Network Models

8

There are several kinds of linear-programming models that exhibit a special structure that can be exploited in the construction of efficient algorithms for their solution. The motivation for taking advantage of their structure usually has been the need to solve larger problems than otherwise would be possible to solve with existing computer technology. Historically, the first of these special structures to be analyzed was the transportation problem, which is a particular type of network problem. The development of an efficient solution procedure for this problem resulted in the first widespread application of linear programming to problems of industrial logistics. More recently, the development of algorithms to efficiently solve particular large-scale systems has become a major concern in applied mathematical programming.

Network models are possibly still the most important of the special structures in linear programming. In this chapter, we examine the characteristics of network models, formulate some examples of these models, and give one approach to their solution. The approach presented here is simply derived from specializing the rules of the simplex method to take advantage of the structure of network models. The resulting algorithms are extremely efficient and permit the solution of network models so large that they would be impossible to solve by ordinary linear-programming procedures. Their efficiency stems from the fact that a pivot operation for the simplex method can be carried out by simple addition and subtraction without the need for maintaining and updating the usual tableau at each iteration. Further, an added benefit of these algorithms is that the optimal solutions generated turn out to be *integer* if the relevant constraint data are integer.

8.1 THE GENERAL NETWORK-FLOW PROBLEM

A common scenario of a network-flow problem arising in industrial logistics concerns the distribution of a single homogeneous product from plants (origins) to consumer markets (destinations). The total number of units produced at each plant and the total number of units required at each market are assumed to be known. The product need not be sent directly from source to destination, but may be routed through intermediary points reflecting warehouses or distribution centers. Further, there may be capacity restrictions that limit some of the shipping links. The objective is to minimize the variable cost of producing and shipping the products to meet the consumer demand.

The sources, destinations, and intermediate points are collectively called *nodes* of the network, and the transportation links connecting nodes are termed *arcs*. Although a production/distribution problem has been given as the motivating scenario, there are many other applications of the general model. Table E8.1 indicates a few of the many possible alternatives.

A numerical example of a network-flow problem is given in Fig 8.1. The nodes are represented by numbered circles and the arcs by arrows. The arcs are assumed to be *directed* so that, for instance, material can be sent from node 1 to node 2, but not from node 2 to node 1. Generic arcs will be denoted by i–j, so that 4–5 means the arc *from* node 4 *to* node 5. Note that some pairs of nodes, for example 1 and 5, are not connected directly by an arc.

Table 8.1 Examples of Network Flow Problems

	Urban transportation	*Communication systems*	*Water resources*
Product	Buses, autos, etc.	Messages	Water
Nodes	Bus stops, street intersections	Communication centers, relay stations	Lakes, reservoirs, pumping stations
Arcs	Streets (lanes)	Communication channels	Pipelines, canals, rivers

Figure 8.1 exhibits several additional characteristics of network flow problems. First, a flow capacity is assigned to each arc, and second, a per-unit cost is specified for shipping along each arc. These characteristics are shown next to each arc. Thus, the flow on arc 2–4 must be between 0 and 4 units, and each unit of flow on this arc costs \$2.00. The ∞'s in the figure have been used to denote unlimited flow capacity on arcs 2–3 and 4–5. Finally, the numbers in parentheses next to the nodes give the material supplied or demanded at that node. In this case, node 1 is an origin or *source node* supplying 20 units, and nodes 4 and 5 are destinations or *sink nodes* requiring 5 and 15 units, respectively, as indicated by the negative signs. The remaining nodes have no net supply or demand; they are intermediate points, often referred to as *transshipment nodes*.

The objective is to find the minimum-cost flow pattern to fulfill demands from the source nodes. Such problems usually are referred to as *minimum-cost flow* or *capacitated transshipment* problems. To transcribe the problem into a formal linear program, let

$$x_{ij} = \text{Number of units shipped from node } i \text{ to } j \text{ using arc } i\text{–}j.$$

Then the tabular form of the linear-programming formulation associated with the network of Fig. 8.1 is as shown in Table 8.2.

The first five equations are flow-balance equations at the nodes. They state the conservation-of-flow law,

$$\begin{pmatrix}\text{Flow out}\\ \text{of a node}\end{pmatrix} - \begin{pmatrix}\text{Flow into}\\ \text{a node}\end{pmatrix} = \begin{pmatrix}\text{Net supply}\\ \text{at a node}\end{pmatrix}.$$

As examples, at nodes 1 and 2 the balance equations are:

$$\begin{aligned} x_{12} + x_{13} &= 20 \\ x_{23} + x_{24} + x_{25} - x_{12} &= 0. \end{aligned}$$

It is important to recognize the special structure of these balance equations. Note that there is one balance equation for each node in the network. The flow variables x_{ij} have only 0, +1, and −1 coefficients in these

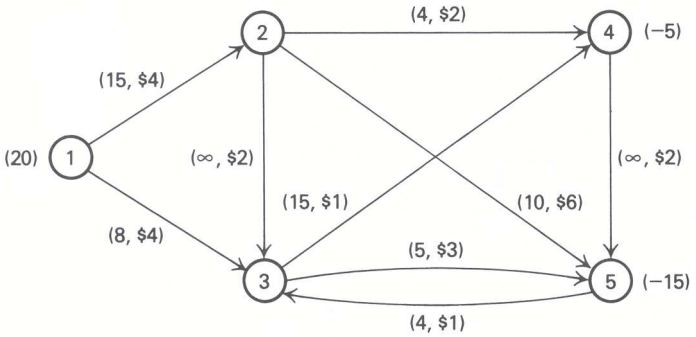

Figure 8.1 Minimum-cost flow problem.

Table 8.2 Tableau for Minimum-Cost Flow Problem

	x_{12}	x_{13}	x_{23}	x_{24}	x_{25}	x_{34}	x_{35}	x_{45}	x_{53}	*Righthand side*
Node 1	1	1								20
Node 2	−1		1	1	1					0
Node 3		−1	−1			1	1		−1	0
Node 4				−1		−1		1		−5
Node 5					−1		−1	−1	1	−15
Capacities	15	8	∞	4	10	15	5	∞	4	
Objective function	4	4	2	2	6	1	3	2	1	(Min)

equations. Further, each variable appears in exactly two balance equations, once with a $+1$ coefficient, corresponding to the node from which the arc emanates; and once with a -1 coefficient, corresponding to the node upon which the arc is incident. This type of tableau is referred to as a *node–arc incidence matrix*; it completely describes the physical layout of the network. It is this particular structure that we shall exploit in developing specialized, efficient algorithms.

The remaining two rows in the table give the upper bounds on the variables and the cost of sending one unit of flow across an arc. For example, x_{12} is constrained by $0 \leq x_{12} \leq 15$ and appears in the objective function as $2x_{12}$. In this example the lower bounds on the variables are taken implicitly to be zero, although in general there may also be nonzero lower bounds.

This example is an illustration of the following general *minimum-cost flow* problem with n nodes:

$$\text{Minimize } z = \sum_i \sum_j c_{ij} x_{ij},$$

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = b_i \quad (i = 1, 2, \ldots, n), \qquad \text{[Flow balance]}$$

$$l_{ij} \leq x_{ij} \leq u_{ij}. \qquad \text{[Flow capacities]}$$

The summations are taken only over the arcs in the network. That is, the first summation in the ith flow-balance equation is over all nodes j such that i–j is an arc of the network, and the second summation is over all nodes k such that k–i is an arc of the network. The objective function summation is over arcs i–j that are contained in the network and represents the total cost of sending flow over the network. The ith balance equation is interpreted as above: it states that the flow out of node i minus the flow into i must equal the net supply (demand if b_i is negative) at the node. u_{ij} is the upper bound on arc flow and may be $+\infty$ if the capacity on arc i–j is unlimited. ℓ_{ij} is the lower bound on arc flow and is often taken to be zero, as in the previous example. In the following sections we shall study variations of this general problem in some detail.

8.2 SPECIAL NETWORK MODELS

There are a number of interesting special cases of the minimum-cost flow model that have received a great deal of attention. This section introduces several of these models, since they have had a significant impact on the development of a general network theory. In particular, algorithms designed for these specific models have motivated solution procedures for the more general minimum-cost flow problem.

The Transportation Problem

The transportation problem is a network-flow model without intermediate locations. To formulate the problem, let us define the following terms:

a_i = Number of units available at source i ($i = 1, 2, \ldots, m$);
b_j = Number of units required at destination j ($j = 1, 2, \ldots, n$);
c_{ij} = Unit transportation cost from source i to destination j
($i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$).

For the moment, we assume that the total product availability is equal to the total product requirements; that is,

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j.$$

Later we will return to this point, indicating what to do when this supply–demand balance is not satisfied. If we define the decision variables as:

x_{ij} = Number of units to be distributed from source i to destination j
($i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$),

we may then formulate the transportation problem as follows:

$$\text{Minimize } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}, \tag{1}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1, 2, \ldots, m), \tag{2}$$

$$\sum_{i=1}^{m} (-x_{ij}) = -b_j \qquad (j = 1, 2, \ldots, n), \tag{3}$$

$$x_{ij} \geq 0 \qquad (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n) \tag{4}$$

Expression (1) represents the minimization of the total distribution cost, assuming a linear cost structure for shipping. Equation (2) states that the amount being shipped from source i to all possible destinations should be equal to the total availability, a_i, at that source. Equation (3) indicates that the amounts being shipped to destination j from all possible sources should be equal to the requirements, b_j, at that destination. Usually Eq. (3) is written with positive coefficients and righthand sides by multiplying through by minus one.

Let us consider a simple example. A compressor company has plants in three locations: Cleveland, Chicago, and Boston. During the past week the total production of a special compressor unit out of each plant has been 35, 50, and 40 units respectively. The company wants to ship 45 units to a distribution center in Dallas, 20 to Atlanta, 30 to San Francisco, and 30 to Philadelphia. The unit production and distribution costs from each plant to each distribution center are given in Table E8.3. What is the best shipping strategy to follow?

The linear-programming formulation of the corresponding transportation problem is:

$$\begin{aligned}\text{Minimize } z = {} & 8x_{11} + 6x_{12} + 10x_{13} + 9x_{14} + 9x_{21} + 12x_{22} + 13x_{23} \\ & + 7x_{24} + 14x_{31} + 9x_{32} + 16x_{33} + 5x_{34},\end{aligned}$$

Table 8.3 Unit Production and Shipping Costs

	Distribution centers				
Plants	Dallas	Atlanta	San Francisco	Phila.	*Availability* (units)
Cleveland	8	6	10	9	35
Chicago	9	12	13	7	50
Boston	14	9	16	5	40
Requirements (units)	45	20	30	30	[125]

subject to:

$$\begin{aligned}
x_{11} + x_{12} + x_{13} + x_{14} &= 35,\\
x_{21} + x_{22} + x_{23} + x_{24} &= 50,\\
x_{31} + x_{32} + x_{33} + x_{34} &= 40,\\
-x_{11} - x_{21} - x_{31} &= -45,\\
-x_{12} - x_{22} - x_{32} &= -20,\\
-x_{13} - x_{23} - x_{33} &= -30,\\
-x_{14} - x_{24} - x_{34} &= -30,
\end{aligned}$$

$$x_{ij} \geq 0 \qquad \text{(i=1,2,3; j=1,2,3,4).}$$

Because there is no ambiguity in this case, the same numbers normally are used to designate the origins and destinations. For example, x_{11} denotes the flow from source 1 to destination 1, although these are two distinct nodes. The network corresponding to this problem is given in Fig. 8.2.

The Assignment Problem

Though the transportation model has been cast in terms of material flow from sources to destinations, the model has a number of additional applications. Suppose, for example, than n people are to be assigned to n jobs and that c_{ij} measures the performance of person i in job j. If we let

$$x_{ij} = \begin{cases} 1 & \text{if person } i \text{ is assigned to job } j,\\ 0 & \text{otherwise,} \end{cases}$$

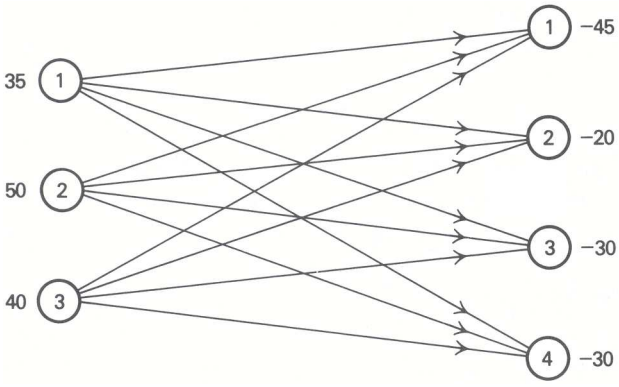

Figure 8.2 Transportation network.

we can find the optimal assignment by solving the optimization problem:

$$\text{Maximize } z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij},$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad (i = 1, 2, \ldots, n),$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n),$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

The first set of constraints shows that each person is to be assigned to exactly one job and the second set of constraints indicates that each job is to be performed by one person. If the second set of constraints were multiplied by minus one, the equations of the model would have the usual network interpretation.

As stated, this assignment problem is formally an integer program, since the decision variables x_{ij} are restricted to be zero or one. However, if these constraints are replaced by $x_{ij} \geq 0$, the model becomes a special case of the transportation problem, with one unit available at each source (person) and one unit required by each destination (job). As we shall see, network-flow problems have integer solutions, and therefore formal specification of integrality constraints is unnecessary. Consequently, application of the simplex method, or most network-flow algorithms, will solve such integer problems directly.

The Maximal Flow Problem

For the maximal flow problem, we wish to send as much material as possible from a specified node s in a network, called the *source*, to another specified node t, called the *sink*. No costs are associated with flow. If v denotes the amount of material sent from node s to node t and x_{ij} denotes the flow from node i to node j over arc i–j the formulation is:

$$\text{Maximize } v,$$

subject to:

$$\sum_{j} x_{ij} - \sum_{k} x_{ki} = \begin{cases} v & \text{if } i = s \text{ (source)}, \\ -v & \text{if } i = t \text{ (sink)}, \\ 0 & \text{otherwise}, \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

As usual, the summations are taken only over the arcs in the network. Also, the upper bound u_{ij} for the flow on arc i–j is taken to be $+\infty$ if arc i–j has unlimited capacity. The interpretation is that v units are supplied at s and consumed at t.

Let us introduce a fictitious arc t–s with unlimited capacity; that is, $u_{ts} = +\infty$. Now x_{ts} represents the variable v, since x_{ts} simply returns the v units of flow from node t back to node s, and no formal external supply of material occurs. With the introduction of the arc t–s, the problem assumes the following special form of the general network problem:

$$\text{Maximize } x_{ts},$$

subject to:

$$\sum_{j} x_{ij} - \sum_{k} x_{ki} = 0 \qquad (i = 1, 2, \ldots, n),$$

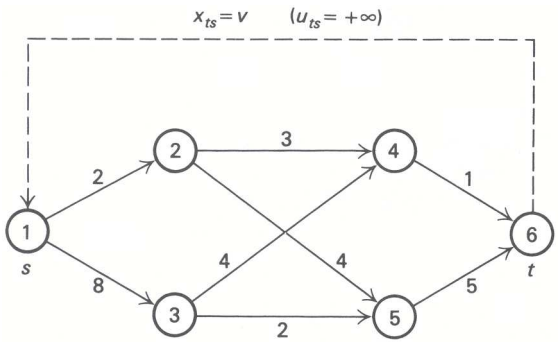

Figure 8.3

$$0 \leq x_{ij} \leq u_{ij} \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

Let us again consider a simple example. A city has constructed a piping system to route water from a lake to the city reservoir. The system is now underutilized and city planners are interested in its overall capacity. The situation is modeled as finding the maximum flow from node 1, the lake, to node 6, the reservoir, in the network shown in Fig. 8.3.

The numbers next to the arcs indicate the maximum flow capacity (in 100,000 gallons/day) in that section of the pipeline. For example, at most 300,000 gallons/day can be sent from node 2 to node 4. The city now sends 100,000 gallons/day along each of the paths 1–2–4–6 and 1–3–5–6. What is the maximum capacity of the network for shipping water from node 1 to node 6?

The Shortest-Path Problem

The shortest-path problem is a particular network model that has received a great deal of attention for both practical and theoretical reasons. The essence of the problem can be stated as follows: Given a network with distance c_{ij} (or travel time, or cost, etc.) associated with each arc, find a path through the network from a particular origin (source) to a particular destination (sink) that has the shortest total distance. The simplicity of the statement of the problem is somewhat misleading, because a number of important applications can be formulated as shortest- (or longest-) path problems where this formulation is not obvious at the outset. These include problems of equipment replacement, capital investment, project scheduling, and inventory planning. The theoretical interest in the problem is due to the fact that it has a special structure, in addition to being a network, that results in very efficient solution procedures. (In Chapter 11 on dynamic programming, we illustrate some of these other procedures.) Further, the shortest-path problem often occurs as a subproblem in more complex situations, such as the subproblems in applying decomposition to traffic-assignment problems or the group-theory problems that arise in integer programming.

In general, the formulation of the shortest-path problem is as follows:

$$\text{Minimize } z = \sum_i \sum_j c_{ij} x_{ij},$$

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} 1 & \text{if } i = s \text{ (source)}, \\ 0 & \text{otherwise}, \\ -1 & \text{if } i = t \text{ (sink)} \end{cases}$$

$$x_{ij} \geq 0 \quad \text{for all arcs } i\text{–}j \text{ in the network.}$$

We can interpret the shortest-path problem as a network-flow problem very easily. We simply want to send one unit of flow from the source to the sink at minimum cost. At the source, there is a net supply of one unit; at the sink, there is a net demand of one unit; and at all other nodes there is no net inflow or outflow.

As an elementary illustration, consider the example given in Fig. 8.4, where we wish to find the shortest distance from node 1 to node 8. The numbers next to the arcs are the distance over, or cost of using, that arc. For the network specified in Fig. 8.4, the linear-programming tableau is given in Tableau 1.

Tableau 8.4 Node–Arc Incidence Tableau for a Shortest-PathProblem

	x_{12}	x_{13}	x_{24}	x_{25}	x_{32}	x_{34}	x_{37}	x_{45}	x_{46}	x_{47}	x_{52}	x_{56}	x_{58}	x_{65}	x_{67}	x_{68}	x_{76}	x_{78}	*Relations*	RHS
Node 1	1	1																	=	1
Node 2	−1		1	1	−1						−1								=	0
Node 3		−1			1	1	1												=	0
Node 4			−1			−1		1	1	1									=	0
Node 5				−1				−1			1	1	1	−1					=	0
Node 6									−1			−1		1	1	1	−1		=	0
Node 7							−1			−1					−1		1	1	=	0
Node 8													−1			−1		−1	=	−1
Distance	5.1	3.4	0.5	2.0	1.0	1.5	5.0	2.0	3.0	4.2	1.0	3.0	6.0	1.5	0.5	2.2	2.0	2.4	=	z (min)

8.3 THE CRITICAL-PATH METHOD

The Critical-Path Method (CPM) is a project-management technique that is used widely in both government and industry to analyze, plan, and schedule the various tasks of complex projects. CPM is helpful in identifying which tasks are critical for the execution of the overall project, and in scheduling all the tasks in accordance with their prescribed *precedence relationships* so that the total project completion date is minimized, or a target date is met at minimum cost.

Typically, CPM can be applied successfully in large construction projects, like building an airport or a highway; in large maintenance projects, such as those encountered in nuclear plants or oil refineries; and in complex research-and-development efforts, such as the development, testing, and introduction of a new product. All these projects consist of a well specified collection of tasks that should be executed in a certain prescribed sequence. CPM provides a methodology to define the interrelationships among the tasks, and to determine the most effective way of scheduling their completion.

Although the mathematical formulation of the scheduling problem presents a network structure, this is not obvious from the outset. Let us explore this issue by discussing a simple example.

Suppose we consider the scheduling of tasks involved in building a house on a foundation that already exists. We would like to determine in what sequence the tasks should be performed in order to minimize

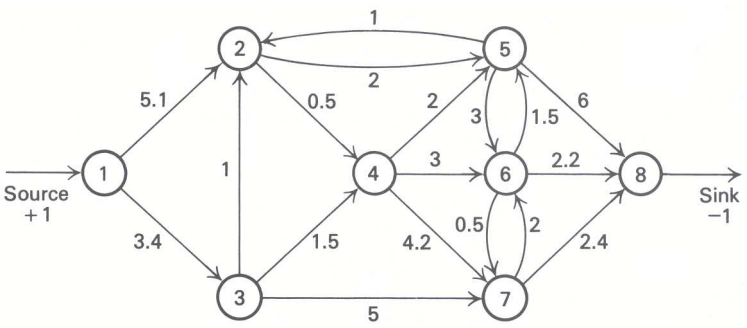

Figure 8.4 Network for a shortest-path problem.