest value allowed by the validity conditions (2.9). It creates at least one admissible arc at i on which a push operation can be carried out next.

Discussion

1. At any stage of the algorithm, if node i is an active node, clearly either a push or a relabel operation can be carried out at it.

2. If g, d are the present preflow, distance label pair, and a push operation is carried out leading to the preflow g^1 (a relabel operation is carried out leading to distance label vector d^1), then it can be verified that g^1, d (g, d^1) satisfy (2.9). So, (2.9) will hold throughout the algorithm.

3. If g, d are the present pair at some stage of the algorithm, there exists no chain from $\breve{s}$ to $\breve{t}$ in the residual network $G(g)$. This can be seen from the following. If there is a chain from $\breve{s}$ to $\breve{t}$ in $G(g)$, there must be a simple chain. Suppose $1 = \breve{s}, 2, \ldots, l+1 = \breve{t}$ is the sequence of nodes on a simple chain in $G(g)$. From (2.9), we have

$d(i) \leqq d(i+1)+1$ for $i = 1$ to l. Hence $d(\breve{s}) = d(1) \leqq d(\breve{t})+l < n$ since $d(\breve{t}) = 0$ and $l \leqq n-1$, which contradicts $d(\breve{s}) = n$.

4. If g, d are the present pair and i is an active node, then there is a chain from i to $\breve{s}$ in $\mathrm{G}(g)$. Suppose this is not the case. Let $\mathbf{X} = \{j : j \in \mathcal{N}$ such that there is a chain from i to j in $\mathrm{G}(g)\}$. So, $\breve{s} \notin \mathbf{X}$, and hence $\overline{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X} \neq \emptyset$. From the definition of $\mathrm{G}(g)$ and $\mathbf{X}$, we must therefore have $g_{pq} = k_{pq}$ for all $(p,q) \in \mathcal{A}$ satisfying $p \in \mathbf{X}, q \in \overline{\mathbf{X}}$; and $g_{pq} = 0$ for all $(p,q) \in \mathcal{A}$ satisfying $p \in \overline{\mathbf{X}}, q \in \mathbf{X}$. Now $e(\mathbf{X}) = g(\mathcal{N}, \mathbf{X}) - g(\mathbf{X}, \mathcal{N}) = g(\overline{\mathbf{X}}, \mathbf{X}) - g(\mathbf{X}, \overline{\mathbf{X}}) = -g(\mathbf{X}, \overline{\mathbf{X}})$ by the above, and since $g \geqq 0$ we have $e(\mathbf{X}) \leqq 0$. But $e(i) > 0$ since $i \in \mathbf{X}$ is an active node and $e(p) \geqq 0$ for all p as g is a preflow, there is a contradiction in $e(\mathbf{X}) \leqq 0$.

5. For all $i \in \mathcal{N}, d(i)$ never decreases during the algorithm, since distance labels change only when relabeling is done, and from the facts mentioned in the relabel operations.

6. Throughout the algorithm $d(i) \leqq 2n-1$ for all $i \in \mathcal{N}$. Since $d(\breve{t}) = 0, d(\breve{s}) = n$ always, this statement is true for $\breve{s}$ and $\breve{t}$. Let $i \neq \breve{s}$ or $\breve{t}$. Suppose i is active at some stage. Then by 4, there exists a simple chain on which the sequence of nodes is $i_o = i, i_1, i_2, \ldots, i_l = \breve{s}$, say, in the residual network $\mathrm{G}(g)$ at that stage. So $l \leqq n-1$, and (i_r, i_{r+1}) is an arc in $\mathrm{G}(g)$ for all $r = 0$ to $l-1$. So by (2.9), $d(i_r) \leqq d(i_{r+1}) + 1$. Hence $d(i) = d(i_0) \leqq d(i_l) + l \leqq d(\breve{s}) + n - 1 = 2n-1$. So for all active nodes $d(i) \leqq 2n-1$ always. Since the algorithm changes distance labels for only active nodes, the statement holds for all $i \in \mathcal{N}$.

7. The total number of relabeling operations carried out is at most $(2n-1)$ per node and $(2n-1)(n-2)$ during the entire algorithm. A relabeling operation carried out at a node i increases $d(i)$ by at least one. So, these facts follow from 6.

8. The total number of saturating push operations carried out is at most $2nm$. Consider an arc $(i,j) \in \mathcal{A}$ and a saturating push

from i to j on it. The next push operation on this arc (which will decrease the flow on it, so it will be from j to i) cannot happen until $d(j)$ increases by at least 2. And similarly for the next time in the other direction again. Since $d(i) + d(j) \geqq 1$, when the first push between i and j occurs, and $d(i) + d(j) \leqq 4n - 3$ when the last such push occurs (by 6), the above fact implies that the total number of saturating pushes on (i, j) is at most $2n - 1$. So, the total number of saturating pushes over all edges is at most $(2n - 1)m < 2nm$.

9. The total number of nonsaturating pushing operations is at most $4n^{\mathbf{2}}m$ in the algorithm. A nonsaturating push at an active node i with the admissible arc (i, j) in $\mathrm{G}(g)$, makes node i inactive, and at that time $d(j) = d(i) - 1$, hence $d(i)$ must have been $\geqq 1$. Hence if we define $L = \sum(d(i) : \text{over active nodes } i)$; L decreases by at least 1 in each nonsaturating push operation.

   Consider a saturating push at an active node i with the admissible arc (i, j) in $\mathrm{G}(g)$. This might make node j active, hence by 6, this operation may increase L by at most $2n - 1$. So by 8, the total increase in L due to saturating push operations is at most $(2n - 1)2nm$.

   In a relabeling operation carried out on a node i, its distance label increases by $\gamma \geqq 1$, and it increases L by at most γ. By 5 and 6, the total increase in $d(i)$ for any node i during the entire algorithm by relabeling operations is at most $2n$. Since relabeling operations are carried out only at nodes $\neq \check{s}$ or $\check{t}$, the total increase in L by relabeling operations during the entire algorithm is at most $2n(n - 2)$.

   Initially $L \geqq 0$, and at termination, $L = 0$. Hence, the total of decreases in L during the algorithm and the total number of nonsaturating pushes, is $\leqq$ the total of increases in L during the algorithm, which is at most $(2n - 1)2nm + 2n(n - 2) \leqq 4n^{\mathbf{2}}m$.

10. From 7, 8, and 9 we see that the total number of basic operations carried out in this algorithm is at most $0(n^{\mathbf{2}}m)$.

The running time of this algorithm depends on the order in which the push and relabel operations are applied and on the other details of the implementation. The simple scheme which selects an active node, maintains the set of residual arcs incident at it in some order, carries out the push operation at this node using these arcs in order, one after the other, and then relabels that node, can be shown to have running time of $0(n^{\mathbf{2}}m)$. It has been shown that maintaining the set of active nodes as a queue and selecting the node for push/relabel operations using a first-in first-out rule, the worst case running time of the algorithm improves to $0(n^{\mathbf{3}})$. Using different rules for selecting nodes for push/relabel operations (for example, selecting the active node with the highest distance label, etc.) and exploiting the other flexible features of this algorithm, many different versions of the algorithm with improved complexity bounds have been obtained. From a theoretical worst case computational complexity aspect, the best version so far, based on the dynamic tree data structures, has a running time bound of $0(nm \log(n^{\mathbf{2}}/m))$.

2.6 Phase 1 For Problems With $\ell \neq 0$

Let $G = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ be a directed single commodity flow network with $0 \leqq \ell \leqq k$. In this section we will discuss an algorithm for the Phase 1 problem of finding a feasible flow vector in G, if one exists.

Let G^1 be the network G with an additional artificial arc $(\breve{t}, \breve{s})$ with lower bound 0 and capacity $+\infty$. If f is a feasible flow vector of value v in G, define $f^1_{\breve{t}\breve{s}} = v$, and if $f^1 = (f, f^1_{\breve{t}\breve{s}})$, then f^1 is a feasible circulation in G^1. Conversely, given any feasible circulation f^1 in G^1, let $f^1_{\breve{t}\breve{s}}$ be the flow amount in f^1 on the artificial arc $(\breve{t}, \breve{s})$, and f the vector obtained by deleting the entry $f^1_{\breve{t}\breve{s}}$ from f^1. Then f is a feasible flow vector in G of value $f^1_{\breve{t}\breve{s}}$. Hence, the problem of finding a feasible flow vector in G is equivalent to that of finding a feasible circulation in G^1. For this problem in G^1, the original source and sink nodes $\breve{s}, \breve{t}$ do not play any special role, they are like any other node since it is a circulation problem. We will again transform this problem into that of finding

a maximum value flow on a further augmented network G^* in which the lower bounds for all the arc flows are 0. In G, for $i \in \mathcal{N}$ define $\alpha_i = \ell(\mathcal{N}, i)$, $\beta_i = \ell(i, \mathcal{N})$. Add artificial source and sink nodes s^* and t^* to G^1. Introduce an artificial arc (s^*, i) with capacity α_i for each $i \in \mathcal{N}$ such that $\alpha_i \neq 0$, and an artificial arc (i, t^*) with capacity β_i for each $i \in \mathcal{N}$ such that $\beta_i \neq 0$. Change the capacity on each original arc $(i, j) \in \mathcal{A}$ to $k_{ij} - \ell_{ij}$. Make the lower bounds for flows on all the arcs 0. Let G^* be the resulting network. Notice that the source and sink nodes in G^* are the artificial nodes s^*, t^* respectively (the source and the sink nodes $\breve{s}, \breve{t}$ in G are transit nodes in G^* like all other nodes in $\mathcal{N}$).

Beginning with the initial feasible flow vector of 0, find a maximum value flow, f^*, say, in G^*. Let the value of f^* be v^*. The following conclusions hold.

1. Find $\ell(\mathcal{N}, \mathcal{N})$ in the original network G. If $v^* < \ell(\mathcal{N}, \mathcal{N})$, there exists no feasible circulation in G^1, and consequently no feasible flow vector in G (see Theorem 2.10 below).

2. If $v^* = \ell(\mathcal{N}, \mathcal{N})$, define $\hat{f}_{ij} = f^*_{ij} + \ell_{ij}$ for each $(i, j) \in \mathcal{A}$. Then $\hat{f} = (\hat{f}_{ij} : (i, j) \in \mathcal{A})$ is a feasible flow vector in G, and $(\hat{f}, f^*_{\breve{t},\breve{s}})$ is a feasible circulation in G^1.

THEOREM 2.10 *A feasible circulation in* G^1*, and consequently a feasible flow vector in* G*, exist iff* v^**, the maximum value in* G^* *is* $\ell(\mathcal{N}, \mathcal{N})$*.*

Proof Suppose f^* is a maximum value flow vector in G^* of value $v^* = \ell(\mathcal{N}, \mathcal{N})$. Then all the arcs of the form $(s^*, i), (i, t^*)$ in G^* are saturated in f^*, and the statements in 2 above can be verified to be true.

To show the converse, assume that $f = (f_{ij})$ is a feasible flow vector of value v in G. Define a flow vector f^* in G^* by: $f^*_{ij} = f_{ij} - \ell_{ij}$ for all $(i, j) \in \mathcal{A}$, $f^*_{\breve{t}\breve{s}} = v$, and $f^*_{s^*i} = \alpha_i, f^*_{it^*} = \beta_i$ for all such arcs in G^*. Verify that f^* is feasible for G^* and that it saturates all the arcs of the form $(s^*, i), (i, t^*)$. This implies that f^* is a maximum value flow in G^* and that its value is $\ell(\mathcal{N}, \mathcal{N})$. ∎

As an example consider the network G in Figure 2.21, left. Data on the arcs is lower bound, capacity in that order. The corresponding networks G^1, G^* are drawn in Figures 2.21 right and 2.22. A maximum value flow in G^* is entered in Figure 2.22, it saturates all the arcs of the form $(s^*, i), (i, t^*)$ and thus satisfies the condition in Theorem 2.10. The feasible flow vector in G constructed from this maximum value flow in G^* is $(f_{12}, f_{13}, f_{23}, f_{32}, f_{24}, f_{34}) = (3, 4, 1, 2, 4, 3)$.

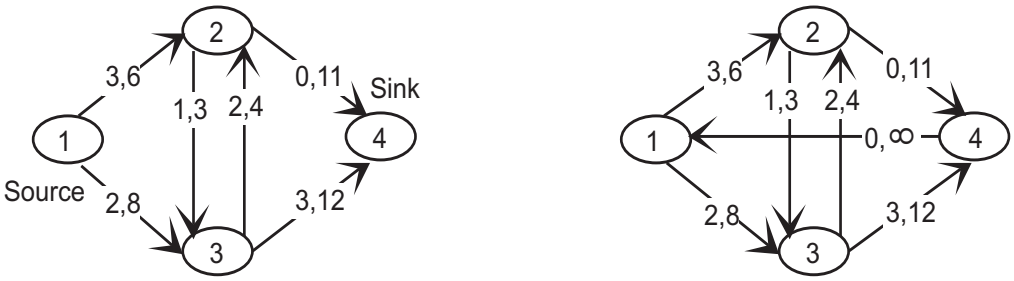


Figure 2.21:

THEOREM 2.11 *CONDITIONS FOR THE EXISTENCE OF A FEASIBLE CIRCULATION* *Let* $G = (\mathcal{N}, \mathcal{A}, \ell, k)$ *be a directed single commodity flow network, where* $0 \leqq \ell \leqq k$. *A feasible circulation exists in* G *iff*

$$k(\mathbf{X}, \bar{\mathbf{X}}) \geqq \ell(\bar{\mathbf{X}}, \mathbf{X}), \text{ for all } \mathbf{X} \subset \mathcal{N}, \bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X} \tag{2.10}$$

Proof Introduce the artificial source and sink nodes, s^*, t^*, and arcs (s^*, i) with capacity $\ell(\mathcal{N}, i)$, (i, t^*) with capacity $\ell(i, \mathcal{N})$, for each $i \in \mathcal{N}$. Make the lower bounds on all the arcs zero, and change the capacity on $(i, j) \in \mathcal{A}$ to $k_{ij} - \ell_{ij}$. Let the resulting network be G^* $= (\mathcal{N}^*, \mathcal{A}^*, 0, k^*, s^*, t^*)$. From Theorem 2.10 we know that a feasible circulation exists in G iff the maximum flow value from s^* to t^* in G^* is $\ell(\mathcal{N}, \mathcal{N})$. A necessary and sufficient condition for this is that the capacity of every cut separating s^* and t^* in G^* is $\geqq \ell(\mathcal{N}, \mathcal{N})$. Let $[\ \mathbf{X}^*, \bar{\mathbf{X}}^*\ ]$ be such a cut, and let $\mathbf{X} = \mathbf{X}^* \backslash \{s^*\}$, $\bar{\mathbf{X}} = \mathcal{N} \backslash\ \mathbf{X}$. The

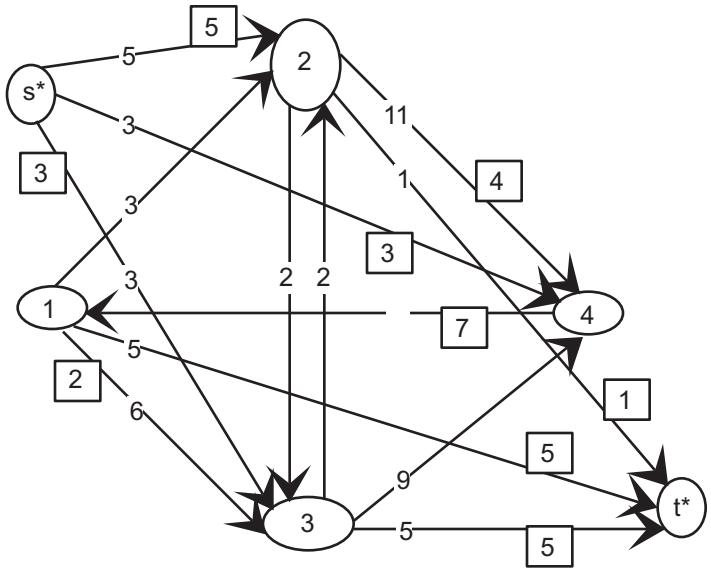


Figure 2.22: Network G*. Capacities are entered on the arcs. Nonzero flows are shown in boxes by the side of the arcs.

capacity of the cut [$\mathbf{X}^*, \bar{\mathbf{X}}^*$] in G* is $\sum(k^*_{ij}$: over $(i,j) \in \mathcal{A}^*$, with $i \in \mathbf{X}^*$, $j \in \bar{\mathbf{X}}^*) = \sum(\ k_{ij} - \ell_{ij}$: over $(i,j) \in \mathcal{A}$ with $i \in \mathbf{X}$, $j \in \bar{\mathbf{X}}) + \sum(\ \ell(\mathcal{N}, j)$: over $j \in \bar{\mathbf{X}}) + \sum(\ell(i, \mathcal{N})$: over $i \in \mathbf{X}) = k(\mathbf{X}, \bar{\mathbf{X}})$ - $\ell(\mathbf{X}, \bar{\mathbf{X}}) + \ell(\mathcal{N}, \bar{\mathbf{X}}) + \ell(\mathbf{X}, \mathcal{N}) = k(\mathbf{X}, \bar{\mathbf{X}}) + \ell(\bar{\mathbf{X}}, \bar{\mathbf{X}}) + \ell(\mathbf{X}, \mathcal{N})$. This cut capacity is $\geq \ell(\mathcal{N}, \mathcal{N})$, iff (2.10) holds. Therefore, a feasible circulation exists in G iff (2.10) holds. ∎

Let $|\mathcal{N}| = n$. There are $2^{\mathbf{n}}$ conditions in (2.10). Thus the result in Theorem 2.11 is not practically useful for verifying the existence of feasible circulations in G unless n is small. Fortunately, the problem of finding either a feasible circulation in G, or a subset $\mathbf{X} \subset \mathcal{N}$ violating (2.10) can be carried out with at most $O(n^{\mathbf{3}})$ effort by applying the methods discussed earlier to find a maximum value flow in the augmented network G* discussed in the proof of Theorem 2.11. Suppose f^* is a maximum value flow in G* of value v^*. If $v^* = \ell(\mathcal{N}, \mathcal{N})$, define $f = (f_{ij})$ by $f_{ij} = f^*_{ij} + \ell_{ij}$ for each $(i,j) \in \mathcal{A}$. It is a feasible circulation in G. If $v^* < \ell(\mathcal{N}, \mathcal{N})$ (this happens if some of the arcs of the form $(s^*, i), (i, t^*)$ remain unsaturated in f^*), let $[\mathbf{Y}^*, \bar{\mathbf{Y}}^*]$ be a minimum ca-

pacity cut in G*. Then the subset $\mathbf{X} = \mathbf{Y}^* \backslash \{s^*\}$ can be verified to violate (2.10).

Conditions (2.10) have a practical interpretation. They require a sufficient escape capacity from the set of nodes $\mathbf{X}$ to disperse the flow forced into the set by the lower bound constraints on arc flows. They provide a useful infeasibility analysis procedure. If $\mathbf{X} \subset \mathcal{N}$ violates (2.10), either the capacities on arcs in $(\mathbf{X}, \bar{\mathbf{X}})$ have to be increased or the lower bounds on arcs in $(\bar{\mathbf{X}}, \mathbf{X})$ have to be reduced in order to remedy the situation.

Necessary and sufficient conditions for the existence of feasible flow vectors in any single commodity flow network can be derived by applying Theorem 2.11 to an appropriate modification of the network. As an example consider the directed single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, V)$, where V is the specified vector of exogenous flows at the nodes. Let $\mathbf{S} = \{i : V_i > 0\}$, $\mathbf{N} = \{i : V_i < 0\}$. **S, N** are the sets of source and sink nodes respectively in G. A flow vector in G is feasible if it satisfies the constraints in (1.20). Modify G by introducing the artificial nodes s, t, arcs (s, i) with lower bound and capacity equal to V_i for each $i \in \mathbf{S}$, arcs (j, t) with lower bound and capacity both equal to $|V_j|$ for each $j \in \mathbf{N}$, and the arc (t, s) with lower bound 0 and capacity ∞. Let $\hat{G} = (\hat{\mathcal{N}}, \hat{\mathcal{A}}, \hat{\ell}, \hat{k})$ denote the modified network. Clearly a feasible flow vector exists in G iff a feasible circulation exists in $\hat{G}$. Condition (2.10) corresponding to $\mathbf{X} = \{s, t\}$, $\mathcal{N}$ respectively lead to $V(\mathbf{S}) \geqq -V(\mathbf{N})$, $-V(\mathbf{N}) \geqq V(\mathbf{S})$. Since $V_i = 0$ for all $i \in \mathcal{N} \backslash (\mathbf{S} \cup \mathbf{N})$, these conditions are equivalent to

$$V(\mathcal{N}) = 0 \tag{2.11}$$

which is the same as (1.21). Let $\mathbf{X} \subset \mathcal{N}, \bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X}$. Condition (2.10) corresponding to the subset of nodes $\mathbf{X}$, or $\mathbf{X} \cup \{s\}$, or $\mathbf{X} \cup \{s, t\}$, can be verified to lead to

$$k(\mathbf{X}, \bar{\mathbf{X}}) - \ell(\bar{\mathbf{X}}, \mathbf{X}) \geqq V(\mathbf{X}), \text{ for all } \mathbf{X} \subset \mathcal{N}, \bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X} \tag{2.12}$$

Thus (2.11) and (2.12) are the necessary and sufficient conditions for the existence of a feasible circulation in $\hat{G}$, and consequently a

feasible flow vector in G. If (2.11) holds, the procedure discussed above to find a feasible circulation in $\hat{G}$, either finds a feasible flow vector in G or a subset $\mathbf{X}$ violating (2.12).

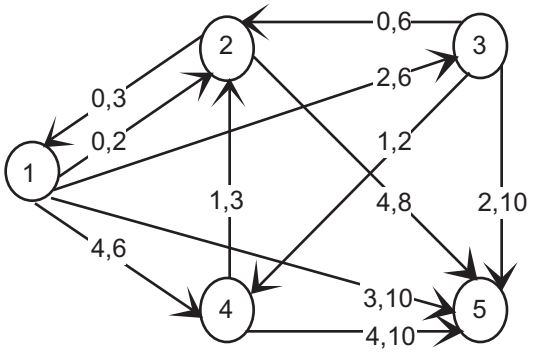


Figure 2.23: Data on the arcs is lower bound, capacity, in that order.

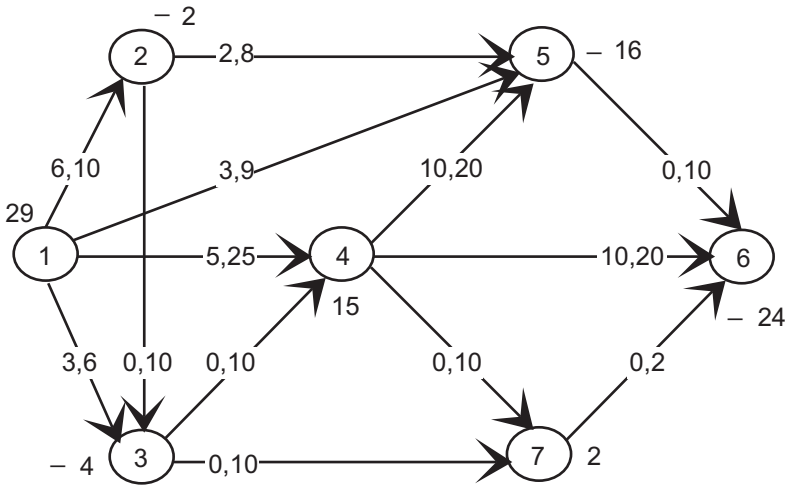


Figure 2.24:

Exercises

2.10 Find a maximum value flow from 1 to 5 on the network in Figure 2.23.

2.11 Find feasible flow vectors in the networks in Figures 2.24 and 2.25. Data on the arcs is lower bound, capacity in that order. The exogenous flow at each node is entered by its side.

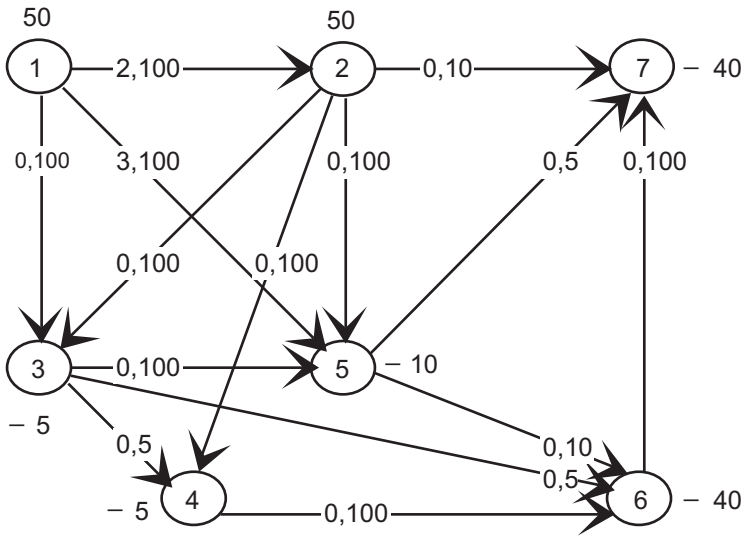


Figure 2.25: Data on the arcs is lower bound, capacity, in that order.

2.12 G $= (\mathcal{N}, \mathcal{A}, 0, k)$ is a directed single commodity flow network. **S, N** are the subsets of source, sink nodes respectively, and all the nodes in $\mathcal{N} \backslash (\mathbf{S} \cup \mathbf{N})$ are transient nodes. For each source node $i \in \mathbf{S}$, a positive quantity a_i is given; it is the maximum amount of material available at i for shipping out. For each sink node $j \in \mathbf{N}$, a positive quantity b_j is given; it is the minimum amount of material required to be delivered at j. Prove that a feasible flow vector satisfying all these constraints exists in G iff

$$k(\mathbf{X}, \bar{\mathbf{X}}) \geqq b(\mathbf{N} \cap \bar{\mathbf{X}}) - a(\mathbf{S} \cap \bar{\mathbf{X}}), \text{ for all } \mathbf{X} \subset \mathcal{N}, \bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X}$$

Also, in this problem, if all a_i, b_j, k_{ij} are integers and a feasible flow vector exists, then prove that an integer feasible flow vector exists.

2.13 G $= (\mathcal{N}, \mathcal{A}, \ell, k)$ is a directed single commodity network with $\ell \leqq k$. For each $i \in \mathcal{N}$, a_i and b_i are given integers satisfying $a_i \leqq b_i$. It

is required to find a flow vector $f = (f_{ij})$ in G satisfying $\ell \leqq f \leqq k$ and $a_i \leqq f(i, \mathcal{N}) - f(\mathcal{N}, i) \leqq b_i$ for each $i \in \mathcal{N}$. Prove that a flow vector satisfying these conditions exists in G iff

$$k(\mathbf{X}, \bar{\mathbf{X}}) \geqq \ell(\bar{\mathbf{X}}, \mathbf{X}) + \text{max. } \{a(\mathbf{X}), -b(\bar{\mathbf{X}})\}, \text{ for all } \mathbf{X} \subset \mathcal{N}, \bar{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X}$$

Discuss an efficient procedure that will either find a feasible flow vector if one exists, or determine a subset of nodes $\mathbf{X}$ violating the above condition.

2.14 Consider a round robin tournament between n baseball teams, with each team playing every other team exactly β times. For $i = 1$ to n, let α_i be the number of wins for the ith team at the conclusion of the tournament. Derive necessary and sufficient conditions on a given set of nonnegative integers $\alpha_1, \ldots, \alpha_n$ in order that they represent a possible win record (D. Gale).

2.15 $\tilde{f}$ is a non-integral circulation in a directed single commodity flow network G. Prove that there exists an integral circulation $\bar{f}$ in G satisfying $|\bar{f}_{ij} - \tilde{f}_{ij}| < 1$ for all arcs (i, j) in G. Discuss an efficient algorithm for finding such an $\bar{f}$.

How to find a maximum value flow in G with $\ell \neq 0$ using the preflow push algorithm

There is one significant difference between networks with zero lower bounds and those with nonzero lower bounds. If the lower bound vector is zero and the capacity vector is nonnegative, there is always a feasible flow vector, since the vector 0 is itself feasible. If the lower bound vector is nonzero, it is possible that there is no feasible flow vector.

The preflow push algorithm discussed in Section 2.5 is based on preflows. It obtains a sequence of preflows and terminates only when the preflow becomes a feasible flow vector, at which time it will be a

maximum value flow. That's why it cannot detect infeasibility directly and is thus applied directly only for solving the maximum value flow problem in networks with zero lower bounds. Let *Algorithm 1* refer to any such algorithm. Here we show that Algorithm 1 can be used to find the maximum value flow in the network $G = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ with $\ell \neq 0$, by using it in two phases. This two-phase procedure from Yi and Murty [1991] is the following:

PROCEDURE

Phase I In this phase, try to find a feasible flow vector in G. As discussed earlier, this problem can be transformed into that of finding a maximum value flow in an augmented network G^* in which lower bounds for all arc flows are zero. Because of this, a maximum value flow in G^* can be found by Algorithm 1 directly. From this either conclude that there is no feasible flow vector in G and terminate or obtain a feasible flow vector in it.

Suppose a feasible flow vector $\bar{f} = (\bar{f}_{ij})$ of value $\bar{v}$ has been found in G.

Phase II Find a maximum value feasible flow vector in the residual network $G(\bar{f}) = (\mathcal{N}, \mathcal{A}(\bar{f}), 0, \kappa = (\kappa_{ij}), \breve{s}, \breve{t})$ with the same nodes $\breve{s}, \breve{t}$ as source, sink nodes. Since the lower bounds on all arc flows are zero in $G(\bar{f})$; this can be carried out by Algorithm 1. Let $\bar{h} = (\bar{h}_{pq})$ be the maximum value flow vector obtained in $G(\bar{f})$, and ω its value. Lower bounds in the residual network $G(\bar{f})$ are 0, and for a pair of nodes p, q, if there are arcs (p, q) and (q, p) both with positive flows in $\bar{h}$, then the flows on them can be canceled (i.e., replace the larger of $\bar{h}_{pq}$, $\bar{h}_{qp}$ by their difference and the smaller by 0) and at least one of these flows converted to 0, leading to another feasible flow vector in $G(\bar{f})$ of the same maximum value. We assume that this is done. So, without loss of generality, we assume that for any pair of nodes, the vector $\bar{h}$ has positive flows in at most one of the two orientations for arcs joining them.

Define a flow vector $\hat{f} = (\hat{f}_{ij})$ in G, where for $(i, j) \in \mathcal{A}$, $\hat{f}_{ij}$ is given by

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} + \bar{h}_{ij} & \text{if there is a } + \text{ labeled arc } (i,j) \text{ in } G(\bar{f}), \text{ corresponding to } (i,j) \in \mathcal{A}, \text{ with } \bar{h}_{ij} > 0. \\ \bar{f}_{ij} - \bar{h}_{ji} & \text{if there is a } - \text{ labeled arc } (j,i) \text{ in } G(\bar{f}), \text{ corresponding to } (i,j) \in \mathcal{A}, \text{ with } \bar{h}_{ji} > 0. \\ \bar{f}_{ij} & \text{otherwise} \end{cases}$$

Then $\hat{f}$ is a maximum value feasible flow vector in G and its value is $\bar{v} + \omega$. Terminate.

THEOREM 2.12 *The flow vector $\hat{f}$ obtained in Phase II of the above procedure is a maximum value feasible flow vector in* G.

Proof Since $\bar{f}$ is a feasible flow vector in G of value $\bar{v}$, and $\bar{h}$ is a feasible flow vector of value ω in $G(\bar{f})$, the fact that $\hat{f}$ is a feasible flow vector of value $\bar{v} + \omega$ follows from the flow conservation equations satisfied by $\bar{f}$ and $\bar{h}$ in the respective networks G and $G(\bar{f})$ and by the definition of upper bounds in $G(\bar{f})$.

Now, to show that $\hat{f}$ is a mximum value flow in G, suppose it is not true. Then there must exist an FAP from $\breve{s}$ to $\breve{t}$ wrt $\hat{f}$ in G. Suppose it is $\mathcal{P}$. We will now show that using $\mathcal{P}$ we can construct an FAP from $\breve{s}$ to $\breve{t}, \mathcal{P}_1$, in $G(\bar{f})$ wrt $\bar{h}$.

1. If (i,j) is a forward arc on $\mathcal{P}$ with $\hat{f}_{ij} = \bar{f}_{ij}$, we have $\bar{f}_{ij} < k_{ij}$, so arc (i,j) exists in $G(\bar{f})$ with + label and capacity $k_{ij} - \bar{f}_{ij} > 0$, and since $\hat{f}_{ij} = \bar{f}_{ij}$, we must have $\bar{h}_{ij} = 0$. So put (i,j) as a forward arc on $\mathcal{P}_1$.

2. If (i,j) is a forward arc on $\mathcal{P}$ with $\hat{f}_{ij} > \bar{f}_{ij}$, from the definition of $\hat{f}$, (i,j) must be a + labeled arc in $G(\bar{f})$ with $\bar{h}_{ij} = \hat{f}_{ij} - \bar{f}_{ij} > 0$, and since $\hat{f}_{ij} = \bar{f}_{ij} + \bar{h}_{ij} < k_{ij}$, we have $\bar{h}_{ij} < k_{ij} - \bar{f}_{ij} = \kappa_{ij}$. So put (i,j) as a forward arc on $\mathcal{P}_1$.

3. If (i,j) is a forward arc on $\mathcal{P}$ with $\hat{f}_{ij} < \bar{f}_{ij}$, from the definition of $\hat{f}$, (j,i) must be a $-$ labeled arc in $G(\bar{f})$ with $\bar{h}_{ji} > 0$. Put (j,i) as a reverse arc on $\mathcal{P}_1$.

4. If (i,j) is a reverse arc on $\mathcal{P}$ with $\hat{f}_{ij} = \bar{f}_{ij}$, we have $\bar{f}_{ij} > \ell_{ij}$, so arc (j,i) must be a $-$ labeled arc in $G(\bar{f})$ with capacity $\kappa_{ji} = \bar{f}_{ij} - \ell_{ij} > 0$, and from the definition of $\hat{f}$, $\bar{h}_{ji} = 0$. Put (j,i) as a forward arc on $\mathcal{P}_1$.

5. If (i,j) is a reverse arc on $\mathcal{P}$ with $\hat{f}_{ij} > \bar{f}_{ij}$, from the definition of $\hat{f}$, (i,j) must be a $+$ labeled arc in $G(\bar{f})$ with $\bar{h}_{ij} = \hat{f}_{ij} - \bar{f}_{ij} > 0$. Put (i,j) as a reverse arc on $\mathcal{P}_1$.

6. If (i,j) is a reverse arc on $\mathcal{P}$ with $\hat{f}_{ij} < \bar{f}_{ij}$, from the definition of $\hat{f}$, (j,i) must be a $-$ labeled arc in $G(\bar{f})$ with $\bar{h}_{ji} = \bar{f}_{ij} - \hat{f}_{ij} > 0$; and since $\bar{f}_{ij} - \bar{h}_{ji} = \hat{f}_{ij} > \ell_{ij}$, we have $\bar{h}_{ji} < \bar{f}_{ij} - \ell_{ij} = \kappa_{ji}$. Put (j,i) as a forward arc on $\mathcal{P}_1$.

It can be verified that the path $\mathcal{P}_1$ constructed using statements 1 to 6 above is a path from $\breve{s}$ to $\breve{t}$ in $G(\bar{f})$, with the forward, reverse orientations for arcs on it as specified in these statements, and that it is an FAP from $\breve{s}$ to $\breve{t}$ in $G(\bar{f})$ with respect to $\bar{h}$. This contradicts the hypothesis that $\bar{h}$ is a maximum value flow from $\breve{s}$ to $\breve{t}$ in $G(\bar{f})$. So there does not exist any FAP from $\breve{s}$ to $\breve{t}$ in G with respect to $\hat{f}$. Hence $\hat{f}$ is a maximum value flow in G. ∎

The theorem shows that the two-phase procedure described here always finds a maximum value feasible flow vector in the given network G.

2.7 Sensitivity Analysis

Let $G = (\mathcal{N}, \mathcal{A}, \ell = 0, k, \breve{s}, \breve{t})$ be a directed single commodity flow network. Consider a particular arc $(i,j) \in \mathcal{A}$. Sensitivity analysis in G deals with the problem of deriving the maximum value flow as a function of k_{ij} as it varies from 0 to ∞, while all the other data remains unchanged.

To avoid confusion, denote k_{ij} by ξ and let $v(\xi)$ be the maximum value of flow in G as a function of ξ. Let $\mathbf{U}_1$ ($\mathbf{U}_2$) denote the set of all cuts separating $\breve{s}$ and $\breve{t}$ in G that contain (i,j) as a forward arc (do not contain (i,j) as a forward arc).

The capacity of any cut in $\mathbf{U}_2$ is unaffected by changes in the capacity ξ of arc (i, j). Suppose the minimum capacity among cuts in $\mathbf{U}_2$ is c_2; c_2 is defined to be $+\infty$ if $\mathbf{U}_2 = \emptyset$.

Let c_1 be the minimum capacity of cuts in $\mathbf{U}_1$ when $\xi = 0$. Since all cuts in $\mathbf{U}_1$ contain (i, j) as a forward arc, the minimum capacity among cuts in $\mathbf{U}_1$ as a function of ξ is $c_1 + \xi$.

Hence the minimum capacity of cuts separating $\check{s}$ and $\check{t}$ in G, as a function of ξ is min. $\{c_1 + \xi, c_2\}$. Thus $v(\xi) =$ min. $\{c_1 + \xi, c_2\}$.

Therefore, if $c_1 \geqq c_2, V(\xi) = c_2$ for every $\xi \geqq 0$. In this case there is a minimum capacity cut separating $\check{s}$ and $\check{t}$ in G which does not contain (i, j) as a forward arc, for every $\xi \geqq 0$.

Suppose $c_1 < c_2$. In this case $v(\xi) = c_1 + \xi$ for $0 \leqq \xi \leqq c_2 - c_1$, and in this interval for ξ there is a minimum capacity cut separating $\check{s}$ and $\check{t}$ in G, which contains (i, j) as a forward arc. For $\xi > c_2 - c_1, v(\xi) = c_2$. So, $v(\xi)$ increases as ξ increases from 0 to $c_2 - c_1$, and beyond $c_2 - c_1$ it does not change. The number $c_2 - c_1$ is therefore called the **critical capacity** of arc (i, j) and denoted by k_{ij}^*. So, we have $c_1 = v(0), c_2 = v(\infty)$, and $v(\xi) =$ min. $\{v(0) + \xi, v(\infty)\}$, for all $\xi \geqq 0$. See Figure 2.26.

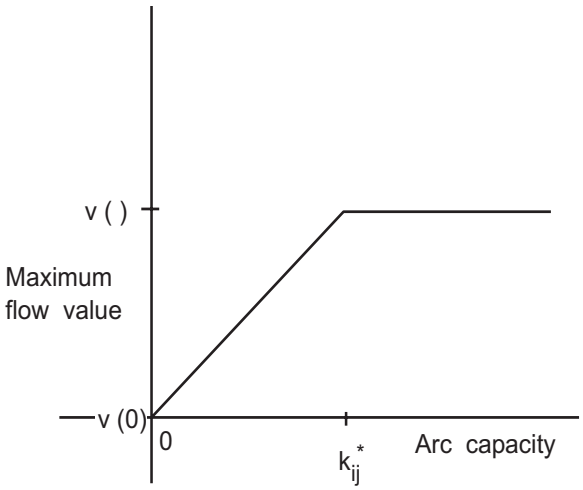


Figure 2.26: The maximum flow value as a function of the capacity of arc (i, j).k_{ij}^* is the critical capacity.

To compute the critical capacity $k^*_{ij} = v(\infty) - v(0)$, we have to solve two maximum value flow problems, one with the capacity of this arc set at ∞, and the other with this capacity set at 0. We have the following facts.

1. Whenever $k_{ij} < k^*_{ij}, (i, j)$ is a forward arc in every minimum capacity cut separating $\check{s}$ and $\check{t}$. If $k_{ij} > k^*_{ij}, (i, j)$ is not a forward arc in any minimum capacity cut separating $\check{s}$ and $\check{t}$. If $k_{ij} = k^*_{ij}$, there exists a minimum capacity cut separating $\check{s}$ and $\check{t}$ that contains (i, j) as a forward arc, and another that does not.

2. Destroying an arc in a network is equivalent to reducing its capacity to 0. The amount by which the maximum flow value would decrease if arc (i, j) is destroyed is min. $\{k_{ij}, k^*_{ij}\}$.

2.8 Exercises

2.16 In the single commodity flow network in Figure 2.27 all lower bounds are zero, and capacities are entered on the arcs. Find a maximum value flow, a minimum capacity cut, and the critical capacities of arcs (3, 5), (2, 6).

2.17 Discuss an efficient scheme for finding an arc, the destruction of which reduces the maximum flow value from the source to the sink the most.

2.18 Let k_{ij}, k^*_{ij} be the present capacity and critical capacities respectively of arc (i, j). Maximum $\{k^*_{ij} - k_{ij}, 0\}$ is the *scope* for increasing the maximum flow value by developing the arc (i, j). Discuss an efficient scheme for finding an arc with maximum scope from those in a specified subset of arcs.

2.19 Let $(i, j), (p, q)$ be two arcs in a directed single commodity flow network G $= (\mathcal{N}, \mathcal{A}, 0, k, \check{s}, \check{t})$. Let $v(\xi, \eta)$ denote the maximum flow value in G as a function of $\xi = k_{ij}, \eta = k_{pq}$, in the region $\xi \geqq 0, \eta \geqq 0$, while all the other data remains unchanged. Prove that $v(\xi, \eta) = \min \{v(0, 0)+\xi+\eta, v(0, \infty)+\xi, v(\infty, 0)+\eta, v(\infty, \infty)\}$. Using this, show that

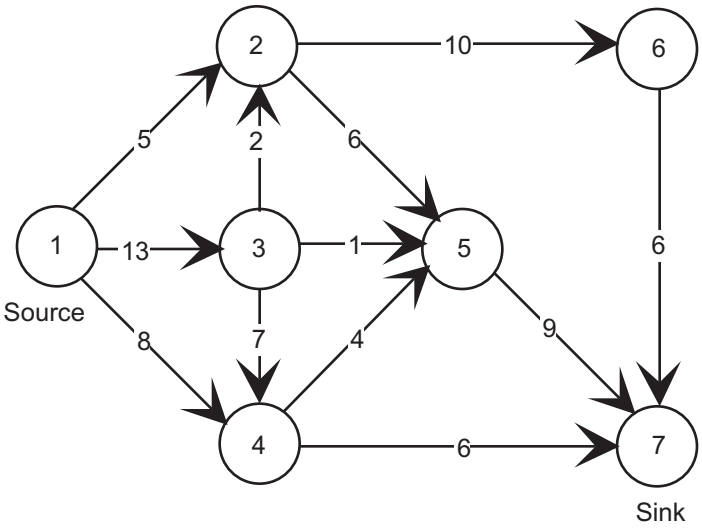


Figure 2.27:

in the nonnegative quadrant of the $\xi, \eta-$ plane, $v(\xi, \eta)$ is a piecewise linear function, dividing this quadrant into at most four convex regions in each of which $v(\xi, \eta)$ is linear. Illustrate with a numerical example.

2.20 Using the notation of Exercise 2.19, prove that for all rectangles with vertices $(\xi, \eta), (\xi + h, \eta), (\xi, \eta + r), (\xi + h, \eta + r)$ in the $\xi, \eta-$ nonnegative quadrant, the difference quotient $(v(\xi+h, \eta+r)-v(\xi+h, \eta)-v(\xi, \eta+r)+v(\xi, \eta))/hr$ always has the same sign.

2.21 Let $[\mathbf{X}, \bar{\mathbf{X}}]$ be a cut, and f a feasible flow vector in the directed single commodity flow network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, V)$. Prove that the net flow across the cut $[\mathbf{X}, \bar{\mathbf{X}}]$ in f is $V(\mathbf{X})$.

2.22 $\bar{f} = (\bar{f}_{ij})$ is a feasible flow vector of value $\bar{v}$ in a directed single commodity flow network $\mathrm{G} = (\mathcal{N}, \mathcal{A}, \ell, k, \check{s}, \check{t})$ with $0 < \ell < k < \infty$ and $|\mathcal{A}| = m$. δ is a given small positive number, much smaller than any $k_{ij} - \ell_{ij}$. We plant a rooted tree at $\check{s}$ and grow it by the following labeling rules.

Forward labeling If i is labeled, j is unlabeled, and $(i, j) \in \mathcal{A}$ and $\bar{f}_{ij} \leqq k_{ij} - \delta$; label j with $(i, +)$.

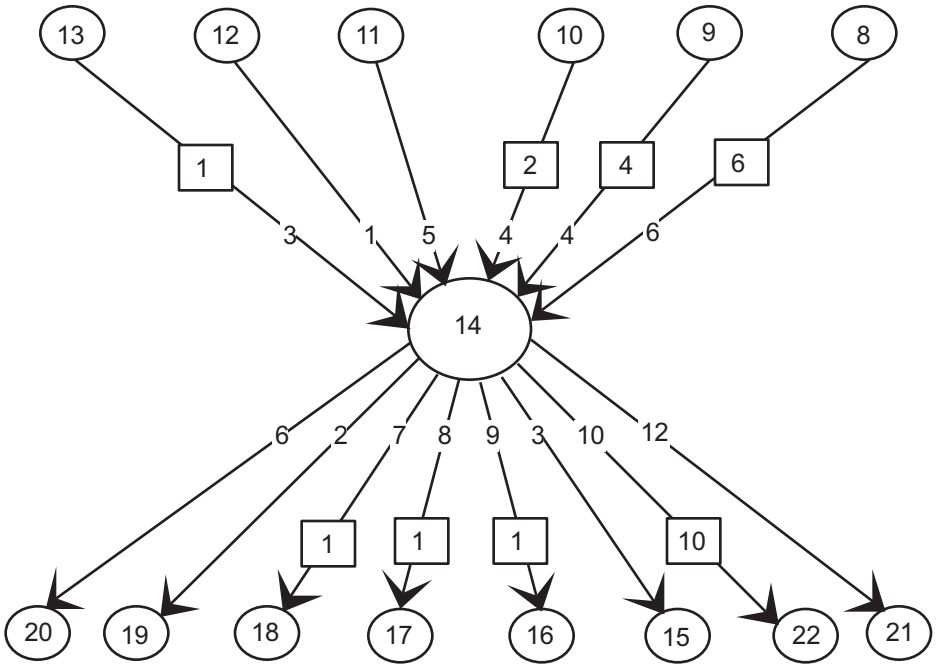


Figure 2.28:

Reverse labeling If i is labeled, j is unlabeled, and $(j,i) \in \mathcal{A}$ and $\bar{f}_{ij} \geqq \ell_{ij} + \delta$; label j with $(i,-)$.

The tree growth routine has terminated without $\breve{t}$ ever getting labeled. $\mathbf{X}$ is the set of labeled nodes, and $\bar{\mathbf{X}}$ its complement at termination. Prove that the maximum flow value in G is $\leqq \bar{v} + m\delta$, and that the minimum capacity of cuts separating $\breve{s}$ and $\breve{t}$ in G is $\geqq$ capacity of $[\mathbf{X}, \bar{\mathbf{X}}]$ - $m\delta$.

2.23 In Figure 2.28, we show the arcs incident at the reference node 14 in a layered network in which a maximal flow is being found by the MKM algorithm. The number on each arc is its capacity in this layered network. Nonzero flow amounts at this stage are entered in little boxes by the side of the arcs. Of the nodes shown, only 10 to 20 are in the set $\mathbf{Y}$ at this stage, 8, 9, 21, 22 are not. Clearly, the arcs incident at nodes 8, 9, 21, 22 are not in the set $\mathbf{F}$ at this stage. Compute the reference potential at this stage. Do flow pushing and pulling at node 14, and

indicate the new flow amounts on the arcs shown in Figure 2.28. Also indicate how much flow pushing or pulling has to be carried out at each of the adjacent node of 14 as this step is continued.

2.24 Discuss the main difference in the strategies employed by the following algorithms: (a) Ford-Fulkerson labeling algorithm, (b) Edmonds-Karp labeling algorithm, (c) Dinic's algorithm for finding a maximum value flow in a directed single commodity flow network. Are all three algorithms guaranteed to solve the problem always? If not, mention the conditions under which they can solve the problem. What is the worst case computational complexity of each of these algorithms?

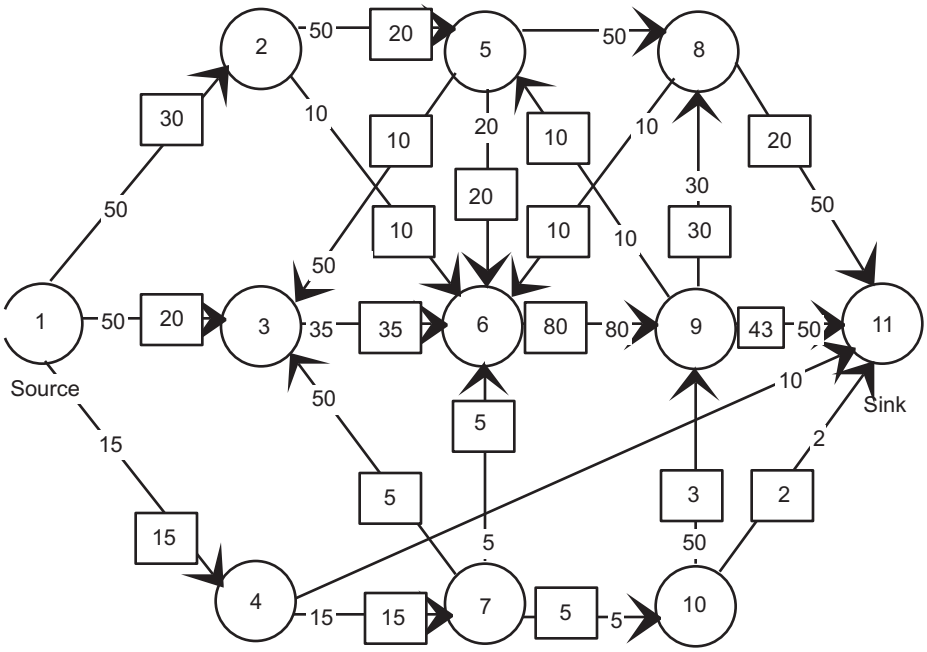


Figure 2.29:

Consider the network in Figure 2.29. All lower bounds are zero, and the capacities are entered on the arcs. Nonzero flow amounts in a feasible flow vector are marked in little boxes by the side of the arcs. Draw the layered network wrt this flow vector, and obtain the referent

by carrying out any pruning that is necessary. Use the MKM algorithm to find a maximal flow vector in the referent. Augment the flow vector in the original network using this maximal flow.

2.25 Consider the following variant of the initial version of the labeling method for finding the maximum value flow in the directed single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, \check{s}, \check{t})$ beginning with the feasible flow vector $f^o = (f_{ij}^0)$, called "*Capacity.*" At each stage choose the next node to be labeled by the following procedure: let $f = (f_{ij})$ be the feasible flow vector in G at that stage. Define $\mathbf{E}^+ = \{(i,j) : i$ labeled, j unlabeled, $(i,j) \in \mathcal{A}$, and $f_{ij} < k_{ij}\}$, $\mathbf{E}^- = \{(i,j) : j$ labeled, i unlabeled, $(i,j) \in \mathcal{A}$ and $f_{ij} > \ell_{ij}\}$. Define $\varepsilon_{ij} = k_{ij} - f_{ij}$ for $(i,j) \in \mathbf{E}^+$, or $f_{ij} - \ell_{ij}$ for $(i,j) \in \mathbf{E}^-$. Select the next arc to be made in-tree to be $(p,q) \in \mathbf{E}^+ \cup \mathbf{E}^-$ satisfying $\varepsilon_{pq} =$ maximum $\{\varepsilon_{ij} : (i,j) \in \mathbf{E}^+ \cup \mathbf{E}^-\}$. If $(p,q) \in \mathbf{E}^+$, label q with the label $(p,+)$; and if $(p,q) \in \mathbf{E}^-$, label p with $(q,-)$. Prove the following about this algorithm "Capacity."

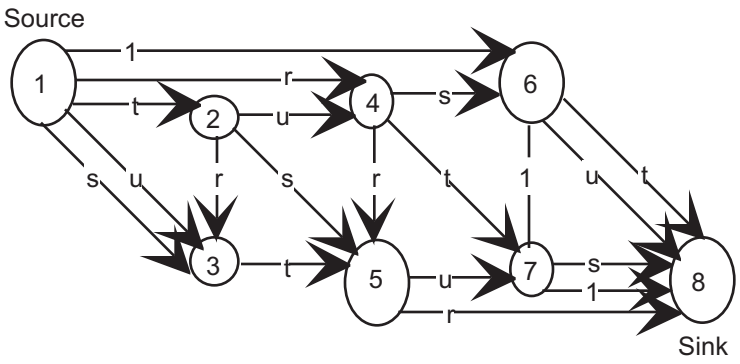


Figure 2.30:

(i) In each iteration, the FAP obtained gives the largest possible flow augmentation among all FAPs from $\check{s}$ to $\check{t}$ wrt f at that stage.

(ii) When ℓ, k, f^o are all integer vectors, it terminates with the maximum value flow vector after at most $0(m(\log m + \log \overline{c}))$ flow augmentations, where $m = |\mathcal{A}|$ and $\overline{c}$ is the average of $(k_{ij} - \ell_{ij})$ over arcs in G if all k_{ij} are finite, or the logarithm of the finite

capacity of any cut in G if some k_{ij} is infinite. This establishes that the algorithm is polynomially bounded.

(iii) Consider the network $G_r = (\mathcal{N}, \mathcal{A}, 0, k^r, \check{s}, \check{t})$ which is the same as the network discussed in Example 2.1, except that the capacity of arc (i, j) is now $k_{ij}^r = \lfloor 2^r k_{ij} \rfloor$, where k_{ij} is the real capacity of (i, j) as defined in Example 2.1. Here $\log(\overline{c}) = O(r)$. Beginning with the zero flow vector in prove that the algorithm "Capacity" takes $0(r)$ flow augmentations to find the maximum value flow in G_r, establishing that on networks with integer data and integer initial feasible flow vector, the computational effort needed by this algorithm may continue to grow with the size of the capacity data, indefinitely, on the same network.

(iv) Consider the network in Figure 2.30 with capacity data entered on the arcs, where $\alpha = \frac{1}{2}(-1+\sqrt{5}), r = \alpha, s = \alpha/2, t = (1+\alpha)/2$ and $u = 1/2$. All lower bounds are 0. Beginning with the initial flow vector $f^o = 0$, prove that the algorithm "Capacity" produces an infinite sequence of feasible flow vectors converging to a maximum value flow vector in an infinite number of iterations. This establishes that on networks with nonrational data "Capacity" may not find a maximum value flow within a finite number of iterations.

(v) Prove that the sequence of flows constructed by "Capacity" always converges to a maximum value flow vector, whether the data is rational or not.

(Queyranne [1980])

2.26 Rectilinear Distance Facility Location Located at $(a_i, b_i), i =$ 1 to m in $\mathbb{R}^2$ are m existing factories. We need to determine $(x_j, y_j), j =$ 1 to n, optimum locations for n new facilities to minimize the weighted sum of rectilinear distances

$$z(x, y) = \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ji}(|x_j - a_i| + |y_j - b_i|) + \frac{1}{2} \sum_{j=1}^{n} \sum_{r=1}^{n} v_{jr}(|x_j - x_r| + |y_j - y_r|)$$

where $x = (x_1 \ldots, x_n)^T, y = (y_1, \ldots, y_n)^T, w_{ji} \geqq 0, v_{jr} = v_{rj} \geqq 0$ for all $r, j, i \cdot z(x, y)$ can be written as $g(x) + h(y)$, so problem equivalent to minimizing $g(x)$ and $h(y)$ separately. We consider the problem of minimizing $g(x)$.

$$g(x) = \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ji}|x_j - a_i| + \frac{1}{2} \sum_{j=1}^{n} \sum_{r=1}^{n} v_{jr}|x_j - x_r|.$$

(i) Prove that there exists an $x^* = (x_j^*)$ minimizing $g(x)$ in which $x_j^* \in \{a_1, \ldots, a_m\}$ for all $j = 1$ to n.

(ii) Consider the special case in which $a_i = i, i = 1$ to m. Let $f(x)$ be the value of $g(x)$ in this special case. So, consider the problem:

$$\text{minimize} \quad f(x)$$

$$\text{subject to} \quad x_j \in \{1, \ldots, m\}, \ j = 1 \text{ to } n \tag{2.13}$$

For any $p \in \{1, \ldots, m\}$ and $x = (x_j)$ feasible to (2.13), let $\mathbf{S}_p = \{j : x_j \leqq p\}, \overline{\mathbf{S}}_p = \{j : x_j > p\}$. Prove that $f(x) = \sum_{p=1}^{m-1} c_p(\mathbf{S}_p, \overline{\mathbf{S}}_p)$ where

$c_p(\mathbf{S}_p, \overline{\mathbf{S}}_p) = \sum_{j \in \mathbf{S}_p} \sum_{i=p+1}^{m} w_{ji} + \sum_{j \in \overline{\mathbf{S}}_p} \sum_{i=1}^{p} w_{ji} + \sum_{j \in \mathbf{S}_p} \sum_{r \in \overline{\mathbf{S}}_p} v_{jr}$.

Consider the undirected network $\mathrm{G}_p = (\mathcal{N}, \mathcal{A}, 0, k^p)$ with $\mathcal{N} = \{s, 1, \ldots, n, t\}, \mathcal{A} = \{(s; j), (j; t) : \text{for all } j = 1 \text{ to } n\} \cup \{(j_1; j_2); \text{for all } j_j \neq j_2 \in \{1, \ldots, n\}\}$, called the p *locale network*, where the capacity vector k^p is given by the following:

$$k_{s;j}^p = \sum_{i=1}^{p} w_{ji}, k_{j;t}^p = \sum_{i=p+1}^{m} w_{ji}, j = 1 \text{ to } n$$

$$k_{j_1;j_2}^p = v_{j_1,j_2} \quad \text{for all} \quad j_1 \neq j_2 \in \{1, \ldots, n\}.$$

If $\mathbf{X} \cup \overline{\mathbf{X}}$ is a partition of $\{1, \ldots, n\}$, show that the capacity of the cut $(\{s\} \cup \mathbf{X}; \{t\} \cup \overline{\mathbf{X}})$ in G_p is $c_p(\mathbf{X}, \overline{\mathbf{X}})$. Prove that $x^o = (x_j^o)$ is an optimum solution of (2.13) iff for each $p = 1$ to $m - 1$, the cut $(\{s\} \cup \mathbf{X}_p; \{t\} \cup \overline{\mathbf{X}}_p)$, where $\mathbf{X}_p = \{j : x_j^o \leqq p\}, \overline{\mathbf{X}}_p = \{j : x_j^o \geqq p + 1\}$, is a minimum capacity cut in G_p. Conversely, if $(\{s\} \cup \mathbf{Y}_p; \{t\} \cup \overline{\mathrm{Y}}_p)$ is a minimum capacity cut in G_p, show that there

exists an optimum solution of (2.13), $x^* = (x_j^*)$ satisfying $x_j^* \leqq p$ for all $j \in \mathbf{Y}_p$ and $x_j^* \geqq p+1$ for all $j \in \overline{\mathrm{Y}}_p$. From this, show that (2.13) can be solved as a minimum capacity cut problem, if $m = 2$.

(iii) Consider (2.13) when $m \geqq 3$. Let $N_1, \ldots, N_q$ be a partition of $\{1, \ldots, n\}$ and let $B_1, \ldots, B_q$ be integers satisfying $1 \leqq B_1 \leqq B_2 \leqq \cdots \leqq B_q \leqq m$. Consider the problem:

$$\begin{gathered} \text{minimize} \quad f(x) \\ \text{subject to} \quad B_s \leqq x_j \leqq B_{s+1}, \text{ for } \; j \in N_s, s = 1 \text{ to } \; q \qquad (2.14) \\ x_j \text{ integer for all } \; j. \end{gathered}$$

Prove that for any $s = 1$ to q, the optimum values of x_j for $j \in N_s$ are independent of the actual locations of those facilities (new and old) located outside the interval B_s to B_{s+1}, but are dependent on which of these facilities are located at points $\leqq B_s$ and which are located at points $\geqq B_{s+1}$.

	w_{ji}			
old facility	$i=1$	2	3	4
new facility $j=1$	3	2	3	1
2	1	2	1	2
3	9	1	2	2

	v_{jr}		
$r=$	1	2	3
$j=1$	·	3	2
2	3	·	1
3	2	1	·

Using this, show that (2.14) can be decomposed into q problems of the same form as (2.13), where for each s all new facilities $j \in \cup(N_p; p = 1 \text{ to } s-1)$ and $j \in \cup(N_p : p = s+1 \text{ to } q)$ are treated as old facilities located at B_s, B_{s+1}, respectively and

old facilities $i \leqq B_s$, and $i \geqq B_{s+1}$ are treated as old facilities located at B_s, B_{s+1} respectively. From these results, develop an algorithm for solving (2.13) by finding a minimum capacity cut in each of the p local networks G_p, for $p = 1$ to $m-1$. Solve the numerical problem (2.13) with $m = 4, n = 3$, and the data in the tables given above.

(iv) Now consider the original problem of minimizing $g(x)$. Assume that $a_i s$ are ordered so that $a_1 < a_2 < \ldots < a_m$. If $x^o = (x_j^o)$ is an optimum solution of (2.13), then prove that $x^* = (x_j^*)$, where $x_j^* = a_i$ when $x_j^o = i$, is an optimum solution of this problem.

(Picard and Ratliff [1978])

2.27 The Sharing Problem G $= (\mathcal{N}, \mathcal{A}, 0, k)$ is a single commodity directed flow network with $\mathbf{S} \subset \mathcal{N}$ as the set of source nodes with a_i units material available at source node $i \in \mathbf{S}$; and $\mathbf{D} \subset \mathcal{N}$ as the set of sink nodes with w_j being the weight of sink node $j \in \mathbf{D}$ for allocation of material in a shortage situation. Under shortage, an equitable distribution of material should attempt to maximize the minimum weighted net flow reaching the sink nodes in $\mathbf{D}$. Discuss an approach for solving this problem. Solve the numerical problem in Figure 2.31 using this approach(Brown [1979]).

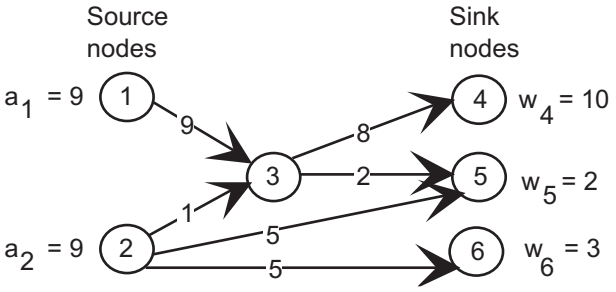


Figure 2.31: All lower bounds are 0. Capacities are entered on the arcs.

2.28 $\mathbf{D}$ is a finite set of points. For each $d \in \mathbf{D}$, c_d is the cost of choosing it. $c = (c_d) > 0$ is given. $\triangle$ is a class of subsets of $\mathbf{D}$.

For each element $\sigma \in \triangle, p_\sigma$ is the profit of choosing that element. $p = (p_\sigma) > 0$ is given. A *selection* is a collection of elements from $\triangle$ together with all points of $\mathbf{D}$ which belong to this collection. If $\{\sigma_1, \ldots, \sigma_r\} \subset \triangle$ is the set of elements in a selection, its value is defined to be $\sum_{i=1}^{r} p_{\sigma_i} - \sum(c_d : \text{over } d \in \bigcup_{i=1}^{r} \sigma_i)$. It is required to find a selection of maximum value. Consider a directed bipartite network $G = (\mathcal{N}_1, \mathcal{N}_2; \mathcal{A}, 0, k)$ with $\mathcal{N}_1 = \{\check{s}\} \cup \mathbf{D}$, $\mathcal{N}_2 = \{\check{t}\} \cup \triangle$ where $\check{s}, \check{t}$ are source and sink nodes, and arcs $(\check{s}, \sigma)$ with capacity $k_{s\sigma} = p_\sigma$, for each $\sigma \in \triangle$, $(d, \check{t})$ with capacity $k_{dt} = c_d$, for each $d \in \mathbf{D}$, and (d_i, σ_j) with capacity ∞, for each $d_i \in \sigma_j$, for each $\sigma_j \in \triangle$. Show that there is a one-to-one correspondence between selections, and cuts in G separating $\check{s}$ and $\check{t}$ which contain no forward arcs of the type (d_i, σ_j). Using this, show that a maximum value selection corresponds to a minimum capacity cut separating $\check{s}$ and $\check{t}$ in G, and hence can be found efficiently by finding a maximum value flow from $\check{s}$ to $\check{t}$ in G.

As an application, consider the following problem of a transport undertaking. They are considering the installation of freight handling terminals at locations d_1, d_2, d_3, and d_4. The cost of installing the terminals at any of these locations is \$5 million. The existence of terminals at certain pairs of terminals permits a service to be operated between those terminals, which is associated with a net profit. These pairs are: (d_1, d_2) with a profit of \$2 million, $(d_1; d_3)$ with a profit of \$9 million, $(d_2; d_3)$ with a profit of \$4 million, and $(d_3; d_4)$ with a profit of \$6 million. Find a selection of services and terminals to maximize excess of profit over cost. (Rhys [1970], Balinski [1970] both of Chapter 1, Murchland [1968])

2.29 Let $G = (\mathcal{N}, \mathcal{A}, 0, k, 1, n)$ be a directed single commodity flow network with $|\mathcal{N}| = n$, and nodes $1, n$ as the source, sink nodes respectively, with $k > 0$. Any partition of the nodes in $\mathcal{N}$ into $\mathbf{X}, \overline{\mathbf{X}}$ with $1 \in \mathbf{X}, n \in \overline{\mathbf{X}}$, can be represented by the 0-1 vector $x = (x_1, x_2, \ldots, x_n)$ defined on $\mathcal{N}$ by

$$x_i = \begin{cases} 1, & \text{if } i \in \mathbf{X} \\ 0, & \text{if } i \in \overline{\mathbf{X}} \end{cases}$$

and conversely any $0 - 1$ vector x defined on $\mathcal{N}$ with $x_1 = 1, x_n = 0$ defines such a partition of $\mathcal{N}$.

(i) If $[\mathbf{X}, \overline{\mathbf{X}}]$ is a cut separating 1 and n in G, associated with the 0-1 vector $x = (x_i)$ on $\mathcal{N}$, prove that its capacity $k(\mathbf{X}, \overline{\mathbf{X}}) = \sum(k_{ij}x_i(1 - x_j);\ \text{over}(i, j) \in \mathcal{A})$.

(ii) From the above result, show that the quadratic 0-1 integer programming problem, where $q_{jr} \geqq 0$ for all j and r, can be solved very efficiently by solving a maximum value flow problem on a related network. Based on this, develop an efficient algorithm for this quadratic programming problem.

$$\text{minimize } \sum_{j=1}^{n} p_j y_j - \sum_{j=1}^{n}\sum_{r=1}^{n} q_{jr} y_j y_r$$
$$\text{subject to } y_j = 0 \text{ or } 1 \text{ for all } j = 1 \text{ to } n$$

(Picard and Ratliff [1975])

2.30 Let $\mathbf{\Gamma} = \{1, \ldots, n\}, \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ is a family of subsets of $\mathbf{\Gamma}$ each of cardinality $\geqq 2$. Consider the following 0-1 nonlinear programming problem.

$$\begin{aligned} \text{maximize } z(x) &= \sum_{t=1}^{r} a_t(\mathbf{\Pi}_{j\in\mathbf{S}_t} x_j) + \sum_{j=1}^{n} c_j x_j \\ \text{subject to } x_j &\in \{0, 1\}, \text{ for all } j \in \Gamma. \end{aligned} \tag{2.15}$$

where $(a_1, \ldots, a_r) > 0$. Now define new variables $y_1, \ldots, y_r$, with y_t corresponding to the set $\mathbf{S}_t$ for $t = 1$ to r. Show that (2.15) is equivalent to the following 0-1 integer program (2.16). Show that (2.16) is a "selection problem" as it is defined in Exercise 2.28. In (2.16) if $c_j \geqq 0$ for some j, prove that $x_j = 1$ in an optimum solution. Hence such variables can be fixed equal to 1 and the problem size reduced. In the sequel we assume that $c_j < 0$ for all j. Create a network G with node set $\mathcal{N} = \{\check{s}, \mathbf{S}_1, \ldots, \mathbf{S}_r, 1, \ldots, n, \check{t}\}$. Arcs in it are $(\check{s}, \mathbf{S}_t)$ with capacity a_t for $t = 1$ to r; $(\mathbf{S}_t, j)$ for each $j \in \mathbf{S}_t$, $t = 1$ to r with capacity ∞; and $(j, \check{t})$ with capacity $-c_j$ for $j = 1$ to n. All lower bounds in G are 0.

$$\text{maximize } \sum_{t=1}^{r} a_t y_t + \sum_{j=1}^{r} c_j x_j$$

$$\text{subject to } y_t \leq x_j, \text{ for all } j \in \mathbf{S}_t, \text{ for all } t = 1 \text{ to } r \tag{2.16}$$

$$y_t = 0 \text{ or } 1 \text{ for all } t = 1 \text{ to } r$$
$$x_j = 0 \text{ or } 1 \text{ for all } t = 1 \text{ to } n.$$

Prove that the variables x_j which take on a value of 1 in an optimum solution of (2.15) or (2.16) correspond to labeled vertices $j \in \mathcal{N}$ after a maximum value flow in G from $\breve{s}$ to $\breve{t}$ has been found by any of the labeling algorithms. Solve the following numerical problem using this approach.

$$\begin{aligned} \max z(x) = \; & 2x_1x_2 + 2x_1x_2x_3 + 6x_1x_2x_4 \\ & + x_2x_3x_4 + x_1x_2x_3x_4 - 2x_1 \\ & - x_2 - 5x_3 - 2x_4 \end{aligned}$$

$$\text{subject to } x_j = 0 \text{ or } 1, \text{ for } j = 1 \text{ to } 4.$$

(Picard and Queyranne [1982a])

2.31 Let G = $(\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ be a directed connected single commodity flow network with $k > \ell$. Among all minimum capacity cuts separating $\breve{s}$ and $\breve{t}$ in G, it is required to find one satisfying one of the following additional properties: (i) it has the smallest number of forward arcs, (ii) it has the smallest number of reverse arcs, or (iii) it has the smallest number of arcs. Develop efficient algorithms for these problems (Hamacher [1982]).

2.32 Consider the connected directed single commodity flow network G = $(\mathcal{N},\ \mathcal{A},\ 0,\ k,\ \breve{s},\ \breve{t})$ where $k > 0$. Let f^* be a maximum value feasible flow vector in G of value v^*, and f a feasible flow vector of value $v < v^*$. Define a feasible flow vector g in the residual network wrt f, G $(f) = (\mathcal{N}, \mathcal{A}(f), 0, \kappa, \breve{s}, \breve{t})$ by the following:

1. For each $(i,j) \in \mathcal{A}$ satisfying $0 < f_{ij} < k_{ij}$, we have $(i,j) \in \mathcal{A}(f)$ and $(j,i) \in \mathcal{A}(f)$, make g_{ij} = maximum $\{0, f^*_{ij} - f_{ij}\}$, g_{ji} = maximum $\{0, f_{ij} - f^*_{ij}\}$.

2. For each $(i,j) \in \mathcal{A}$ satisfying $f_{ij} = 0$, we have $(i,j) \in \mathcal{A}(f)$, make $g_{ij} = f^*_{ij}$.
3. For each $(i,j) \in \mathcal{A}$ satisfying $f_{ij} = k_{ij}$, we have $(j,i) \in \mathcal{A}(f)$, make $g_{ji} = -(f^*_{ij} - f_{ij})$.

Prove that g is a maximum value feasible flow vector in $G(f)$. Prove that if the node partition $[\mathbf{X}, \overline{\mathbf{X}}]$ defines minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G, then it defines a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in $G(f)$ (Ramachandran [1987]).

2.33 The Maximum Flow Problem is Not Easier in an Acyclic Network Than in a General Network. Let $G = (\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$ be a connected directed single commodity flow network. Without any loss of generality we assume that every node and arc in G lies on at least one chain from $\breve{s}$ to $\breve{t}$ (otherwise such things could be deleted).

Find a depth first search spanning tree $\mathbb{T}$ rooted at $\breve{s}$ in G and let $\mathbf{B}$ be the set of arcs in G that are back arcs wrt $\mathbb{T}$. Get a new network $G^* = (\mathcal{N}\mathcal{A}^*, 0, k^*, \breve{s}, \breve{t})$ from G, and a feasible flow vector g in it by doing the following for each $(i,j) \in \mathbf{B}$: Replace (i,j) by (j,i) but keep its capacity the same, k_{ij}. Define the flow g_{ji} on this new arc to be its capacity k_{ij}. Introduce new arcs $(\breve{s}, j)$, $(i, \breve{t})$ also of the same capacity k_{ij}, and make the flow on both of them equal to this capacity. Verify that the resulting network G^* is acyclic, that the flow defined on it, g, has value $\sum(k_{ij} : \text{over } (i,j) \in \mathbf{B})$ and that it saturates all the newly introduced arcs.

Let $G^*(g)$ be the residual network of G^* wrt g. Verify that $G^*(g)$ is G together with some additional arcs either incident into $\breve{s}$ or incident out of $\breve{t}$. So, any partition of the nodes in $\mathcal{N}$ that defines a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ for $G^*(g)$ also defines a minimum capacity cut for G. By Exercise 2.32, the node partition that induces a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G^* induces also a minimum capacity cut for $G^*(g)$. Using these facts show that the problem of finding a minimum capacity cut separating the source and sink in the general directed network G reduces in linear time to a corresponding problem in an acyclic network. Then show that the maximum value flow problem is a general directed network reduces in linear time to a corresponding problem in an acyclic network (Ramachandran [1987]).

2.34 Consider the single commodity flow network $G = (\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$. A subset $\mathbf{A} \subset \mathcal{A}$, $|\mathbf{A}| = r$ is said to be the set of r most vital arcs in this network, if the simultaneous removal of the arcs in $\mathbf{A}$ results in the greatest decrease in the maximum flow value in the remaining network, among all subsets of arcs of cardinality r. Prove that the r most vital arcs in G are the r largest capacity arcs in a particular cut separating $\breve{s}$ and $\breve{t}$ in G. Develop an algorithm for finding such a set of arcs (Ratliff, Sicilia and Lubore [1975]).

2.35 Consider the following discrete quadratic programming problem. Define $\mathcal{N} = \{1, \ldots, n\}, \mathcal{A} = \{(i,j) : i,j \in \mathcal{N}$ such that $d_{ij} \neq 0\}$, G = $(\mathcal{N}, \mathcal{A}, k)$, where for $(i,j) \in \mathcal{A}, k_{i;j} = d_{ij}$. Define the capacity of a cut in G to be the sum of k_{ij} over (i,j) in the cut. Show that this quadratic program is equivalent to the problem of finding a maximum capacity cut in G.

$$\text{minimize} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j$$

$$\text{subject to } x_i \in \{-1, 1\}, \text{ for each } i = 1 \text{ to } n.$$

(Barahona [1982])

2.36 Let $G = (\mathcal{N}, \mathcal{A}, 0, k)$ be a connected undirected single commodity flow network, with $k > 0$. Let $v(x,y)$ denote the maximum flow value in this network with x as the source node and y as the sink node. Hence $v(x,y)$ is a positive valued function defined for pairs of distinct nodes of G, which will be called a *flow value function* of G. Prove the following:

- **(i)** $v(x,y) = v(y,x)$, for all $x \neq y \in \mathcal{N}$.
- **(ii)** $v(x,y) \geqq \min\{v(x,z), v(z,y)\}$, for every three distinct nodes x, y, z in $\mathcal{N}$.
- **(iii)** $v(x_1, x_r) \geqq \min\{v(x_1,x_2), v(x_2,x_3), \ldots, v(x_{r-1}, x_r\}$, for any sequence of distinct nodes $x_1, \ldots, x_r$ in G.
- **(iv)** From (ii) show that among $v(x,y), v(x,z)$ and $v(z,y)$, at least two must be equal, and the third is $\geqq$ their common value.

(v) Let $|\mathcal{N}| = n$. Let H be the complete undirected network on $\mathcal{N}$ with $v(x;y)$ as the length of the edge $(x;y)$, for $x \neq y \in \mathcal{N}$. Let $\mathbb{T}$ be a maximum length spanning tree in H. Using (iii) show that the length of every out-of-tree edge in H must be equal to the length of some in-tree edge. Thus prove that of the $n(n-1)$ flow values $v(x,y)$ in G, there are at most $(n-1)$ numerically distinct values.

(vi) Given any positive symmetric function $v(\cdot,\cdot)$ defined over pairs of distinct nodes in $\mathcal{N}$, satisfying the "traingle" inequality in (ii), prove that a spanning tree spanning the nodes in $\mathcal{N}$ can be constructed and capacities of edges in this tree defined in such a way that $v(\cdot,\cdot)$ is the flow value function on this tree.

(vii) Devise a scheme to determine the flow value function $v(\cdot,\cdot)$ on G by the successive solution of at most $(n-1)$ maximum value flow problems.

(Gomory and Hu [1961])

2.37 The Allocation of Specialists to Hospitals in a Region There are four hospitals, h_1, h_2, h_3, h_4, in a region. There are seven specialties for which there are plans to develop additional facilities (i.e., hospital beds dedicated to those specialties) in the region, these are s_1 to s_7 as indicated in the table given below. Lower and upper bounds are imposed on the total number of beds allocated to (i) each speciality in each hospital, and (ii) to each hospital. Determine one feasible allocation of speciality beds to hospitals satisfying all these constraints, using a network formulation. Discuss how one can determine an "optimum allocation" among all feasible allocations of specialty beds to hospitals in this problem (Duncan [1979]).

Specialty	Lower/upper bounds on no. of beds allocated to specialty in hospital h_1	h_2	h_3	h_4	Total beds allocated to specialty
ENT s_1	17/28	7/11	-	-	28
Dental surgery s_2	3/5	1/2	-	-	5
Plastic surgery s_3	24/39	10/15	-	-	39
Rheumatology s_4	4/10	2/5	2/8	4/9	14
T. & O. surgery s_5	80/97	34/51	-	-	131
General surgery s_6	51/136	50/71	34/92		183
General medicine s_7	51/139	40/73	50/111	28/75	187
Lower/upper bounds on number of beds allocated to hospital	0/247	0/176	0/183	0/100	

T. & O. is Traumatic and Orthopedic

2.38 There are n jobs ordered as $1, 2, \ldots, n$, to be processed on either of two available machines. Job i has processing time of p_i, q_i respectively, depending on whether it is processed on machine A or B. In the subset of jobs assigned to each machine, they can only be processed in the order of lowest number job first. The flow time of a job is defined to be the duration of time lapse from the beginning (i.e., time point 0) till its processing is completed. It is required to assign the jobs to the two machines so as to minimize the sum of flow times of all the jobs. Define $x_i = 1$, if job i is assigned to machine A, 0 if it is assigned to B. The 0-1 assignment vector is $x = (x_1, \ldots, x_n)^T$. Show that the total flow time corresponding to the assignment x is $x^T C x + \sum_{j=1}^{n} w_j$, where $C = (c_{ij})$ is a symmetric matrix given by

$$w_j = \sum_{i=1}^{j} q_j$$

$$c_{ij} = \begin{cases} (p_i + q_i)/2, \text{ if } j > i \\ -[w_j + (n - j + 1)q_j - (p_j + q_j)], \text{ if } j = i, i = 1 \text{ to } n \\ (p_j + q_j)/2, \text{ if } j < i \end{cases}$$

So, the problem of finding an optimum assignment of jobs to ma-

chines A, B is equivalent to

$$\begin{aligned}&\text{minimize } x^T C x\\ &\text{subject to } x_i = 0 \text{ or } 1 \text{ for all } i\end{aligned}$$

Augment the matrix C into a symmetric matrix $C' = (C'_{ij})$ of order $(n+1) \times (n+1)$ by adding a dummy row (row 0) and dummy column (column 0) so that the sum of all entries in each row and in each column of C' is zero. Thus

$$C'_{oi} = C'_{io} = -[\sum_{j=1}^{i}((p_j - q_j)/2) + (n - i + 1)(p_i - q_i)/2],\ i = 1 \text{ to } n$$

$$C'_{oo} = -\sum_{i=1}^{n} C'_{oi},\ \text{and } C'_{ij} = C_{ij} \text{ for } i, i = 1 \text{ to } n.$$

Let $X = (x_0, x_1, \ldots, x_n)^T$. Show that the above optimum assignment problem is equivalent to

$$\begin{aligned}&\text{minimize} X^T C' X\\ &\text{subject to } x_i = 0 \text{ or } 1,\ i = 0, 1, \ldots, n.\end{aligned}$$

Formulate this as the problem of finding a maximum capacity cut in a network for which the set of nodes is $\mathcal{N} = \{0, 1, \ldots, n\}$. Using the special property that $C'_{ij} = r_i = (p_i + q_i)/2$ for all $j > i$, develop an efficient direct algorithm for solving this maximum capacity cut problem. Discuss how to solve the job assignment problem using this algorithm.

Solve the numerical problem with $n = 5$ and $p = (p_i) = (2, 6, 7, 9, 8), q = (q_i) = (4, 4, 11, 3, 14)$ using this approach (Lakshminarayanan, Lakshmanan, Papineau, and Rochette [1979]).

2.39 Let G $= (\mathcal{N}, \mathcal{A}, 0, k, \check{s}, \check{t})$ be a directed single commodity flow network, with the capacity vector $k \geq 0$. Define $y = (y_{ij} :\ (i,j) \in \mathcal{A})$ to be a vector of variables associated with the arcs in G. Consider the following problem

$$\text{minimize } \sum(k_{ij} y_{ij} :\ \text{over } (i,j) \in \mathcal{A})$$

$$\text{subject to } \sum(y_{ij} : \text{ over } (i,j) \in \mathcal{C}) \geqq 1, \text{for chains } \mathcal{C} \text{ from } \breve{s} \text{ to } \breve{t} \text{ in G}$$

$$y_{ij} = 0 \text{ or } 1 \text{ for all } (i,j) \in \mathcal{A}.$$

Show that this is the arc-chain formulation of the problem of finding a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G. Show that the LP relaxation obtained by replacing the 0-1 constraints on y by $y \geqq 0$ has an optimum solution which satisfies $y_{ij} = 0$ or 1 for all (i,j). Give an interpretation for the dual of the LP relaxation discussed above, as an alternative formulation of the maximum value flow problem in G.

2.40 Let G $= (\mathcal{N}, \mathcal{A}, 0, k, \breve{s}, \breve{t})$ be a directed connected single commodity flow network. Suppose we are given a feasible flow vector of maximum value from $\breve{s}$ to $\breve{t}$ in G. Then the labeling procedure discussed in the proof of Theorem 2.3 can be used to generate a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G with a computational effort of at most $0(|\mathcal{A}|)$. Conversely suppose we are given a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G. Is there a procedure that can use this information to generate a maximum value flow from in G efficiently? (Picard and Queyranne [1982b])

2.41 Maximum Weighted Closure of a Network Let G $= (\mathcal{N}, \mathcal{A})$ be a directed network with $w = (w_i : i \in \mathcal{N})$ as the vector of vertex weights that may be of any sign. A *closure* of G is any subset $\mathbf{U} \subset \mathcal{N}$ satisfying the property that $i \in \mathbf{U}$ and $(i,j) \in \mathcal{A}$ imply that $j \in \mathbf{U}$ also. A closure of G is also called a *hereditary subset*, or *initial subset* or a *selection*. The closure has application in the selection of contingent investments. Suppose we are given a set of projects (represented by nodes in a network) and a set of contingency relations among them. Project i is contingent to project j means that if we decide to select project i, then we must also select project j (this is represented by an arc (i,j) in the network). Every project is associated with a net profit (which may be negative for a project presumably useful to the selection of other more profitable projects). The problem of selecting the subset of projects to implement in order to maximize net profit then becomes that of finding a maximum weight closure in the network. Define the variables y_i for $i \in \mathcal{N}$ by $y_i = 1$ if i is included in the closure, or y_i

$= 0$ otherwise. Show that the problem of finding a maximum weight closure in G is equivalent to the 0-1 nonlinear program.

$$\begin{array}{rcl} \text{maximize } \sum(w_i y_i & : & \text{over } i \in \mathcal{N}) \\ \text{subject to } y_i(1-y_j) & = & 0, \text{ for each } (i,j) \in \mathcal{A} \\ y_i & = & 0 \text{ or } 1 \text{ for all } i \in \mathcal{N}. \end{array}$$

When λ is sufficiently large ($\lambda > 1 + \sum_{i \in \mathcal{N}} |w_i|$), show that this problem is equivalent to the 0-1 quadratic program (QP) given below. Augment G into a network G$'$ by the following procedure. Make the capacity of all the arcs in G equal to λ. Introduce a new source node $\breve{s}$ and a new sink node $\breve{t}$. For each $i \in \mathcal{N}$ if $w_i \geqq 0$ include on arc $(\breve{s}, i)$ with capacity w_i; if $w_i < 0$ include $(i, \breve{t})$ with capacity $-w_i$. The lower bounds on all the arcs in G$'$ is 0. Let $[\mathbf{X}, \overline{\mathbf{X}}]$ be a minimum capacity cut separating $\breve{s}$ and $\breve{t}$ in G$'$. Show that the incidence vector of $\mathbf{X} \backslash \{s\}$ is an optimum solution of the 0-1 QP below, and that $\mathbf{X} \backslash \{s\}$ is a maximum weight closure of G.

$$\begin{array}{rcl} \text{maximize } \sum\limits_{i \in \mathcal{N}} w_i y_i & - & \lambda \sum (y_i(1-y_j) : \text{ over } (i,j) \in \mathcal{A}) \\ y_i & = & 0 \text{ or } 1 \text{ for all } \ i \in N. \end{array}$$

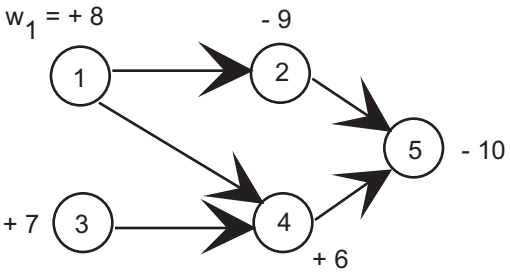


Figure 2.32:

Find a maximum weight closure in the network in Figure 2.32 with the minimum cut approach outlined above (Picard and Queyranne [1982b]).

2.42 Activity Selection Game **A** is a set of activities. Selection of activity i yields a profit w_i (of arbitrary sign), and selection of activity i without the selection of activity j leads to a penalty of $\lambda_{ij} \geqq 0$. It is required to find a subset of activities $\mathbf{U} \subset \mathbf{A}$ which yields the maximum net profit. Show that this problem is equivalent to finding a $y = (y_i : \ i \in \mathbf{A})$ that solves

$$\text{maximize } [\sum_{i \in \mathrm{A}} w_i y_i - \sum_{i,j \in \mathrm{A}, i \neq j} \lambda_{ij} y_i (1 - y_j)]$$

$$\text{subject to } y_i = 0 \text{ or } 1 \text{ for all } i \in \mathbf{A}.$$

Show how this problem can be transformed into a minimum capacity cut problem in a network, using the approach discussed in Exercises 2.29 and 2.41 (Picard and Queyranne [1982 b], Topkis [1980]).

2.43 Binary Posynomial Maximization Let $y = (y_1, \ldots, y_n)^T$ be a vector of binary variables and let $\mathbf{S}_1, \ldots, \mathbf{S}_r$ be r distinct nonempty subsets of $\{1, \ldots, n\}$. The function $P(y) = \sum_{t=1}^{r} c_t (\Pi_{i \in \mathbf{S}_t} y_i)$ is said to be a posynomial in y if $c_t > 0$ for all $t = 1$ to r. Maximizing $P(y)$ has the trivial solution $y = \mathbf{e}$, the vector of all 1's. However, the problem

$$\text{maximize } P(y) - by$$

$$\text{subject to } y_i = 0 \text{ or } 1 \text{ for all } \ i$$

where b is an arbitrary real $n-$vector, is nontrivial, it is called the binary posynomial maximization problem.

Generate a directed network $\mathrm{G} = (\mathcal{N}, \mathcal{A})$ with node weight vector w by the following procedure: Associate a node in $\mathcal{N}$ with the set $\mathbf{S}_t$ for each $t = 1$ to r, make its weight $w_t = c_t$ if $|\mathbf{S}_t| \geqq 2$, or $c_t - b_i$ if $\mathbf{S}$ is the singleton set $\{y_i\}$. For each $i = 1$ to n such that the singleton set $\{y_i\}$ is not among the sets $\mathbf{S}_1, \ldots, \mathbf{S}_r$, associate a node in $\mathcal{N}$ with its weight $-b_i$. Let $\mathcal{N}_1$, be the set of nodes introduced so far. For each $\mathbf{Q} \subset \{1, \ldots, n\}$ associated with which there is a node in $\mathcal{N}_1$, introduce a new node associated with the set $\mathbf{Q} \backslash \{j\}$ if it is not there already, and an arc directed from the node associated with $\mathbf{Q}$ to this node, for each $j \in \mathbf{Q}$. Make the weights of all the nodes for which the weight is not

defined above equal to zero. $\mathcal{N}$ is the set of all the nodes and $\mathcal{A}$ is the set of all the arcs defined above.

Show that the above problem is equivalent to the maximum weight closure problem (defined in Exercise 2.41) in G (Picard and Queyranne [1982 b]).

2.44 Let G = $(\mathcal{N}, \mathcal{A}, 0, k, \check{s}, \check{t})$ be a directed connected network with $k > 0$. Using the transformation of the minimum capacity cut problem into a binary quadratic programming problem discussed in Exercise 2.29 and the fact that the maximum capacity cut problem is NP-hard, show that cardinality constrained minimum cut problems (i.e., the two problems, finding a minimum capacity cut $[(\mathbf{X}, \bar{\mathbf{X}})]$ separating $\check{s}$ and $\check{t}$ in G with either the constraint that $|(\mathbf{X}, \bar{\mathbf{X}})| = r$ or the constraint $|\mathbf{X}| = r$, for specified r) are NP-hard problems (Picard and Queyranne [1982 b]).

2.45 A nursing staff scheduling problem There are three departments in a hospital, each of which operates on three shifts daily, with staff members working one shift per day. The daily requirements are spelled out in the table given below. It is required to determine the minimum number of nurses to staff these three departments subject to the constraints and bounds described above. Formulate this as a flow problem and find an optimum solution for it.

Shift	Minimum no. of nurses required in shift	Lower bound-upper bound for nurses in shift in department 1	2	3
1	26	6-8	11-12	7-12
2	24	4-6	11-12	7-12
3	19	2-4	10-12	5-7
minimum no. of nurses required in dept. over all these shifts put together		13	32	22

(Khan and Lewis [1987])

2.46 Another Version of a Maximum Cut Problem Let G = $(\mathcal{N}, \mathcal{A})$ be a directed network with $c = (c_{ij})$ as the vector of arc weights.

Define a cut in G to be a partition of $\mathcal{N}$ into $[\mathbf{X}, \overline{\mathbf{X}}]$, and its value to be $c(\mathbf{X}, \overline{\mathbf{X}}) - c(\overline{\mathbf{X}}, \mathbf{X},)$. Define a node $i \in \mathcal{N}$ to be an *overall source* if $c(i, \mathcal{N}) > c(\mathcal{N}, i)$, *overall sink* if $c(\mathcal{N}, i,) > c(i, \mathcal{N})$, and *neutral* if it is neither an overall source nor an overall sink. Under these definitions, prove that a cut $[\mathbf{X}, \overline{\mathbf{X}}]$ is a maximum value cut in G if every overall source is in the set $\mathbf{X}$ and every overall sink is in the set $\overline{\mathbf{X}}$. From this result construct an efficient algorithm to find a maximum value cut in G by the definition given here (Farley and Proskurowski [1982]).

2.47 A Maximum Value Cut Problem in a Tree Let $\mathbb{T}$ be a tree in which every line is an arc. Let $c = (c_{ij})$ be the vector of arc weights in $\mathbb{T}$. Define a cut in $\mathbb{T}$ to be a partition of its nodes into $(\mathbf{X}, \overline{\mathbf{X}})$, and its value to be $c(\mathbf{X}, \overline{\mathbf{X}})$. Develop a linear-time algorithm to find a maximum value cut in $\mathbb{T}$ under this definition (Farley and Proskurowski [1982]).

2.48 G $= (\mathcal{N}, \mathcal{A}, 0, k)$ is a connected directed capacitated network. For a pair of distinct nodes i, j in $\mathcal{N}$, let $v_{i,j}$ denote the capacity of minimum capacity cut separating i and j. For $i_1, \ldots, i_r \in \mathcal{N}$, prove that $v_{i_1,i_r} \geqq \min. \{v_{i_t,i_{t+1}} : \ t = 1, \ldots, r-1\}$. Prove that the set of distinct values of $v_{ij}, i \neq j \in \mathcal{N}$ is at most $|\mathcal{N}| - 1$.

2.49 G $= (\mathcal{N}, \mathcal{A}, 0, k, \check{s}, \check{t})$ is a directed single commodity flow network. $\mathbf{E}= \{(i_1, j_1), \ldots, (i_r, j_r)\} \subset \mathcal{A}$. It is required to check whether $\mathbf{E}$ is a subset of a set of forward arcs of a cut separating $\check{s}$ and $\check{t}$ in G, and if so to find such a cut of a minimum capacity. Develop necessary conditions, and an efficient algorithm for this problem based on a maximum value flow formulation.

2.50 Preemptive Scheduling of Jobs with Due Dates $T_1, \ldots, T_n$ are n tasks to be processed on a single machine. Task T_i is decomposed into two subtasks denoted by M_i (mandatory subtask) and O_i (optional subtask). The processing times of M_i, O_i are m_i, σ_i respectively. The time at which M_i becomes ready for execution is r_i, and O_i becomes ready for execution when M_i is completed. The deadline for T_i (the time at which T_i must be completed) is d_i. For each i, M_i must be executed to completion, but preemption is allowed. O_i can be executed

any time within the interval between its ready time and the deadline. It is terminated at the deadline d_i even if it is not completed.

A schedule is an assignment of the tasks $M_i, O_i, i = 1$ to n to the machine in disjoint intervals of time. There are precedence constraints specified for processing the tasks. This defines a precedence successor relationship among them. If T_j is a successor of T_i, then execution of M_j cannot begin until M_i is completed.

A schedule is said to be a valid schedule if each M_i is executed to completion in the time interval $[r_i, \infty]$ and the precedence constraints between all tasks are obeyed. A valid schedule is said to be feasible if each M_i is completed in the interval $[r_i, d_i]$. Let p_i denote the machine time allotted to O_i in a feasible schedule (since the processing of O_i is terminated at d_i, p_i may be $\leq \sigma_i$). If $p_i = \sigma_i$, the task T_i is said to be precisely scheduled in this schedule. Otherwise, if $p_i < \sigma_i$, the error of the task T_i in this schedule is defined to be $\varepsilon_i = \sigma_i - p_i$ (this portion of O_i is essentially discarded if this schedule is implemented). The total error in the schedule is $\sum_{i=1}^{n} \varepsilon_i$. A feasible schedule is said to be a precise schedule if all the ε_i are zero in it.

Let A_i denote the set of all successors (i.e., descendents) of T_i in the precedence relationship. Define $\bar{d}_i = \min\{d_i, \min\{d_j : \ j \in A_i\}\}$. Working with the modified deadlines $\bar{d}_i$ instead of d_i allows the precedence constraints to be ignored temporarily (from an invalid schedule in which portions of T_i are scheduled after some portion of T_j for $T_j \in A_i$, a valid schedule can be constructed by appropriate exchange of the time segments).

Let $a_1, \ldots, a_g$ be the strictly increasing sequence of all the distinct entries in the set $\{r_1, \ldots, r_n, \bar{d}_1, \ldots, \bar{d}_n\}$, so $g \leq 2n$. This sequence divides time into $g - 1$ intervals $[a_h, a_{h+1}]$ for $h = 1$ to $g - 1$. The length of the hth interval is $t_h = a_{h+1} - a_h$.

Define a directed network $G(\delta) = (\mathcal{N}, \mathcal{A})$ by the following. $\mathcal{N}$ contains a source node $\check{s}$, a sink node $\check{t}$, two nodes called T_i and T_i^1 for each task T_i, nodes called M_i, O_i for each i, a node called $[a_h, a_{h+1}]$ representing the hth interval defined above for each h, and another vertex called I. $\mathcal{A}$ contains the following arcs: $(\check{s}, T_i)$ for each $i = 1$ to n with capacity $\tau_i = m_i + \sigma_i$; (T_i, M_i) with capacity m_i and (T_i, O_i) with capacity σ_i for $i = 1$ to n; (M_i, T_i^1) with capacity m_i, and (O_i, T_i^1)

with capacity σ_i, for $i = 1$ to n; $(T_i^1, [a_h, a_{h+1}])$ with capacity t_h for each h such that $r_i \leq a_h$ and $\bar{d}_i \geq a_{h+1}$ (this implies that the task T_i can be scheduled in this interval); $([a_h, a_{h+1}], \breve{t})$ with capacity t_h for each h; (O_i, I) with capacity σ_i for each $i = 1$ to n; and $(I, \breve{t})$ with capacity δ. All lower bounds for arch flows in $\mathrm{G}(\delta)$ are 0.

Define $\mathrm{G}^1(\delta) = (\mathcal{N}, \mathcal{A}^1)$ to be the network obtained from $\mathrm{G}(\delta)$ by deleting all the arcs (T_i, O_i) and (O_i, T_i^1) for $i = 1$ to n, from it.

(i) Show that a feasible schedule exists if $v =$ maximum flow value from $\breve{s}$ to $\breve{t}$ in $\mathrm{G}^1(0)$, is $\sum_{i=1}^n m_i$. If $v < \sum_{i=1}^n m_i$, no feasible schedule exists. Given a feasible flow vector in $G^1(0)$ of value $\sum_{i=1}^n m_i$, discuss a procedure for generating a feasible schedule from it.

(ii) Assume that a feasible schedule exists. Show that a precise schedule exists if $F =$ the maximum flow value from $\breve{s}$ to $\breve{t}$ in $\mathrm{G}(0)$ is $\sum_{i=1}^n (m_i + \sigma_i) = u$. Given a feasible flow vector in $\mathrm{G}(0)$ of value u, discuss a procedure for generating a precise schedule from it.

(iii) Suppose a feasible schedule exists but not a precise schedule. Let $\delta = u - F$. Show that a feasible schedule with minimum total error can be constructed from a maximum value flow in $\mathrm{G}(\delta)$ and that its total error is δ.

(iv) Find a feasible schedule with minimum total error in the problem with the following data, using this approach. $n = 5$. T_1 precedes both T_2 and T_3, T_2 precedes T_4, and T_3 precedes T_5.

	r_i	d_i	m_i	O_i
$i = 1$	0.0	0.6	0.2	0.2
2	0.2	0.7	0.1	0.3
3	0.4	1.0	0.2	0.3
4	1.2	1.5	0.1	0.2
5	0.6	2.0	0.5	0.3

(v) Suppose we are given a weight w_i which measures the relative importance of the task $T_i, i = 1$ to n. Develop a formulation for

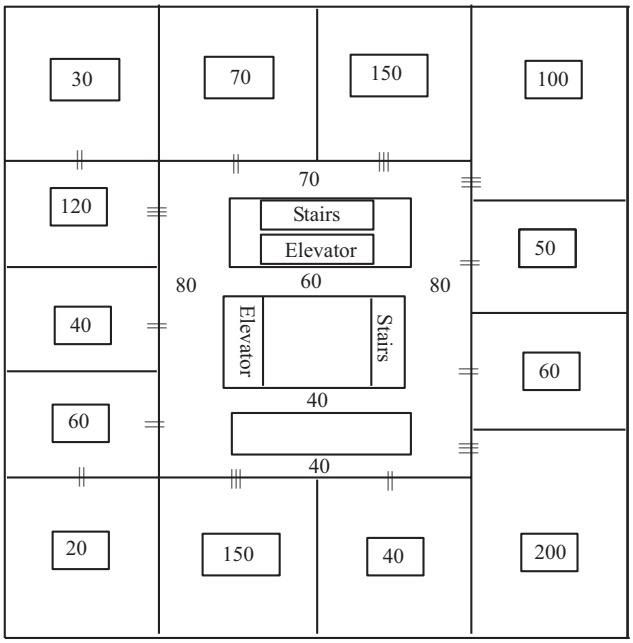


Figure 2.33: Floor plan of third (top) Floor.

the problem of finding a feasible schedule that minimizes the total weighted error, $\sum_{i=1}^{n} w_i \varepsilon_i$, as a minimum cost flow problem.

(Shih, Liu, Chung and Gillies [1989])

2.51 A large building, or building complex, occupied by hundreds of people at the same time, may contain several floors, have several corridors in each floor, and have a combination of elevators and stairways connecting the floors. Hotels, hospitals, schools and universities, libraries, large office complexes, malls, entertainment facilities, etc., are examples of such buildings. In an emergency situation, such a building has to be evacuated in a short time period. In evaluating building designs, an important characteristic is the number of people that can be moved out of the building per unit time using all available exits. This depends on the capacities of the corridors, doors, stairways, and ele-

vators of the building; it also depends on the distribution, or expected concentration of people in various areas of the building.

In a network model of emergency evacuation, the corridors, a point on a line where two corridors intersect, the points where a corridor leads to a stairway or an elevator, or the rooms, halls, or the labs where people collect can all be represented as nodes. In Figure 2.33, the floor plan of the third floor of a three-floor building is shown, with all the relevant data.

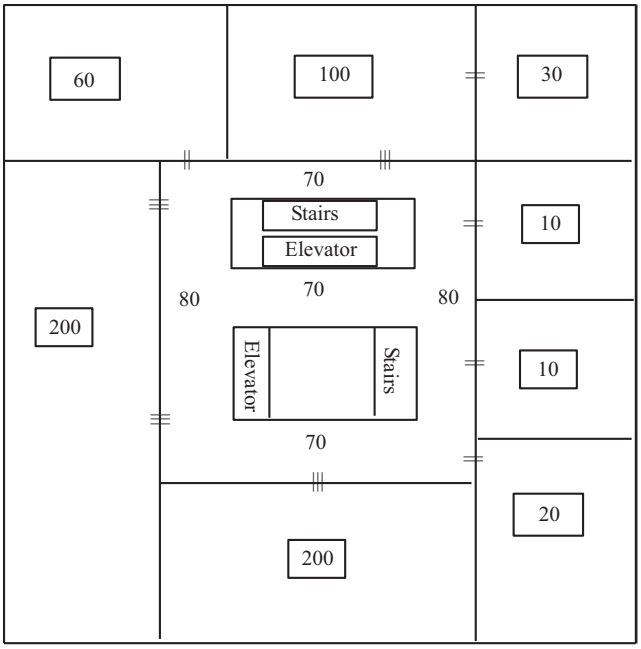


Figure 2.34: Floor plan of second Floor.

Each room contains a little box with the number of people expected in that room entered inside it. Doors are marked with two or three lines (|| or |||) with capacities of evacuating 50 and 75 persons/minute respectively. Each stairway has a capacity of evacuating 100 persons/minute, while each elevator has a capacity of evacuating 12 person/minute. The capacities of corridors (persons/minute) are entered along the corridor. The floor plans of the second and first (ground) floors are given in Figures 2.34 and 2.35 respectively.

The entrance door marked with four lines (||||) has a capacity of evacuating 500 persons/minute. Formulate the problem of determining the maximum number of persons that can be evacuated from this building per minute in case of an emergency as a network flow problem, clearly showing the network on which the problem is posed.

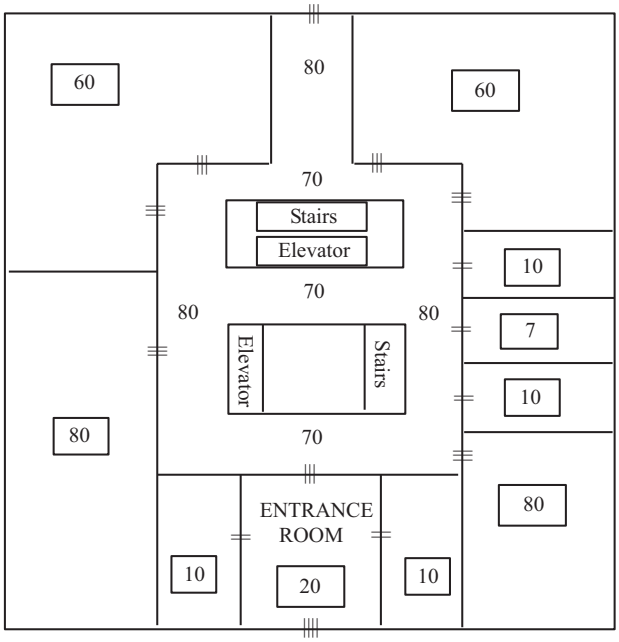


Figure 2.35: Floor plan of First (ground) Floor.

2.52 There are $p = nm$ students to be formed into n groups of m students each. Each group will work on a separate project. For every pair of distinct students i, j we are given $u_{ij} > 0$, the *utility* of assigning both of them to the same group, whichever group it may be. The $p \times p$ matrix $u = (u_{ij})$ is given. It is required to form the groups so that the total utility is as large as possible. Give a formulation for this problem and check whether it is NP-hard.

2.53 Hall's Theorem $\mathbf{C} = \{\mathbf{S}_1, \ldots, \mathbf{S}_r\}$ is a class of nonempty subsets of $\mathbf{\Gamma} = \{1, \ldots, n\}$. A subset Δ of r distinct elements of $\mathbf{\Gamma}, \Delta =$

$\{i_1, \ldots, i_r\}$ is said to be an SDR (*system of distinct representatives*) for the class $\mathbf{C}$ if $i_h \in \mathbf{S}_h$ for each $h = 1$ to r, in this case i_h is said to represent $\mathbf{S}_h$ for $h = 1$ to r. As an example, for the class $\{\mathbf{S}_1 = \{1, 2, 3, 4\}, \mathbf{S}_2 = \{3, 4, 5, 6\}\}$, the set $\{1, 4\}$ is an SDR as 1 can represent $\mathbf{S}_1$ and 4 can represent $\mathbf{S}_2$ in it. But the set $\{3\}$ is not an SDR for this class even though 3 is in both $\mathbf{S}_1$ and $\mathbf{S}_2$.

Formulate the problem of finding an SDR for $\mathbf{C}$ as a flow problem. Draw the networks, and find the SDRs for the classes $\{\mathbf{S}_1 = \{2, 4, 5\}, \mathbf{S}_2 = \{1, 5\}, \mathbf{S}_3 = \{3, 4\}, \mathbf{S}_4 = \{3, 4\}\}$, $\{\mathbf{S}_1 = \{1, 2\}, \mathbf{S}_2 = \{2\}, \mathbf{S}_3 = \{2, 3, 4, 5\}, \mathbf{S}_4 = \{1, 2\}\}$ respectively, by solving the corresponding flow problems. Prove that an SDR exists for the class $\mathbf{C}$ iff every union of u sets of $\mathbf{C}$ contains at least u distinct elements, for $u = 1,2, \ldots, r$. (Ford and Fulkerson [1962 of Chapter 1])

2.54 $G = (\mathcal{N}, \mathcal{A}, \ell, k)$ is a directed connected single commodity flow network and (p, q) is a selected arc in G. It is required to find a feasible circulation $f = (f_{ij})$ in G which (i) maximizes f_{pq}, (ii) minimizes f_{pq}. Discuss methods for solving these problems. Apply your methods to solve these problems in the network in Figure 2.23 with $(p, q) = (4, 1)$.

2.55 Let $f^1 = (f^1_{ij}), f^2 = (f^2_{ij})$ be two feasible flow vectors in $G = (\mathcal{N}, \mathcal{A}, \ell, k, \breve{s}, \breve{t})$ of values v_1, v_2 respectively. Generate a flow vector h in the residual network $G(f^1)$ by the following rules: (a) For each $(i, j) \in \mathcal{A}$ satisfying $f^1_{ij} > f^2_{ij}$, we must have $\ell_{ij} \leqq f^2_{ij} < f^1_{ij}$, so the arc (j, i) exists in $G(f^1)$ with a $-$ label, make $h_{ji} = f^1_{ij} - f^2_{ij}$. (b) For each $(i, j) \in \mathcal{A}$ satisfying $f^1_{ij} < f^2_{ij}$, we must have $f^1_{ij} < f^2_{ij} \leqq k_{ij}$, so the arc (i, j) exists in $G(f^1)$ with a $+$ label, make $h_{ij} = f^2_{ij} - f^1_{ij}$. (c) On all the arcs in $G(f^1)$ on which a flow amount is not defined by (a), (b) above, make the flow in the vector h to be zero. Then show that h is a feasible flow vector in the residual network $G(f^1)$ with flow value from $\breve{s}$ to $\breve{t}$ of $v_2 - v_1$.

Similarly, if $f^1 = (f^1_{ij})$ is a feasible flow vector in G of value v_1, and $h = (h_{pq})$ a feasible flow vector in the residual network $G(f^1)$ of value ω from $\breve{s}$ to $\breve{t}$, define a flow vector $\hat{f} = (\hat{f}_{ij})$ in G, where $\hat{f}_{ij}$ is given by

$$\hat{f}_{ij} = \begin{cases} f^1_{ij} + h_{ij} & \text{if there is a + labeled arc } (i,j) \text{ in } G(f^1), \text{ corresponding to } (i,j) \in \mathcal{A}, \text{ with } h_{ij} > 0. \\ f^1_{ij} - h_{ji} & \text{if is a - labeled arc } (j,i) \text{ in } G(f^1), \text{ corresponding to } (i,j) \in \mathcal{A}, \text{ with } h_{ji} > 0. \\ f^1_{ij} & \text{otherwise} \end{cases}$$

Then show that $\hat{f}$ is a feasible flow vector in G and its value is $v_1 + \omega$.

Use these results to provide an alternate proof of Theorem 2.12

2.56 The Minimum Value Flow Problem Consider the single commodity flow network $G = (\mathcal{N}, \mathcal{A}, \ell, k, \check{s}, \check{t})$ with $0 \leqq \ell \leqq k$, in which some or all of the capacities k_{ij} may be infinite.

(i) Assuming that $k = \infty$ and $\ell > 0$, develop an efficient algorithm for finding a feasible flow vector in G, if one exists; by specializing the methods of Section 2.6 as applied to G. In this case show that the maximum value flow from $\check{s}$ to $\check{t}$ in G is infinite.

(ii) When, $\ell \neq 0$, $f = 0$ is not a feasible flow vector in G. So, in this case, the problem of finding a *minimum value flow* (i.e., one which has the least possible value among all feasible flow vectors in G) is of interest. Given a feasible flow vector $\bar{f} = (\bar{f}_{ij})$ of value $\bar{v}$ in G, a path $\mathcal{P}$ from $\check{s}$ to $\check{t}$ in G satisfying

$$\begin{array}{rl} (i,j) \text{ is a forward arc on} \mathcal{P} & \text{implies } \bar{f}_{ij} > \ell_{ij} \\ (i,j) \text{ is a reverse arc on} \mathcal{P} & \text{implies } \bar{f}_{ij} < k_{ij} \end{array}$$

is known as a *flow reduction path* (FRdP) from $\check{s}$ to $\check{t}$ wrt $\bar{f}$. Given a FRdP $\mathcal{P}$ from $\check{s}$ to $\check{t}$ wrt $\bar{f}$, the flow reduction step using it generates the new flow vector $\hat{f} = (\hat{f}_{ij})$ where $\epsilon = \text{min. } \{\bar{f}_{ij} - \ell_{ij}$:

over (i,j) forward on $\mathcal{P}\} \cup \{k_{ij} - \bar{f}_{ij}$: over (i,j) reverse on $\mathcal{P}\}$, and

$$\hat{f}_{ij} = \begin{cases} \bar{f}_{ij} - \epsilon & \text{if } (i,j) \text{ is forward on } \mathcal{P} \\ \bar{f}_{ij} + \epsilon & \text{if } (i,j) \text{ is reverse on } \mathcal{P} \\ \bar{f}_{ij} & \text{otherwise} \end{cases}$$

Show that the new flow vector $\hat{f}$ is a feasible flow vector in G of value $\hat{v} = \bar{v} - \epsilon$.

Prove that a feasible flow vector f in G is a minimum value feasible flow vector iff there exists no FRdP from $\breve{s}$ to $\breve{t}$ wrt it. Using this result, develop algorithms analogous to those discussed in this chapter, to find a minimum value feasible flow vector in G. Apply your algorithm to find minimum value feasible flow vectors in the networks in Figures 2.22, 2.25, 2.26, and 2.27.

(iii) Prove that if a feasible flow vector exists in G, then the minimum value in G is equal to the maximum of $\ell(\mathbf{X}, \overline{\mathbf{X}}) - k(\overline{\mathbf{X}}, \mathbf{X})$ over all $\mathbf{X} \subset \mathcal{N}$, $\overline{\mathbf{X}} = \mathcal{N} \backslash \mathbf{X}$, such that $\breve{s} \in \mathbf{X}$, $\breve{t} \in \overline{\mathbf{X}}$. This result is the analogue of Theorem 2.4 to the minimum value flow problem.

Comment 2.1 Ford and Fulkerson [1962 of Chapter 1] state that the problem of maximizing flow from one point to another in a capacity-constrained network was posed to them in the spring of 1955 by T. E. Harris and General F.S. Ross. Shortly afterwards the maximum flow minimum cut theorem was established by Ford and Fulkerson in their 1956 paper listed in this chapter's references. The relationship of this theorem to the duality theorem of linear programming was recognized almost at the same time. The scanning version of the labeling algorithm was devised by Ford and Fulkerson in 1957. Johnson [1966 of Chapter 5] suggested a supplemental rule to guarantee finite termination of this algorithm for arbitrary data. Edmonds and Karp [1972] demonstrated that if the node to scan from the list is selected by the FIFO rule, then the worst case computational complexity of this algorithm is of order $O(nm^2)$ where n, m are the number of nodes, arcs in the network. This is the shortest augmenting path implementation.

Zadeh [1972] constructed examples to show that this bound is tight in dense networks.

Multipath methods using all available shortest augmenting paths in each iteration were introduced by Dinic [1970] and refined by Malhotra, Kumar and Maheswari [1978], and others. Methods based on preflows were introduced by Karzanov [1974]. Goldberg and Tarjan [1986, 1988] developed simple and elegant preflow-push algorithms based on push/relabel steps. These methods have several advantages.

These are the algorithms that have given good computational performance. However, the literature on algorithms for the maximum value flow problem is vast. Cherkasky [1977] and Galil [1980] made improvements on Karzanov algorithm. Galil and Naamad [1980] and Sleater and Tarjan [1983] discuss improvements in Dinic's algorithm using new data structures. Gabow [1985] discusses an approach based on scaling. Cheriyan and Maheswari [1987] and Ahuja, Orlin, and Tarjan [1987] discuss variants of the Goldberg-Tarjan algorithm. Goldfarb and Hao [1990] developed an $O(mn^2)$ complexity primal simplex variant for this problem. Ramachandran [1987] has shown that in a theoretical worst-case sense, the minimum capacity cut and the maximum value flow problems are not easier in an acyclic network than in a network that is not acyclic.

So far, no one has succeeded in developing an algorithm with the worst case computational complexity of $O(nm)$ for the maximum value flow problem in sparse networks. So, the search goes on.

For results of computational studies comparing the various algorithms, see Cheung [1980], Glover, Klingman, Mote, and Whitman [1979], Imai [1983], and Hamacher [1979]. But these studies were carried out before the development of preflow-push algorithms. Currently the computational performance of preflow-push algorithms is being evaluated by several groups.

Accounts of the contributions of D. R. Fulkerson can be found in Billera and Lucas [1978] and Hoffman [1978].

A variety of applications of the minimum capacity cut problem are discussed in Picard and Ratliff [1975] and Picard and Queyranne [1982a, 1982b]. Several other applications of the maximum value flow, minimum capacity cut problems are discussed in many papers in the

following references, notable among those are included among the exercises given above.

Necessary and sufficient conditions for the feasibility of flow and circulation problems are due to Gale [1957] and Hoffman [1960].

2.9 References

R. K. AHUJA and J. B. ORLIN, Sept.-Oct. 1989, "A Fast and Simple Algorithm for the Maximum Flow Problem," *OR*, 37, no. 5 (748-759).

R. K. AHUJA, J. B. ORLIN, and R. E. TARJAN, 1987, "Improved Time Bounds for the Maximum Flow Problem," Sloan School of Management, MIT, Cambridge, MA.

F. BARAHONA, 1982, "On the Computational Complexity of the Ising Spin Models," *Journal of Physics A: Mathematics and General*, 15(3241-3250).

L. J. BILLERA and W. F. LUCAS, 1978, "Delbert Ray Fulkerson: August 14, 1924 - January 10, 1976," *MPS*, 8(1-16).

J. R. BROWN, Mar.-Apr. 1979, "The Sharing Problem," *OR*, 27, no. 2(324-340).

V. CABOT, R. L. FRANCIS, and M. A. STARY, 1970, "A Network Flow Solution to a Rectilinear Distance Facility Location Problem," *AIIE Transactions*, 2(132-141).

J. CHERIYAN and S. N. MAHESWARI, 1989, "Analysis of Preflow Push Algorithms for Maximum Network Flow," *SIAM J. on Computing*, 18(1057-1086).

R. V. CHERKASKY, 1977, "Algorithms of Construction of Maximal Flow in Networks with Complexity $O(n^2\sqrt{m})$ Operations," *Mathematical Methods of Solution of Economical Problems*, 7(112-125).

T. CHEUNG, 1980, "Computational Comparison of Eight Methods for the Maximum Network Flow Problem," *ACM Transactions on Mathematical Software*, 6(1-16).

E. A. DINIC, 1970, "Algorithms for Solution of a Problem of Maximum Flow in Networks with Power Estimation," *Soviet Mathematics Doklady*, 11(1277-1280).

I. B. DUNCAN, Nov. 1979, "The Allocation of Specialities to Hospitals in a Health District," *JORS*, 30, no. 11(953-961).

J. EDMONDS and R. M. KARP, 1972, "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems," *JACM*, 19(248-264).

P. ELIAS, A. FEINSTEIN, and C. E. SHANNON, 1956, "Note on Maximum Flow Through a Network," *IRE Transactions on Information Theory, IT-2*(117-119).

A. M. FARLEY and A. PROSKUROWSKI, Dec. 1982, "Directed Maximal-Cut Problems," *IPL*, 15, no. 5(238-241).

L. R. FORD Jr. and D. R. FULKERSON, 1956, "Maximum Flow Through a Network," *Canadian Journal of Mathematics*, 8(399-404).

H. N. GABOW, 1985, "Scaling Algorithms for Network Problems," *Journal of Computer and System Sciences*, 31(148-168).

D. GALE, 1957, "A Theorem on Flows in Networks," *Pacific Journal of Mathematics*, 7(1073-1082).

Z. GALIL, 1980, "An $O(n^{\mathbf{5}/3}m^{\mathbf{2}/3})$ Algorithm for the Maximal Flow Problem," *Acta Informatica*, 14(221-242).

Z. GALIL and A. NAAMAD, 1980, "An $O(nmlog^{\mathbf{2}}n)$ Algorithm for the Maximal Flow Problem," *Journal of Computer and System Sciences*, 21(203-217).

G. GALLO, M. D. GRIGORIADIS, and R. E. TARJAN, Feb. 1989, "A Fast Parametric Maximum Flow Algorithm," *SIAM J. on Computing*, 18, no. 1(30-55).

F. GLOVER, D. KLINGMAN, J. MOTE and D. WHITMAN, 1979, "Comprehensive Computer Evaluation and Enhancement of Maximum Flow Algorithms," Graduate School of Business Administration, University of Colorado, Boulder, CO, extended abstract in *DAM*, 2 (1980)(251-254).

A. V. GOLDBERG and R. E. TARJAN, 1986, "A New Approach to the Maximum Flow Problem," *Proceedings of the 18th Symposium on the Theory of Computing*,(136-146).

A. V. GOLDBERG and R. E. TARJAN, Oct. 1988, "A New Approach to the Maximum Flow Problem," *JACM*, 35, no. 4(921-940).

D. GOLDFARB and J. HAO, Aug. 1990, "A Primal Simplex Algorithm that Solves the Maximum Flow Problem in at Most nm Pivots and $O(n^{\mathbf{2}}m)$ Time," *MP*, 47, no. 3(353-365).

R. E. GOMORY and T. C. HU, Dec. 1961, "Multi-terminal Network Flows," *Journal of SIAM*, 9, no. 4(551-570).

D. GUSFIELD, C. MARTEL and D. FERNANDEZ-BACA, 1987, "Fast Algorithms for Bipartite Network Flow," *SIAM J. on Computing*, 16(237-251).

H. HAMACHER, 1979, "Numerical Investigations on the Maximal Flow Algorithm of Karzanov," *Computing*, 22(17-29).

H. HAMACHER, 1982, "Determining Minimal Cuts with a Minimal Number of Arcs," *Networks*, 12, no.4(493-504).

A. J. HOFFMAN, 1960,"Some recent applications of the Theory of Linear Inequalities to Extremal Combinatorial Analysis," *Proceedings of the Symposia on Applied*

Math., 10(113-128).

A. J. HOFFMAN, 1978, "Ray Fulkerson's Contributions to Polyhedral Combinatorics," *MPS*, 8(17-23).

H. IMAI, 1983, "On the Practical Efficiency of Various Flow Algorithms," *Journal of the OR Society of Japan*, 26(61-82).

A. V. KARZANOV, 1974, "Determining the Maximal Flow in a Network by the Method of Preflows," *Soviet Mathematics Doklady*, 15(434-437).

M. R. KHAN and D. A. LEWIS, 1987, "A Network Model forNursing Staff Scheduling," *Zeittschrift fur Operations Research*, 31, no. 6(B161- B171).

S. LAKSHMINARAYANAN, R. LAKSHMANAN, R. L. PAPINEAU and R. ROCHETTE, Aug. 1979,"Order Preserving Allocation of Jobs to Two Non-identical Parallel Machines: A Solvable Case of the Maximum Cut Problem," *INFOR*, 17, no. 3(230-241).

V. M. MALHOTRA, M. P. KUMAR and S. N. MAHESWARI, 1978, "An O($n^{\mathbf{3}}$) Algorithm for Finding Maximum Flows in Networks," *IPL*, 7(277-278).

N. MEGIDDO and Z. GALIL, 1979, "On Fulkerson's Conjecture About Consistent Labeling Processes," *MOR*, 4(265-267).

J. D. MURCHLAND, 1968, "Rhy's Combinatorial Station Selection Problem," London Graduate School of Business.

J. C. PICARD and H. D. RATLIFF, 1975, "Minimum Cuts and Related Problems," *Networks*, 5(357-370).

J. C. PICARD and H. D. RATLIFF, May-June 1978, "A Cut Approach to the Rectilinear Distance Facility Location Problem," *OR*, 26, no. 3(422-433).

J. C. PICARD and M. QUEYRANNE, 1982a, "A Network Flow Solution to Some Nonlinear 0-1 Programming Problems with Applications to Graph Theory," *Networks*, 12(141-159).

J. C. PICARD and M. QUEYRANNE, Nov. 1982b, "Selected Applications of Maximum Flows and Minimum Cuts in Networks," *INFOR*, 20, no. 4(394-422).

M. QUEYRANNE, 1980, "Theoretical Efficiency of the Algorithm ' Capacity' for the Maximum Flow Problem," *MOR*, 5(258-266).

V. RAMACHANDRAN, 1987, "The Complexity of Minimum Cut and Maximum Flow Problems in an Acyclic Network," *Networks*, 17, no. 4(387-392).

H. D. RATLIFF, G. T. SICILIA and S. H. LUBORE, Jan. 1975, "Finding the n Most Vital Links in Flow Networks," *MS*, 21, no. 5(531-539).

W. K. SHIH, J. W. S. LIU, J. Y. CHUNG and D. W. GILLIES, July 1989, "Scheduling Tasks with Ready Times and Deadlines to Minimize Average Error," *Operating*

Systems Review, 23, no. 3(14-28).

D. SLEATOR and R. E. TARJAN, 1983, "A Data Structure for Dynamic Trees," *Journal of Computer and System Sciences*, 24(362-391).

D. SLEATOR and R. E. TARJAN, 1985, "Self-adjusting Binary Search Trees," *JACM*, 32(652-686).

R. E. TARJAN, 1984, "A Simple Version of Karzanov's Blocking Flow Algorithm," *OR Letters*, 2(265-268).

D. M. TOPKIS, 1980, "Activity Selection Games and the Minimum Cut Problem," Technical report, Bell Labs. Holmdel, NJ.

A. TUCKER, May 1977, "A Note on Convergence of the Ford-Fulkerson Flow Algorithm," *MOR*, 2(143-144).

G. R. R. WAISSI, 1985, "Acyclic Network Generation and Maximal Flow Algorithms for Single Commodity Flow," Ph. D. dissertation, Dept. of Civil Engineering, University of Michigan, Ann Arbor, Mich.

R. D. WOLLMER, "Removing Arcs From A Network," *OR*, 12, no. 6(934-940).

T. YI and K G MURTY, 1991, "Finding Maximum Flows in Networks with Nonzero Lower Bounds Using Preflow Methods," Tech. report, IOE Dept., University of Michigan, Ann Arbor, Mich.

N. ZADEH, 1972, "Theoretical Efficiency of Edmonds-Karp Algorithm For Computing Maximal Flows," *JACM*, 19(184-192).

Index

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