Applications of Network Optimization

July 1992 WP #3458

Ravindra K. Ahuja*
Thomas L. Magnanti*
James B. Orlin*
M.R. Reddy*

* see page bottom for complete address

Ravindra K. Ahuja
Department of Industrial & Management Engineering
Indian Institute of Technology
Kanpur - 208 016, INDIA

Thomas L. Magnanti
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 02139

James B. Orlin
Sloan School of Management
Massachusetts Institute of Technology
Cambridge, MA 01239

M.R. Reddy
Department of Industrial & Management Engineering
Indian Institute of Technology
Kanpur - 208 016, INDIA

APPLICATIONS OF NETWORK OPTIMIZATION

Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin, and M. R. Reddy

ABSTRACT

Network optimization has always been a core problem domain in operations research, as well as in computer science, applied mathematics, and many fields of engineering and management. Network optimization problems arise in a variety of situations, and often in situations that apparently are quite unrelated to networks. These applications are scattered throughout the literature and until recently no single paper, book, or any other reference, summarized these applications. Consequently, the research and practitioner community has not fully appreciated the richness of these applications. This paper attempts to partially satisfy this important need by presenting a collection of applications of the following fundamental network optimization problems: the shortest path problem, the maximum flow problem, the minimum cost flow problem, assignment and matching problems, and the minimum spanning tree problem. We describe 25 applications of these problems and provide references for more than 100 additional applications. This paper is intended to provide an appreciation for the pervasiveness of network optimization problems. We hope that this paper will stimulate researchers and practitioners to model more decisions problems within the framework of network optimization.

1. INTRODUCTION

Everywhere we look in our daily lives, networks are apparent. Electrical and power networks bring lighting and entertainment into our homes. Telephone networks permit us to communicate with each other almost effortlessly within our local communities and across regional and international borders. National highway systems, rail networks, and airline service networks provide us with the means to cross great geographical distances to accomplish our work, to see our loved ones, and to visit new places and enjoy new experiences. Manufacturing and distribution networks give us access to life's essential foodstock and to consumer products. And computer networks, such as airline reservation systems, have changed the way we share information and conduct our business and personal lives. In all of these problem domains, and in many more, we wish to send some flow (electricity, a consumer product, a person or a vehicle, a message) from one point to another in an underlying physical network, and to do so as efficiently as possible, both to provide good service to the users of the network and to use the underlying (and typically expensive) transmission facilities effectively. These problems constitute the domain of network optimization problems.

Network optimization problems in many other settings, do not correspond to any direct physical systems. For example, sometimes the nodes and arcs have a temporal dimension that models activities that take place over time. Many scheduling applications have this flavor. Network optimization problems also arise in surprising ways for problems that on the surface might not appear to involve networks at all. In any event, networks model a variety of problems in project, machine and crew scheduling; location and layout; warehousing and distribution; production planning and control; and social, medical, and defense contexts; and in the physical and life sciences. The operations research, applied mathematics, computer science, and engineering communitites have made many successes on this front, and a wide variety of practical problems can be formulated and solved as network optimization problems. These applications, however, are scattered throughout the literature, and until recently no single paper, book or any other reference summarized these applications. Consequently, the research and practitioner communities lacked awareness of these applications. This paper helps to fill this need by describing 25 applications and providing references for over 100 additional applications of the following fundamental network optimization problems: (1) the shortest path problem; (2) the maximum flow problem; (3) the minimum cost flow problem; (4) assignment and matching problems; and (5) the minimum spanning tree problem. We have selected these problem categories on account of their importance to the operations research community. The applications for these problems can be partitioned into two category: (i) optimization problems that are transformable into one of the network optimization problems cited in this list; and (ii) optimization problems that contain embedded network structure and are solvable by solving a sequence of network optimization problems. We have limited the scope of our discussion in this paper to applications in the first category. The 25 applications described in this paper arise in the following domains:

1. Approximating piecewise linear functions.
2. DNA sequence alignment.
3. Production planning problems.
4. System of difference constraints.
5. Telephone operator scheduling.
6. Matrix rounding problem.

7 Baseball elimination problem.
8. Open pit mining.
9. Scheduling on uniform parallel machines.
10. Tanker scheduling problem.
11. Leveling mountainous terrain
12. Reconstructing the left ventricle from X-ray projections.
13. Optimal loading of a hopping airplane.
14. Directed Chinese postman problem.
15. Models for building evacuation.
16. Personell assignment.
17. Paring stereo speakers.
18. Locating objects in space.
19. Matching moving objects.
20. Determing chemical bonds.
21. Designing physical systems.
22. Measuring homogeneity of bimetallic objects.
23. Reducing data storage.
24. All pairs minimax path problem.
25. Cluster analysis.

The applications described in this paper are drawn from important operations research, computer science and applied mathematics journals. The integer programming bibliographies by Kastning [1976], Hausman [1978] and Von Randow [1982, 1985] have helped us to identify these applications. We have excerped these applications from our fortcoming book, "*Network Flows: Theory, Algorithms, and Applications*," written by the first three authors of this paper. This book describes a total of 150 applications of the network optimization problems and provides many more references for the shortest paths, maximum flows, minimum cost flows, assignments and matchings, minimum spanning trees, convex cost flows, generalized flows, and multicommodity flows.

2. PRELIMINARIES

In this section, we present some basic notation and definitions of graph theory that we use in this paper. We also present a mathematical programming formulation of the minimum cost flow problem, which is the core network optimization problem studied in this paper.

Let G = (N, A) be a directed network defined by a set N of n *nodes*, and a set A of m *directed arcs*. Each arc (i, j) $\in$ A has an associated *cost* c_{ij} per unit flow on that arc. We assume that the flow cost varies linearly with the amount of flow. Each arc (i, j) $\in$ A also has a *capacity* u_{ij} denoting the maximum amount that can flow on the arc, and a *lower bound* l_{ij} denoting the minimum amount that must flow on the arc. We associate with each node i $\in$ N an integer number b(i) representing its supply/demand. If b(i) > 0, then node i is a *supply node*; if b(i) < 0, then node i is a *demand node*; and if b(i) = 0, then node i is a *transshipment node*.

The minimum cost flow problem is easy to state: we wish to determine a least cost shipment of a commodity through a network that will satisfy the flow demands at the demand nodes from available supplies at other nodes. The decision variables in the minimum cost flow problem are arc flows and we represent the flow on an arc $(i, j) \in A$ by x_{ij}. The minimum cost flow problem is an optimization model formulated as follows:

$$\text{Minimize} \sum_{(i, j) \in A} c_{ij} x_{ij} \tag{1a}$$

subject to

$$\sum_{\{j : (i, j) \in A\}} x_{ij} - \sum_{\{j : (j, i) \in A\}} x_{ji} = b(i), \text{ for all } i \in N, \tag{1b}$$

$$l_{ij} \le x_{ij} \le u_{ij}, \text{ for all } (i, j) \in A, \tag{1c}$$

with data satisfying the feasibility condition $\sum_{i=1}^{n} b(i) = 0$ (that is, the total supply must equal the total demand). We refer to the constraints in (1b) as the *mass balance constraints*. The mass balance constraints state that the outflow into each node minus its inflow must equal the supply/demand of the node. The flow must also satisfy the lower bound and capacity constraints (1c) which we refer to as the *flow bound constraints*. The flow bounds typically model physical capacities or restrictions imposed upon the flows' operating ranges. In most applications, the lower bounds on arc flows are zero; therefore, if we do not state lower bounds explicitly, we assume they have value zero.

We now collect together several basic definitions and describe some notation. A *path* in $G = (N, A)$ is a sequence of distinct nodes and arcs $i_1, (i_1, i_2), i_2, (i_2, i_3), i_3, \ldots, (i_{r-1}, i_r), i_r$ satisfying the property that either $(i_k, i_{k+1}) \in A$ or $(i_{k+1}, i_k) \in A$ for each $k = 1, \ldots, r-1$. For simplicity of notation, we often refer to a path as a sequence of nodes $i_1 - i_2 - \ldots - i_k$ when its arcs are apparent from the problem context. A *directed path* is defined similarly except that for any two consecutive nodes i_k and i_{k+1} on the path, the path must contain the arc (i_k, i_{k+1}). A *directed cycle* is a directed path together with the arc (i_r, i_1), and a *cycle* is a path together with the arc (i_r, i_1) or (i_1, i_r).

A graph $G' = (N', A')$ is a *subgraph* of $G = (N, A)$ if $N' \subseteq N$ and $A' \subseteq A$. A graph $G' = (N', A')$ is a spanning subgraph of $G = (N, A)$ if $N' = N$ and $A' \subseteq A$. Two nodes i and j are said to be *connected* if the graph contains at least one undirected path from i to j. A graph is said to be *connected* if all pairs of its nodes are connected; otherwise, it is *disconnected*. The connected subgraphs of a graph are called its *components*. A tree is a connected graph that contains no cycle. A subgraph T is a *spanning tree* of G if T is a tree of G containing all its nodes. A *cut* of G is any set $Q \subseteq A$ with the property that the graph $G' = (N, A-Q)$ is disconnected, and no subset of Q has this property. A cut partitions the graph into two sets of nodes, X and N-X. We shall sometimes represent the cut Q as the node partition [X, N-X]. A cut [X, N-X] is an *s-t cut* for two specially designated nodes s and t if $s \in X$ and $t \in N-X$.

3. SHORTEST PATH PROBLEM

The shortest path problem is among the simplest network optimization problems. In this problem, we wish to find a path of minimum cost (length) from a specified *source node* s to another specified *sink node* t assuming that each arc (i, j) $\in$ A has an associated cost (or length) c_{ij}. In the formulation of the minimum cost flow problem given in (1), if we set b(s) = 1, b(t) = -1, and b(i) = 0 for all other nodes, then the solution to this problem will send one unit of flow from node s to node t along the shortest path. If we want to determine shortest paths from the source node s to every other node in the network, then in the minimum cost flow problem, we set b(s) = (n-1) and b(i) = -1 for all other nodes. We can set each arc capacity u_{ij} to any number larger than (n-1). The minimum cost flow solution would then send one unit flow from node s to every other node i along a shortest path.

Shortest path problems are alluring to both researchers and to practitioners for several reasons: (i) they arise frequently in practice since in a wide variety of application settings we wish to send some material (for example, a computer data packet, a telephone call, a vehicle) between two specified points in a network as quickly, as cheaply, or as reliably as possible; (ii) they are easy to solve efficiently; (iii) as the simplest network models, they capture many of the most salient core ingredients of network flows and so they provide both a benchmark and a point of departure for studying more complex network models; and (iv) they arise frequently as subproblems when solving many combinatorial and network optimization problems. In this section, we describe a few applications of the shortest path problem that are indicative of its range of applications. The applications arise in applied mathemetics, biology, computer science, production planning, and work force scheduling. We conclude the section by producing references for many additional applications in a wide variety of fields.

Application 1. Approximating Piecewise Linear Functions (Imai and Iri [1986])

Numerous applications encountered within many different scientific fields use piecewise linear functions. On several occasions, because these functions contain a large number of breakpoints, they are expensive to store and to manipulate (for example, even to evaluate). In these situations, it might be advantageous to replace the piecewise linear function by another approximating function that uses fewer breakpoints. By approximating the function, we will generally be able to save on storage space and on the cost of using the function; we will, however, incur a cost because of the inaccuracy of the approximating function. In making the approximation, we would like to make the best possible tradeoff between these conflicting costs and benefits.

Let $f_1(x)$ be a piecewise linear function of a scalar x. We represent the function in the two-dimensional plane: it passes through n points $a_1 = (x_1,y_1)$, $a_2 = (x_2,y_2)$,..., $a_n = (x_n,y_n)$. Suppose that we have ordered the points so that $x_1 \le x_2 \le \cdots \le x_n$. We assume that the function varies linearly between every two consecutive points x_i and x_{i+1}. We consider situations in which n is very large and for practical reasons we wish to approximate the function $f_1(x)$ by another function $f_2(x)$ that passes through only a subset of the points a_1, a_2, ..., a_n (including a_1 and a_n). As an example, consider Figure 1(a): in this figure, we have approximated a function $f_1(x)$ passing through 10 points by a function $f_2(x)$ (drawn with dotted lines) passing through only 5 of the points.

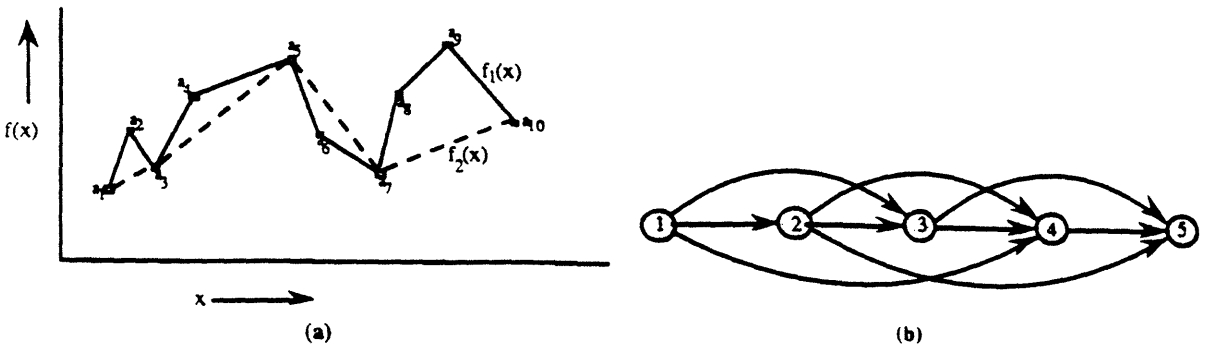


Figure 1. Approximating precise linear functions.
(a) Approximating a function $f_1(x)$ passing through 10 points by a function $f_2(x)$ passing through only 5 points.
(b) Corresponding shortest path problem.

This approximation results in a savings in storage space and in the use of the function. For purposes of illustration, assume that we can measure these costs by a per unit cost α associated with any single interval used in the approximation (which is defined by two points a_i and a_j). As we have noted, the approximation also introduces errors which have an associated penalty. We assume that the error of an approximation is proportional to the sum of the squared errors between the actual data points and the estimated points, i.e., the penalty is $\beta\sum_{i=1}^{p}[f_1(x_i) - f_2(x_i)]^2$ for some constant β. Our decision problem is to identify the subset of points to be used to define the approximation function $f_2(x)$ so we incur the minimum total cost as measured by the sum of the cost of storing and the cost of the errors imposed by the approximation.

We formulate this problem as a shortest path problem on a network G with n nodes, numbered 1 through n, as follows. The network contains an arc (i, j) for each pair of nodes i and j. Figure 1(b) gives an example of the network with n = 4 nodes. The arc (i, j) in this network signifies that we approximate the linear segments of the function $f_1(x)$ between the points $a_i, a_{i+1}, \ldots, a_j$ by one linear segment joining the points a_i and a_j. The cost c_{ij} of the arc (i, j) has two components: the storage cost α and the penalty associated with approximating all the points between a_i and a_j by the corresponding points lying on the line joining a_i and a_j. In the interval $[x_i, x_j]$, the approximating function is $f_2(x) = (x - x_i)[f_1(x_j) - f_1(x_i)]/(x_j - x_i)$ and so the total cost in this interval is

$$c_{ij} = \alpha + \beta\,[\Sigma_{k=i}^{j}\,(f_1(x_k) - f_2(x_k))^2].$$

Each directed path from node 1 to node n in G corresponds to a function $f_2(x)$, and the cost of this path equals the total cost for storing this function and for using it to approximate the original function. For example, the path 1-3-4 corresponds to the function $f_2(x)$ passing through the points a_1, a_3 and a_4. As a consequence of these observations, we see that the shortest path from node 1 to node n specifies the optimal set of points needed to define the approximating function $f_2(x)$.

Application 2. DNA Sequence Alignment (Waterman [1988])

Scientists model strands of DNA as a sequence of letters drawn from the alphabet {A, C, G, T}. Given two sequences of letters, say $B = b_1 b_2 ... b_p$ and $D = d_1 d_2 ... d_q$ of possibly different lengths, molecular biologists are interested in determining how similar or dissimilar these sequences are to each other. (These sequences are subsequences of a genome and typically contain several thousand letters.) A natural way of measuring the dissimilarity between the two sequences B and D is to determine the minimum "cost" required to transform sequence B into sequence D. To transform B into D, we can perform the following operations: (i) insert an element in B (at any place in the sequence) at a "cost" of α units; (ii) delete an element from B (at any place in the sequence) at a "cost" of β units; and (iii) mutate an element b_i into an element d_j at a "cost" of $g(b_i, d_j)$ units. Needless to say, it is possible to transform the sequence B into the sequence D in many ways and so identifying a minimum cost transformation is a nontrivial task. We show how we can solve this problem using dynamic programming, which we can also view as solving a shortest path problem on an appropriately defined network.

Suppose that we conceive of the process of transforming the sequence B into the sequence D as follows. First, add or delete elements from the sequence B so that the modified sequence, say B', has the same number of elements as D. Next "align" the sequences B' and D to create a one-to-one alignment between their elements. Finally, mutate the elements in the sequence B' so that this sequence becomes identical with the sequence D. As an example, suppose that we wish to transform the sequence B = AGTT into the sequence D = CTAGC. One possible transformation is to delete one of the elements T from B and add two new elements at the beginning, giving the sequence B' = ⊕⊕AGT (we denote any new element by a placeholder ⊕ and later assign a letter to this placeholder). We then align B' with D, as shown in Figure 2, and mutate the element T into C so that the sequences become identical. Notice that because we are free to assign values to the newly added elements, they do not incur any mutation cost. The cost of this transformation is $\beta+2\alpha+g(T,C)$.

```
B' = ⊕ ⊕ A G T          C T A G C
                   ⇒
D = C T A G C           C T A G C
```

Figure 2. Transforming the sequence B into the sequence D.

We now describe a dynamic programming formulation of this problem. Let f(i, j) denote the minimum cost of transforming the subsequence $b_1 b_2 ... b_i$ into the subsequence $d_1 d_2 ... d_j$. We are interested in the value f(p,q), which is the minimum cost of transforming B into D. To determine f(p,q), we will determine f(i,j) for all i = 0, 1, ..., p, and for all j = 0, 1, ..., q. We can determine these intermediate quantities f(i, j) using the following recursive relationships:

$$f(i,0) = \beta\, i \text{ for all } i; \tag{2a}$$

$$f(0,j) = \alpha\, j \text{ for all } j; \text{ and} \tag{2b}$$

$$f(i,j) = \min\{f(i-1, j-1) + g(b_i, d_j),\ f(i, j-1) + \alpha,\ f(i-1, j) + \beta\} \tag{2c}$$

We now justify this recursion. The cost of transforming a sequence of i elements into a null sequence is the cost of deleting i elements. The cost of transforming a null sequence into a sequence of j elements is the cost of adding j elements. Next consider f(i,j). Let B' denote the optimal aligned sequence of B (i.e., the sequence just before we create the mutation of B' to transform it into D). At this point, B' satisfies exactly one of the following three cases:

Case 1. *B' contains the letter b_i which is aligned with the letter d_j of D* (as shown in Figure 3(a)). In this case, f(i,j) equals the optimal cost of transforming the subsequence $b_1b_2...b_{i-1}$ into $d_1d_2...d_{j-1}$ and the cost of transforming the element b_i into d_j. Therefore, $f(i,j) = f(i-1, j-1) + g(b_i, d_j)$.

Case 2. *B' contains the letter b_i which is not aligned with the letter d_j* (as shown in Figure 3(b)). In this case, b_i is to the left of d_j and so a newly added element must be aligned with b_j. In this case, f(i,j) equals the optimal cost of transforming the subsequence $b_1b_2...b_i$ into $d_1d_2...d_{j-1}$ plus the cost of adding a new element to B. Therefore, $f(i,j) = f(i,j-1) + \alpha$.

Case 3. *B' does not contains the letter b_i*. In this case, we must have deleted b_i from B and so the optimal cost of the transformation equals the cost of deleting this element and transforming the remaining sequence into D. Therefore, $f(i,j) = f(i-1,j) + \beta$.

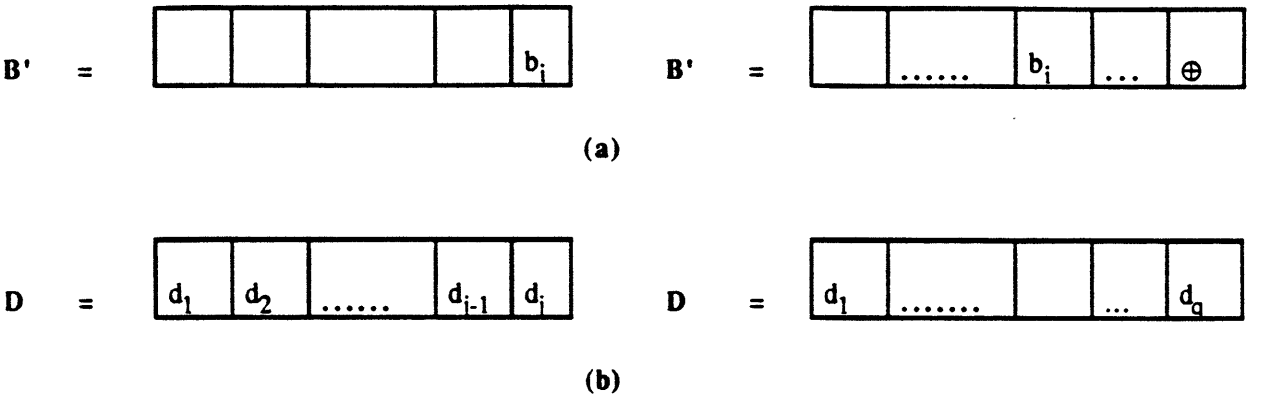


Figure 3. Explaining the dynamic programming recursion.

The preceding discussion justifies the recursive relationships specified in (2). We can use these relationships to compute f(i,j) for increasing values of i and, for a fixed value of i, for increasing values of j. This method allows us to compute f(p, q) in O(pq) time, that is time proportioned to the product of the number of elements in the two sequences.

We can alternatively formulate the DNA sequence alignment problem as a shortest path problem. In Figure 4, we show the shortest path network for this formulation for a situation with p = 3 and q = 3. For simplicity, in this network we let g_{ij} denote $g(b_i, d_j)$. We can establish the correctness of this formulation by applying an induction argument based upon the induction hypothesis that the shortest path length from node 0^0 to node i^j equals f(i,j). The shortest path from node 0^0 to node i^j must contain one of the following arcs as the last arc in the path: (i) arc $(i\text{-}1^{j-1}, i^j)$; (ii) arc (i^{j-1}, i^j), or (iii) arc $(i\text{-}1^j, i^j)$. In these three cases, the lengths of these paths will be $f(i-1,j-1) + g(b_i, d_j)$, $f(i,j-1) + \alpha$, and $f(i-1, j) + \beta$. Clearly,

the shortest path length f(i,j) will equal the minimum of these three numbers, which is consistent with the dynamic programming relationships stated in (2).

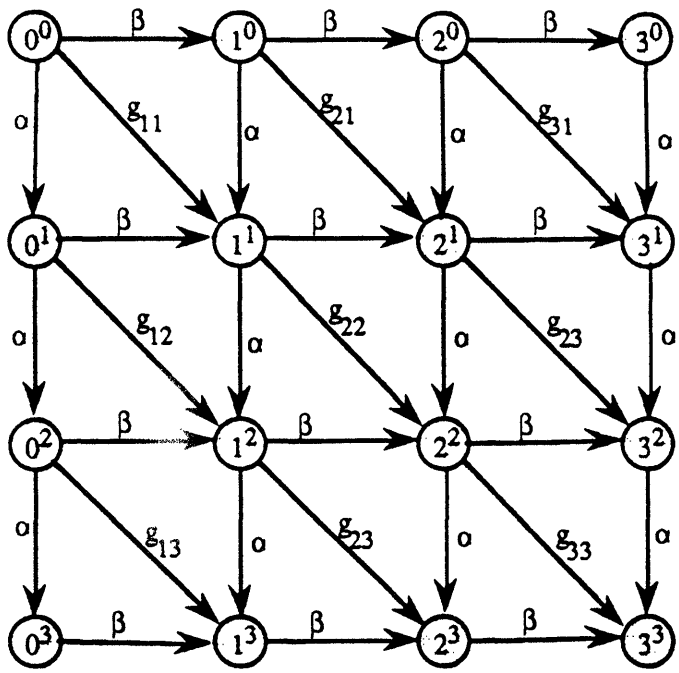


Figure 4. Sequence alignment problem as a shortest path problem.

This application shows a relationship between shortest paths and dynamic programming. We have seen how to solve the DNA sequence alignment problem through the dynamic programming recursion or by formulating and solving it as a shortest path problem on an acyclic network. The recursion we use to solve the dynamic programming problem is just a special implementation of one of the standard algorithms for solving shortest path problems on acyclic networks. This observation provides us with a concrete illustration of the meta statement that "(deterministic) dynamic programming is a special case of the shortest path problem". Accordingly, shortest path problems model the enormous range of applications in many disciplines that are solvable by dynamic programming.

Application 3. Production Planning Problems ([Veinott and Wagner [1962], Zangwill [1969], Evans [1977])

Many optimization problems in production and inventory planning are network optimization models. All of these models address a basic *economic order quantity* issue: when we plan a production run of any particular product, how much should we produce? Producing in large quantities reduces the time and cost required to set up equipment for the individual production runs; on the other hand, producing in large quantities also means that we will carry many items in inventory awaiting purchase by customers. The *economic order quantity* strikes a balance between the set up and inventory costs to find the production plan that achieves the lowest overall costs. The models that we consider in this section all attempt to balance the production and inventory carrying costs while meeting known demands that vary throughout a given planning horizon. We study one of the simplest models : a single product, single stage model with concave costs and backordering.

In this model, we assume that the production cost in each period is a concave function of the level of production. In practice, the production x_j in the j^{th} period frequently incurs a fixed cost F_j (independent of the level of production) and a per unit production cost c_j. Therefore, for each period j, the production cost is 0 for $x_j = 0$ and $F_j + c_j x_j$ if $x_j > 0$, which is a concave function of the production level x_j. The production cost might also be concave due to other economies of scale in production. In this model, we also permit backordering, which implies that we might not fully satisfy the demand of any period from the production in that period or from current inventory, but could fulfill the demand from production in future periods. We assume that we do not lose any customer whose demand is not satisfied on time and who must wait until his or her order materializes. Instead, we incur a penalty cost for backordering any item. We assume that the inventory carrying and backordering costs are linear, and that we have no capacity imposed upon production, inventory, or backordering volumes.

In this model, we wish to meet a prescribed demand d_j for each of K periods j = 1, 2, ..., K, by either producing an amount x_j in period j, by drawing upon the inventory I_{j-1} carried from the previous period, and/or by backordering the item from the next period. Figure 5(a) shows the network for modeling this problem. The network has K+1 nodes: the j^{th} node, for j = 1, 2, ..., K, represents the j^{th} planning period; node 0 represents the "source" of all production. The flow on the *production arc* (0, j) prescribes the production level x_j in period j, and the flow on *inventory carrying arc* (j, j+1) prescribes the inventory level I_j to be carried from period j to period j +1, and the flow B_j on the backordering arc (j, j-1) represents the amount backordered from the next period.

The network flow problem in Figure 5(a) is a concave cost flow problem, because the cost of flow on every production arc is a concave function. The following well known result about the concave cost flow problems helps us to solve this problem: These problems always have an optimal spanning tree solution (see, for example, Ahuja, Magnanti and Orlin [1993]). Figure 5(b) shows an instance of the spanning tree solution. This result implies the following property, known as the *production property*: In the optimal solution, each time we produce, we produce enough to meet the demand for an integral number of contiguous periods. Moreover, in no period do we both produce and carry inventory from the previous or next period.

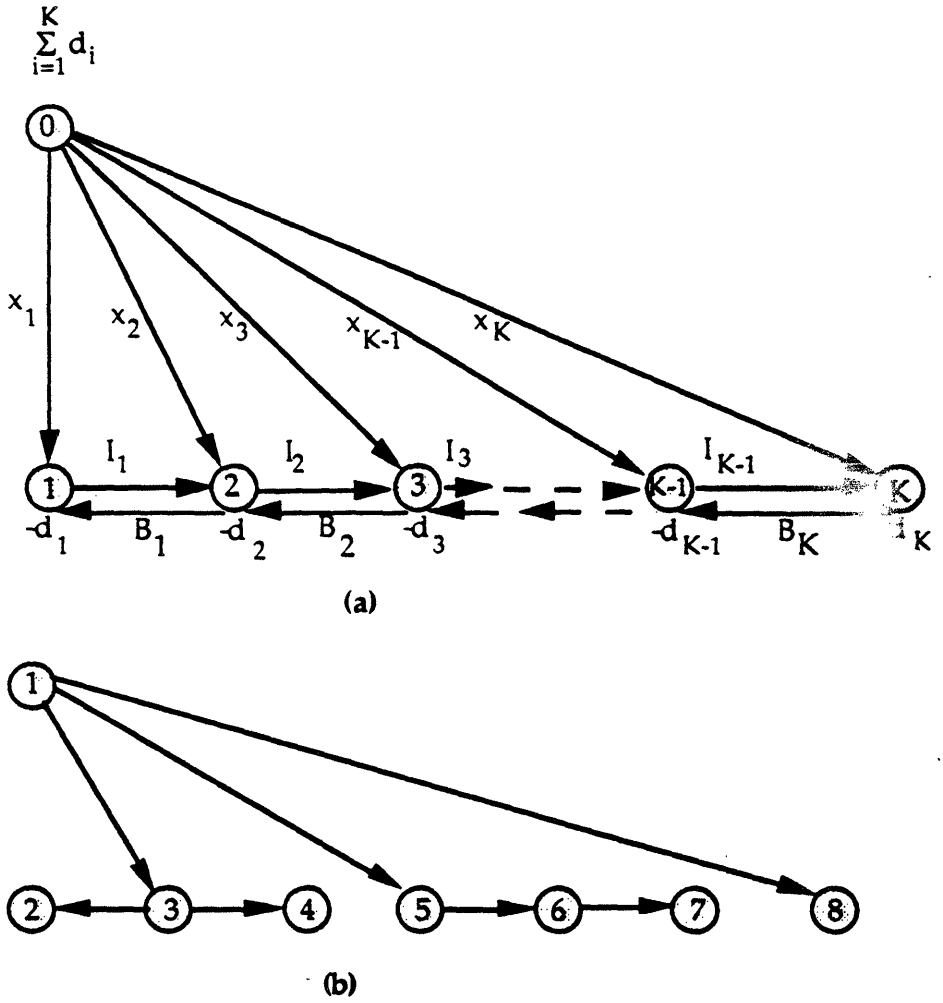


Figure 5. Production planning problem.
(a) Underlying network.
(b) Graphical structure of a spanning tree solution.

The production property permits us to solve the production planning problem very efficiently as a shortest path problem on an auxiliary network G' defined as follows. The network G' consists of nodes 1 through K+1 and contains an arc (i, j) for every pair of nodes i and j with i < j. We set the cost of arc (i, j) equal to the sum of the production, inventory carrying and backorder carrying costs incurred in satisfying the demands of periods i, i+1,..., j-1 by producing in some period k between i and j-1; we select this period k that gives the least possible cost. In other words, we vary k from i to j-1, and for each k, we compute the cost incurred in satisfying the demands of periods i through j-1 by the production in period k; the minimum of these values defines the cost of arc (i, j) in the auxiliary network G'. Observe that for every production schedule satisfying the production property, G' contains a directed path from node 1 to node K+1 with the same objective function value, and vice-versa. Therefore, we can obtain the optimal production schedule by solving a shortest path problem.

Several variants of the production planning problem arise in practice. If we impose capacities on the production, inventory, or backordering arcs, then the production property does not hold and we can not formulate this problem as a shortest path problem. In this case, however, if the production cost in each

period is linear, then the problem becomes a minimum cost flow model. The minimum cost flow model also models multistage situations, where the product passes through a sequence of operations. We treat each production operation as a separate stage and require that the product pass through each of the stages before its production is complete. In a further multi-item generalization where the common manufacturing facilities are used to manufacture multiple products in multiple stages; this problem is a multicommodity flow problem. The references cited for this application describe these various generalizations.

Application 4. System of Difference Constraints (Bellman [1958])

In some linear programming applications, with constraints of the form $Ax \leq b$, the n×m constraint matrix A contains one +1 and one -1 in each row; all the other entries are zero. Suppose that the k^{th} row has a +1 entry in column j_k and a -1 entry in column i_k; the entries in the vector b have arbitrary signs. This linear program defines the following set of m *difference constraints* in the n variables x = (x(1), x(2), x(n)):

$$x(j_k) - x(i_k) \leq b(k), \text{ for each } k=1,\ldots, m. \tag{3}$$

We wish to determine whether the system of difference constraints given by (3) has a feasible solution, and if so, we want to identify one. This model arises in a variety of applications; in the following application we describe the use of this model in the telephone operator scheduling, an additional application arises in the scaling of data (Orlin and Rothblum [1985]) and just-in-time scheduling (Elmaghraby [1978] and Levner and Nemirovsky [1991]).

Each system of difference constraints has an associated graph G, which we call a *constraint graph*. The constraint graph has n nodes corresponding to the n variables and m arcs corresponding to the m difference constraints. We associate an arc (i_k, j_k) of length b(k) in G with the constraint $x(j_k) - x(i_k) \leq b(k)$. As an example, consider the following system of constraints whose corresponding graph is shown in Figure 6(a).

$$x(3) - x(4) \leq 5, \tag{4a}$$

$$x(4) - x(1) \leq -10, \tag{4b}$$

$$x(1) - x(3) \leq 8, \tag{4c}$$

$$x(2) - x(1) \leq -11, \tag{4d}$$

$$x(3) - x(2) \leq 2. \tag{4e}$$

Figure 6. Graph corresponding to a system of difference constraints.

To model the problem use two well known results about shortest paths: (i) the optimality conditions for the shortest path distances $d(\cdot)$, in the network $G = (N, A)$ satisfy the optimality criteria $d(j) - d(i) \le c_{ij}$ for all $(i, j) \in A$; and (ii) the shortest path distances exist if and only if the network G does not contain a negative cycle (see Cormen, Leiserson, and Rivest [1990] and Ahuja, Magnanti, and Orlin [1993]). Notice the similarity between the difference inequalities (4) and the shortest path optimality conditions. The first result that the network for which (4) become the shortest path optimality conditions is given in Figure 6(a). The network shown in Figure 6(a) contains a negative cycle 1-2-3 of length -1, and the corresponding constraints (i.e., $x(2) - x(1) \le -11$, $x(3) - x(2) \le 2$ and $x(1) - x(3) \le 8$) are inconsistent because summing these constraints yields $0 \le -1$. Therefore, we conclude that the system of difference constraints given by (4) has no feasible solution.

We can detect the presence of a negative cycle in a network by using a label correcting algorithm. Label correcting algorithms require that all the nodes in the network are reachable by a directed path from some node, which we use as the source node for the shortest path problem. To satisfy this requirement, we introduce a new node s and join it to all the nodes in the network with arcs of zero cost. For our example, Figure 6(b) shows the modified network. Since all of the arcs incident to node s are directed out of this node, node s is not contained in any directed cycle and so the modification does not create any new directed cycles, and so does not introduce any cycles with negative costs. The label correcting algorithms either indicate the presence of a negative cycle or provide the shortest path distances. In the former case, the system of difference constraints has no solution, and in the latter case, the shortest path distances constitute a solution of (4).

Application 5. Telephone Operator Scheduling (Bartholdi, Orlin and Ratliff [1980])

As an application of the system of difference constraints, consider the following telephone operator scheduling problem. A telephone company needs to schedule operators around the clock. Let b(i) for i = 0, 1, 2, . . ., 23, denote the minimum number of operators needed for the i^{th} hour of the day (here b(0) denotes number of operators required between midnight and 1 AM). Each telephone operator works in a shift of 8 consecutive hours and a shift can begin at any hour of the day. The telephone company wants to determine a "cyclic schedule" that repeats daily, i.e., the number of operators assigned to the shift starting at 6 AM and ending at 2 PM is the same for each day. The optimization problem requires that we identify the

fewest operators needed to satisfy the minimum operator requirement for each hour of the day. If we let y_i denote the number of workers whose shift begins at the i^{th} hour, then we can state the telephone operator scheduling problem as the following optimization model:

$$\text{Minimize} \quad \sum_{i=0}^{23} y_i \tag{5a}$$

subject to

$$y_{i-7} + y_{i-6} + \ldots + y_i \geq b(i), \text{ for all } i = 8 \text{ to } 23, \tag{5b}$$

$$y_{17+i} + \ldots + y_{23} + y_0 + \ldots + y_i \geq b(i), \text{ for all } i = 0 \text{ to } 7, \tag{5c}$$

$$y_i \geq 0 \text{ for all } i = 0 \text{ to } 23. \tag{5d}$$

Notice that this linear program has a very special structure because the associated constraint matrix contains only 0's and 1's and the 1's or 0's in each row appear consecutively. In this application, we study a restricted version of the telephone operator scheduling problem: we wish to determine whether some feasible schedule uses p or fewer operators. We convert this restricted problem into a system of difference constraints by redefining the variables. Let $x(0) = y_0$, $x(1) = y_0 + y_1$, $x(2) = y_0 + y_1 + y_2$, ... , and $x(23) = y_0 + y_2 + \ldots + y_{23} = p$. Now notice that we can rewrite each constraint in (5b) as

$$x(i) - x(i-8) \geq b(i), \text{ for all } i = 8 \text{ to } 23, \tag{6a}$$

and each constraints in (5c) as

$$x(23) - x(16+i) + x(i) = p - x(16-i) + x(i) \geq b(i), \text{ for all } i = 0 \text{ to } 7. \tag{6b}$$

Finally, the nonnegativity constraints (5d) become

$$x(i) - x(i-1) \geq 0. \tag{6c}$$

By virtue of this transformation, we have reduced the restricted version of the telephone operator scheduling problem into a problem of finding a feasible solution of the system of difference constraints.

Additional Applications

Some additional applications of the shortest path problem include (1) *knapsack problem* (Fulkerson [1966]); (2) *tramp steamer problem* (Lawler [1966]); (3) *allocating inspection effort on a production line* (White [1969]); (4) *reallocation of housing* (Wright [1975]); (5) *assortment of steel beams* (Frank [1965]); (6) *compact book storage in libraries* (Ravindran [1971]); (7) *concentrator location on a line* (Balakrishnan, Magnanti and Wong [1989]); (8) *manpower planning problem* (Clark and Hasting [1977]); (9) *equipment replacement* (Veinott and Wagner [1962]); (10) *determining minimum project duration* (Elmaghraby [1978]); (11) *assembly line balancing* (Gutjahr and Nemhauser [1964]); (12) *optimal improvement of transportation networks* (Goldman and Nemhauser [1967]); (13) *machining process optimization* (Szadkowski [1970]); (14) *capacity expansion* (Luss [1979]); (15) *routing in computer communication networks* (Schwartz and Stern [1980]); (16) *scaling of matrices* (Golitschek and Schneider

[1984]); (17) *city traffic congestion* (Zawack and Thompson [1987]); (18) *molecular confirmation* (Dress and Havel [1988]); (19) *order picking in an isle* (Goetschalckx and Ratliff [1988]); and (20) *robot design* (Haymond, Thornton, and Warner [1988]. Shortest path problems often arise as important subroutines within algorithms for solving many different types of network optimization problems. These applications are too numerous to mention.

4. MAXIMUM FLOW PROBLEM

The maximum flow problem is in a sense a complementary model to the shortest path problem. The shortest path problem models situations in which flow incurs a cost, but is not restricted by any capacities; in contrast, in the maximum flow problem flow incurs no costs, but is restricted by flow bounds. The maximum flow problem seeks a feasible solution, which sends the maximum amount of flow from a specified source node s to another specified sink node t. If we interpret u_{ij} as the maximum flow rate of arc (i, j), then the maximum flow problem identifies the maximum steady-state flow that the network can send from node s to node t per unit time. We can formulate this problem as a minimum cost flow problem in the following manner. We set b(i) = 0 for all $i \in N$, $c_{ij} = 0$ for all $(i, j) \in A$, and introduce an additional arc (t, s) with cost $c_{ts} = -1$ and flow bound $u_{ts} = \infty$. Then the minimum cost flow solution maximizes the flow on arc (t, s); but since any flow on arc (t, s) must travel from node s to node t through the arcs in A (since each b(i) = 0), the solution to the minimum cost flow problem will maximize the flow from node s to node t in the original network.

The maximum flow problem arises in a wide variety of situations and in several forms. Examples of the maximum flow problem include determining the maximum steady state flow of (i) petroleum products in a pipeline network, (ii) cars in a road network; (iii) messages in a telecommunication network; and (iv) electricity in an electrical network. Sometimes the maximum flow problem occurs as a subproblem in the solution of more difficult network problems such as the minimum cost flow problem or the generalized flow problem. The maximum flow problem also arises in a number of combinatorial applications that on the surface might not appear to be maximum flow problems at all. In this section, we describe a few such applications.

Application 6. Matrix Rounding Problem (Bacharach [1966])

This matrix rounding application is concerned with consistent rounding of the elements, row sums, and column sums of a matrix. We are given a pxq matrix of *real* numbers $D = \{d_{ij}\}$, with row sums α_i and column sums β_j. We can round any real number to the next smaller integer $\lfloor a \rfloor$ or to the next larger integer $\lceil a \rceil$, and the decision to round up or down is entirely up to us. The matrix rounding problem requires that we round the matrix elements, and the row and column sums of the matrix so that the sum of the rounded elements in each row equals the rounded row sum, and the sum of the rounded elements in each column equals the rounded column sum. We refer to such a rounding as a *consistent rounding*.

We shall formulate this problem and some of the subsequent following applications as a problem known as a *feasible flow problem*. In the feasible flow problem, we wish to determine a flow x in a network G = (N, A) satisfying the following constraints:

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i), \quad \text{for } i \in N, \tag{7a}$$

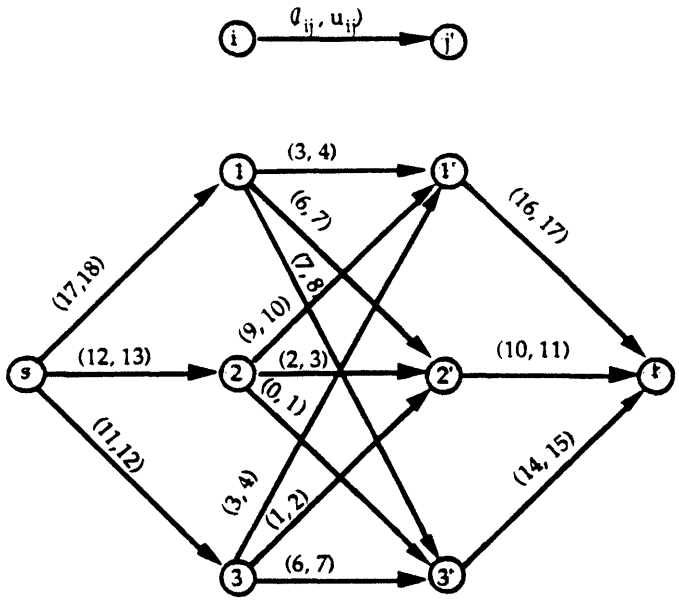
$$0 \le x_{ij} \le u_{ij}, \quad \text{for all } (i,j) \in A. \tag{7b}$$

We assume that $\sum_{i\in N} b(i) = 0$. We can solve the feasible flow problem by solving a maximum flow problem defined on an augmented network as follows. We introduce two new nodes, a source node s and a sink node t. For each node i with $b(i) > 0$, we add an arc (s, i) with capacity b(i), and for each node i with $b(i) < 0$, we add an arc (i, t) with capacity -b(i). We refer to the new network as the *transformed network*. Then we solve a maximum flow problem from node s to node t in the transformed network. It is easy to show that the problem (7) has a feasible solution if and only if the maximum flow saturates all the arcs emanating from the source node.

We show how we can discover such a rounding scheme by solving a feasible flow problem for a network with nonnegative lower bounds on arc flows. Figure 7(b) shows the maximum flow network for the matrix rounding data shown in Figure 7(a). This network contains a node i corresponding to each row i and a node j' corresponding to each column j. Observe that this network contains an arc (i, j') for each matrix element d_{ij}, an arc (s, i) for each row sum, and an arc (j', t) for each column sum. The lower and the upper bounds of arc (k, *l*) corresponding to the matrix element, row sum, or column sum of value v_{ij} are $\lfloor v_{ij} \rfloor$ and $\lceil v_{ij} \rceil$, respectively. It is easy to establish a one-to-one correspondence between the consistent roundings of the matrix and feasible integral flows in the associated network. We know that there is a feasible integral flow since the original matrix elements induce a feasible fractional flow, and maximum flow algorithms produce all integer flows. Consequently, we can find a consistent rounding by solving a maximum flow problem.

				row sum
	3.1	6.8	7.3	17.2
	9.6	2.4	0.7	12.7
	3.6	1.2	6.5	11.3
column sum	16.3	10.4	14.5	

(a)



(b)

Figure 7. (a) Matrix rounding problem.
(b) Associated network.

This matrix rounding problem arises is several application contexts. For example, the U.S. Census Bureau uses census information to construct millions of tables for a wide variety of purposes. By law, the bureau has an obligation to protect the source of its information and not disclose statistics that could be attributed to any particular individual. We might disguise the information in a table as follows. We round off each entry in the table, including the row and column sums, either up or down to a multiple of a constant k (for some suitable value of k), so that the entries in the table continue to add to the (rounded) row and column sums, and the overall sum of the entries in the new table adds to a rounded version of the overall sums in the original table. This Census Bureau problem is the same as the matrix rounding problem discussed earlier except that we need to round each element to a multiple of $k \geq 1$ instead of rounding it to a multiple of 1. We solve this problem by defining the associated network as before, but now defining the lower and upper bounds for any arc with an associated real number 'α' as the greatest multiple of k less than or equal to α and the smallest multiple of k greater than or equal to α.

Application 7. Baseball Elimination Problem (Schwartz [1966])

At a particular point in the baseball season, each of n + 1 teams in the American League, which we number as 0, 1, ..., n, has played several games. Suppose that team i has won w_i of the games that it has already played and that g_{ij} is the number of games that teams i and j have yet to play with each other. No game ends in a tie. An avid and optimistic fan of one of the teams, say the Boston Red Sox, wishes to know if his team still has a chance to win the league title. We say that we can *eliminate* a specific team 0, the Red Sox, if for every possible outcome of the unplayed games, at least one team will have more wins than the Red Sox. Let w_{max} denote w_0 plus the total number of games team 0 has yet to play, which, in the best of all possible worlds, is the number of victories the Red Sox can achieve. Then, we cannot eliminate team 0 if in some outcome of the remaining games to be played throughout the league, w_{max} is

at least as large as the possible victories of every other team. We want to determine whether we can or cannot eliminate team 0.

We can transform this baseball elimination problem into a feasible flow problem on a bipartite network shown in Figure 8, whose node set is $N_1 \cup N_2$. The node set for this network contains (i) a set N_1 of n *team nodes* indexed 1 through n, (ii) n(n-1)/2 *game nodes* of the type i-j for each $1 \le i \le j \le n$, and (iii) a *source node* s. Each game node i-j has a demand of g_{ij} units and has two incoming arcs (i, i-j) and (j, i-j) The flows on these two arcs represent the number of victories for team i and team j, respectively, among the additional g_{ij} games that these two teams have yet to play against each other (which is the required flow into the game node i-j). The flow x_{si} on the source arc (s, i) represents the total number of additional games that team i wins. We cannot eliminate team 0 if this network contains a feasible flow x satisfying the conditions

$$w_{max} \ge w_i + x_{si}, \text{ for all } i = 1, ..., n.$$

which we can rewrite as

$$x_{si} \le w_{max} - w_i, \text{ for all } i = 1, ..., n.$$

This observation explains the capacities of arcs shown in the figure. We have thus shown that if the feasible flow problem shown in Figure 8 admits a feasible flow, then we cannot eliminate team 0; otherwise, we can eliminate this team and our avid fan can turn his attention to other matters.

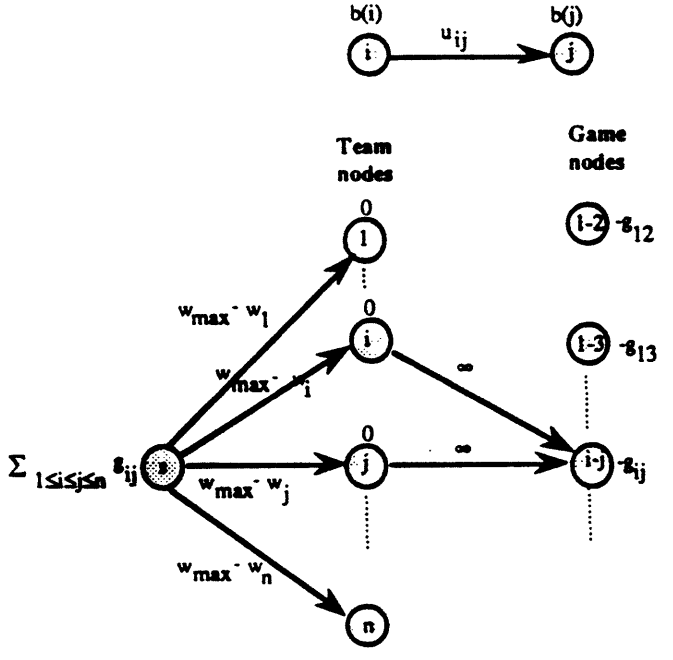


Figure 8. Network formulation of the baseball elimination problem.

Application 8. Open Pit Mining (Johnson [1968])

Before describing the open pit mining problem, we first define an underlying general model known as the *maximum closure problem.* A *closure* of a directed network G = (N, A) is a subset of nodes without any outgoing arcs; that is, a subset $N_1 \subseteq N$ satisfying the property that if i belongs to N_1 and (i, j) $\in$ A, then j also belongs to N_1. A closure might have more than one component. Suppose we associate a node weight w_i (of arbitrary sign) with each node i of G. In the *maximum weight closure problem,* we wish to find a closure N_1 with the largest possible weight $w(N_1)$ defined as $w(N_1) = \Sigma_{i \in N_1} w_i$. As an example, the network shown in Figure 9(a) has the closures {3, 4, 5}, {4, 5}, {5}, {2, 5} and {1, 2, 4, 5}; the maximum weight closure for this network is {3, 4, 5}.

We can transform the maximum weight closure problem defined on the network G = (N, A) into a maximum flow problem on a slightly augmented network G' = (N', A') in the following manner. We introduce a source node s and for each node i $\in$ N with $w_i > 0$, we create an arc (s, i) with capacity w_i. We also introduce a sink node t and for each node i $\in$ N with $w_i < 0$, we create an arc (i, t) with capacity $-w_i$. We then set the capacity of every original arc (i, j) $\in$ A equal to ∞. (In fact, any integer greater than $\Sigma_{i \in N} |w_i|$ would suffice). Figure 9(b) shows the transformed network for the maximum weight closure problem shown in Figure 9(a). It is possible to show that for every closure N_1 in Figure 9(a), the network G' in Figure 9(b) has a finite capacity s-t cut [S, $\bar{S}$] satisfying the equality $w(N_1) + u[S, \bar{S}] = \Sigma_{i \in N} w_i$ = a constant; and the converse is also true. This relationship implies that the minimum capacity cut in Figure 9(b) will yield a maximum weight closure in Figure 9(a).

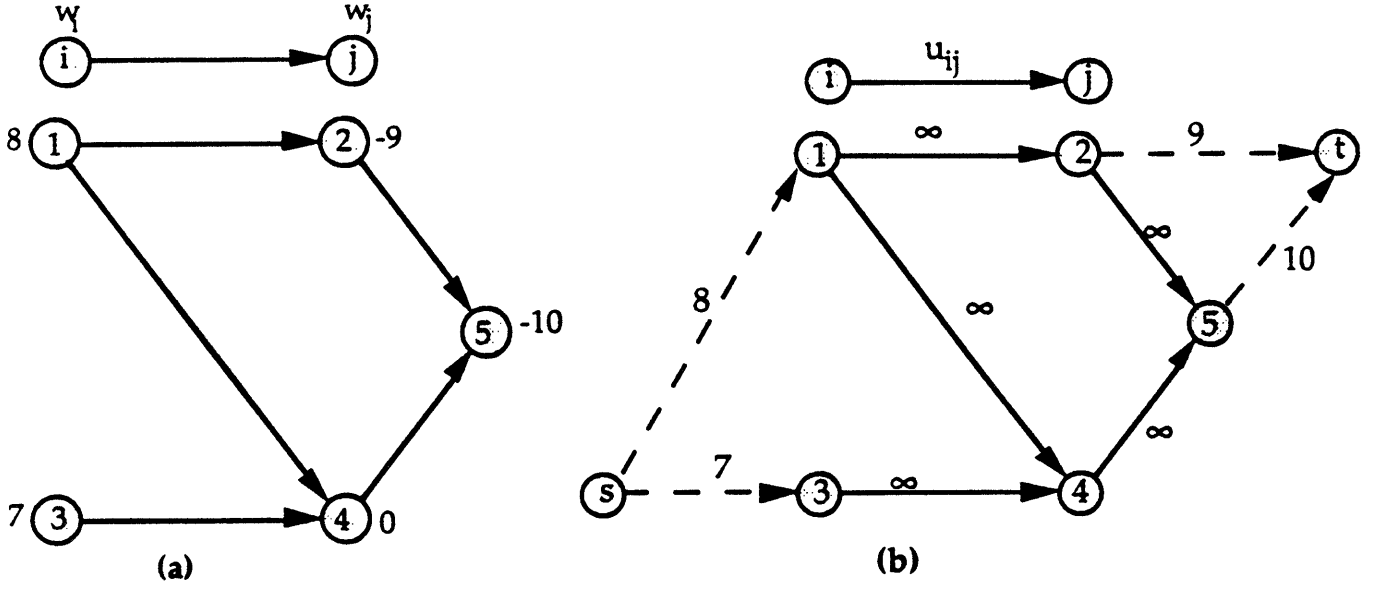


Figure 9. (a) Maximum weight closure problem.
(b) Transformed network G'.

We now return to the open pit mining problem, which is a problem of considerable importance in determining the optimal contour of an open-pit mine. In an open-pit mine, we might divide the potential mining region into blocks. The provisions of any given mining technology, and perhaps the geography of the mine, impose restrictions on how we can remove the blocks. For example, we can never remove a block until we have removed every block that lies immediately above it (see Figure 10); restrictions on the "angle" of mining the blocks might impose similar precedence conditions. Moreover, every block i has an economic measure w_i representing the net profit from removing that block (value of the ore contained in the block minus the cost of exploiting and processing the block). In the open pit mining problem, we wish to identify a set of blocks that maximizes the net profit. We model this problem as a maximum weight closure problem by representing each block as a node; if we must remove block j before removing block i, then we include the arc (i, j) in the network. If we want to remove a contour B of blocks, then every block that we need to remove before removing a block in B must also lie in B. That is, the nodes defined by B have no outgoing arcs and, therefore, define a closure of the network. Therefore, the open pit mining problem is a special case of the maximum weight closure problem.

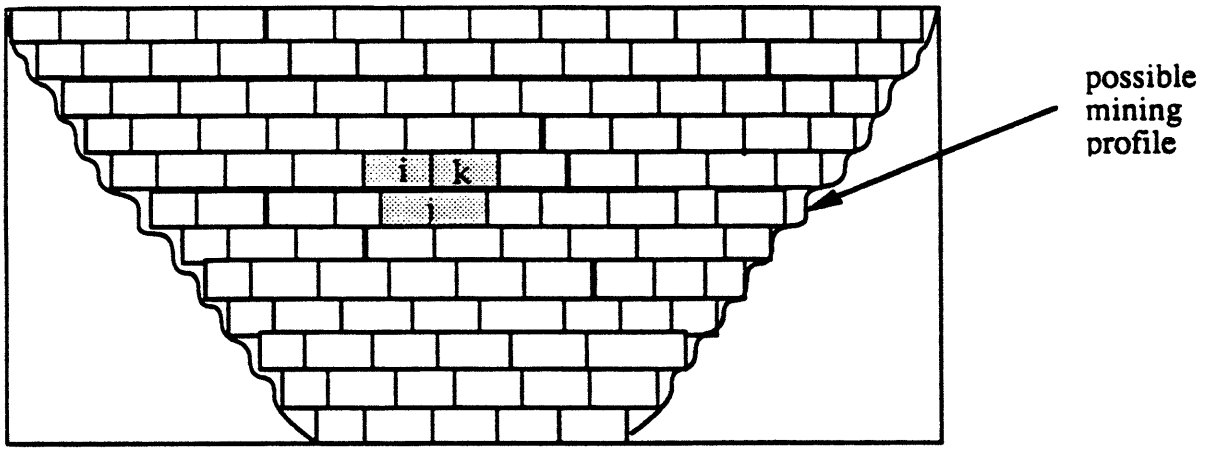


Figure 10. An open pit mine; we must remove blocks i and k before removing block j.

The maximum weight closure problem arises in several other applications. These applications include (1) *selecting freight handling terminals* (Rhys [1970]); (2) *optimal destruction of military targets* (Orlin [1987]); and (3) *the fly away kit problem* (Mamer and Smith [1982]). The survey paper of Picard and Queyranne [1982] cites additional application of the maximum weight closure problem.

Application 9. Scheduling on Uniform Parallel Machines (Federgruen and Groenevelt [1986])

In this application, we consider the problem of scheduling a set J of jobs on M uniform parallel machines. Each job $j \in J$ has a processing requirement p_j (denoting the number of machine days required to complete the job); a release date r_j (representing the beginning of the day when job j becomes available for processing); and a due date $d_j \geq r_j + p_j$ (representing the beginning of the day by which the job must be completed). We assume that a machine can work on only one job at a time and that each job can be processed by at most one machine at a time. However, we allow *preemptions*, i.e., we can interrupt a job

and process it on different machines on different days. The scheduling problem is to determine a feasible schedule that completes all jobs before their due dates or to show that no such schedule exists.

This type of preemptive scheduling problem arise in batch processing systems when each batch consists of a large number of units. The feasible scheduling problem, described in the preceding paragraph, is a fundamental problem in this situation and can be used as a subroutine for more general scheduling problems, such as the maximum lateness problem, the (weighted) minimum completion time problem, and the (weighted) maximum utilization problem.

To illustrate the formulation of the feasible scheduling problem as a maximum flow problem, we shall use the scheduling data described in Figure 11.

job (j)	1	2	3	4
Processing time (p_j)	1.5	1.25	2.1	3.6
Release time (r_j)	3	1	3	5
Due date (d_j)	5	4	7	9

Figure 11. A scheduling problem.

First, we rank all the release and due dates, r_j and d_j for all j, in ascending order and determine $P \leq 2|J| - 1$ mutually disjoint intervals of dates between consecutive milestones. Let $T_{k,l}$ denote the interval that starts at the beginning of date k and ends at the beginning of date $l+1$. For our example, this order of release and due dates is 1, 3, 4, 5, 7, 9. We have five intervals represented by $T_{1,2}$, $T_{3,3}$, $T_{4,4}$, $T_{5,6}$ and $T_{7,8}$. Notice that within each interval $T_{k,l}$, the set of available jobs (that is, those released, but not yet due) does not change: we can process all jobs j with $r_j \leq k$ and $d_j \geq l + 1$ in the interval.

We formulate the scheduling problem as a maximum flow problem on a bipartite network G as follows. We introduce a source node s, a sink node t, a node corresponding to each job j, and a node corresponding to each interval $T_{k,l}$, as shown in Figure 12. We connect the source node to every job node j with an arc with capacity p_j, indicating that we need to assign a minimum of p_j machine days to job j. We connect each interval node $T_{k,l}$ to the sink node t by an arc with capacity $(l-k+1)M$, representing the total number of machine days available on the days from k to l. Finally, we connect a job node j to every interval node $T_{k,l}$ if $r_j \leq k$ and $d_j \geq l+1$ by an arc with capacity $(l-k+1)$ which represents the maximum number of machines that we can allot to job j on the days from k to l. We next solve a maximum flow problem on this network: the scheduling problem has a feasible schedule if and only if the maximum flow value equals $\Sigma_{j \in J} p_j$ (alternatively, for every node j, the flow on arc (s, j) is p_j). The validity of this formulation is easy to establish by showing a one-to-one correspondence between feasible schedules and flows of value equal to $\Sigma_{j \in J} p_j$ from the source to the sink.

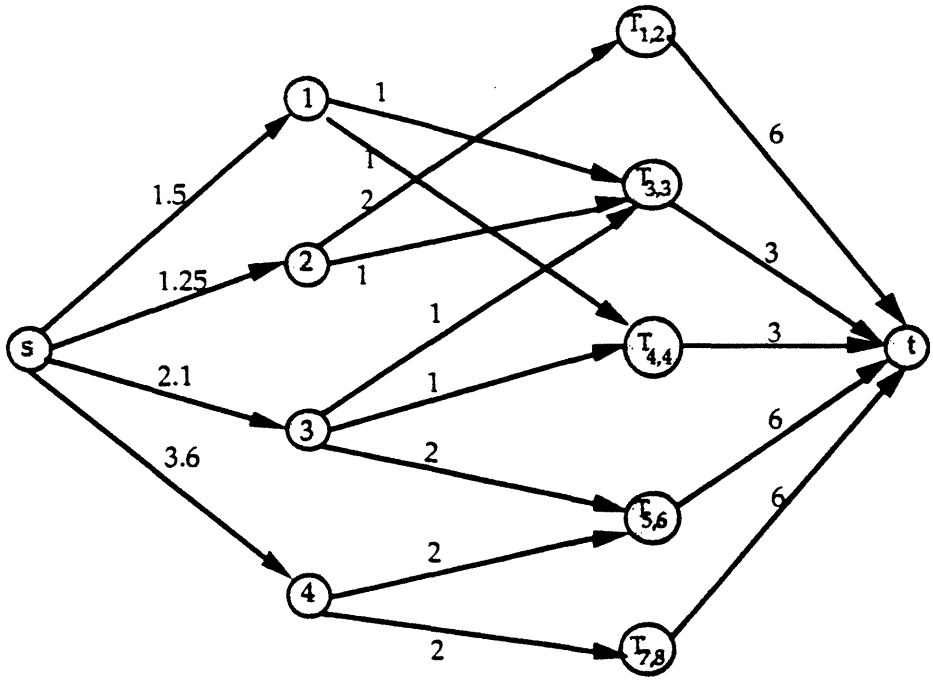


Figure 12. Network for scheduling uniform parallel machines.

Application 10. Tanker Scheduling Problem (Dantzig and Fulkerson [1954])

A steamship company has contracted to deliver perishable goods between several different origin-destination pairs. Since the cargo is perishable, the customers have specified precise dates (i.e., delivery dates) when the shipments must reach their destinations. (The cargoes may not arrive early r late.) The steamship company wants to determine the minimum number of ships needed to meet the delivery dates of the shiploads.

To illustrate a modeling approach for this problem, we consider an example with four shipments; each shipment is a full shipload with the characteristics shown in Figure 13(a). For example, as specified by the first row in this figure, the company must deliver one shipload available at port A and destined for port C on day 3. Figures 13(b) and 13(c) show the transit times for the shipments (including allowances for loading and unloading the ships) and the return times (without a cargo) between the ports.

Ship-ment	Origin	Desti-nation	Delivery date
1	Port A	Port C	3
2	Port A	Port C	8
3	Port B	Port D	3
4	Port B	Port C	6

(a)

	C	D
A	3	2
B	2	3

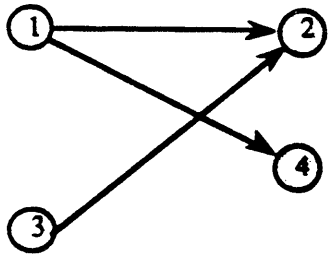
(b)

	A	B
C	2	1
D	1	2

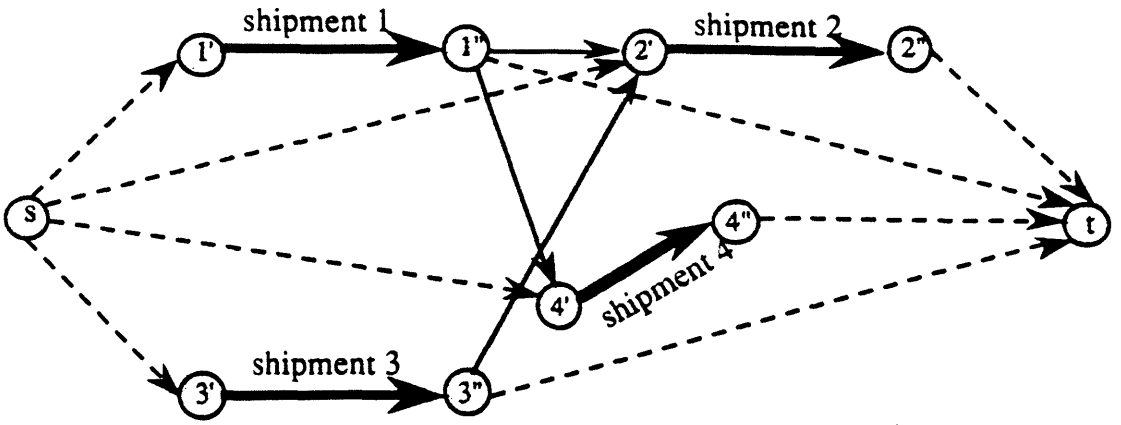
(c)

Figure 13. Data for the tanker scheduling problem.

We solve this problem by constructing a network shown in Figure 14(a). This network contains a node for each shipment and an arc from node i to node j if it is possible to deliver shipment j after completing shipment i; that is, the start time of shipment j is no earlier than the delivery time of shipment i plus the travel time from the destination of shipment i to the origin of shipment j. A directed path in this network corresponds to a feasible sequence of shipment pickups and deliveries. The tanker scheduling problem requires that we identify the minimum number of directed paths that will contain each node in the network on exactly one path.



(a)



(b)

Figure 14. Network formulation of the tanker scheduling problem.
(a) Network of feasible sequences of two consecutive shipments.
(b) Maximum flow model.

We can transform this problem into the framework of the maximum flow problem as follows. We split each node i into two nodes i' and i" and add the arc (i', i"). We set the lower bound on each arc (i', i"), called the *shipment arc*, equal to one so that at least unit of flow passes through this arc. We also add a source node s and connect it to the origin of each shipment (to represent putting a ship into service), and we add a sink node t and connect each destination node to it (to represent taking a ship out of service). We set the capacity of each arc in the network to value one. Figure 14(b) shows the resulting network for our example. In this network, each directed path from the source node s to the sink node t corresponds to a feasible schedule for a single ship. As a result, a feasible flow of value v in this network decomposes into schedules of v ships, and our problem reduces to identifying a feasible flow of minimum value. We note that the zero flow is not feasible because shipment arcs have unit lower bounds. We can solve this problem, which is known as the *minimum value problem*, in the following manner. We first establish a feasible flow in the network by solving a maximum flow problem. We then send flow from node t to node s until no more flow can be sent. The solution at this point is an optimal solution of the minimum flow problem.

Several other applications bear a close resemblence to the tanker scheduling problem and can be solved using the same technique. We next briefly introduce some of these applications.

Optimal coverage of sporting events. A group of reporters wants to cover a set of sporting events in an olympiad. The sports events are held in several stadiums throughout a city. We know the starting time of each event, its duration, and the stadium where it is held. We are also given the travel times between different stadiums. We want to determine the least number of reporters required to cover the sporting events.

Airline scheduling problem. An airline has p flight legs that it wishes to service by the fewest possible planes. To do so, it must determine the most efficient way to combine these legs into flight schedules. The starting time for flight i is a_i and the finishing time is b_i. The plane requires r_{ij} time to return from the point of destination of flight i to the point of origin of flight j.

Machine setup problem. A job shop needs to perform eight tasks on a particular day. We know the start and end times of each task. The workers must perform these tasks according to this schedule and so that exactly one worker performs each task. A worker cannot work on two jobs at the same time. We also know the setup time (in minutes) required for a worker to go from one task to another. We wish to find the minimum number of workers to perform the tasks.

The maximum flow problem arises in many other applications, including: (1) *problem of representatives* (Hall [1956]); (2) *distributed computing on a two-processor model* (Stone [1977]); (3) *the tournament problem* (Ford and Johnson [1959]); (4) *police patrol problem* (Khan [1979]); (5) *nurse staff scheduling* (Khan and Lewis [1987]); (6) *solving a system of equations* (Lin [1986]); (7) *statistical security of data* (Gusfield [1988], Kelly, Golden and Assad [1992]); (8) *minimax transportation problem* (Ahuja [1986]); (9) *network reliability testing* (Van Slyke and Frank [1972]); (10) *maximum dynamic flows* (Ford and Fulkerson [1958]); (11) *preemptive scheduling on machines with different speeds* (Martel [1982]); and (12) *multifacility rectilinear distance location problem* ([Picard and Ratliff [1978]). The following papers describe additional applications of the maximum flow problem or provide additional references: Berge and Ghouila-Houri [1962]; McGinnis and Nuttle [1978], Picard and Queyranne [1982], Abdallaoui [1987], Gusfield, Martel and Fernandez-Baca [1987], Gusfield and Martel [1989], and Gallo, Grigoriadis and Tarjan [1989].

5. MINIMUM COST FLOW PROBLEM

The minimum cost flow model is the most fundamental of all network optimization problems. This problem, described in Section 1, is concerned with determining a least cost shipment of a commodity through a network that will satisfy demands at certain nodes from available supplies at other nodes. This model has a number of familiar applications: the distribution of a product from manufacturing plants to warehouses, or from warehouses to retailers; the flow of raw material and intermediate goods through the various machining stations in a production line; the routing of automobiles through an urban street network; and the routing of calls through the telephone system. Minimum cost flow problems arise in almost all industries, including agriculture, communications, defense, education, energy, health care, manufacturing, medicine, retailing, and transportation. Indeed, minimum cost flow problems are pervasive in practice. In this section, by considering a few selected applications, we illustrate some of these possible uses of minimum cost flow problems.

Application 11. Leveling Mountainous Terrain (Farley [1980])

This application was inspired by a common problem facing civil engineers when they are building road networks through hilly or mountainous terrain. The problem concerns the distribution of earth from high points to low points of the terrain in order to produce a leveled road bed. The engineer must determine a plan for leveling the route by specifying the number of truck loads of earth to move between various locations along the proposed road network.

We first construct a *terrain graph* which is an undirected graph whose nodes represent locations with a demand for earth (low points) or locations with a supply of earth (high points). An arc of this graph represents an available route for distributing the earth and the cost of this arc represents the cost of per truck

load of moving earth between the two points. (A *truckload* is the basic unit for redistributing the earth.) Figure 15 shows a portion of the terrain graph.

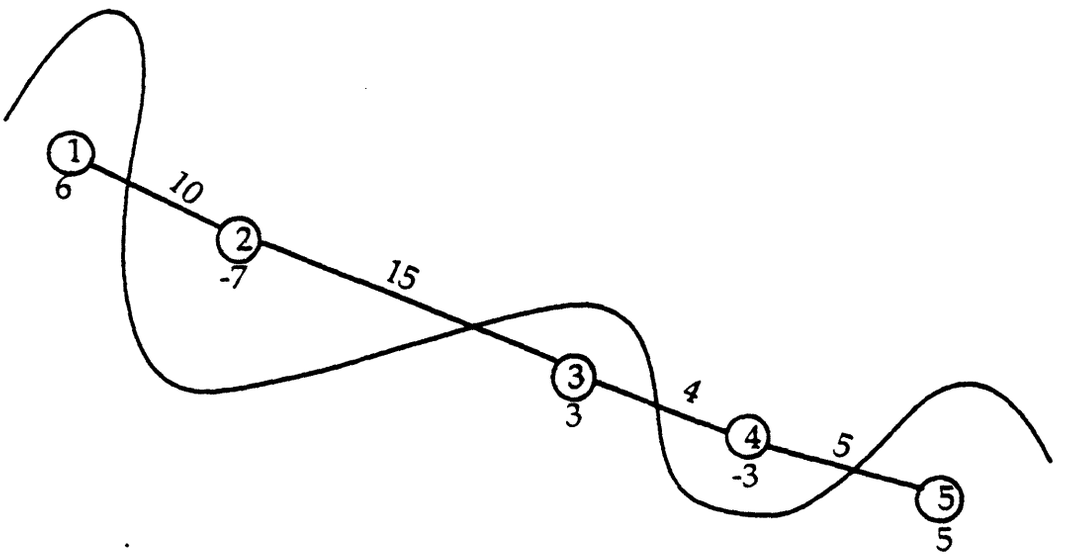


Figure 15. A portion of the terrain graph.

A leveling plan for a terrain graph is a flow (set of truckloads) that meets the demands at nodes (levels the low points) by the available supplies (by earth obtained from high points) at the minimum cost (for the truck movements). This model is clearly a minimum cost flow problem in the terrain graph.

Application 12. Reconstructing the Left Ventricle from X-ray Projections (Slump and Gerbrands [1982])

This application describes a minimum cost flow model for reconstructing the three-dimensional shape of the left ventricle from biplane angiocardiograms that the medical profession uses to diagnose heart diseases. To conduct this analysis, we first reduce the three-dimensional reconstruction problem into several two-dimensional problems by dividing the ventricle into a stack of parallel cross sections. Each two-dimensional cross section consists of one connected region of the left ventricle. During a cardiac catheterization, doctors inject a dye known as Roentgen contrast agent into the ventricle; by taking X-rays of the dye, they would like to determine what portion of the left ventricle is functioning properly (that is, permitting the flow of blood). Conventional biplane X-ray installations do not permit doctors to obtain a complete picture of the left ventricle; rather, these X-rays provide one-dimensional projections that record the total intensity of the dye along two axes (see Figure 16). The problem is to determine the distribution of the cloud of dye within the left ventricle, and thus the shape of the functioning portion of the ventricle, assuming that the dye mixes completely with the blood and fills the portions that are functioning properly .

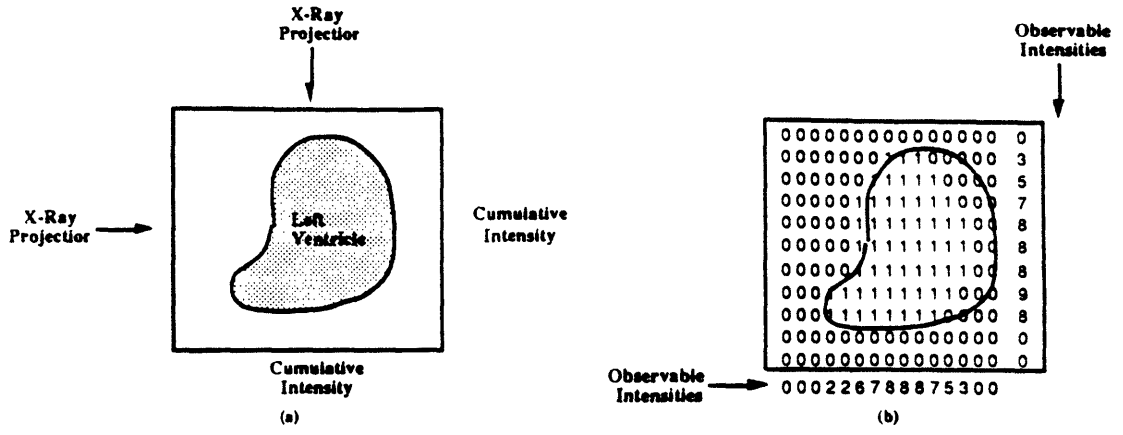


Figure 16. Using X-Ray projections to measure a left ventricle.

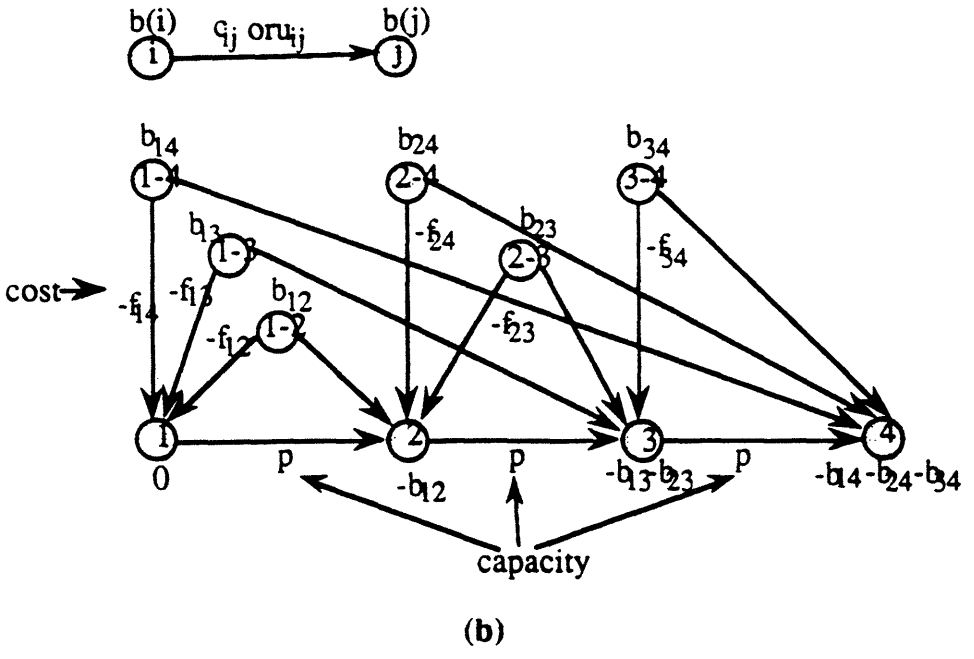
We can conceive of a cross section of the ventricle as a p x r binary matrix: a 1 in a position indicates that the corresponding segment allows blood to flow and a 0 indicates that it doesn't permit blood to flow. The angiocardiograms gives the cumulative intensity of the contrast agent in two planes which we can translate into row and column sums of the binary matrix. The problem is then to construct the binary matrix given its row and column sums. This problem is a special case the feasible flow problem that we discussed in Application 6.

Typically, the number of feasible solutions for such problems are quite large; and, these solutions might differ substantially. To constrain the feasible solutions, we might use certain facts from our experience that indicate that a solution is more likely to contain certain segments rather than others. Alternatively, we can use *a priori* information: for example, after some small time interval, the cross sections might resemble cross sections determined in a previous examination. Consequently, we might attach a probability p_{ij} that a solution will contain an element (i, j) of the binary matrix and might want to find a feasible solution with the largest possible cumulative probability. This problem is equivalent to a minimum cost flow problem.

Application 13. Optimal Loading of a Hopping Airplane (Gupta [1985] and Lawania [1990])

A small commuter airline uses a plane, with a capacity to carry at most p passengers, on a "hopping flight" as shown in Figure 17(a). The hopping flight visits the cities 1, 2, 3, ..., n, in a fixed sequence. The plane can pick up passengers at any node and drop them off at any other node. Let b_{ij} denote the number of passengers available at node i who want to go to node j, and let f_{ij} denote the fare per passenger from node i to node j. The airline would like to determine the number of passengers that the plane should carry between the various origins to destinations in order to maximize the total fare per trip while never exceeding the plane capacity.

(a)



(b)

Figure 17. Formulating the hopping plane flight problem as a minimum cost flow problem.

Figure 17(b) shows a minimum cost flow formulation of this hopping plane flight problem. The network contains data for only those arcs with nonzero costs and with finite capacities: any arc without an associated cost has a zero cost; any arc without an associated capacity has an infinite capacity. Consider, for example, node 1. Three types of passengers are available at node 1, those whose destination is node 2, node 3, or node 4. We represent these three types of passengers by the nodes 1-2, 1-3 and 1-4 with supplies b_{12}, b_{13}, and b_{14}. A passenger available at any such node, say 1-3, either boards the plane at its origin node by flowing through the arc (1-3, 1), and thus incurring a cost of $-f_{13}$ units, or never boards the plane which we represent by the flow through the arc (1-3, 3). This formulation correctly models the hopping plane application.

Application 14. Directed Chinese Postman Problem (Edmonds and Johnson [1973])

Leaving from his home post office, a postman needs to visit the households on each block in his route, delivering and collecting letters, and then returning to the post office. He would like to cover this route by travelling the minimum possible distance. Mathematically, this problem has the following form: Given a network G = (N, A) whose arcs (i, j) have an associated nonnegative length c_{ij}, we wish to identify a walk of minimum length that starts at some node (the post office), visits each arc of the network at least once, and returns to the starting node. This problem has become known as the *Chinese postman problem* because it was first discussed by a Chinese mathematician, K. Mei-Ko. The Chinese postman problem arises in other application settings as well; for instance, in patrolling streets by a police officer, routing of street sweepers and household refuse collection vehicles, fuel oil delivery to households, and the spraying of roads with sand during snow storms. In this application, we discuss the Chinese postman problem on directed networks.

In the Chinese postman problem on directed networks, we are interested in a closed (directed) walk that traverses each arc of the network at least once. The network need not contain any such walk! Figure 1 shows an example. The network contains the desired walk if and only if every node in the network is reachable from every other node; that is, it is *strongly connected*. Since we can determine in O(m) time the strong connectedness of a network, we shall henceforth assume that the network is strongly connected.

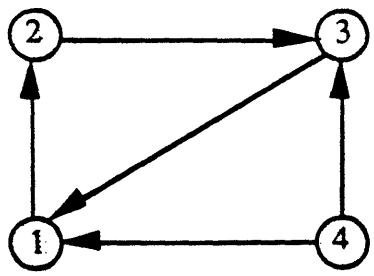


Figure 18. Network containing no feasible solution for the Chinese postman problem

In an optimal walk, a postman might traverse arcs more than once. The minimum length walk minimizes the sum of lengths of the repeated arc traversals. Let x_{ij} denote the number of times the postman traverses an arc (i, j) in a walk. Any walk of the postman must satisfy the following conditions:

$$\Sigma_{\{j\,:\,(i,\,j)\,\in\,A\}}\, x_{ij} - \Sigma_{\{j\,:\,(j,\,i)\,\in\,A\}}\, x_{ji} = 0, \text{ for all } i \in N, \tag{8a}$$

$$x_{ij} \geq 1, \text{ for all } (i, j) \in A. \tag{8b}$$

The constraints (8a) state that the postman enters a node the same number of times that he leaves it. The constraints (8b) state that the postman must visit each arc at least once. Any solution x satisfying (8a) and (8b) defines a postman's walk. We can construct such a walk in the following manner. We replace each arc (i, j) with flow x_{ij} with x_{ij} copies of the arc, each carrying a unit flow. Let A' denote the resulting arc set. Since the outflow equals inflow for each node in the flow x, once we have transformed the network, the outdegree of each node will equal its indegree. This implies that we can decompose the arc set A' into a set of at most m directed cycles. We can connect these cycles together to form a closed walk as follows. The postman starts at some node in one of the cycles, say W_1, and visits the nodes (and arcs) of W_1 in order until he either returns to the node he started from, or encounters a node that also lies in a directed cycle not yet visited, say W_2. In the former case, the walk is complete; and in the latter case, the postman visits cycle W_2 first before resuming his visit of the nodes in W_1. While visiting nodes in W_2, the postman follows the same policy, i.e., if he encounters a node lying on another directed cycle W_3 not yet visited, then he visits W_3 first before visiting the remaining nodes in W_2, and so on. We illustrate this method on a numerical example. Let A' be as indicated in Figure 19(a). This solution decomposes into three directed cycles W_1, W_2 and W_3. As shown in Figure 19(b), the postman starts at node a and visits the nodes in the following order: a-b-d-g-h-c-d-e-b-c-f-a.

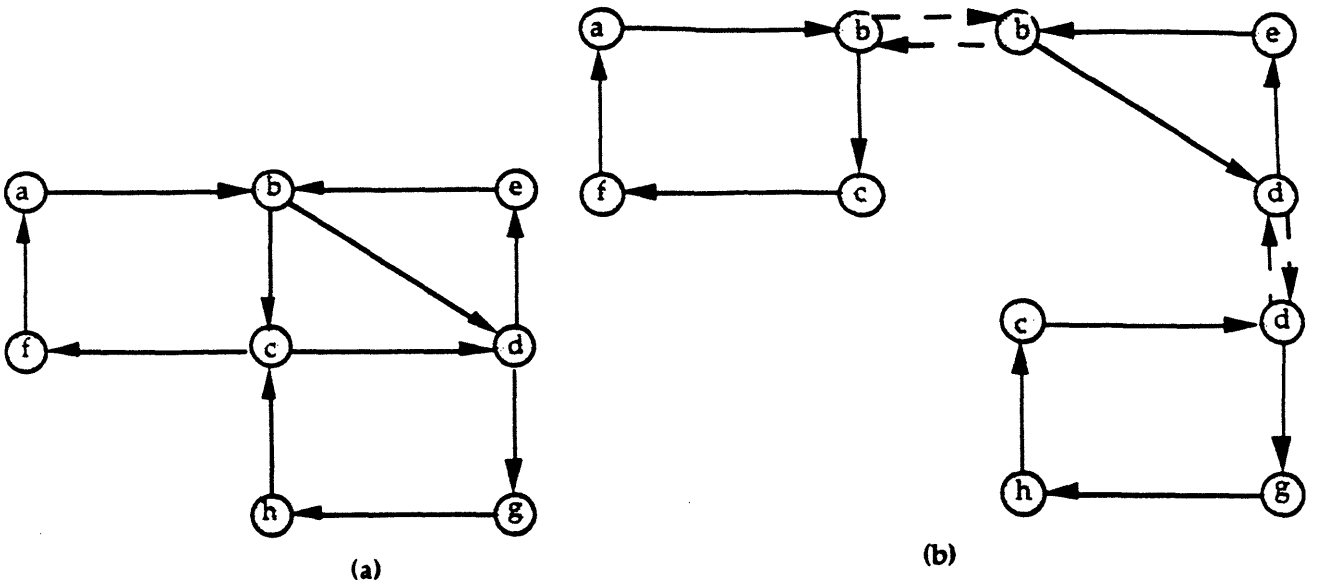


Figure 19. Constructing a closed walk for the postman.

This discussion shows that the solution x defined by a feasible walk for the postman satisfies (8), and, conversely, every feasible solution of (8) defines a walk of the postman. The length of a walk equals $\Sigma_{(i,j)\in A} c_{ij}x_{ij}$. Therefore, the Chinese postman problem seeks a solution x that minimizes $\Sigma_{(i,j)\in A} c_{ij}x_{ij}$, subject to the set of constraints (8). This problem is clearly an instance of the minimum cost flow problem.

Application 15. Models for Building Evacuation (Chalmet, Francis and Sounders [1982])

In large metropolitan areas, among the criteria they use to design large buildings, architects must ensure sufficient capabilities to evacuate buildings quickly, to respond, for example, to a fire, an earthquake, a toxic or natural gas leak, a power blackout, a bomb threat, or a civil defense emergency. As an aid to them in their design efforts, the architects would like to be able to develop an evacuation plan and assess the evacuation time for any particular design. We show how to model this building evacuation problem as a *dynamic flow problem* and solve it using a minimum cost flow algorithm. Whereas ordinary (static) network flow problems optimize the steady state flow in the network (e.g., identify the maximum steady state flow that can be sent from node s to node t per unit time), dynamic network flow problems optimize the transient flow in the network (e.g., identify the maximum total flow that can be sent from node s to node t within a given period of p units). In our description, we present a highly simplified version of the building evacuation problem. The reference for this application provides a more realistic description of the problem.

We first construct a static network for the building. The nodes of this network represent locations of the building such as work centers, offices, hallways, elevator stops, staircases, and building exits; the arcs represent the passages between these locations. Those locations of the building that house a significant number of people are source nodes in the network, and the building exits are the sink nodes. The supply of a source node equals an estimate of the number of people in the location that the node represents. The capacity of an arc is the number of people that can pass through the associated passage way per unit time. For example, if we anticipate that at most 60 persons per minute can pass by every point in a stairwell, and the

length of the time period in our model is 10 seconds, then the capacity of the stairwell is 60/6 = 10 units. We estimate the travel time of an arc and represent it as an integral number of time periods. For example, specifying two time periods for descending one floor in a stairwell means that we allow twenty seconds. Though the static model might have multiple sources and multiple sinks (exits), we can transform it into a model with single source and single sink by using the standard technique of introducing a *super-source* node and a *super-sink* node. Therefore, we assume that the network contains a single source s with a given supply, say B, and a single sink t.

The building evacuation problem is to identify the minimum period r within which the building can be completely evacuated. In other words, we wish to identify the minimum value of r so that the total flow from node s to node t is at least B. We can formulate this problem as a minimum cost flow problem in a *time-expanded network* G^P. We first choose a sufficiently large number p so that we can evacuate the building within p time periods. For a given network G = (N, A), we form the network G^P as follows. We make p copies $i^1, i^2, \ldots, i^P$ of each node i. Node i^k in the time-expanded network represents node i of the original network at time k. We include arc (i^k, j^l) of capacity u_{ij} in the time-expanded network whenever $(i, j) \in A$ and $l - k = \tau_{ij}$; the arc (i^k, j^l) in the time-expanded network represents the potential movement of a commodity from node i to node j in time τ_{ij}. It is easy to see that any static flow in G^P from source nodes $s^1, s^2, \ldots, s^P$ to the sink nodes $t^1, t^2, \ldots, t^P$ is equivalent to a dynamic flow in G, and vice-versa. The minimum time required to evacuate the building is the smallest index r satisfying the property that the maximum flow from the nodes $s^1, s^2, \ldots, s^r$ to the nodes $t^1, t^2, \ldots, t^r$ is at least B. We fomulate this problem as a minimum cost flow problem in the following manner. We introduce a node s* of supply B and join it to each node s^h, $1 \le h \le p$ using arcs of zero cost and sufficiently large capacity. We next introduce a node t* of demand B and join each node t^h, $1 \le h \le p$ to node t* using arcs of cost h and sufficiently large capacity. The minimum cost flow in this network would use the minimum index arcs (t^h, t^*) to send send flow into the sink node, and would take minimum time to evacuate the building. For a faster algorithm to solve this problem, see Burkard, Dlaska, and Klincz [1991].

Additional Applications

A complete list of additional applications of the minimum cost flow problem is too vast to mention here. A partial list of additional references is: (1) *distribution problems* (Glover and Klingman [1976]); (2) *racial balancing of schools* (Belford and Ratliff [1972]); (3) *scheduling with consecutive ones in columns* (Veinott and Wagner [1962]); (4) *linear programs with consecutive circular ones in rows* (Bartholdi, Ratliff and Orlin [1980]); (5) *the entrepreneur's problem* (Prager [1957]); (6) *optimal storage policy for libraries* (Evans [1984]); (7) *zoned warehousing* (Evans [1984]); (8) *allocation of contractors to public works* (Cheshire, McKinnon and Williams [1984]); (9) *phasing out capital equipment* (Daniel [1973]); (10) *the terminal assignment problem* (Esau and Williams [1966]); (11) *capacitated maximum spanning trees* (Garey and Johnson [1979]); (12) *caterer problem* (Jacobs [1954]); (13) *allocating receivers to transmitters* (Dantzig [1962]); (14) *faculty-course assignment* (Mulvey [1979]); (15) *automatic karotyping of chromosomes* (Tso et al. [1991]); (16) *just-in-time scheduling* (Elmaghraby [1978] and Levner and Nemirovsky [1991]); (17) *time-cost tradeoff in project management* (Fulkerson [1961] and Kelley [1961]); (18) *warehouse layout* (Francis and White [1976]); (19) *rectilinear distance facility location* (Cabot, Francis and Stary [1970]); (20) *dynamic lot-sizing* (Zangwill [1969]); (21) *multistage production-inventory planning* (Evans [1977]); (22)

mold allocation (Love and Vemuganti [1978]); (23) *a parking model* (Dirickx and Jennergren [1975]); (24) *the network interdiction problem* (Fulkerson and Harding [1977]); (25) *truck scheduling* (Gavish and Schweitzer [1974]); and (26) *optimal deployment of firefighting companies* (Denardo, Rothblum and Swersey [1988]); (27) *warehousing and distribution of a seasonal product* (Jewell [1957]); (28) *economic distribution of coal supplies in the gas industry* (Berrisford [1960]); (29) *upsets in round robin tournaments* (Fulkerson [1965]); (30) *optimal container inventory and routing* (Horn [1971]); (31) *distribution of empty rail containers* (White [1972]); (32) *optimal defense of a network* (Picard and Ratliff [1973]); (33) *telephone operator scheduling* (Segal [1974]); (34) *multifacility minimax location problem with rectilinear distances* (Dearing and Francis [1974]); (35) *cash management problems* (Srinivasan [1974]); (36) *multiproduct multifacility production-inventory planning* (Dorsey, Hodgson and Ratliff [1975]); (37) *"hub" and "wheel" scheduling problems* (Arisawa and Elmaghraby [1977]); (38) *warehouse leasing problem* (Lowe, Francis and Reinhardt [1979]); (39) *multi-attribute marketing models* (Srinivasan [1979]); (40) *material handling systems* (Maxwell and Wilson [1981]); (41) *microdata file merging* (Barr and Turner [1981]); (42) *determining service districts* (Larson and Odoni [1981]); (43) *control of forest fires* (Kourtz [1984]); (44) *allocating blood to hospitals from a central blood bank* (Sapountzis [1984]; (45) *market equilibrium problems* (Dafermos and Nagurney [1984]); (46) *automatic chromosome classifications* (Tso [1986]); (47) *city traffic congestion problem* (Zawack and Thompson [1987]); (48) *satellite scheduling* (Servi [1989]); (49) *determining k disjoint cuts in a network* (Wagner [1990]); and (50) *controlled rounding of matrices* (Cox and Ernst [1982]).

6. ASSIGNMENTS AND MATCHING

A *matching* in a graph G = (N, A) is a set of arcs with the property that every node is incident to at most one arc in the set; thus a matching induces a pairing of (some of) the nodes in the graph using the arcs in A. In a matching, each node is matched with at most one other node, and some nodes might not be matched with any other node. Suppose that each arc (i, j) in the network has an associated cost c_{ij}. The *matching problem* seeks a matching that minimizes the total cost of the arcs in the matching. Matching problems on a bipartite graphs (i.e., on a graph $G = (N_1 \cup N_2, A)$ where $N = N_1 \cup N_2$ and each arc $(i, j) \in A$ has $i \in N_1$ and $j \in N_2$) are called *bipartite matching problems* and those not on necessarily bipartite graphs are called *nonbipartite matching problems*. There are two further ways of categorizing matching problems: *cardinality matching problems*, that maximize the number of pairs of nodes matched, and *weighted matching problems* that maximize or minimize the weight of the matching. The weighted matching problem on a bipartite graph is also known as the *assignment problem*.

The matching problem arises in many different problem settings since we often wish to find the best way to pair objects or people together to achieve some desired goal. We first describe several such direct applications, followed by some nondirect applications of the assignment problems.

Application 16. Personnel Assignment (Machol [1970], Ewashko and Dudding [1971]), Meggido and Tamir [1978]).

In many different problem contexts, we wish to assign people to objects: for example, to jobs, machines, rooms, or each other. Each assignment has a "value" and we wish to make the assignments so

that we maximize the sum of these values. To illustrate the range of these contexts, in this application we consider six different applications relating to personnel assignment.

I. A firm has hired n graduates to fill n vacant jobs. Based upon aptitude tests, college grades, and letters of recommendation, the firm has assigned a *proficiency index* u_{ij} for placing candidate i in job j. The objective is to identify an assignment that maximizes the total proficiency score over all jobs. This problem is clearly an application of the assignment problem.

II. A swimming coach must select from his or her eight best swimmers a medley relay team of four, each of whom will then swim one of the following four strokes: back, breast, butterfly, and free-style. The coach knows the time of each swimmer in each stroke. The problem is to identify the team of the four best swimmers out of the eight that are available. The sum of times obtained by optimally matching four out of the eight swimmers to the four strokes gives the minimum feasible relay time, and the corresponding team is the best team. In this version of the assignment problem, $|N_1| > |N_2|$; nevertheless, by adding "dummy nodes," we can easily transform this problem easily into an equivalent one in which both the node sets N_1 and N_2 have the same size.

III. In the armed forces, many men and women are qualified to perform specific jobs, or postings. The armed forces would like to assign the service personnel to postings in order to minimize moving costs. General rules specify the needed qualifications of the personnel for the postings and identify jobs that need to be filled. Policy rules determine allowable assignments that reflect job qualifications and personnel requirements. For an allowable assignment (i.e., satisfying the general and policy rules), the *posting cost* is the dollar cost of moving the individual, his or her family, and his or her belongings to the new residence. In this case, the assignment problem would find an allowable assignment that minimizes the total posting cost.

IV. During World War II, the Royal Air Force (RAF) of Britain contained many pilots from foreign countries who spoke different languages and had different levels of training. The RAF had to assign two pilots to each plane, always assigning pilots with compatible languages and training to the same plane. For obvious reasons, the RAF wanted to fly as many planes as possible. We can formulate this problem as a maximum cardinality matching problem by defining a graph whose nodes represent pilots and with two nodes joined by an arc if the corresponding pilots are compatible.

V. A hostel manager wants to assign pairs of roommates to rooms of his or her hostel. The nationality, religion, cultural background, and hobbies determine pairs of roommates that are compatible. In this scenerio, the problem of finding the maximum number of compatible pairs is a maximum cardinality matching problem.

VI. Suppose that an airline wishes to divide its 2p airplane pilots, linearly ordered by seniority (with no ties), into m teams each containing a captain and a first officer. The captain of each team must have seniority over the first officer. Each pilot i has a measure, α_i, of his or her effectiveness as a captain and another, β_i, measure of his effectiveness as a first officer. We seek an assignment of pilots to teams that will maximize the total measure of effectiveness summed over all the teams. This problem is an instance

of the maximum weight matching problem: we represent each pilot as a node and define the cost of an arc (i, j) as $\alpha_j + b_i$, if pilot j is more senior than pilot i, and as $\alpha_i + \beta_j$ if pilot i is more senior than pilot j.

Application 17. Pairing Stereo Speakers (Mason and Philpott [1988])

As a part of its manufacturing process, a manufacturer of stereo speakers must pair individual speakers before it can sell them as a set. The performance of the two speakers depends upon their frequency response. In order to measure the quality of the pairs, the company generates matching coefficients for each possible pair. It calculates these coefficients by summing the absolute differences between the responses of the two speakers at twenty discrete frequencies, thus giving a matching coefficient value between 0 and 30,000. Bad matches yield a large coefficient, and a good pairing produces a low coefficient.

The manufacturer typically uses two different objectives in pairing the speakers: (i) finding as many pairs as possible whose matching coefficients do not exceed a specification limit; or (ii) pairing speakers within specification limits in order to minimize the total sum of the matching coefficients. The first objective minimizes the number of pairs outside of specification, and so the number of speakers that the firm must sell at a reduced price. This model is an application of the nonbipartite cardinality matching problem on an undirected graph: the nodes of this graph represent speakers and arcs join two nodes if the matching coefficients of the corresponding speakers are within the specification limit. The second model is an application of the nonbipartite weighted matching problem.

Application 18. Locating Objects in Space (Brogan [1989])

This application concerns locating objects in space. To identify an object in (three-dimensional) space, we could use two infrared sensors, located at geographically different sites. Each sensor provides an angle of sight of the object and, hence, the line on which the object must lie. The unique intersection of the two lines provided by the two sensors (provided that the two sensors and the object are not co-linear) determines the unique location of the object in space.

Consider now the situation in which we wish to determine the locations of p objects using two sensors. The first sensor would provide us with a set of lines $L_1, L_2, ..., L_p$ for the p objects and the second sensor would provide us a different set of lines $L_1', L_2', ..., L_p'$. To identify the location of the objects—using the fact that if two lines correspond to the same object, then the lines intersect one-another—we need to match the lines from the first sensor to the lines from the second sensor. In practice, two difficulties limit the use of this approach. First, a line from a sensor might intersect more than one line from the other sensor, so the matching is not unique. Second, two lines corresponding to the same object might not intersect because the sensors make measurement errors in determining the angle of sight. We can overcome this difficulty in most situations by formulating this problem as an assignment problem.

In the assignment problem, we wish to match the p lines from the first sensor with the p lines from the second sensor. We define the cost c_{ij} of the assignment (i, j) as the minimum Euclidean distance between the lines L_i and L_j'. We can determine c_{ij} using standard calculations from geometry. If the lines L_i and L_j' correspond to the same object, then c_{ij} would be close to zero. An optimal solution of the assignment problem would provide an excellent matching of the lines. Simulation studies have found that

in most circumstances, the matching produced by the assignment problem defines the correct location of the objects.

Application 19. Matching Moving Objects (Brogan [1989] and Kolitz [1991])

In several different application contexts, we might wish to estimate the speeds and the directions of movement of a set of p objects (e.g., enemy fighter planes, missiles) that are moving in space. Using the method described in the preceding application, we can determine the location of the objects at any point in time. One plausible way to estimate the objects movement directions and speeds is to take two snapshots of the objects at two distinct times and then to match one set of points with the other set of points. If we correctly match the points, then we can assess the speed and direction of movement of the objects. As an example, consider Figure 20 which denotes the objects at time 1 by squares and the objects at time 2 by circles.

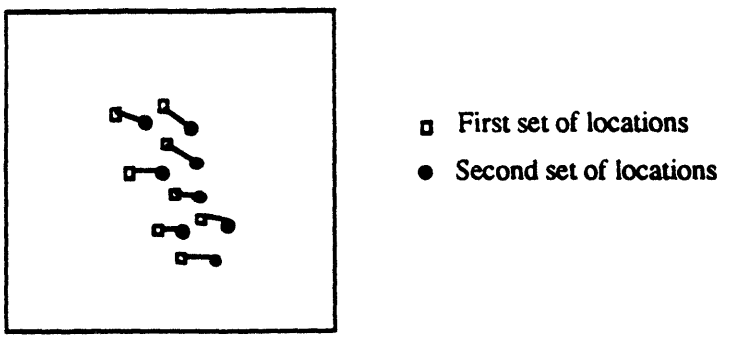


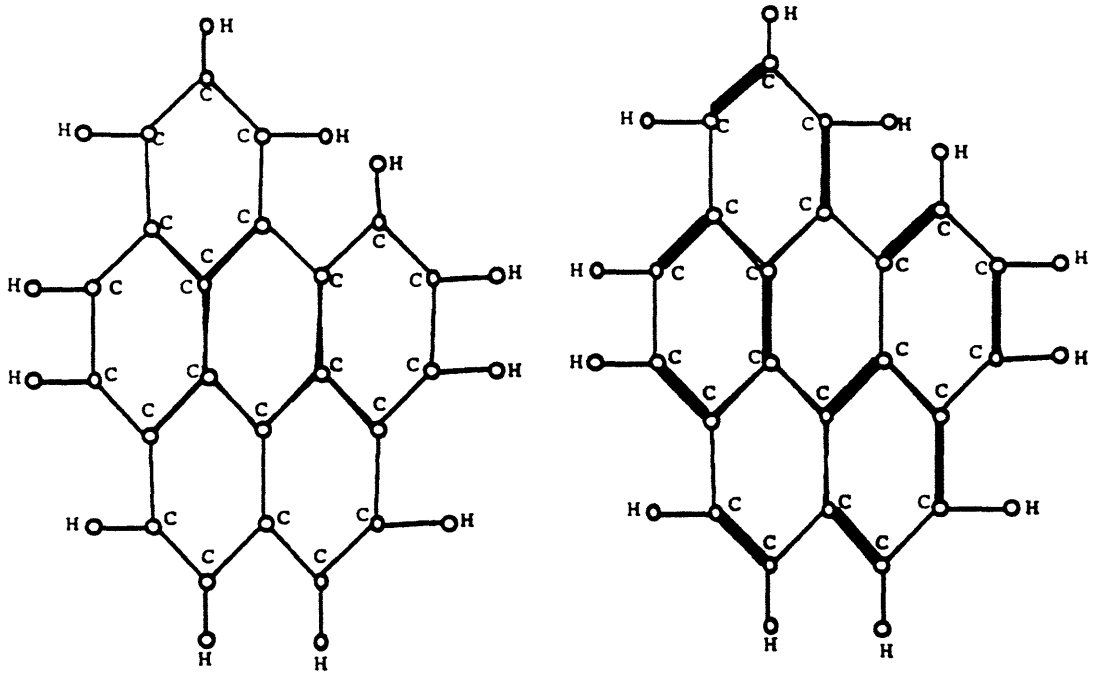
Figure 20. Two snapshots of a set of 8 objects.

Let (x_i, y_i, z_i) denote the coordinates of object i at time 1 and (x_i', y_i', z_i') denote the coordinates of the same object at time 2. We could match one set of points with the other set of points in many ways. Minimizing the sum of the squared Euclidean distances between the matched points is reasonable in this scenario because it attaches a higher penalty to larger distances. If we take the snapshots of the objects at two times that are sufficiently close to each other, then the optimal assignment will often match the points correctly. In this application of the assignment problem, we let $N_1 = \{1, 2, ..., p\}$ denote the set of objects at time 1, let $N_2 = \{1', 2', ..., p'\}$ denote the set of objects at time 2, and we define the cost of an arc (i, j') as $[(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2]$. The optimal assignment in this graph will specify the desired matching of the points. From this matching, we obtain an estimate of the movement directions and velocities of the individual objects.

Application 20. Determining Chemical Bonds (Dewar and Longuet-Higgins [1952])

Matching problems arise in the field of chemistry as chemists attempt to determine the possible atomic structures of various molecules. Figure 21(a) specifies the partial chemical structure of a molecule of some hydrocarbon compound. The molecule contains carbon atoms (denoted by nodes with the letter "C" next to them) and hydrogen atoms (denoted by nodes with the letter "H" next to them). Arcs denote bonds between atoms. The bonds between the atoms, which can be either single or double bonds, must satisfy the

"valency requirements" of all the nodes. (The valency of an atom is the sum of its bonds.) Carbon atoms must have a valency of 4 and hydrogen atoms a valency of 1.



(a) (b)

Figure 21. Determining the chemical structure of a hydrocarbon.

In the partial structure shown in Figure 21(a), each arc depicts a single bond and, consequently, each hydrogen atom has a valency of 1, but each carbon atom has a valency of only 3. We would like to determine which pairs of carbon atoms to connect by a double bond so that each carbon atom has valency 4. We can formulate this problem of determining some feasible structure of double bonds to determining whether or not there exists a maximum cardinality matching in which all nodes are matched, in the network obtained by deleting the hydrogen atoms and those carbon atoms with valency 4. Figure 21(b) gives one feasible bonding structure of the compound; the bold lines in this network denote double bonds between the atoms.

Additional Applications

Additional applications of the matching problems include: (1) *rewiring of typewriters* (Machol [1961]); (2) *dual completion of oil wells* (Devine [1973]); (3) *optimal depletion of inventory* (Derman and Klein [1959]); (4) *scheduling of parallel machines* (Horn [1973]); (5) *solving shortest path problems in directed and undirected networks* (Hoffman and Markowitz [1963] and Edmonds [1967]); (6) *undirected Chinese postman problem* (Edmonds and Johnson [1973]); (7) *discrete location problems* (Francis and White [1976]); (8) *two-processor scheduling* (Fujii, Kasami and Ninomiya [1969]); (9) *determining the rank of a matrix* (Anderson [1975]); (10) *vehicle and crew scheduling* (Carraresi and Gallo [1984]); and (11) *making matrices optimally sparse* (Hoffman and McCormick [1984]).

7. MINIMUM SPANNING TREE PROBLEM

As we noted previously, a spanning tree is a tree (i.e., a connected acyclic graph) that spans (touches) all the nodes of an undirected network. The cost of a spanning tree is the sum of the costs (or lengths) of its arcs. In the minimum spanning tree problem, we wish to identify a spanning tree of minimum cost (or length). Minimum spanning tree problems generally arise in one of two ways, directly or indirectly. In some *direct* applications, we wish to connect a set of points using the least cost or least length collection of arcs. Frequently, the points represent physical entities that need to be connected to each other. In *indirect* applications, we either (i) wish to connect some set of points using a measure of performance that on the surface bears little resemblance to the minimum spanning tree objective (sum of arc costs), or (ii) the problem itself bears little resemblance to an "optimal tree" problem—these instances often require creativity in modeling so that they become a minimum spanning tree problem. In this section, we consider several direct and indirect applications.

Application 21. Designing Physical Systems (Borůvka [1926], Prim [1957], Loberman and Weinberger [1957], and Dijkstra [1959])

The design of physical systems can be a complex task involving an interplay between performance objectives (such as throughput and reliability), design costs and operating economics, and available technology. In many settings, the major criterion is fairly simple: we need to design a network that will connect geographically dispersed system components or that will provide the needed infrastructure that will permit users to communicate with each other. In many of these settings, the system need not have any redundancy, so we are interested in the simplest possible connection, namely, a spanning tree. This type of application arises in numerous problem settings including the building of certain types of highways, computer networks, leased-line telephone networks, or railroads, or the installation of cable television lines, or high voltage electrical power transmission lines. The following applications are a few other problem settings in which this type of minimum spanning tree problem arises.

I. Connect terminals in cabling the panels of electrical equipment. How should we wire the terminals to use the least possible length of wire?

II. Constructing a pipeline network to connect a number of towns using the smallest possible total length of pipeline.

III. Linking isolated villages in a remote region, that are connected by roads, but not yet by telephone service. In this instance, we wish to determine along which stretches of roads we should place telephone lines, using the minimum possible total miles of the lines, to link every pair of villages.

IV. Constructing a digital computer system, composed of high frequency circuitry, when it is important to minimize the lengths of wires between different components to reduce both capacitance and delay line effects. Since all components must be connected, we obtain a spanning tree problem.

V. Connecting a number of computer sites by high speed lines. Each line is available for leasing at a certain monthly cost, and we wish to determine a configuration that connects all the sites at the least possible cost.

Each of these applications is a direct application of the minimum spanning tree problem. We next describe several indirect applications.

Application 22. Measuring Homogeneity of Bimetallic Objects (Shier [1982], and Filliben, Kafadar and Shier [1983])

This application shows how a minimum spanning tree problem can be used to determine the degree to which a bimetallic object is homogeneous in composition. To use this approach, we measure the composition of the bimetallic object at a set of sample points. We then construct a network with nodes corresponding to the sample points and with an arc connecting physically adjacent sample points. We assign a cost with each arc (i, j) equal to the product of the physical (Euclidean) distance between the sample points i and j and a homogeneity factor between 0 and 1. This homogeneity factor is 0 if the composition of the corresponding samples is exactly alike, and is 1 if the composition is very different; otherwise, it is a number between 0 and 1. Note that this measure gives greater weight to two points if they are different and are far apart. The cost of the minimum spanning tree is a measure of the homogeneity of the bimetallic object. The cost of the tree is 0 if all the sample points are exactly alike, and high cost values imply that the material is quite nonhomogeneous.

Application 23. Reducing Data Storage (Kang et al. [1977])

In several different application contexts, we wish to store data specified in the form of a two-dimensional array more efficiently than storing all the elements of the array (that is, to save memory space). We assume that the rows of the array have many similar entries and differ only at a few places. One such situation arises in the sequence of aminoacids in a protein found in the mitochondria of different animals and higher plants.

Since the entities in the rows are similar, one approach for saving memory is to store one row, called the *reference row*, completely, and to store only the differences between some of the rows so that we can derive each row from these differences and the reference row. Let c_{ij} denote the number of different entries in rows i and j; that is, if we are given row i, then by making c_{ij} changes to the entries in this row we can obtain row j, and vice-versa. Suppose that the array contains four rows, represented by R_1, R_2, R_3 and R_4, and we decide to treat R_1 as a reference row. Then one plausible solution is to store the differences between R_1 and R_2, R_2 and R_4, and R_1 and R_3. Clearly, from this solution, we can obtain rows R_2 and R_3 by making c_{12} and c_{13} changes to the elements in row R_1. Having obtained row R_2, we can make c_{24} changes to the elements of this row to obtain R_4.

It is easy to see that it is sufficient to store differences between those rows that correspond to arcs of a spanning tree. These differences permit us to obtain each row from the reference row. The total storage requirement for a particular storage scheme will be the length of the reference row (which we can take as the row with the least amount of data) plus the sum of the differences between the rows. Therefore, a minimum spanning tree would provide the least cost storage scheme.

Application 24. All Pairs Minimax Path Problem (Hu [1961])

In a network G = (N, A) with arc costs c_{ij}'s, we define the *value* of a path P from node k to node *l* as the maximum cost arc in P. The *all pairs minimax path problem* requires that we determine, for every pair [k, *l*] of nodes, a minimum value path from node k to node *l*. We show how to solve the all pairs minimax path problem on an undirected graph by solving a single minimum spanning tree problem.

The minimax path problem arises in a variety of situations. As an example, consider a spacecraft that is about to enter the earth's atmosphere. The craft passes through different pressure and temperature zones that we can represent by arcs of a network. It needs to fly along a trajectory that will bring the craft to the surface of the earth while keeping the maximum temperature to which the surface of the craft is exposed as low as possible. As an alternative, we might wish to select a path that will minimize the maximum deceleration during the descent. Other possible examples of the minimax path problem are: (i) in traveling through a desert, we want to minimize the length of the longest stretch between rest areas; and (ii) in traveling in a wheelchair, a person might wish to minimize the maximum ascent along the path segments.

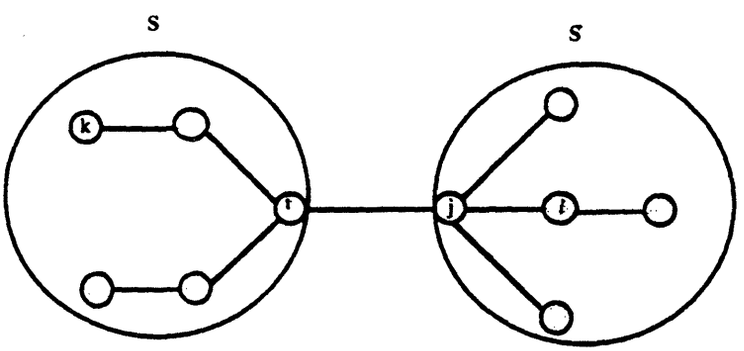


Figure 22. Cut formed by deleting the arc (p, q) from a spanning tree.

We now transform the all pairs minimax path problem into a minimum spanning tree problem. Let T^* be a minimum spanning tree of G. Let P denote the unique path in T^* between a node pair [p, q] and let (i, j) denote the maximum cost arc in P. Observe that the value of the path P is c_{ij}. By deleting arc (i, j) from T^*, we partition the node set N into two subsets and therefore define a cut [S, $\bar{S}$] with $i \in S$ and $j \in \bar{S}$ (see Figure 22). The optimality of the tree T^* implies that this cut satisfies the following property:

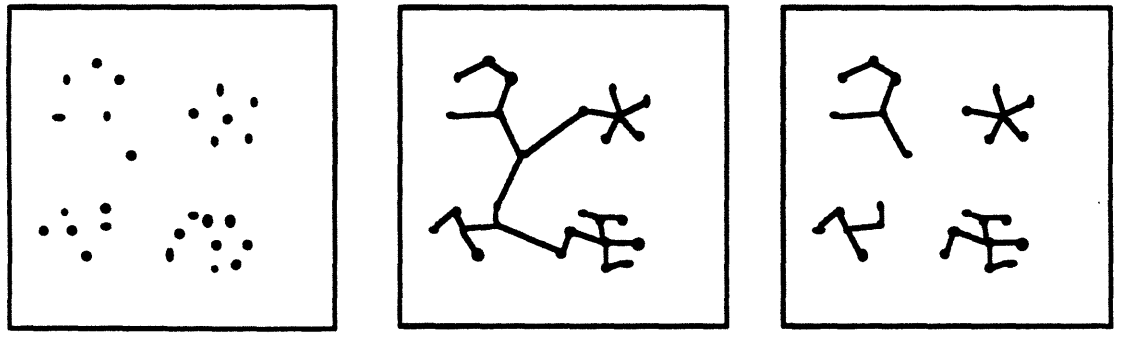
$$c_{ij} \le c_{kl} \text{ for each arc } (k,l) \in [S, \bar{S}], \tag{9}$$

for otherwise by replacing the arc (i, j) by an arc (k, *l*) we can obtain a spanning tree of smaller cost. Now, consider any path P' from node p to node q. This path must contain at least one arc (k, *l*) in [S, $\bar{S}$]. The conditions (9) implies that the value of the path P' will be at least c_{ij}. Since c_{ij} is the value of the path P, P must be a minimum value path from node k to node *l*. This observation establishes the fact that the unique path between any pair of nodes in T^* is the minimum value path between that pair of nodes.

Application 25. Cluster Analysis (Gower and Ross [1969], and Zahn [1971])

In this application, we describe the use of spanning tree problems to solve a class of problems that arises in the context of cluster analysis. The essential issue in cluster analysis is to partition a set of data into "natural groups"; the data points within a particular group of data, or a cluster, should be more "closely related" to each other than the data points not in that cluster. Cluster analysis is important in a variety of disciplines that rely upon empirical investigations. Consider, for example, an instance of a cluster analysis arising in medicine. Suppose we have data on a set of 350 patients, measured with respect to 18 symptoms. Suppose, further, that a doctor has diagnosed all of these patients as having the same disease which is not well understood. The doctor would like to know if he or she can develop a better understanding of this disease by categorizing the symptoms into smaller groupings using cluster analysis. Doing so might permit the doctor to find more natural disease categories to replace or subdivide the original disease.

Suppose we are interested in finding a partition of a set of n points in two-dimensional Euclidean space into clusters. A popular method for solving this problem is by using Kruskal's algorithm for solving the minimum spanning tree problem (see Lawler [1976]). Kruskal's algorithm maintains a forest (i.e., a collection of node-disjoint trees) and adds arcs in a nondecreasing order of their costs. We can regard the components of the forest at intermediate steps as different clusters. These clusters are often excellent solutions for the clustering problem and, moreover, we can obtain them very efficiently. Kruskal's algorithm can be thought of providing n partitions: the first partition contains n clusters, each cluster containing a single point, and the last partition contains just one cluster containing all the points. Alternatively, we can obtain n partitions by starting with a minimum spanning tree and deleting tree arcs one by one in nonincreasing order of their lengths. We illustrate the later approach using an example. Consider a set of 27 points shown in Figure 23(a). Suppose that Figure 23(b) shows a minimum spanning tree for these points. Deleting the three largest length arcs from the minimum spanning tree gives a partition with four clusters shown in Figure 23(c).



(a) (b) (c)

Figure 23. Identifying clusters by finding a minimum spanning tree.

Analysts can use the information obtained from the preceding analysis in several ways. The procedure we have described yields n partitions. Out of these, we might select the "best" partition by simple visualization or by defining an appropriate objective function value. A good choice of the objective

function depends upon the underlying features of the particular clustering application. We might note that this analysis is not limited to points in two-dimensional space; we can easily extend it to multi-dimensional space if we define inter-point distances appropriately.

Additional Applications

Some additional applications of the minimum spanning tree problem arise in the following situations: (1) *optimal message passing* (Prim [1957]); (2) *solving a special case of the traveling salesman problem* (Gilmore and Gomory [1964]); (3) *chemical physics* (Stillinger [1967]); (4) *Lagrangian relaxation techniques* (Held and Karp [1970]); (5) *network reliability analysis* (Van Slyke and Frank [1972]); (6) *pattern classification* (Dude and Hart [1973]); (7) *picture processing* (Osteen and Lin [1974]); and (8) *network design* (Magnanti and Wong [1984]). The survey paper of Graham and Hell [1985] provides references for additional applications of the minimum spanning tree problem.

8. SUMMARY

In this paper, we describe several applications of the following network optimization problems: the shortest path problem, the maximum flow problem, the minimum cost flow problem, the assignment and matching problems, and the minimum spanning tree problem. We have adapted these applications from Ahuja, Magnanti and Orlin [1993] which describes many more applications of these problems and as well as for the convex cost flow problem, generalized network flow problem, and the multicommodity flow problem. Additional sources of references on applications are the survey papers of Bennington [1974], Glover and Klingman [1976], Bodin et al. [1983], Aronson [1989], and Glover, Klingman and Phillips [1990]. The book by Gondran and Minoux [1984] also describes a variety of applications of network flow problems.

This paper provides network flow models for optimization problems arising in engineering, management and computer science, manufacturing, production and inventory planning, scheduling, communication, distribution and transportation systems, physical and medical sciences, social sciences and public policy, defence, and applied mathematics. The coverage of applications in this paper, as broad as it is, is far from being exhaustive. The literature contains many other applications, some that amplify on the themes we have presented and some that treat new problem domains. We hope that our coverage in this paper gives an appreciation for the power of network optimization as a modeling tool that embraces many application domains, and in doing so, helps to affirm the power of operations research as a field that has an important impact on practice.

REFERENCES

ABDALLAOUI, G. 1987. Maintainability of a grade structure as a transportation problem. *Journal of the Operational Research Society* 38, 367-369.

AHUJA, R.K. 1986. Algorithms for the minimax transportation problem. *Naval Research Logistics Quarterly* 33, 725-740.

AHUJA, R.K., T. L. MAGNANTI, AND J.B. ORLIN. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., New Jersey. (In Press)

ANDERSON, W.N. 1975. Maximum matching and the rank of a matrix. *SIAM Journal on Applied Mathematics* **28**, 114-123.

ARISAWA, S., AND S.E. ELMAGHRABY. 1977. The "hub" and "wheel" scheduling problems. *Transportation Science* **11**, 124-146.

ARONSON, J. E. 1989. A survey of dynamic network flows. *Annals of Operations Research* **20**, 1-66.

BACHARACH, M. 1966. Matrix rounding problems. *Management Science* **9**, 732-742.

BALAKRISHNAN, A., T.L. MAGNANTI, AND R.T. WONG. 1989. Models for capacity expansion in local excess telecommunication networks. Working Paper #3048-89-MS, Sloan School of Management, MIT, Cambridge.

BARR, R.S., AND J.S. TURNER. 1981. Microdata file merging through large scale network technology. *Mathematical Programming Study* **15**, 1-22.

BARTHOLDI, J.J., J.B. ORLIN AND H.D. RATLIFF. 1980. Cyclic scheduling via integer programs with circular ones. *Operations Research* **28**, 1074-1085.

BELFORD, P.C., AND H.D. RATLIFF. 1972. A network-flow model for racially balancing schools. *Operations Research* **20**, 619-628.

BELLMAN, R. 1958. On a routing problem. *Quarterly of Applied Mathematics* **16**, 87-90.

BENNINGTON, G.E. 1974. Applying network analysis. *Industrial Engineering* **6**, 17-25.

BERGE, C., AND A. GHOUILA-HOURI. 1962. *Programming, Games and Transportation Networks.* John Wiley & Sons.

BERRISFORD, H.G. 1960. The economic distribution of coal supplies in the gas industry: An application of the linear programming transport theory. *Operations Research Quarterly* **11**, 139-150.

BODIN, L.D., B.L. GOLDEN, A.A. ASSAD, AND M.O. BALL. 1983. Routing and scheduling of vehicles and crews : The state of the art. *Computers and Operations Research* **10**, 69-211.

BORŮVKA, O. 1926. Příspevěk k řešení otázky ekonomické stavby elektrovodních sití. *Elektrotechnický obzor* **15**, 153-154.

BROGAN, W.L. 1989. Algorithm for ranked assignments with application to multiobject tracking. *Journal of Guidance*, 357-364.

BURKARD, R.E., K. DLASKA, B. KLINCZ. 1991. The quickest flow problem. Report No. 188, Institut fur Mathematik, Technische Universitat Graz, Austria.

CABOT, A.V., R.L. FRANCIS, AND M.A. STARY. 1970. A network flow solution to a rectilinear distance facility location problem. *AIIE Transactions* **2**, 132-141.

CARRARESI, P., AND G. GALLO. 1984. Network models for vehicle and crew scheduling. *European Journal of Operational Research* **16**, 139-151.

CHALMET, L.G., R.L. FRANCIS AND P.B. SAUNDERS. 1982. Network models for building evacuation. *Management Science* **28**, 86-105.

CHESHIRE, M., K.I.M. McKINNON, AND H.P. WILLIAMS. 1984. The efficient allocation of private contractors to public works. *Journal of Operational Research Quarterly* **35**, 705-709.

CLARK, J.A., AND N.A.J. HASTINGS. 1977. Decision networks. *Operational Research Quarterly* **20**, 51-68.

CORMEN, T.H., C.L. LEISERSON, AND R.L. RIVEST. 1990. *Introduction to Algorithms.* The MIT Press and McGraw-Hill Book Company.

COX, L. H., AND L.R. ERNST. 1982. Controlled rounding. *INFOR* **20**, 423-432.

DAFERMAS, S., AND A. NAGURNEY. 1984. A network formulation of market equilibrium problems and variational inequalities. *Operations Research Letters* **5**, 247-250.

DANIEL, R.C. 1973. Phasing out capital equipment. *Operations Research Quarterly* **24**, 113-116.

DANTZIG, G.B. 1962. *Linear Programming and Extensions.* Princeton University Press, Princeton, NJ.

DANTZIG, G.B., AND D.R. FULKERSON. 1954. Minimizing the number of tankers to meet a fixed schedule. *Naval Research Logistics Quarterly* **1**, 217-222.

DEARING, P.M., AND R.L. FRANCIS. 1974. A network flow solution to a multifacility minimax location problem involving rectilinear distances. *Transportation Science* **8**, 126-141.

DENARDO, E.V., U.G. ROTHBLUM, AND A.J. SWERSEY. 1988. A transportation problem in which costs depend on the order of arrival. *Management Science* **34**, 774-783.

DERMAN, C., AND M. KLEIN. 1959. A note on the optimal depletion of inventory. *Management Science* **5**, 210-214.

DEVINE, M. V. 1973. A model for minimizing the cost of drilling dual completion oil wells. *Management Science* **20**, 532-535.

DEWAR, M.S.J., AND H.C. LONGUET-HIGGINS. 1952. The correspondence between the resonance and molecular orbital theories. *Proceedings of the Royal Society of London* **A214**, 482-493.

DIJKSTRA, E. 1959. A note on two problems in connexion with graphs. *Numeriche Mathematics* **1**, 269-271.

DIRICKX, Y.M.I., AND L.P. JENNERGREN. 1975. An analysis of the parking situation in the downtown area of West Berlin. *Transportation Research* **9**, 1-11.

DORSEY, R.C., T.J. HODGSON AND H.D. RATLIFF. 1975. A network approach to a multifacility multi-product production scheduling problem without backordering. *Management Science* **21**, 813-822.

DRESS, A.W.M., AND T.F. HAVEL. 1988. Shortest path problems and molecular conformation. *Discrete Applied Mathematics* **19**, 129-144.

DUDE, R.O., AND P.E. HART. 1973. *Pattern Classification and Science Analysis.* Wiley , New York.

EDMONDS, J. 1967. An introduction to matching. Mimeographed notes, Engineering Summer Conference, The University of Michigan, Ann Arbor.

EDMONDS, J., AND E.L. JOHNSON. 1973. Matching, Euler tours and the Chinese postman. *Mathematical Programming* **5**, 88-124.

ELMAGHRABY, S.E. 1978. *Activity Networks : Project Planning and Control by Network Models.* Wiley-Interscience.

ESAU, L.R., AND K.C. WILLIAMS. 1966. On teleprocessing system design II. *IBM Systems Journal* **5**, 142-147.

EVANS, J.R. 1977. Some network flow models and heuristics for multiproduct production and inventory planning. *AIIE Transactions* **9**, 75-81.

EVANS, J.R. 1984. The factored transportation problem. *Management Science* **30**, 1021-1024.

EWASHKO, T.A., AND R.C. DUDDING. 1971. Application of Kuhn's Hungarian assignment algorithm to posting servicemen. *Operations Research* **19**, 991.

FARLEY, A.R. 1980. Levelling terrain trees : A transshipment problem. *Information Processing Letters* **10**, 189-192.

FEDERGRUEN, A., AND H. GROENEVELT. 1986. Preemptive scheduling on uniform machines by network flow techniques. *Management Science* **32**, 341-349.

FILLIBEN, J.J., K. KAFADAR, AND D.R. SHIER. 1983. Testing for homogenity of two-dimensional surfaces. *Mathematical Modelling* **4**, 167-189.

FORD, L.R., AND D.R. FULKERSON. 1958. Constructing maximal dynamic flows from static flows. *Operations Research* **6**, 419-433.

FORD, L.R., AND S.M. JOHNSON. 1959. A tournament problem. *The American Mathematical Monthly* **66**, 387-389.

FORD, L.R., AND D.R. FULKERSON. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

FRANCIS, R.L., AND J.A. WHITE. 1976. *Facility Layout and Location*. Prentice-Hall, Englewood Cliffs, NJ.

FRANK, C.R. 1965. A note on the assortment problem. *Management Science* **11**, 724-726.

FUJII, M., T. KASAMI, AND K. NINOMIYA. 1969. Optimal sequencing of two equivalent processors. *SIAM Journal on Applied Mathematics* **17**, 784-789. Erratum, same journal **18**, 141.

FULKERSON, D.R. 1961. A network flow computation for project cost curve. *Managemence Science* **7**, 167-178.

FULKERSON, D.R. 1965. Upsets in a round robin tournaments. *Canadian Journal of Mathematics* **17**, 957-969.

FULKERSON, D.R. 1966. Flow networks and combinatorial operations research. *American Mathematical Monthly* **73**, 115-138.

FULKERSON, D.R., AND G.C. HARDING. 1977. Maximizing the minimum source-sink path subject to a budget constraint. *Mathematical Programming* **13**, 116-118.

GALLO, G., M.D. GRIGORIADIS, AND R.E. TARJAN. 1989. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing* **18**, 30-55.

GAREY, M.S., AND D.S. JOHNSON. 1979. *Computers and Intractability : A Guide to the Theory of NP-Completeness*. W.H. Freeman.

GAVISH, B., AND P. SCHWEITZER. 1974. An algorithm for combining truck trips. *Transportation Science* **8**, 13-23.

GILMORE, P.C., AND R.E. GOMORY. 1964. Sequencing a one state-variable machine : A solvable case of the travelling salesman problem. *Operations Research* **12**, 655-679.

GLOVER, F., AND D. KLINGMAN. 1976. Network applications in industry and government. *AIIE Transactions* **9**, 363-376.

GLOVER, F., D. KLINGMAN, AND N. PHILLIPS. 1990. Netform modeling and applications. *Interfaces* **20**, 7-27.

GOETSCHALCKX, M., AND H.D. RATLIFF. 1988. Order picking in an aisle. *IIE Transactions* **20**, 53-62.

GOLDMAN, A.J., AND G.L. NEMHAUSER. 1967. A transport improvement problem transformable to a best-path problem. *Transportation Science* **1**, 295-307.

GOLITSCHEK, M.V., AND H. SCHNEIDER. 1984. Applications of shortest path algorithms to matrix scalings. *Numerische Mathematik* **44**, 111-126.

GONDRAN, M., AND M. MINOUX. 1984. *Graphs and Algorithms*. Wiley-Interscience.

GOWER, J.C., AND G.J.S. ROSS. 1969. Minimum spanning trees and single linkage cluster analysis. *Applied Statistics* **18**, 54-64.

GRAHAM, R.L., AND P. HELL. 1985. On the history of minimum spanning tree problem. *Annals of the History of Computing* **7**, 43-57.

GUPTA, S.K. 1985. *Linear Programming and Network Models*. Affiliated East-West Press Private Limted, New Delhi, India.

GUSFIELD, D. 1988. A graph theoretic approach to statistical data security. *SIAM Journal on Computing* **17**, 552-571.

GUSFIELD, D., C. MARTEL, AND D. FERNANDEZ-BACA. 1987. Fast algorithms for bipartite network flow. *SIAM Journal on Computing* **16**, 237-251.

GUSFIELD, D., AND C. MARTEL. 1989. A fast algorithm for the generalized parametric minimum cut problem and applications. Technical Report CSE-89-21, Computer Science Division, University of California, Davis, CA.

GUTJAHR, A.L., AND G.L. NEMHAUSER. 1964. An algorithm for the line balancing problem. *Management Science* **11**, 308-315.

HALL, M. 1956. An algorithm for distinct representatives. *American Mathematical Monthly* **63**, 716-717.

HAUSMAN, H. 1978. *Integar Programming and Related Areas : A Classified Bibliography*. Lecture Notes in Economics and Mathematical Systems, Vol. 160, Springer-Verlag, Berlin.

HAYMOND, R.E., J.R. THORNTON, AND D.D. WARNER. 1988. A shortest path algorithm in robotics and its implementation on the FPS T-20 hypercube. *Annals of Operations Research* **14**, 305-320.

HELD, M., AND R. KARP. 1970. The traveling salesman problem and minimum spanning trees. *Operations Research* **18**, 1138-1162.

HOFFMAN, A.J., AND H.M. MARKOWITZ. 1963. A note on shortest path, assignment and transportation problems. *Naval Research Logistics Quarterly* **10**, 375-379.

HOFFMAN, A.J., AND S.T. McCORMICK. 1984. A fast algorithm that makes matrices optimally sparse. In *Progress in Combinatorial Optimization*, Academic Press, Canada.

HORN, W.A. 1971. Determining optimal container inventory and routing. *Transportation Science* **5**, 225-231.

HORN, W.A. 1973. Minimizing average flow time with parallel machines. *Operations Research* **21**, 846-847.

HU, T.C. 1961. The maximum capacity route problem. *Operations Research* **9**, 898-900.

IMAI, H., AND M. IRI. 1986. Computational-geometric methods for polygonal approximations of a curve. *Computer Vision, Graphics and Image Processing* **36**, 31-41.

JACOBS, W.W. 1954. The caterer problem. *Naval Research Logistics Quarterly* **1**, 154-165.

JEWELL, W.S. 1957. Warehousing and distribution of a seasonal product. *Naval Research Logistics Quarterly* **4**, 29-34.

JOHNSON, T.B. 1968. Optimum pit mine production scheduling. Technical Report, University of California, Berkeley.

KANG, A.N.C., R.C.T. LEE, C.L. CHANG, AND S.K. CHANG. 1977. Storage reduction through minimal spanning trees and spanning forests. *IEEE Transactions on Computers* **C-26**, 425-434.

KASTNING, C. 1976. *Integer Programming and Related Areas : A Classified Bibliography*. Lecture Notes in Economics and Mathematical Systems, Vol. 128, Springer-Verlag, Berlin.

KELLY, J.P., B.L. GOLDEN, AND A.A. ASSAD. 1992. Cell suppression: Disclosure protection for sensitive tabular data. *Networks* **22**, 397-417.

KELLY, J.R., 1961. Critical path planning and scheduling : Mathematical basis. *Operations Research* **9**, 296-320.

KHAN, M.R. 1979. A capacitated network formulation for manpower scheduling. *Industrial Management* **21**, 24-28.

KHAN, M.R., AND D.A. LEWIS. 1987. A network model for nursing staff scheduling. *Zeitschrift für*

KOLITZ, S. 1991. Personal communication.

KOURTZ, P. 1984. A network approach to least cost daily transfers of forest fire control resources. *INFOR* **22**, 283-290.

LARSON, R.C., AND A.R. ODONI. 1981. *Urban Operations Research*. Prentice-Hall, Englewood Cliffs, NJ.

LAWANIA. 1990. Personal communication.

LAWLER, E.L. 1966. Optimal cycles in doubly weighted linear graphs. *Theory of Graphs : International Symposium*, Dunod, Paris, and Gordon and Breach, New York, pp. 209-213.

LEVNER, E.V., AND A.S. NEMIROVSKY. 1991. A network flow algorithm for just-in-time project scheduling. Memorandum COSOR 91-21, Dept. of Mathematics and Computing Science, Eindhoven Univesity of Technology, Eindhoven, The Netherlands.

LIN, T.F. 1986. A system of linear equations related to the transportation problem with application to probability theory. *Discrete Applied Mathematics* **14**, 47-56.

LOBERMAN, H., AND A. WEINBERGER. 1957. Formal procedures for connecting terminals with a minimum total wire length. *Journal of ACM* **4**, 428-437.

LOVE, R.R., AND R.R. VEMUGANTI. 1978. The single-plant mold allocation problem with capacity and changeover restriction. *Operations Research* **26**, 159-165.

LOWE, T.J., R.L. FRANCIS, AND E.W. REINHARDT. 1979. A greedy network flow algorithm for a warehouse leasing problem. *AIIE Transactions* **11**, 170-182.

LUSS, H. 1979. A capacity expansion model for two facilities. *Naval Research Logistics Quarterly* **26**, 291-303.

MACHOL, R. E. 1961. An application of the assignment Problem. *Operations Research* **9**, 585-586.

MACHOL, R. E. 1970. An application of the assignment Problem. *Operations Research* **18**, 745-746.

MAGNANTI, T.L., and R. WONG. 1984. Network design and transportation planning: Models and algorithms. *Transportation Science* **18**, 1-55.

MAMER, J.W., AND S.A. SMITH. 1982. Optimizing field repair kits based on job completion rate. *Management Science* **28**, 1328-1334.

MARTEL, C. 1982. Preemptive scheduling with release times, deadlines, and due times. *Journal of ACM* **29**, 812-829.

MASON, A.J., AND A.B. PHILPOTT. 1988. Pairing stereo speakers using matching algorithms. *Asia-Pacific Journal of Operational Research* **5**, 101-116.

MAXWELL, W.L., AND R.C. WILSON. 1981. Dynamic network flow modelling of fixed path material handling systems. *AIIE Transactions* **13**, 12-21.

McGINNIS, L.F., AND H.L.W. NUTTLE. 1978. The project coordinators problem. *OMEGA* **6**, 325-330.

MEGGIDO, N., AND A. TAMIR. 1978. An O(n log n) algorithm for a class of matching problems. *SIAM Journal on Computing* **7**, 154-157.

MULVEY, J.M. 1979. Strategies in modeling : A personal scheduling example. *Interfaces* **9**, 66-76.

ORLIN, Donna. 1987. Optimal weapons allocation against layered defenses. *Naval Research Logistics* **34**, 605-617.

ORLIN, J.B., AND U.G. ROTHBLUM. 1985. Computing optimal scalings by parametric network algorithms. *Mathematical Programming* **32**, 1-10.

OSTEEN, R.E., AND P.P. LIN. 1974. Picture skeletons based on eccentricities of points of minimum spanning trees. *SIAM Journal on Computing* **3**, 23-40.

PICARD, J.C., AND H.D. RATLIFF. 1973. Minimum cost cut equivalent networks. *Management Science* **19**, 1087-1092.

PICARD, J.C., AND H.D. RATLIFF. 1978. A cut approach to the rectilinear distance facility location problem. *Operations Research* **26**, 422-433.

PICARD, J.C., AND M. QUEYRANNE. 1982. Selected applications of minimum cuts in networks. *INFOR* **20**, 394-422.

PRAGER, W. 1957. On warehousing problems. *Operations Research* **5**, 504-512.

PRIM, R.C. 1957. Shortest connection networks and some generalizations. *Bell System Technical Journal* **36**, 1389-1401.

RAVINDRAN, A. 1971. On compact book storage in libraries. *Opsearch* **8**, 245-252.

RHYS, J.M.W. 1970. A selection problem of shared fixed costs and network flows. *Management Science* **17**, 200-207.

SAPOUNTZIS, C. 1984. Allocating blood to hospitals from a central blood bank. *European Journal of Operational Research* **16**, 157-162.

SCHWARTZ, B.L. 1966. Possible winners in partially completed tournaments. *SIAM Review* **8**, 302-308.

SCHWARTZ, M., AND T.E. STERN. 1980. Routing techniques used in computer communication networks. *IEEE Transactions on Communications* COM-**28**, 539-552.

SEGAL, M. 1974. The operator-scheduling problem : A network flow approach. *Operations Research* **22**, 808-824.

SERVI, L.D. 1989. A network flow approach to a satellite scheduling problem. Research Report, GTE Laboratories, Inc., Waltham, MA.

SHIER, D.R. 1982. Testing for homogeneity using minimum spanning trees. *The UMAP Journal* **3**, 273-283.

SLUMP, C.H., AND J.J. GERBRANDS. 1982. A network flow approach to reconstruction of the left ventricle from two projections. *Computer Graphics and Image Processing* **18**, 18-36.

SRINIVASAN, V. 1974. A transshipment model for cash management decisions. *Management Science* **20**, 1350-1363.

SRINIVASAN, V. 1979. Network models for estimating brand-specific effects in multiattribute marketing models. *Management Science* **25**, 11-21.

STILLINGER, F.H. 1967. Physical clusters, surface tension, and critical phenomenon. *Journal of Chemical Physics* **47**, 2513-2533.

STONE, H.S. 1977. Multiprocessor scheduling with the aid of network flow algorithms. *IEEE Transactions on Software Engineering* **3**, 85-93.

SZADKOWSKI. 1970. An approach to machining process optimization. *International Journal of Production Research* **9**, 371-376.

TSO, M. 1986. Network flow models in image processing. *Journal of Operational Research Society* **37**, 31-34.

TSO, M., P. KLEINSCHMIDT, I. MITTERREITER, AND J. GRAHAM. 1991. An efficient transportation algorithm for automatic chromosome karotyping. *Pattern Recognition Letters* **12**, 117-126.

VAN SLYKE, R., AND H. FRANK. 1972. Network reliability analysis: Part I. *Netwoks* **1**, 279-290.

VEINOTT, A.F., AND H.M. WAGNER. 1962. Optimal capacity scheduling - Part I and II. *Operations Research* **10**, 518-547.

VON RANDOW, R. 1982. *Integer Programming and Related Areas : A Classified Bibliography 1978-1981*. Lecture Notes in Economics and Mathematical Systems, Vol. 197, Springer-Verlag, Berlin.

VON RANDOW, R. 1985. *Integer Programming and Related Areas : A Classified Bibliography 1981-1984*. Lecture Notes in Economics and Mathematical Systems, Vol. 243, Springer-Verlag, Berlin.

WAGNER, D.K. 1990. Disjoint (s,t)-cuts in a network. *Networks* **20**, 361-371.

WATERMAN, M.S. 1988. *Mathematical Methods for DNA Sequences*. CRC Press.

WHITE, L.S. 1969. Shortest route models for the allocation of inspection effort on a production line. *Management Science* **15**, 249-259.

WHITE, W.W. 1972. Dynamic transshipment networks : An algorithm and its application to the distribution of empty containers. *Networks* **2**, 211-230.

WRIGHT, J.W. 1975. Reallocation of housing by use of network analysis. *Operational Research Quarterly* **26**, 253-258.

ZAHN, C.T. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computing* **C20**, 68-86.

ZANGWILL, W.I. 1969. A backlogging model and a multi-echelon model of a dynamic economic lot size production system - A network approach. *Management Science* **15**, 506-527.

ZAWACK, D.J., AND G.L. THOMPSON 1987. A dynamic space-time network flow model for city traffic congestion. *Transportation Science* **21**, 153-162.