FLOW NETWORKS AND COMBINATORIAL OPERATIONS RESEARCH

D. R. FULKERSON, RAND, Santa Monica, California

PART I: FLOWS IN NETWORKS

1. Maximal flow. A *directed network* (graph) $G = [N; \mathcal{A}]$ consists of a finite collection N of elements $1, 2, \cdots, n$ together with a subset $\mathcal{A}$ of the ordered pairs (i, j) of distinct elements of N. The elements of N will be called *nodes*; members of $\mathcal{A}$ are *arcs*. Figure 1.1 shows a directed network having four nodes and six arcs (1, 2), (1, 3), (2, 3), (2, 4), (3, 2), and (3, 4).

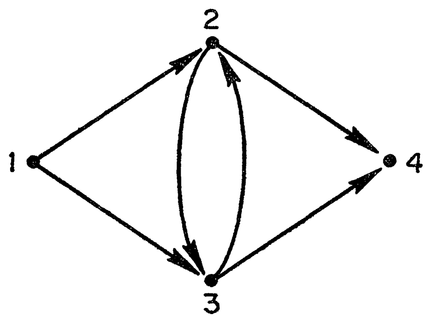

FIG. 1.1

Sometimes we shall also consider *undirected networks*, for which the set $\mathcal{A}$ consists of unordered pairs of nodes. For emphasis, these will then be termed *links*.

Suppose that each arc (i, j) of a directed network has associated with it a nonnegative number c_{ij}, the *capacity* of (i, j), to be thought of as representing the maximal amount of some commodity that can arrive at j from i along (i, j) per unit time in a steady-state situation. Then a natural question is: What is the maximal amount of commodity-flow from some node to another via the entire network? (For example, one might think of a network of city streets, the commodity being cars, and ask for a maximal traffic flow from some point to another.) We may formulate the question mathematically as follows. Let 1 and n be the two nodes in question. A *flow, of amount v,* from 1 *to n* in $G = [N; \mathcal{A}]$ is a function x from $\mathcal{A}$ to real numbers (a vector x having components x_{ij} for(i,j) in $\mathcal{A}$) that satisfies the linear equations and inequalities

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} v, & i = 1, \\ -v, & i = n, \\ 0, & \text{otherwise}, \end{cases} \tag{1.1}$$

$$0 \leqq x_{ij} \leqq c_{ij}, \qquad (i, j) \text{ in } \mathcal{A}. \tag{1.2}$$

In (1.1) the sums are of course over those nodes for which x is defined. We call 1 the *source*, n the *sink*. A *maximal flow* from source to sink is one that maximizes the variable v subject to (1.1), (1.2).

Figure 1.2 shows a flow from source node 1 to sink node 6 of amount 7. In Figure 1.2, the first number of each pair beside an arc is the arc capacity, the second number the arc flow.

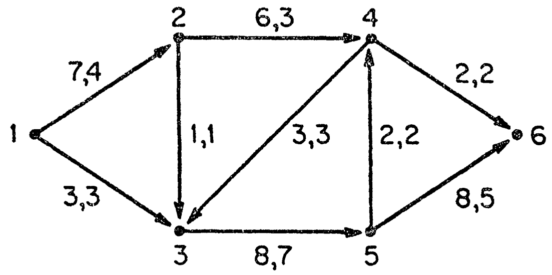

FIG. 1.2

To state the fundamental theorem about maximal flow, we need one other notion, that of a cut. A *cut separating* 1 and n is a partition of the nodes into two complementary sets, I and J, with 1 in I, say, and n in J. The *capacity* of the cut is then

$$\sum_{\substack{i \text{ in } I \\ j \text{ in } J}} c_{ij}. \tag{1.3}$$

(For instance, if $I = \{1, 3, 4\}$ in Fig. 1.2, the cut has capacity $c_{12}+c_{35}+c_{46}=17$.) A cut separating source and sink of minimum capacity is a *minimal* cut, relative to the given source and sink.

Summing the equations (1.1) over i in the source-set I of a cut and using (1.2) shows that

$$v = \sum_{\substack{i \text{ in } I \\ j \text{ in } J}} (x_{ij} - x_{ji}) \leqq \sum_{\substack{i \text{ in } I \\ j \text{ in } J}} c_{ij}. \tag{1.4}$$

In words, for an arbitrary flow and arbitrary cut, the net flow across the cut is the flow amount v, which is consequently bounded above by the cut capacity. Theorem 1.1 below asserts that equality holds in (1.4) for some flow and some cut, and hence the flow is maximal, the cut minimal [**11**].

THEOREM 1.1. *For any network the maximal flow amount from source to sink is equal to the minimal cut capacity relative to the source and sink.*

Theorem 1.1 is a kind of combinatorial counterpart, for the special case of the maximal flow problem, of the duality theorem for linear programs, and can be deduced from it [**5**]. But the most revealing proof of Theorem 1.1 uses a simple "marking" or "labeling" process [**12**] for constructing a maximal flow, which also yields the following theorem.

THEOREM 1.2. *A flow x from source to sink is maximal if and only if there is no flow-augmenting path with respect to x.*

Here we need to say what an x-augmenting path is. First of all, a path from one node to another is a sequence of distinct end-to-end arcs that starts at the first node and terminates at the second; arcs traversed with their direction in going along the path are *forward* arcs of the path, while arcs traversed against their direction are *reverse* arcs of the path. A path from source to sink is x-augmenting provided that $x<c$ on forward arcs and $x>0$ on reverse arcs. For example, the path (1, 2), (2, 4), (5, 4), (5, 6) in Figure 1.2 is an augmenting path for the flow shown there. Figure 1.3 below indicates how such a path can be used to increase the amount of flow from source to sink.

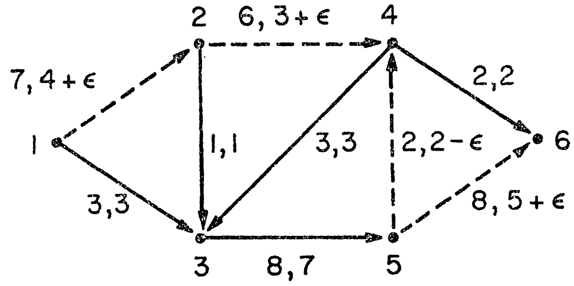

FIG. 1.3

Taking the flow change ϵ along the path as large as possible in Figure 1.3, namely $\epsilon=2$, produces a maximal flow, since the cut $I=\{1, 2, 4\}$, $J=\{3, 5, 6\}$ is then "saturated."

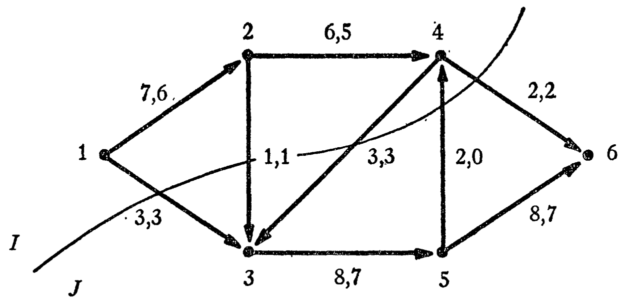

FIG. 1.4

The labeling process of [12] is a systematic and efficient search, fanning out from the source, for a flow augmenting path. If none such exists, the process ends by locating a minimal cut.

The following theorem, of special significance for combinatorial applications, is also a consequence of the procedure sketched above for constructing maximal flow.

THEOREM 1.3. *If all arc capacities are integers, there is an integral maximal flow.*

It is sometimes convenient to alter the constraints (1.2) of the maximal flow problem to

$$l_{ij} \leqq x_{ij} \leqq c_{ij}. \tag{1.5}$$

Here l is a given lower bound function satisfying $l \leqq c$. The analogue of Theorem 1.1 is then

THEOREM 1.4. *If there is a function x satisfying* (1.1) *and* (1.5) *for some number v, then the maximum v subject to these constraints is equal to the minimum of*

$$\sum_{\substack{i \text{ in } I \\ j \text{ in } J}} (c_{ij} - l_{ji}) \tag{1.6}$$

taken over all cuts I, J separating source and sink. On the other hand, the minimum v is equal to the maximum of

$$\sum_{\substack{i \text{ in } I \\ j \text{ in } J}} (l_{ij} - c_{ji}) \tag{1.7}$$

taken over all cuts I, J separating source and sink.

The question of the existence of such a flow x, together with another flow feasibility question, will be discussed in the next section.

2. Feasibility theorems. The constraints of the maximal flow problem are of course always feasible, since $x=0$ satisfies (1.1), (1.2) with $v=0$. By changing the constraints in various ways, interesting feasibility questions arise. Here we shall consider two such, one involving supplies and demands at nodes, the other lower bounds on arc flows, as in (1.5).

Let $G=[N; \mathcal{A}]$ have capacity function c, and let S and T be disjoint subsets of N. With each i in S associate a *supply* $a_i \geqq 0$, with each i in T a *demand* $b_i \geqq 0$, and impose the constraints

$$\begin{aligned} \sum_j (x_{ij} - x_{ji}) &\leqq a_i, \quad i \text{ in } S, \\ &\leqq -b_i, \ i \text{ in } T, \\ &= 0, \quad \text{otherwise}, \end{aligned} \tag{2.1}$$

$$0 \leqq x_{ij} \leqq c_{ij}, \quad (i, j) \text{ in } \mathcal{A}. \tag{2.2}$$

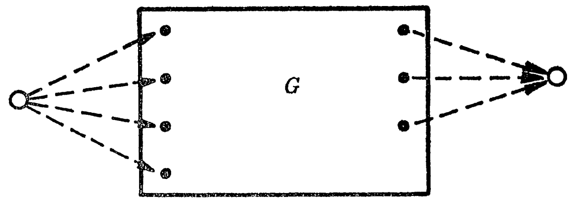

FIG. 2.1

In words, the net flow out of i in S is bounded above by the supply a_i, and the net flow into i in T is bounded below by the demand b_i. When are the supply-demand constraints (2.1), (2.2) feasible?

This question is easily answered by applying Theorem 1.1 to an enlarged network. Extend $G=[N;\ \mathcal{A}]$ to $G^*=[N^*;\ \mathcal{A}^*]$ by adjoining a source 0 and sink $n+1$, together with source arcs $(0,j)$ for j in S, and sink arcs $(i, n+1)$ for i in T. (See Figure 2.1.) The capacity function c^* on $\mathcal{A}^*$ is defined by $c^*_{0,j}=a_j$ for j in S, $c^*_{i,n+1}=b_i$ for i in T, $c^*_{ij}=c_{ij}$ for (i,j) in $\mathcal{A}$. The constraints (2.1) and (2.2) are feasible if and only if the maximal flow amount from source to sink in the enlarged network is at least $\sum_{i \text{ in } T} b_i$, that is, if and only if a maximal flow saturates all sink arcs. Hence we need only construct a maximal flow in order to check the feasibility of (2.1), (2.2). By pushing the analysis a little further, using Theorem 1.1, the following theorem emerges [21].

THEOREM 2.1. *The supply-demand constraints* (2.1), (2.2) *are feasible if and only if, for each subset T' of T, there is a flow $x(T')$ that satisfies the aggregate demand $\sum_{i \text{ in } T'} b_i$ without violating the supply limitations at nodes of S.*

Here satisfying the aggregate demand over T' means that the net flow into the set T' must be at least $\sum_{i \text{ in } T'} b_i$, without regard for the individual demands in T'. The necessity of the condition is of course clear; sufficiency asserts the existence of a single flow x meeting all individual demands provided the flows $x(T')$ exist for all subsets T' of T.

It should be noted that if the functions a, b, c of (2.1), (2.2) are integral valued, and if feasible flows exist, then there is an integral feasible flow. This follows from Theorem 1.3 and the conversion of (2.1), (2.2) to a maximal flow problem. A similar integrity statement holds for the situation of Theorem 1.4, and indeed, for all the flow problems to be discussed in any detail in this survey.

We turn now to a consideration of lower bounds on arc flows, as in (1.5), and pose the resulting feasibility question in terms of circulations, i.e., flows that are source-sink free, instead of flows from source to sink. (One can always add a "return-flow" arc from sink to source to convert to circulations.) Thus we are questioning the feasibility of the constraints

$$\sum_j (x_{ij} - x_{ji}) = 0, \qquad i \text{ in } N, \tag{2.3}$$

$$l_{ij} \leqq x_{ij} \leqq c_{ij}, \qquad (i,j) \text{ in } \mathcal{A}. \tag{2.4}$$

The following theorem answers the question [26]. Its proof can be made to rely on Theorem 1.1 [15].

THEOREM 2.2. *The constraints* (2.3), (2.4) *are feasible if and only if*

$$\sum_{\substack{i \text{ in } I \\ j \text{ in } J}} c_{ij} \geqq \sum_{\substack{i \text{ in } I \\ j \text{ in } J}} l_{ji} \tag{2.5}$$

holds for all partitions I, J of N.

Again the necessity is clear, since (2.5) simply says there must be sufficient escape capacity from the set I to take care of the flow forced into I by the function l. But sufficiency is not obvious.

Other useful flow feasibility theorems have been deduced [**19**, **26**]. In each case Theorem 1.1 can be used as the main tool in a proof.

3. Minimal cost flows. One of the most practical problem areas involving network flows is that of constructing flows satisfying constraints of various kinds and minimizing cost. The standard linear programming transportation problem, which has an extensive literature, is in this category.

We put the problem as follows. Each arc (i, j) of a network $G=[N; \mathcal{A}]$ has a capacity c_{ij} and a cost a_{ij}. It is desired to construct a flow x from source to sink of specified amount v that minimizes the total flow cost

$$\sum_{(i,j) \text{ in } \mathcal{A}} a_{ij}x_{ij} \tag{3.1}$$

over all flows that send v units from source to sink. In many applications one has supplies of a commodity at certain points in a transportation network, demands at others, and the objective is to satisfy the demands from the supplies at minimum cost.

By treating v as a parameter, the method for constructing maximal flows can be used to construct minimal cost flows throughout the feasible range of v. Indeed, the solution procedure can be viewed as one of solving a sequence of maximal flow problems, each on a subnetwork of the original one [**14**]. Another, not essentially different, viewpoint is provided by the following theorem [**1**, **29**].

THEOREM 3.1. *Let x be a minimal cost flow from source to sink of amount v. Then the flow obtained from x by adding $\epsilon>0$ to the flow in forward arcs of a minimal cost x-augmenting path, and subtracting ϵ from the flow in reverse arcs of this path, is a minimal cost flow of amount $v+\epsilon$.*

Here the cost of a path is the sum of arc costs over forward arcs minus the corresponding sum over reverse arcs, i.e., the cost of "sending an additional unit" via the path.

Thus, if all arc costs a_{ij} are nonnegative, for example, one can start with the zero flow and apply Theorem 3.1 to obtain minimal cost flows for increasing v. (The cost profile thereby generated is piecewise linear and convex.) All that is needed to make this an explicit algorithm is a method of searching for a minimal cost flow augmenting path. Various ways of doing this can be described. One such will be given in Part II, section 1.

Another method [**17**] for constructing minimal cost flows poses the problem in circulation format, that is, (3.1) is to be minimized subject to (2.3), (2.4). This construction has a number of advantages, principally in terms of generality and flexibility. For instance, it may be started with any circulation; even (2.4) need not be satisfied initially. Also, no assumption about the cost function is required.

These methods produce integral flows in case the arc capacities (and lower bounds) are integers. Theoretical upper bounds on the computing task, ones that are quite good, are easily obtained in each case. This may be contrasted with the situation for general linear programs, where decent upper bounds on solution methods are unknown.

4. Maximal dynamic flow. Suppose that each arc (i, j) of a network G has not only a capacity, but a transit time t_{ij} as well, and that we are interested in determining the maximum amount of flow that can reach sink n from source 1 in a specified number t of time periods. This dynamic flow problem can always be treated as a static flow problem in a time-expanded version G_t of G. For example, if the given network G is that of Figure 4.1 and if each arc of G has unit transit time, then G_3 is shown in Figure 4.2. (We have included "storage arcs" leading from a location to itself one unit of time later.)

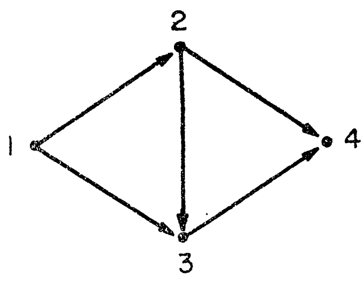

FIG. 4.1

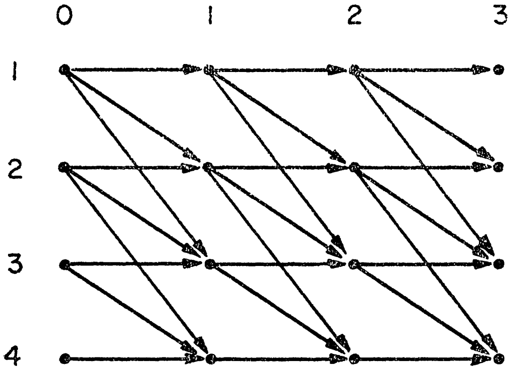

FIG. 4.2

Expanding the network in this way puts one back in the static case. Moreover, arc capacities and transit times can vary with time and this is still so. However, if each capacity and transit time is fixed over time, the problem can be solved in the smaller network G. Specifically, a maximal dynamic flow for t periods can be generated from a static flow x in G of amount v that minimizes the linear form

$$\sum_{(i,j) \text{ in } \mathcal{A}} t_{ij}x_{ij} - (t+1)v \tag{4.1}$$

over all flows in G from source 1 to sink n [14]. By adding the return-flow arc $(n, 1)$ to G with $c_{n1} = \infty$, $t_{n1} = -(t+1)$, the problem may be viewed as one of constructing a circulation that minimizes the "cost" form (4.1).

5. Multi-terminal maximal flow. Heretofore we have phrased statements in terms of directed networks. In this section we confine the discussion to undirected flow networks, by which we merely mean the following. A link (i, j) can carry flow in either direction and has the same flow capacity each way. Thus one can think of an arc (i, j) with capacity c_{ij} and an arc (j, i) with capacity $c_{ji}=c_{ij}$. The assumption of a symmetric capacity function c makes the results described in this section considerably simpler and more appealing than they would otherwise be.

Instead of dealing with a single source and sink, we shift attention to all pairs of distinct nodes taken as terminals for flows. These flows are not to be thought of as occurring simultaneously.

Let v_{ij} denote the maximal flow amount from i to j. Thus the function v is symmetric, $v_{ij}=v_{ji}$, and may be determined explicitly for an n-node network by solving $n(n-1)/2$ maximal flow problems. There is, however, a much simpler way of determining the function v, one that involves the solution of only $n-1$ maximal flow problems; in addition, there is a simple condition in order that a symmetric function v be realizable as the maximal multiterminal flow function of some undirected network [22].

THEOREM 5.1. *A symmetric, nonnegative function v is realizable by an undirected network if and only if v satisfies*

$$v_{ij} \geqq \min(v_{ik}, v_{kj}) \tag{5.1}$$

for all triples i, j, k.

The necessity of the "triangle inequality" (5.1) follows easily from Theorem 1.1.

The condition (5.1) imposes severe limitations on the function v. For instance, among the three functional values appearing in (5.1), two must be equal and the third no smaller than their common value. It also follows that if the network has n nodes, v can take on at most $n-1$ numerically different functional values. It is not altogether surprising, therefore, that v can be determined by a

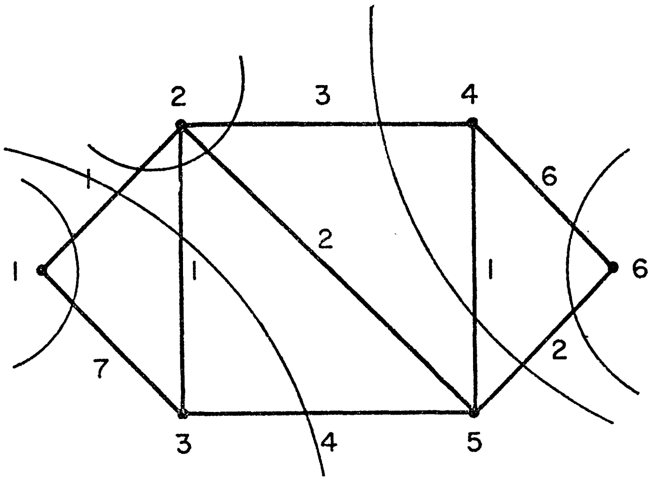

FIG. 5.1

simpler process than solving all single-terminal maximal flow problems. This process systematically picks out precisely $n-1$ cuts in the network having the property that v_{ij} is determined by the minimum one of these cuts separating i and j [22]. For example, in the network of Figure 5.1 the relevant cuts are those shown. Thus, for instance, since nodes 1 and 4 are separated by the three cuts (1/2, 3, 4, 5, 6), (1, 3/2, 4, 5, 6), (1, 2, 3, 5/4, 6) having capacities 8, 6, 6 respectively, then $v_{14}=\min(8, 6, 6)=6$.

6. Other flow problems. The flow problems that have been discussed thus far all have the useful and pleasant feature that the assumption of integral data implies the existence of an integral solution. A number of flow problems that do not share this property have also been studied. Among these we mention flows in networks with gains [29], simultaneous multi-terminal flows [13], and problems involving optimal synthesis of flow networks that meet specified requirements [16, 22]. Methods of solution for such problems are, in general, more complicated than methods for those we have discussed. One rather surprising exception to this statement is the following synthesis problem. Suppose it is desired to construct an undirected network on a specified number of nodes so that $v_{ij} \geqq r_{ij}$ for stipulated requirements r_{ij}, with the total sum of link capacities of the network minimal. A very simple combinatorial method of solution for this synthesis problem is given in [22].

PART II: COMBINATORIAL PROBLEMS

1. Network potentials and shortest chains. Consider a directed network in which each arc (i, j) has associated with it a positive number a_{ij}, which may be thought of as the length of the arc, or the cost of traversing the arc. How does one determine a shortest chain from some node to another? Here we have used *chain* to mean a path containing only forward arcs, the length of the chain being obtained by adding its arc lengths.

While this is a purely combinatorial problem, it may also be viewed as a flow problem simply by imposing a cost a_{ij} per unit flow in (i, j), taking all arc capacities infinite, and asking for a minimal cost flow of one unit from the first node to the second. An integral optimal flow corresponding to $v=1$ singles out a shortest chain.

Many ways of locating shortest chains efficiently have been suggested. We describe one [10]. Like others, it simultaneously finds shortest chains from the first node to all others reachable by chains.

In this method each node i will initially be assigned a number π_i. These node numbers, which we shall refer to as *potentials*, will then be revised in an iterative fashion. Let 1 be the first node. To start, take $\pi_1=0$, $\pi_i=\infty$ for $i\neq 1$. Then search the list of arcs for an arc (i, j) whose end potentials satisfy

$$\pi_i + a_{ij} < \pi_j. \tag{1.1}$$

(Here $\infty + a = \infty$.) If such an arc is found, change π_j to $\pi_j' = \pi_i + a_{ij}$, and search again for an arc satisfying (1.1), using the new node potentials. Stop the process when the node potentials satisfy

$$\pi_i + a_{ij} \geqq \pi_j \tag{1.2}$$

for all arcs.

It is not hard to show that the process terminates, and that when this happens, the potential π_j is the length of a shortest chain from 1 to j. (Here $\pi_j = \infty$ at termination means there is no chain from 1 to j.) A shortest chain from 1 to j can be found by tracing back from j to 1 along arcs satisfying (1.2) with equality (see Figure 1.1).

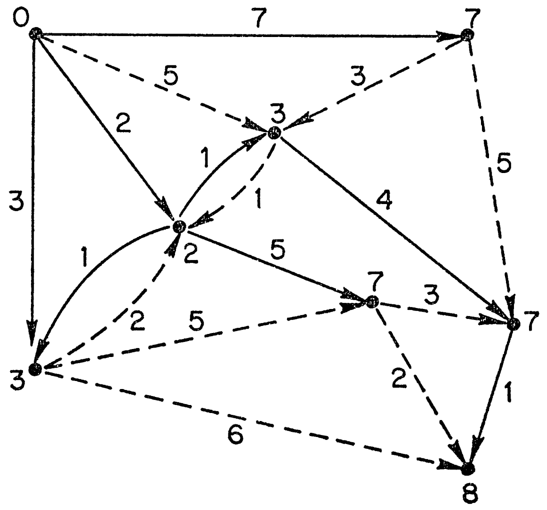

FIG. 1.1

Practical applications that require shortest chains are numerous. For instance, in making up a table of highway distances between cities, a shortest chain between each pair needs to be found. A less obvious application is the discrete version of the problem of determining the least time for an airplane to climb to a given altitude [2]. Some other applications will be discussed in following sections.

While we have assumed positive lengths for the method described above, this assumption can be weakened. Call a chain of arcs leading from a node to itself a *directed cycle*. Then it is enough to suppose that all directed cycle lengths are nonnegative.

If directed cycle costs are nonnegative, the minimum cost flow problem of Part I, section 3, can be solved by repeatedly finding cheapest chains in suitable networks. Because of the assumption on the cost function a, we may start with the zero flow. Thus, using Theorem 3.1, it is enough to reduce the problem of finding a cheapest flow augmenting path with respect to a minimal cost flow x of amount v to that of finding a cheapest chain. Define a new network $G' = [N; \mathcal{A}']$

from the given one $G = [N; \mathcal{A}]$ and the flow x as follows. First note that we may assume $x_{ij} \cdot x_{ji} = 0$, since $a_{ij} + a_{ji} \geqq 0$. Now put (i, j) in $\mathcal{A}'$ if either $x_{ij} < c_{ij}$ or $x_{ji} > 0$ and define a' by

$$a'_{ij} = \begin{cases} a_{ij} & \text{if } \; x_{ij} < c_{ij} \;\text{ and }\; x_{ji} = 0, \\ -a_{ji} & \text{if } \; x_{ji} > 0. \end{cases} \tag{1.3}$$

Thus a chain from source to sink in the new network corresponds to an x-augmenting path in the old, and these have the same cost. Moreover, since x is a minimal cost flow, the function a' satisfies the nonnegative directed cycle condition. Hence the method of this section can be used to construct minimal cost flows of successively larger amounts.

2. Optimal chains in acyclic networks. If the network is acyclic (contains no directed cycles), the shortest chain method of the last section can be modified in such a way that, once a potential is assigned a node, it remains unchanged. One can begin by numbering the nodes so that if (i, j) is an arc, then $i < j$. Such a numbering can be obtained as follows. Since the network is acyclic, there are nodes having only outward-pointing arcs. Number these nodes $1, 2, \cdots, k$ in any order. Next delete these nodes and all their arcs, search the new network for nodes having only outward-pointing arcs, and number these, starting with $k+1$. Repetition of this process leads to the desired kind of numbering (see Figure 2.1).

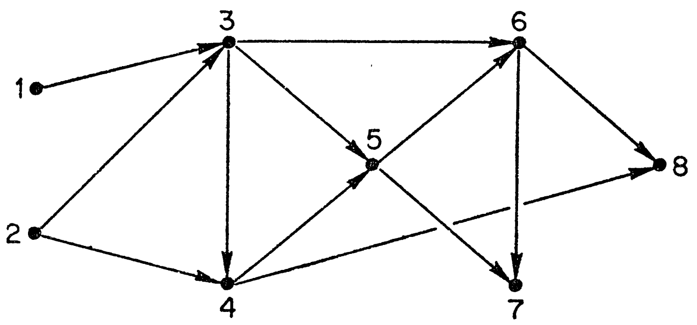

FIG. 2.1

If we wish to find shortest chains from node k to all other nodes reachable from k by chains, the calculation is now trivial. Simply define $\pi_k, \pi_{k+1}, \cdots, \pi_n$ recursively by

$$\begin{cases} \pi_k = 0 \\ \cdot \;\cdot \;\cdot \\ \pi_j = \min\limits_{k \leq i < j} (\pi_i + a_{ij}), \quad j = k+1, \cdots, n. \end{cases} \tag{2.1}$$

Here the minimum is of course taken over i such that (i, j) is an arc.

Longest chains in acyclic networks can be computed by replacing "min" by "max" in (2.1).

The recursion (2.1), of dynamic programming type, can be applied in a number of problems. We shall discuss three such applications in the following sections.

3. The knapsack problem. Suppose there are K objects, the i-th object having weight w_i and value v_i, and that it is desired to find the most valuable subset of objects whose total weight does not exceed W. Thus we wish to maximize

$$\sum_{i \text{ in } S} v_i \tag{3.1}$$

over subsets $S \subseteq \{1, 2, \cdots, K\}$ such that

$$\sum_{i \text{ in } S} w_i \leqq W. \tag{3.2}$$

We take $w_1, w_2, \cdots, w_K, W$ to be positive integers.

This combinatorial problem, commonly referred to as the knapsack problem, can be viewed as one of finding a longest chain in a suitable acyclic network. Let the network have nodes denoted by ordered pairs (i, w), $i=0, 1, \cdots, K$, $w=0, 1, \cdots, W$. The node (i, w) has two arcs leading into it, one from $(i-1, w)$, the other from $(i-1, w-w_i)$, provided these exist. (See Figure 3.1.) The length of the first arc is zero; the other has length v_i. In addition we put in a starting node and join it to all of the nodes $(0, w)$ by arcs of length zero. Then chains from the starting node to (i, w) correspond to subsets of the first i objects whose total weight is at most w, the length of the chain being the value of the subset.

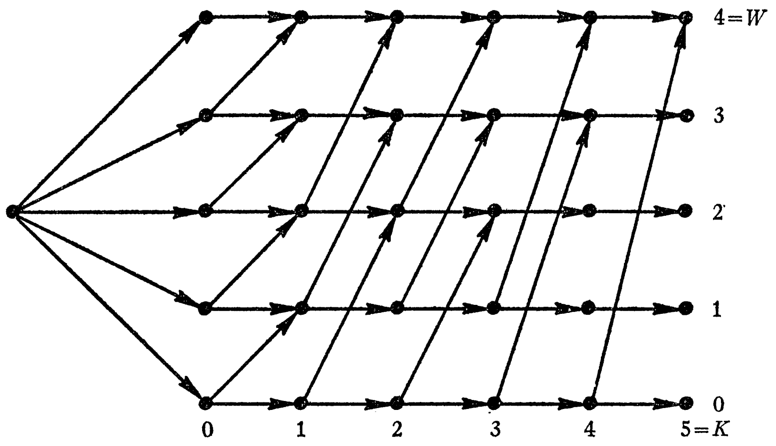

Fig. 3.1

4. Equipment replacement. As equipment deteriorates with age, and improved equipment becomes available on the market, a time may be reached when the purchase cost of new equipment is repaid by its potential future earnings. One is then faced with the problem of determining an optimal replacement policy [9].

For simplicity, consider a single machine and suppose that at the beginning of each of K periods of time it must be decided whether to keep the machine another period or purchase a new one. Let $r(i, t)$ denote the revenue obtainable during period i from a machine which starts the period at age t (the function r may reflect upkeep costs), and let $c(i, t)$ denote the cost of replacing a machine of age t with a new machine if the replacement occurs at the beginning of period i. Thus replacing a machine of age t at period i gives a net return for the period of $r(i, 0)-c(i, t)$.

The acyclic network shown in Figure 4.1 indicates one formulation in terms of chains. Again nodes are points (i, t), $i=0, 1, \cdots, K$, $t=0, 1, \cdots, T$. (Here T is some sufficiently large integer; if we start with a machine of age t, $T=K+t$ will do.) In general, two arcs, reflecting the possibilities of keeping or replacing, lead from (i, t), the "keep" arc going to $(i+1, t+1)$, the "replace" arc to $(i+1, 1)$. The first of these has length $r(i, t)$, the second has length $r(i, 0)-c(i, t)$. We may also put in a sink node with arcs of length zero leading into it from the nodes (K, t), $t=0, 1, \cdots, T$. Then chains from $(0, t)$ to the sink correspond to the various replacement policies starting with a machine of age t, the length of the chain being the total return from the policy.

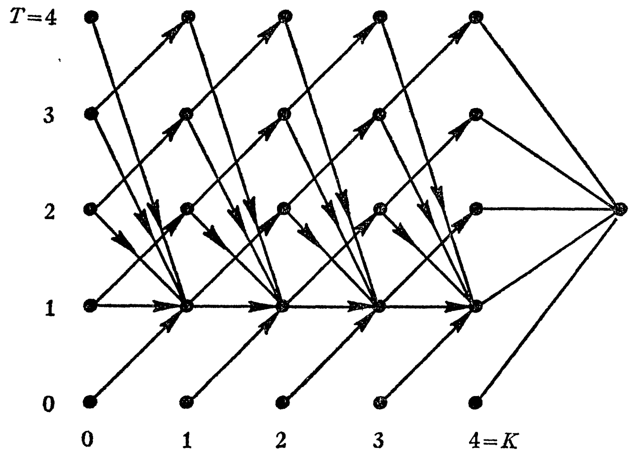

FIG. 4.1

A simpler network for this problem is shown in Figure 4.2. In Figure 4.2 an arc (i, j) corresponds to keeping the machine throughout periods $i, i+1, \cdots, j-1$ and replacing it at the start of period j, the associated length being the return obtained from this action. Thus a longest chain from source 1 to sink K is to be found.

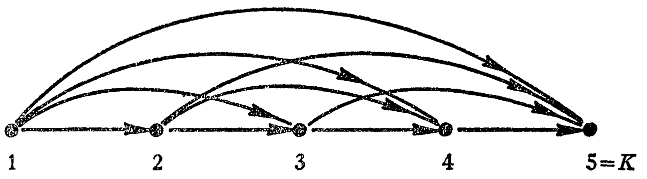

FIG. 4.2

The examples of this section and the preceding section are typical discrete dynamic programming problems. Such problems can always be viewed as seeking optimal chains in appropriate acyclic networks.

5. Project planning. One of the most popular combinatorial applications involving networks deals with the planning and scheduling of large, complicated projects [33]. Suppose that a project of some kind (the construction of a bridge, for example) is broken down into many individual jobs. Certain of these jobs will have to be finished before others can be started. We may depict the order relations among the jobs by means of an acyclic network whose arcs represent jobs. To take a simple case, suppose there are five jobs with the ordering: 1 precedes 3; 1 and 2 precede 4; 1, 2, 3 and 4 precede 5. This may be pictured by the network shown in Figure 5.1.

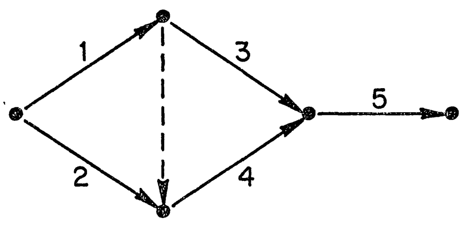

FIG. 5.1

Notice that we have added a "dummy" job, the dotted arc of Figure 5.1, to maintain the proper order relations among the jobs. The use of dummies permits a network representation of this kind for any project (finite partially ordered set).

Assuming that each job has a known duration time (dummies have zero duration time), that the only scheduling restriction is that all inward-pointing jobs at a node must be finished before any outward-pointing job can be started, it follows that the minimum time to complete the entire project is equal to the length of a longest chain of jobs. Hence the minimum project duration time can be calculated easily.

Although a fixed time has been assumed for each job, it may be the case that by spending more money, a job can be expedited. The question then arises: Which jobs should money be spent on and how much, in order that the project be finished by a given date at minimum cost? If the time-cost relation for each job is linear, this problem can be shown to be a minimal cost flow problem of the kind described in Part I, section 3 [**18**, **33**].

6. Minimal chain coverings of acyclic networks. The following question concerning acyclic networks has both theoretical and practical interest: What is the minimum number of chains required to cover a given subset of arcs? We first show how flows may be used to answer this question, and then give a practical interpretation.

Let the subset of arcs be denoted by $\mathcal{A}'$. To rephrase the question, we seek the minimum number of chains in the acyclic network G such that every arc of $\mathcal{A}'$ belongs to at least one of these chains. Theorems 1.3 and 1.4 can be used to provide an answer to the question, as sources in G take all nodes with only outward-pointing arcs; as sinks take all nodes with only inward-pointing arcs. Now place a lower bound of 1 on flow in arcs of $\mathcal{A}'$, 0 on arcs not in $\mathcal{A}'$, and take all arc capacities infinite. Then an integral flow through G of amount v picks out v chains in G that cover all arcs of $\mathcal{A}'$, and the second half of Theorem 1.4 implies

THEOREM 5.1. *The minimum number of chains in an acyclic network needed to cover a subset of arcs is equal to the maximum number of arcs of the subset having the property that no two belong to any chain.*

Theorem 5.1 is a generalization of a known result on chain decompositions of partially ordered sets [**8**].

A practical instance of this situation arises if we think of an airline, say, attempting to meet a fixed flight schedule with the minimum number of planes, all of the same type [**4**]. Let the individual flights be numbered $1, 2, \cdots, n$. Start and finish times $s_i < f_i$ are known for each flight, and the times t_{ij} to return from the destination of the ith flight to the origin of the jth flight are also known. The flights can be partially ordered by saying that i precedes j if $f_i + t_{ij} \leqq s_j$, and the resulting partially ordered set represented by an acyclic network (as in the preceding section). A chain in the network represents a possible assignment of flights to one aircraft. The problem then is to cover the nondummy arcs (those corresponding to actual flights) with the minimum number of chains. Theorem 5.1 asserts that this number is equal to the maximum number of flights, no two of which can be accomplished by a single plane.

Problems of this nature become considerably more complicated if the assumption of a fixed schedule is dropped. For instance, suppose the times s_i, f_i are at our disposal subject to the restriction that $f_i - s_i = t_i$, with the duration times t_i known, as well as the reassignment times t_{ij}. The problem might then be to arrange a schedule completing all flights by a given time and requiring the minimum number of planes, or to finish all flights at the earliest possible time with a fixed number of planes. For such scheduling problems there is very little known in the way of general theoretical results or good computational procedures. However, some special results have been deduced [**27**, **31**].

7. Assignment problems. The following is typical of an important and well-known class of combinatorial problems having network flow formulations. Suppose there are m men and n jobs, and that it is known whether or not man i is qualified to fill job j, $i=1, 2, \cdots, m, j=1, 2, \cdots, n$. When is it possible to fill all jobs with qualified men and how does one determine such an assignment?

Using Theorem 1.3, the problem may be phrased in terms of flows. Corresponding to man i take a source node i, to job j a sink node j, and direct an arc from i to j if man i is qualified for job j. (See Figure 7.1.) Impose a demand of 1 unit at each sink and let each source have a supply of 1 unit. All arc capacities may be taken infinite. The problem of assigning men to jobs thus becomes that of constructing a flow (integral, of course) meeting the demands from the supplies.

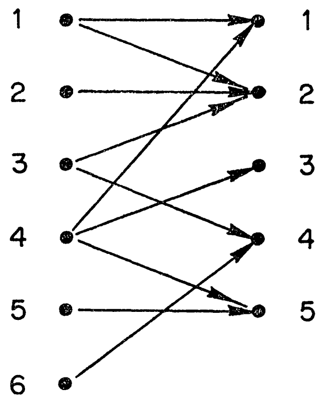

FIG. 7.1

Combinatorial interpretations of Theorems 1.1 and 2.1 for this situation lead in the first instance to a well-known theorem about maximum matchings and minimum covers in bipartite networks [35], and in the second instance to an equally well-known, and equivalent, theorem concerning systems of distinct representatives for subsets of a given set [24].

A more general assignment problem, usually referred to as that of optimal assignment [36, 42], assumes that man i in job j is worth a_{ij} units, and the total worth of an assignment is given by the sum of the numbers a_{ij} taken over the individual man-job matchings in the assignment. The problem then is to construct an assignment of maximal worth. By taking the cost per unit of flow in arc (i, j) to be $-a_{ij}$, the optimal assignment problem is seen to be a special case of the minimal cost flow problem.

More complicated personnel assignment models have been formulated in terms of flow networks. For instance, one which involves the recruiting, training, and retraining of military personnel to meet stipulated requirements in various job specialties over time has been treated in this way [23].

Applications of the optimal assignment model to other kinds of problems are very numerous. We mention one which involves the optimal depletion of inventory [7]. Suppose a stockpile consists of m items of the same kind, and that the age t_i of item i is known. Also known is a function $u(t)$ giving the utility for an item of age t when withdrawn from the stockpile, together with a schedule of demands specifying the times at which items will be required. The problem is to determine that order of item issue which maximizes the total utility while meeting the demand schedule. (For a concrete example, suppose one has m bottles of wine in his cellar, the ages of each being known, and consumes one bottle of wine weekly. The utility function for wine might appear as in Figure 7.2.) The utility of item i issued at time j is given by $u_{ij}=u(t_i+j)$, and hence the problem is to find an assignment of items to times which is optimal in terms of the u_{ij}.

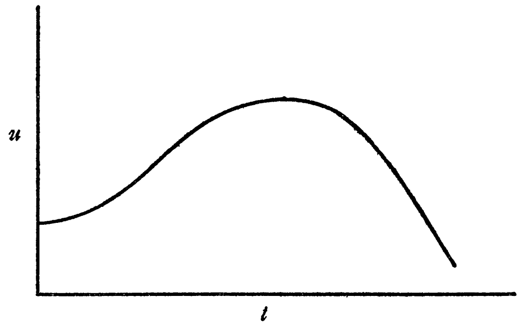

FIG. 7.2

If the utility function is convex, there is a simple rule for solving the problem: Issue the youngest item first, then the next youngest, and so on. This policy, sometimes called LIFO (last in first out), may be shown optimal here by a simple interchange argument. Similarly, if the utility function is concave, the reverse rule FIFO (first in first out) solves the problem. In general, however, no such simple rule works and an optimal assignment needs to be computed.

8. Production and inventory planning. Problems involving dynamic production and inventory programs for a single type item have received considerable study. A very simple deterministic problem in this category is the following. Suppose there are n periods of time with known period demands $b_1, b_2, \cdots, b_n$ for the item, that the unit cost of production in period i is p_i, and the unit cost of storage from period i to $i+1$ is s_i. What pattern of production and storage meets the demands at minimum cost?

The network shown in Figure 8.1 assumes that production in period i can be used to satisfy demand in period i. The ith "production" arc (source arc) has infinite capacity and cost p_i; the ith "storage" arc has infinite capacity and cost s_i; the ith "demand" arc (sink arc) has capacity b_i and zero cost. The problem then is to determine a flow of amount $v=\sum_i b_i$ from source to sink that min-

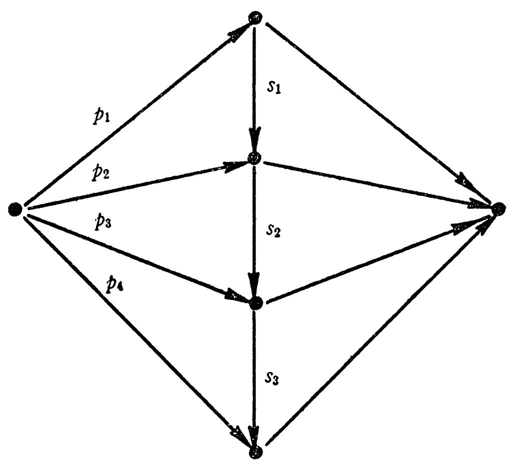

FIG. 8.1

imizes cost. Clearly production and storage capacities may be introduced if desired. But if these are left infinite, there is a very simple rule for solving the problem. For the *i*th demand, compare the chain costs

$$\begin{cases} p_1 + s_1 + \cdots + s_{i-1} \\ p_2 + s_2 + \cdots + s_{i-1} \\ \cdots \\ p_i \end{cases}$$

and take the smallest of these. Then send b_i units of flow along the corresponding chain. An almost equally simple rule works in case each period's production cost is convex in the number of items produced [**30**].

If it is assumed that demands do not have to be satisfied, but that unfulfilled demand in period i results in a penalty cost c_i per unit, we may place a flow cost $-c_i$ on the *i*th sink arc and solve the minimal cost flow problem parametrically in the flow amount v, selecting that v which gives the least cost.

9. Optimal capacity scheduling. The model of this section, proposed and studied in [**41**], is a rather general one which can be shown to include several of those previously discussed here. One version of the model is described in [**41**] as follows: "A decision maker must contract for warehousing capacity over n time periods, the minimal capacity requirement for each period being deterministically specified. His economic problem arises because savings may possibly accrue by his undertaking long-term leasing or contracting at favorable periods of time, even though such commitments may necessitate leaving some of the capacity idle during several periods."

To put the problem mathematically, let d_i be the minimal capacity demand in period i. Let x_{ij}, $i<j$, be the number of units of capacity acquired at the begin-

ning of period i, available for possible use during periods $i, i+1, \cdots, j-1$, and relinquished at the beginning of period j, and let a_{ij} be the associated unit cost. Then the problem is to find $x_{ij} \geqq 0$ that minimize

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n+1} a_{ij}x_{ij} \tag{9.1}$$

subject to the constraints

$$\sum_{i=1}^{k} \sum_{j=k+1}^{n+1} x_{ij} \geqq d_k, \qquad k = 1, 2, \cdots, n. \tag{9.2}$$

To see that the constraints (9.2) describe flows, first rewrite (9.2) as

$$\sum_{i=1}^{k} \sum_{j=k+1}^{n+1} x_{ij} - y_k = d_k, \qquad y_k \geqq 0. \tag{9.3}$$

Next successively subtract the $(k-1)$-st equation from the kth, $k=n, n-1, \cdots, 2$, to obtain an equivalent system of constraints. The result is

$$\begin{aligned} &\sum_{j=2}^{n} x_{ij} - y_1 = d_1, \\ &\cdots \\ &\sum_{j=k+1}^{n} x_{kj} - \sum_{i=1}^{k-1} x_{ik} + y_{k-1} - y_k = d_k - d_{k-1}, \qquad k = 2, \cdots, n, \\ &\cdots \\ -&\sum_{i=1}^{n-1} x_{in} + y_n = -d_n, \end{aligned} \tag{9.4}$$

subject to which the linear form (9.1) is to be minimized.

The corresponding network is shown in Figure 9.1 below.

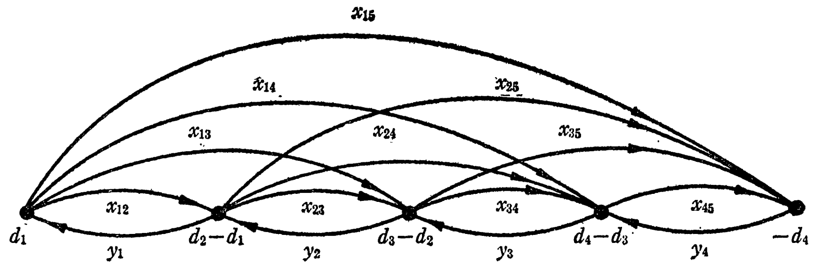

FIG. 9.1

Here x_{ij} is the flow in (i, j) and a_{ij} is the cost per unit of flow; y_i is the flow in $(i+1, i)$, with $a_{i+1,i}=0$. Nodes i for which $d_i-d_{i-1}>0$ are sources with supplies d_i-d_{i-1}; nodes i for which $d_i-d_{i-1}<0$ are sinks with demands $-(d_i-d_{i-1})$.

Referring to Figure 4.2, Part II, and Figure 9.1 above, it is apparent that the equipment replacement problem can be viewed as a special case of capacity scheduling by taking $d_i = 1$, all i.

A number of other situations that can be interpreted in terms of capacity scheduling are described in [41]. Mentioned are models involving checkout and replacement of stochastically failing mechanisms; determination of economic lot sizes, product assortment, and batch queuing policies; labor-force planning; and multi-commodity warehousing decisions.

One application (the dynamic economic lot size model [43]) deals with the problem described in section 8, where production costs now are concave functions of the number of items produced, and demands must be satisfied. Generally speaking, concavity makes minimization problems difficult, but here it can be seen that it is uneconomical to both produce in a period and carry inventory into the period, and hence there is an optimal policy of the following kind. The total time interval is broken into subintervals, with enough production at the beginning of each of these to satisfy its aggregate demand. Thus finding an optimal policy can be formulated in terms of capacity scheduling by letting a_{ij} be the total cost (including storage) associated with producing enough in period i to satisfy the demands for periods $i, i+1, \cdots, j-1$, and by taking all $d_i = 1$. In short, the problem has been reduced to one of finding a cheapest chain from source to sink in the network of Figure 9.1.

10. Minimal spanning trees.

A network combinatorial problem for which there is a particularly simple solution method is that of selecting a minimum spanning subtree from an undirected network each of whose links has a length or cost. We may illustrate this problem with the following example. Imagine a number n of cities on a map and suppose that the cost of installing a communication link between cities i and j is $a_{ij} = a_{ji} \geqq 0$. Each city must be connected, directly or indirectly, to all others, and this is to be done at minimum total cost. Clearly attention can be confined to trees (acyclic and connected networks of links), for if a connected network contains a cycle, removing one link of the cycle leaves the network connected and reduces cost. A minimal cost tree can be found easily as follows [37]. Select the cheapest link, then the next cheapest, and so on, being sure at each stage that no subset of the selected links forms a cycle. After $n-1$ selections, a cheapest tree has been constructed.

For example, in the network of Figure 10.1, this procedure might lead to the minimal cost tree shown in heavy links.

While it is not difficult to prove that this method solves the problem, it is nonetheless remarkable that being greedy at each stage works. There are few extremal combinatorial problems for which it does.

There is an interesting relation between the minimal spanning tree problem and another, which sounds on the surface to be very different. Think of the network in Figure 10.1 as being a highway map, where the number recorded beside each link is the maximum elevation encountered in traversing the link. Suppose someone who plans to drive from i to j dislikes high altitudes and hence

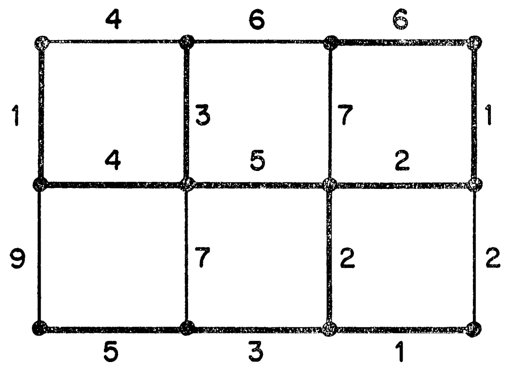

FIG. 10.1

wants to find a path connecting i and j that minimizes the maximum altitude. This problem is related to the shortest chain problem in the sense that methods for solving the latter are easily modified to solve the former, and this is so in either the directed or undirected case [40]. But it is also true in the undirected case that the minimal spanning tree solves the problem, and for all pairs of cities. That is, the unique path in the minimal spanning tree joining a pair of cities minimizes the path height [28]. Here we have used "path height" to mean the maximum number on the path.

There is also a min-max theorem concerning paths and cuts for this problem. Call the minimum link number in a cut the "cut height." Then it may be verified that the minimum height of paths joining two nodes is equal to the maximum height of cuts separating the two.

11. The traveling-salesman problem. Many problems that involve minimal connecting networks, and hence superficially resemble that of the last section, have no known simple solution procedures. For example, consider a network connecting a number of cities, with the length of each link being known, and imagine a traveling salesman who must start at some city, visit each of the others just once, and then return to the starting city. How does the salesman determine an itinerary that minimizes the total distance traveled?

A cycle that passes through every node of a network just once is usually called a Hamiltonian cycle. For brevity, we refer to it as a tour. Thus the problem asks for a shortest tour. (Of course the given network may contain no tour, but the existence question can be subsumed by considering all possible links present, those not corresponding to original ones having very large lengths.)

Not a great deal is known, either theoretically or computationally, about this problem, except that it is hard. On the theoretical side, for instance, there seem to be no simple conditions that are necessary and sufficient for a given network to contain a tour. On the computational side, while many methods for determining a shortest tour have been proposed, it is safe to assert that no one of these would guarantee that a problem involving 100 cities, say, could be solved in a reasonable length of time.

12. Minimal k-connected networks. In considering the synthesis of reliable communication networks with respect to link failure, the following question may be raised. Suppose given the complete, undirected network G on n nodes, where again each link of G has an associated number, the cost of installing a communication link between its end-nodes. For each $k=1, 2, \cdots, n-1$, find a minimal cost k-link-connected spanning subnetwork of G [20]. Here a k-link-connected network is one in which at least k links must be suppressed in order to disconnect the network. Another way of characterizing this property is to say that every pair of nodes is joined by at least k link-disjoint paths. Thus k might be thought of as the "reliability level" of the communication network, and the practical problem is to minimize cost while achieving a stipulated reliability level.

For $k=1$, the problem is that of section 10, and hence is readily solved. For $k=2$ and all link costs 1 or ∞, the problem becomes that of determining whether a given network (the subnetwork of unit cost links) contains a tour, and is thus already difficult. But if all link costs are equal, the answer is known. Here the problem is to determine the minimum number of links required in a k-link-connected network on n nodes. For $k \geqq 2$, there is an obvious lower bound on the number needed, namely $kn/2$ (for even kn) or $kn+1/2$ (for odd kn). These bounds can always be achieved.

This research was sponsored by the United States Air Force under Project RAND—Contract No. AF49 (638)-700.

References

1. R. G. Busacker and P. J. Gowen, A procedure for determining a family of minimal cost network flow patterns, O.R.O. Technical Paper, 15 (1961).
2. T. F. Cartaino and S. E. Dreyfus, Application of dynamic programming to the airplane minimum time-to-climb problem, Aero. Engr. Rev., 16 (1957) 74–77.
3. G. B. Dantzig, Application of the simplex method to a transportation problem, *Activity Analysis of Production and Allocation*, Cowles Commission Monograph 13, Wiley, New York, (1951) 359–373.
4. G. B. Dantzig and D. R. Fulkerson, Minimizing the number of tankers to meet a fixed schedule, Naval Res. Logist. Quart., 1 (1954) 217–222.
5. ———, On the max-flow min-cut theorem of networks, *Linear Inequalities and Related Systems*, Ann. of Math. Study 38, Princeton Univ. Press, (1956) 215–221.
6. G. B. Dantzig, D. R. Fulkerson, and S. Johnson, Solution of a large scale traveling salesman problem, Op. Res. 2, 393–410.
7. C. Derman and M. Klein, A note on the optimal depletion of inventory, Management Sci., 5 (1959) 210–214.
8. R. P. Dilworth, A decomposition theorem for partially ordered sets, Ann. of Math., 51 (1950) 161–166.
9. S. E. Dreyfus, A generalized equipment replacement study, J. Soc. Indust. Appl. Math., 8 (1960) 425–435.
10. L. R. Ford, Jr., Network flow theory, The RAND Corp., P-923 (1956).

11. L. R. Ford, Jr. and D. R. Fulkerson, Maximal flow through a network, Canad. J. Math., 8 (1956) 399–404.

12. ———, A simple algorithm for finding maximal network flows and an application to the Hitchcock problem, Canad. J. Math., 9 (1957) 210–218.

13. ———, A suggested computation for maximal multi-commodity network flows, Management Sci., 5 (1958) 97–101.

14. ———, Constructing maximal dynamic flows from static flows, Op. Res., 6 (1958) 419–433.

15. ———, *Flows in networks*, Princeton Univ. Press, 1962, 194 p.

16. D. R. Fulkerson, Increasing the capacity of a network: The parametric budget problem, Management Sci., 5 (1959) 472–483.

17. ———, An out-of-kilter method for minimal cost flow problems, J. Soc. Indust. Appl. Math., 9 (1961) 18–27.

18. ———, A network flow computation for project cost curves, Management Sci., 7 (1961) 167–178.

19. ———, A network flow feasibility theorem and combinatorial applications, Canad. J. Math., 11 (1959) 440–451.

20. D. R. Fulkerson and L. S. Shapley, Minimal k-arc-connected graphs, The RAND Corp., P-2371 (1961) 11 p.

21. D. Gale, A theorem on flows in networks, Pac. J. Math., 7 (1957) 1073–1082.

22. R. E. Gomory and T. C. Hu, Multi-terminal network flows, J. Soc. Indust. Appl. Math., 9 (1961) 551–571.

23. W. Gorham, An application of a network flow model to personnel planning, The RAND Corp., RM-2587 (1960) 85 p.

24. P. Hall, On representatives of subsets, J. Lond. Math. Soc., 10 (1935) 26–30.

25. F. L. Hitchcock, The distribution of a product from several sources to numerous localities, J. Math. and Phys., 20 (1941) 224–230.

26. A. J. Hoffman, Some recent applications of the theory of linear inequalities to extremal combinatorial analysis, Proc. Symposia Applied Math., 10 (1960).

27. T. C. Hu, Parallel sequencing and assembly line problems, Op. Res., 9 (1961) 841–849.

28. ———, The maximum capacity route problem, Op. Res., 9 (1961) 898–900.

29. W. S. Jewell, Optimal flow through networks with gains, Proc. Second International Conf. on Oper. Res., Aix-en-Provence, France 1960.

30. S. M. Johnson, Sequential production planning over time at minimum cost, Man. Sci., 3 (1957) 435–437.

31. ———, Optimal two- and three-stage production schedules with setup times included, Naval Res. Log. Q., 1 (1954) 61–68.

32. L. Kantorovitch and M. K. Gavurin, The application of mathematical methods in problems of freight flow analysis, *Collection of Problems Concerned with Increasing the Effectiveness of Transports*, Publ. of the Akad. Nauk SSSR, Moscow-Leningrad (1949) 110–138.

33. J. E. Kelley, Critical path planning and scheduling: mathematical basis, Op. Res., 9 (1961) 296–321.

34. T. C. Koopmans and S. Reiter, A model of transportation, *Activity Analysis of Production and Allocation*, Cowles Commission Monograph 13, Wiley, New York, 1951, 229–259.

35. D. König, *Theorie der endlichen und unendlichen Graphen*, Chelsea, 1950, 258 p.

36. H. W. Kuhn, The Hungarian method for the assignment problem, Naval Res. Log. Q., 2 (1955) 83–97.

37. J. B. Kruskal, Jr., On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Amer. Math. Soc., 7 (1956) 48–50.

38. G. J. Minty, Monotone networks, Proc. Roy. Soc., A, 257 (1960) 194–212.

39. A. Orden, The transshipment problem, Man. Sci., 3 (1956) 276–285.

40. M. Pollack, The maximum capacity route through a network, Op. Res., 8 (1960) 733–736.

41. A. F. Veinott, Jr. and H. M. Wagner, Optimal capacity scheduling—I and II, Op. Res., 10 (1962) 518–547.

42. D. F. Votaw, Jr. and A. Orden, The personnel assignment problem, Project SCOOP, Manual 10 (1952) 155–163.

43. H. M. Wagner and T. M. Whitin, Dynamic version of the economic lot-size model, Man. Sci., 5 (1958) 89–96.

QUADRATIC CONGRUENCES WITH AN ODD NUMBER OF SUMMANDS

ECKFORD COHEN, University of Tennessee

1. Introduction. It is well known [6, Chapters 6–11] that the formulas for the number of representations of an integer as a sum of an odd number of squares are, in general, more complicated than in the case of an even number. While this principle remains valid when the representations are taken with respect to a modulus, there are several important special cases of quadratic congruences with an odd number of unknowns for which the number of solutions is expressible by a quite simple formula. It is the purpose of this paper to exhibit some of these cases.

Let n be an integer and r a positive integer. Also, let $a_1, \cdots, a_{2m+1}$ be integers prime to r for $m \geqq 0$, and with $s = 2m+1$, place

$$\alpha = (-1)^m a_1 \cdots a_s, \qquad (\alpha, r) = 1. \tag{1}$$

Denote by $B_m(n, r)$ the number of distinct solutions (mod r) of the congruence,

$$n \equiv a_1 x_1^2 + \cdots + a_{2m+1} x_{2m+1}^2 \pmod{r}, \tag{2}$$

the a_i being subject to the condition in (1). We shall use Q to represent the set of square-free integers and $q(n)$ the characteristic function of Q, defined by $q(n) = 1$ if $n \in Q$ and 0 if $n \notin Q$. The Jacobi symbol (mod r) will be denoted $\psi_r(n)$.

A particularly simple formula for $B_m(n, r)$ occurs when r is odd and n is relatively prime to r. In fact, we have the following elegant formula due, at least in an implicit form, to Minkowski:

$$B_m(n, r) = r^{2m} \sum_{d \mid r} \psi_d(\alpha n) \frac{q(d)}{d^m}, \quad (n, r) = 1, \ r \text{ odd}. \tag{3}$$

For an account of Minkowski's work on quadratic congruences, we refer to Bachmann [1, Chapter 7] and in particular to Sections 2 and 7 of that reference. Regarding the case $s = 1$ of (3), in an equivalent form, we mention Landau [7, Theorems 71 and 87]. The history of the earlier development of the theory of quadratic congruences may be found in Dickson [6, Chapter 10].