# 9

# Network Optimization Models

Networks arise in numerous settings and in a variety of guises. Transportation, electrical, and communication networks pervade our daily lives. Network representations also are widely used for problems in such diverse areas as production, distribution, project planning, facilities location, resource management, and financial planning—to name just a few examples. In fact, a network representation provides such a powerful visual and conceptual aid for portraying the relationships between the components of systems that it is used in virtually every field of scientific, social, and economic endeavor.

One of the most exciting developments in operations research (OR) in recent years has been the unusually rapid advance in both the methodology and application of network optimization models. A number of algorithmic breakthroughs have had a major impact, as have ideas from computer science concerning data structures and efficient data manipulation. Consequently, algorithms and software now are available *and are being used* to solve huge problems on a routine basis that would have been completely intractable two or three decades ago.

Many network optimization models actually are special types of *linear programming* problems. For example, both the transportation problem and the assignment problem discussed in the preceding chapter fall into this category because of their network representations presented in Figs. 8.3 and 8.5.

One of the linear programming examples presented in Sec. 3.4 also is a network optimization problem. This is the Distribution Unlimited Co. problem of how to distribute its goods through the distribution network shown in Fig. 3.13. This special type of linear programming problem, called the *minimum cost flow* problem, is presented in Sec. 9.6. We shall return to this specific example in that section and then solve it with network methodology in the following section.

The third linear programming case study presented in Sec. 3.5 also features an application of the minimum cost flow problem. This case study involved planning the supply, distribution, and marketing of goods at Citgo Petroleum Corp. The OR team at Citgo developed an optimization-based decision support system, using a minimum cost flow problem model for each product, and coupled this system with an on-line corporate database. Each product's model has about 3,000 equations (nodes) and 15,000 variables (arcs), which is a very modest size by today's standards for the application of network opti-

mization models. The model takes in all aspects of the business, helping management decide everything from run levels at the various refineries to what prices to pay or charge. A network representation is essential because of the flow of goods through several stages: purchase of crude oil from various suppliers, shipping it to refineries, refining it into various products, and sending the products to distribution centers and product storage terminals for subsequent sale. As discussed in Sec. 3.5, the modeling system enabled the company to reduce its petroleum products inventory by over $116 million with no drop in service levels. This resulted in a savings in annual interest of $14 million as well as improvements in coordination, pricing, and purchasing decisions worth another $2.5 million each year, along with many indirect benefits.

In this one chapter we only scratch the surface of the current state of the art of network methodology. However, we shall introduce you to four important kinds of network problems and some basic ideas of how to solve them (without delving into issues of data structures that are so vital to successful large-scale implementations). Each of the first three problem types—the *shortest-path problem,* the *minimum spanning tree problem,* and the *maximum flow problem*—has a very specific structure that arises frequently in applications.

The fourth type—the *minimum cost flow problem*—provides a unified approach to many other applications because of its far more general structure. In fact, this structure is so general that it includes as special cases both the shortest-path problem and the maximum flow problem as well as the transportation problem and the assignment problem from Chap. 8. Because the minimum cost flow problem is a special type of linear programming problem, it can be solved extremely efficiently by a streamlined version of the simplex method called the *network simplex method.* (We shall not discuss even more general network problems that are more difficult to solve.)

The first section introduces a prototype example that will be used subsequently to illustrate the approach to the first three of these problems. Section 9.2 presents some basic terminology for networks. The next four sections deal with the four problems in turn. Section 9.7 then is devoted to the network simplex method.

## 9.1 PROTOTYPE EXAMPLE

SEERVADA PARK has recently been set aside for a limited amount of sightseeing and backpack hiking. Cars are not allowed into the park, but there is a narrow, winding road system for trams and for jeeps driven by the park rangers. This road system is shown (without the curves) in Fig. 9.1, where location $O$ is the entrance into the park; other letters designate the locations of ranger stations (and other limited facilities). The numbers give the distances of these winding roads in miles.

The park contains a scenic wonder at station $T$. A small number of trams are used to transport sightseers from the park entrance to station $T$ and back.

The park management currently faces three problems. One is to determine which route from the park entrance to station $T$ has the *smallest total distance* for the operation of the trams. (This is an example of the shortest-path problem to be discussed in Sec. 9.3.)

A second problem is that telephone lines must be installed under the roads to establish telephone communication among all the stations (including the park entrance). Because the installation is both expensive and disruptive to the natural environment, lines

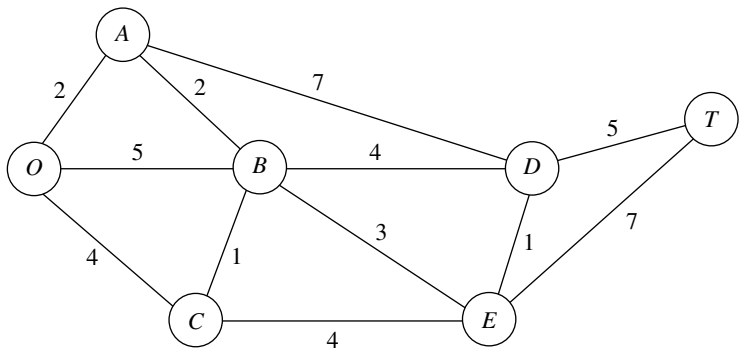



**FIGURE 9.1**
The road system for Seervada Park.

will be installed under just enough roads to provide some connection between every pair of stations. The question is where the lines should be laid to accomplish this with a *minimum* total number of miles of line installed. (This is an example of the minimum spanning tree problem to be discussed in Sec. 9.4.)

The third problem is that more people want to take the tram ride from the park entrance to station *T* than can be accommodated during the peak season. To avoid unduly disturbing the ecology and wildlife of the region, a strict ration has been placed on the number of tram trips that can be made on each of the roads per day. (These limits differ for the different roads, as we shall describe in detail in Sec. 9.5.) Therefore, during the peak season, various routes might be followed regardless of distance to increase the number of tram trips that can be made each day. The question pertains to how to route the various trips to *maximize* the number of trips that can be made per day without violating the limits on any individual road. (This is an example of the maximum flow problem to be discussed in Sec. 9.5.)

## 9.2 THE TERMINOLOGY OF NETWORKS

A relatively extensive terminology has been developed to describe the various kinds of networks and their components. Although we have avoided as much of this special vocabulary as we could, we still need to introduce a considerable number of terms for use throughout the chapter. We suggest that you read through this section once at the outset to understand the definitions and then plan to return to refresh your memory as the terms are used in subsequent sections. To assist you, each term is highlighted in **boldface** at the point where it is defined.

A network consists of a set of *points* and a set of *lines* connecting certain pairs of the points. The points are called **nodes** (or vertices); e.g., the network in Fig. 9.1 has seven nodes designated by the seven circles. The lines are called **arcs** (or links or edges or branches); e.g., the network in Fig. 9.1 has 12 arcs corresponding to the 12 roads in the road system. Arcs are labeled by naming the nodes at either end; for example, *AB* is the arc between nodes *A* and *B* in Fig. 9.1.

The arcs of a network may have a flow of some type through them, e.g., the flow of trams on the roads of Seervada Park in Sec. 9.1. Table 9.1 gives several examples of flow in typical networks. If flow through an arc is allowed in only one direction (e.g., a one-way street), the arc is said to be a **directed arc.** The direction is indicated by adding an arrowhead at the end of the line representing the arc. When a directed arc is labeled by listing two nodes it connects, the *from* node always is given before the *to* node; e.g., an arc that is directed *from* node *A* *to* node *B* must be labeled as *AB* rather than *BA*. Alternatively, this arc may be labeled as $A \rightarrow B$.

If flow through an arc is allowed in either direction (e.g., a pipeline that can be used to pump fluid in either direction), the arc is said to be an **undirected arc.** To help you distinguish between the two kinds of arcs, we shall frequently refer to undirected arcs by the suggestive name of **links.**

Although the flow through an undirected arc is allowed to be in either direction, we do assume that the flow will be one way in the direction of choice rather than having simultaneous flows in opposite directions. (The latter case requires the use of a *pair of directed arcs* in opposite directions.) However, in the process of making the decision on the flow through an undirected arc, it is permissible to make a sequence of assignments of flows in opposite directions, but with the understanding that the actual flow will be the *net flow* (the difference of the assigned flows in the two directions). For example, if a flow of 10 has been assigned in one direction and then a flow of 4 is assigned in the opposite direction, the actual effect is to *cancel* 4 units of the original assignment by reducing the flow in the original direction from 10 to 6. Even for a directed arc, the same technique sometimes is used as a convenient device to reduce a previously assigned flow. In particular, you are allowed to make a fictional assignment of flow in the "wrong" direction through a directed arc to record a reduction of that amount in the flow in the "right" direction.

A network that has only directed arcs is called a **directed network.** Similarly, if all its arcs are undirected, the network is said to be an **undirected network.** A network with a mixture of directed and undirected arcs (or even all undirected arcs) can be converted to a directed network, if desired, by replacing each undirected arc by a pair of directed arcs in opposite directions. (You then have the choice of interpreting the flows through each pair of directed arcs as being simultaneous flows in opposite directions or providing a net flow in one direction, depending on which fits your application.)

When two nodes are not connected by an arc, a natural question is whether they are connected by a series of arcs. A **path** between two nodes is a *sequence of distinct arcs* connecting these nodes. For example, one of the paths connecting nodes *O* and *T* in Fig. 9.1 is the sequence of arcs *OB–BD–DT* ($O \rightarrow B \rightarrow D \rightarrow T$), or vice versa. When some

**TABLE 9.1 Components of typical networks**

| Nodes | Arcs | Flow |
|---|---|---|
| Intersections | Roads | Vehicles |
| Airports | Air lanes | Aircraft |
| Switching points | Wires, channels | Messages |
| Pumping stations | Pipes | Fluids |
| Work centers | Materials-handling routes | Jobs |

of or all the arcs in the network are directed arcs, we then distinguish between directed paths and undirected paths. A **directed path** from node *i* to node *j* is a sequence of connecting arcs whose direction (if any) is *toward* node *j,* so that flow from node *i* to node *j* along this path is feasible. An **undirected path** from node *i* to node *j* is a sequence of connecting arcs whose direction (if any) can be *either* toward or away from node *j.* (Notice that a directed path also satisfies the definition of an undirected path, but not vice versa.) Frequently, an undirected path will have some arcs directed toward node *j* but others directed away (i.e., toward node *i*). You will see in Secs. 9.5 and 9.7 that, perhaps surprisingly, *undirected* paths play a major role in the analysis of *directed* networks.

To illustrate these definitions, Fig. 9.2 shows a typical directed network. (Its nodes and arcs are the same as in Fig. 3.13, where nodes *A* and *B* represent two factories, nodes *D* and *E* represent two warehouses, node *C* represents a distribution center, and the arcs represent shipping lanes.) The sequence of arcs *AB–BC–CE* ($A \rightarrow B \rightarrow C \rightarrow E$) is a directed path from node *A* to *E,* since flow toward node *E* along this entire path is feasible. On the other hand, *BC–AC–AD* ($B \rightarrow C \rightarrow A \rightarrow D$) is *not* a directed path from node *B* to node *D,* because the direction of arc *AC* is away from node *D* (on this path). However, $B \rightarrow C \rightarrow A \rightarrow D$ is an undirected path from node *B* to node *D,* because the sequence of arcs *BC–AC–AD* does *connect* these two nodes (even though the direction of arc *AC* prevents flow through this path).

As an example of the relevance of undirected paths, suppose that 2 units of flow from node *A* to node *C* had previously been assigned to arc *AC*. Given this previous assignment, it now is feasible to assign a smaller flow, say, 1 unit, to the entire undirected path $B \rightarrow C \rightarrow A \rightarrow D$, even though the direction of arc *AC* prevents positive flow through $C \rightarrow A$. The reason is that this assignment of flow in the "wrong" direction for arc *AC* actually just *reduces* the flow in the "right" direction by 1 unit. Sections 9.5 and 9.7 make heavy use of this technique of assigning a flow through an undirected path that includes arcs whose direction is opposite to this flow, where the real effect for these arcs is to reduce previously assigned positive flows in the "right" direction.

A path that begins and ends at the same node is called a **cycle.** In a *directed* network, a cycle is either a directed or an undirected cycle, depending on whether the path involved

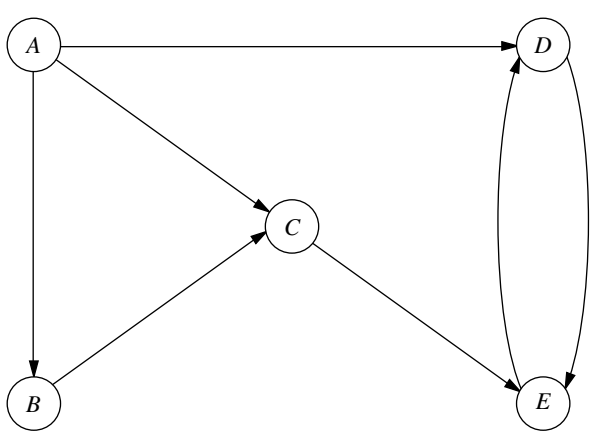



**FIGURE 9.2**
The distribution network for Distribution Unlimited Co., first shown in Fig. 3.13, illustrates a directed network.

is a directed or an undirected path. (Since a directed path also is an undirected path, a directed cycle is an undirected cycle, but not vice versa in general.) In Fig. 9.2, for example, *DE–ED* is a directed cycle. By contrast, *AB–BC–AC* is *not* a directed cycle, because the direction of arc *AC* opposes the direction of arcs *AB* and *BC*. On the other hand, *AB–BC–AC* is an undirected cycle, because $A \rightarrow B \rightarrow C \rightarrow A$ is an undirected path. In the undirected network shown in Fig. 9.1, there are many cycles, for example, *OA–AB–BC–CO*. However, note that the definition of *path* (a sequence of *distinct* arcs) rules out retracing one's steps in forming a cycle. For example, *OB–BO* in Fig. 9.1 does not qualify as a cycle, because *OB* and *BO* are two labels for the *same* arc (link). On the other hand, *DE–ED* is a (directed) cycle in Fig. 9.2, because *DE* and *ED* are distinct arcs.

Two nodes are said to be **connected** if the network contains at least one *undirected* path between them. (Note that the path does not need to be directed even if the network is directed.) A **connected network** is a network where every pair of nodes is connected. Thus, the networks in Figs. 9.1 and 9.2 are both connected. However, the latter network would not be connected if arcs *AD* and *CE* were removed.

Consider a connected network with $n$ nodes (e.g., the $n = 5$ nodes in Fig. 9.2) where all the arcs have been deleted. A "tree" can then be "grown" by adding one arc (or "branch") at a time from the original network in a certain way. The first arc can go anywhere to connect some pair of nodes. Thereafter, each new arc should be between a node that already is connected to other nodes and a new node not previously connected to any other nodes. Adding an arc in this way avoids creating a cycle and ensures that the number of connected nodes is 1 greater than the number of arcs. Each new arc creates a larger **tree,** which is a *connected network* (for some subset of the $n$ nodes) that contains *no undirected cycles.* Once the $(n - 1)$st arc has been added, the process stops because the resulting tree *spans* (connects) all $n$ nodes. This tree is called a **spanning tree,** i.e., a *connected network* for all $n$ nodes that contains *no undirected cycles.* Every spanning tree has exactly $n - 1$ arcs, since this is the *minimum* number of arcs needed to have a connected network and the *maximum* number possible without having undirected cycles.

Figure 9.3 uses the five nodes and some of the arcs of Fig. 9.2 to illustrate this process of growing a tree one arc (branch) at a time until a spanning tree has been obtained. There are several alternative choices for the new arc at each stage of the process, so Fig. 9.3 shows only one of many ways to construct a spanning tree in this case. Note, however, how each new added arc satisfies the conditions specified in the preceding paragraph. We shall discuss and illustrate spanning trees further in Sec. 9.4.

Spanning trees play a key role in the analysis of many networks. For example, they form the basis for the *minimum spanning tree problem* discussed in Sec. 9.4. Another prime example is that (feasible) spanning trees correspond to the BF solutions for the *network simplex method* discussed in Sec. 9.7.

Finally, we shall need a little additional terminology about *flows* in networks. The maximum amount of flow (possibly infinity) that can be carried on a directed arc is referred to as the **arc capacity.** For nodes, a distinction is made among those that are net generators of flow, net absorbers of flow, or neither. A **supply node** (or source node or source) has the property that the flow *out* of the node exceeds the flow *into* the node. The reverse case is a **demand node** (or sink node or sink), where the flow *into* the node exceeds the flow *out* of the node. A **transshipment node** (or intermediate node) satisfies *conservation of flow,* so flow in equals flow out.

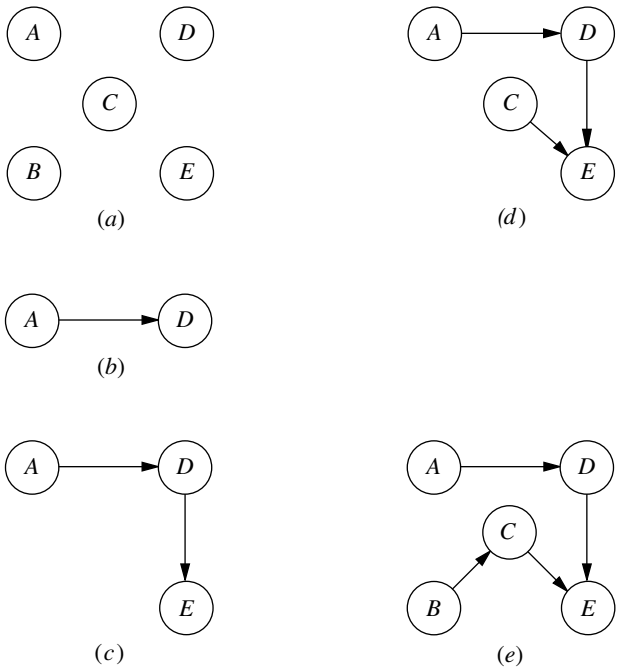



**FIGURE 9.3**
Example of growing a tree one arc at a time for the network of Fig. 9.2: (*a*) The nodes without arcs; (*b*) a tree with one arc; (*c*) a tree with two arcs; (*d*) a tree with three arcs; (*e*) a spanning tree.

## 9.3 THE SHORTEST-PATH PROBLEM

Although several other versions of the shortest-path problem (including some for directed networks) are mentioned at the end of the section, we shall focus on the following simple version. Consider an *undirected* and *connected* network with two special nodes called the *origin* and the *destination.* Associated with each of the *links* (undirected arcs) is a nonnegative *distance.* The objective is to find the shortest path (the path with the minimum total distance) from the origin to the destination.

A relatively straightforward algorithm is available for this problem. The essence of this procedure is that it fans out from the origin, successively identifying the shortest path to each of the nodes of the network in the ascending order of their (shortest) distances from the origin, thereby solving the problem when the destination node is reached. We shall first outline the method and then illustrate it by solving the shortest-path problem encountered by the Seervada Park management in Sec. 9.1.

### Algorithm for the Shortest-Path Problem.

*Objective of* n*th iteration:* Find the *n*th nearest node to the origin (to be repeated for $n = 1, 2, \ldots$ until the *n*th nearest node is the destination.

*Input for* n*th iteration:* $n - 1$ nearest nodes to the origin (solved for at the previous iterations), including their shortest path and distance from the origin. (These nodes, plus the origin, will be called *solved nodes;* the others are *unsolved nodes.*)

*Candidates for* n*th nearest node:* Each solved node that is directly connected by a link to one or more unsolved nodes provides *one* candidate—the unsolved node with the *shortest* connecting link. (Ties provide additional candidates.)

*Calculation of* n*th nearest node:* For each such solved node and its candidate, add the distance between them and the distance of the shortest path from the origin to this solved node. The candidate with the smallest such total distance is the *n*th nearest node (ties provide additional solved nodes), and its shortest path is the one generating this distance.

### Applying This Algorithm to the Seervada Park Shortest-Path Problem

The Seervada Park management needs to find the shortest path from the park entrance (node *O*) to the scenic wonder (node *T*) through the road system shown in Fig. 9.1. Applying the above algorithm to this problem yields the results shown in Table 9.2 (where the tie for the second nearest node allows skipping directly to seeking the fourth nearest node next). The first column (*n*) indicates the iteration count. The second column simply lists the *solved nodes* for beginning the current iteration after deleting the irrelevant ones (those not connected directly to any unsolved node). The third column then gives the *candidates* for the *n*th nearest node (the unsolved nodes with the *shortest* connecting link to a solved node). The fourth column calculates the distance of the shortest path from the origin to each of these candidates (namely, the distance to the solved node plus the link distance to the candidate). The candidate with the smallest such distance is the *n*th nearest node to the origin, as listed in the fifth column. The last two columns summarize the information for this *newest solved node* that is needed to proceed to subsequent iterations (namely, the distance of the shortest path from the origin to this node and the last link on this shortest path).

**TABLE 9.2** Applying the shortest-path algorithm to the Seervada Park problem

| *n* | Solved Nodes Directly Connected to Unsolved Nodes | Closest Connected Unsolved Node | Total Distance Involved | *n*th Nearest Node | Minimum Distance | Last Connection |
|---|---|---|---|---|---|---|
| 1 | *O* | *A* | 2 | *A* | 2 | *OA* |
| 2, 3 | *O*<br>*A* | *C*<br>*B* | 4<br>2 + 2 = 4 | *C*<br>*B* | 4<br>4 | *OC*<br>*AB* |
| 4 | *A*<br>*B*<br>*C* | *D*<br>*E*<br>*E* | 2 + 7 = 9<br>4 + 3 = 7<br>4 + 4 = 8 | *E* | 7 | *BE* |
| 5 | *A*<br>*B*<br>*E* | *D*<br>*D*<br>*D* | 2 + 7 = 9<br>4 + 4 = 8<br>7 + 1 = 8 | *D*<br>*D* | 8<br>8 | *BD*<br>*ED* |
| 6 | *D*<br>*E* | *T*<br>*T* | 8 + 5 = 13<br>7 + 7 = 14 | *T* | 13 | *DT* |

Now let us relate these columns directly to the outline given for the algorithm. The *input for* n*th iteration* is provided by the fifth and sixth columns for the preceding iterations, where the solved nodes in the fifth column are then listed in the second column for the current iteration after deleting those that are no longer directly connected to unsolved nodes. The *candidates for* n*th nearest node* next are listed in the third column for the current iteration. The *calculation of* n*th nearest node* is performed in the fourth column, and the results are recorded in the last three columns for the current iteration.

After the work shown in Table 9.2 is completed, the shortest path *from the destination to the origin* can be traced back through the last column of Table 9.2 as *either* $T \rightarrow D \rightarrow E \rightarrow B \rightarrow A \rightarrow O$ or $T \rightarrow D \rightarrow B \rightarrow A \rightarrow O$. Therefore, the two alternates for the shortest path *from the origin to the destination* have been identified as $O \rightarrow A \rightarrow B \rightarrow E \rightarrow D \rightarrow T$ and $O \rightarrow A \rightarrow B \rightarrow D \rightarrow T$, with a total distance of 13 miles on either path.

MPL

## Using Excel to Formulate and Solve Shortest-Path Problems

This algorithm provides a particularly efficient way of solving large shortest-path problems. However, some mathematical programming software packages do not include this algorithm. If not, they often will include the *network simplex method* described in Sec. 9.7, which is another good option for these problems.

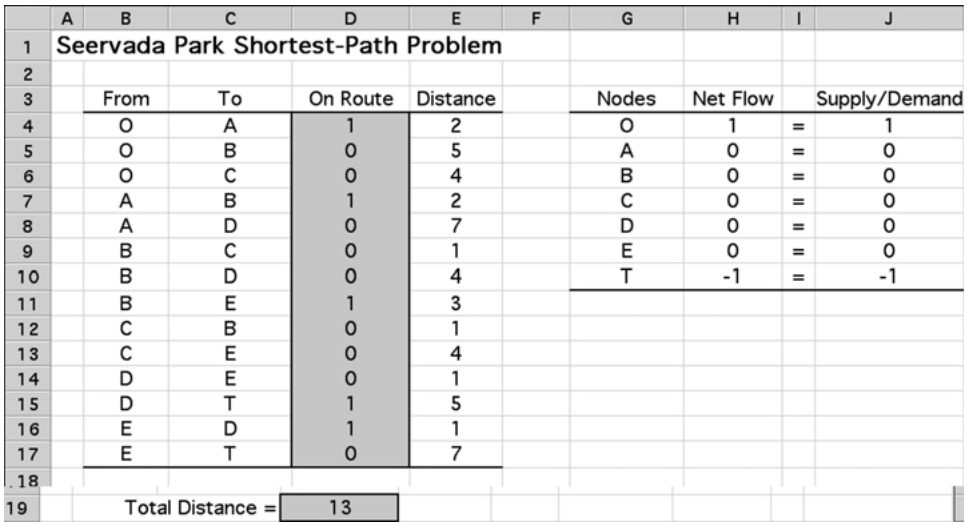
Since the shortest-path problem is a special type of linear programming problem, the general simplex method also can be used when better options are not readily available. Although not nearly as efficient as these specialized algorithms on large shortest-path problems, it is quite adequate for problems of even very substantial size (much larger than the Seervada Park problem). Excel, which relies on the general simplex method, provides a convenient way of formulating and solving shortest-path problems with dozens of arcs and nodes.

Figure 9.4 shows an appropriate spreadsheet formulation for the Seervada Park shortest-path problem. Rather than using the kind of formulation presented in Sec. 3.6 that uses a separate row for each functional constraint of the linear programming model, this formulation exploits the special structure by listing the *nodes* in column G and the *arcs* in columns B and C, as well as the distance (in miles) along each arc in column E. Since each *link* in the network is an *undirected arc,* whereas travel through the shortest path is in one direction, each link can be replaced by a pair of *directed* arcs in opposite directions. Thus, columns B and C together list both of the nearly vertical links in Fig. 9.1 (A–B and D–E) twice, once as a downward arc and once as an upward arc, since either direction might be on the chosen path. However, the other links are only listed as left-to-right arcs, since this is the only direction of interest for choosing a shortest path from the origin to the destination.

A trip from the origin to the destination is interpreted to be a "flow" of 1 on the chosen path through the network. The decisions to be made are which arcs should be included in the path to be traversed. A flow of 1 is assigned to an arc if it is included, whereas the flow is 0 if it is not included. Thus, the decision variables are

$$x_{ij} = \begin{cases} 0 & \text{if arc } i \rightarrow j \text{ is not included} \\ 1 & \text{if arc } i \rightarrow j \text{ is included} \end{cases}$$

for each of the arcs under consideration. The values of these decision variables are entered in the changing cells in column D (cells D4:D17).



| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Seervada Park Shortest-Path Problem | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | From | To | On Route | Distance | | Nodes | Net Flow | | Supply/Demand |
| 4 | | O | A | 1 | 2 | | O | 1 | = | 1 |
| 5 | | O | B | 0 | 5 | | A | 0 | = | 0 |
| 6 | | O | C | 0 | 4 | | B | 0 | = | 0 |
| 7 | | A | B | 1 | 2 | | C | 0 | = | 0 |
| 8 | | A | D | 0 | 7 | | D | 0 | = | 0 |
| 9 | | B | C | 0 | 1 | | E | 0 | = | 0 |
| 10 | | B | D | 0 | 4 | | T | -1 | = | -1 |
| 11 | | B | E | 1 | 3 | | | | | |
| 12 | | C | B | 0 | 1 | | | | | |
| 13 | | C | E | 0 | 4 | | | | | |
| 14 | | D | E | 0 | 1 | | | | | |
| 15 | | D | T | 1 | 5 | | | | | |
| 16 | | E | D | 1 | 1 | | | | | |
| 17 | | E | T | 0 | 7 | | | | | |
| 18 | | | | | | | | | | |
| 19 | | Total Distance = | | 13 | | | | | | |

**FIGURE 9.4**
A spreadsheet formulation for the Seervada Park shortest-path problem, where the changing cells (D4:D17) show the optimal solution obtained by the Excel Solver and the target cell (D19) gives the total distance (in miles) of this shortest path.

Each node can be thought of as having a flow of 1 passing through it if it is on the selected path, but no flow otherwise. The *net flow* generated at a node is the *flow out* minus the *flow in,* so the net flow is 1 at the origin, $-1$ at the destination, and 0 at every other node. These requirements for the net flows are specified in column J of Fig. 9.4. Using the equations at the bottom of the figure, each column H cell then calculates the *actual* net flow at that node by adding the flow out and subtracting the flow in. The corresponding constraints, H4:H10 = J4:J10, are specified in the Solver dialogue box.

The target cell (D19) gives the total distance in miles of the chosen path by using the equation for this cell given at the bottom of Fig. 9.4. The objective of *minimizing* this target cell has been specified in the Solver dialogue box. The solution shown in column D is an optimal solution obtained after clicking on the Solve button. This solution is, of course, one of the two shortest paths identified earlier by the algorithm for the shortest-path algorithm.

### Other Applications

Not all applications of the shortest-path problem involve minimizing the distance traveled from the origin to the destination. In fact, they might not even involve travel at all. The links (or arcs) might instead represent activities of some other kind, so choosing a path through the network corresponds to selecting the best sequence of activities. The numbers giving the "lengths" of the links might then be, for example, the *costs* of the activities, in which case the objective would be to determine which sequence of activities minimizes the total cost.

Here are three categories of applications.

1. Minimize the total *distance* traveled, as in the Seervada Park example.
2. Minimize the total *cost* of a sequence of activities. (Problem 9.3-2 is of this type.)
3. Minimize the total *time* of a sequence of activities. (Problems 9.3-5 and 9.3-6 are of this type.)

It is even possible for all three categories to arise in the *same* application. For example, suppose you wish to find the best route for driving from one town to another through a number of intermediate towns. You then have the choice of defining the best route as being the one that minimizes the total *distance* traveled or that minimizes the total *cost* incurred or that minimizes the total *time* required. (Problem 9.3-1 illustrates such an application.)

Many applications require finding the shortest *directed* path from the origin to the destination through a *directed* network. The algorithm already presented can be easily modified to deal just with directed paths at each iteration. In particular, when candidates for the *n*th nearest node are identified, only directed arcs *from* a solved node *to* an unsolved node are considered.

Another version of the shortest-path problem is to find the shortest paths from the origin to *all* the other nodes of the network. Notice that the algorithm already solves for the shortest path to each node that is closer to the origin than the destination. Therefore, when all nodes are potential destinations, the only modification needed in the algorithm is that it does not stop until all nodes are solved nodes.

An even more general version of the shortest-path problem is to find the shortest paths from *every* node to every other node. Another option is to drop the restriction that "distances" (arc values) be nonnegative. Constraints also can be imposed on the paths that can be followed. All these variations occasionally arise in applications and so have been studied by researchers.

The algorithms for a wide variety of combinatorial optimization problems, such as certain vehicle routing or network design problems, often call for the solution of a large number of shortest-path problems as subroutines. Although we lack the space to pursue this topic further, this use may now be the most important kind of application of the shortest-path problem.

## 9.4 THE MINIMUM SPANNING TREE PROBLEM

The minimum spanning tree problem bears some similarities to the main version of the shortest-path problem presented in the preceding section. In both cases, an *undirected* and *connected* network is being considered, where the given information includes some measure of the positive *length* (distance, cost, time, etc.) associated with each link. Both prob-

lems also involve choosing a set of links that have the *shortest total length* among all sets of links that satisfy a certain property. For the shortest-path problem, this property is that the chosen links must provide a path between the origin and the destination. For the minimum spanning tree problem, the required property is that the chosen links must provide a path between *each* pair of nodes.

The minimum spanning tree problem can be summarized as follows.

1. You are given the *nodes* of a network but *not* the *links.* Instead, you are given the *potential links* and the positive *length* for each if it is inserted into the network. (Alternative measures for the length of a link include distance, cost, and time.)
2. You wish to design the network by inserting enough links to satisfy the requirement that there be a path between *every* pair of nodes.
3. The objective is to satisfy this requirement in a way that minimizes the total length of the links inserted into the network.

A network with $n$ nodes requires only $(n - 1)$ links to provide a path between each pair of nodes. No extra links should be used, since this would needlessly increase the total length of the chosen links. The $(n - 1)$ links need to be chosen in such a way that the resulting network (with just the chosen links) forms a *spanning tree* (as defined in Sec. 9.2). Therefore, the problem is to find the spanning tree with a minimum total length of the links.

Figure 9.5 illustrates this concept of a spanning tree for the Seervada Park problem (see Sec. 9.1). Thus, Fig. 9.5*a* is *not* a spanning tree because nodes *O, A, B,* and *C* are not connected with nodes *D, E,* and *T.* It needs another link to make this connection. This network actually consists of two trees, one for each of these two sets of nodes. The links in Fig. 9.5*b* do *span* the network (i.e., the network is connected as defined in Sec. 9.2), but it is *not* a tree because there are two *cycles* (*O–A–B–C–O* and *D–T–E–D*). It has too many links. Because the Seervada Park problem has $n = 7$ nodes, Sec. 9.2 indicates that the network must have exactly $n - 1 = 6$ links, with *no cycles,* to qualify as a spanning tree. This condition is achieved in Fig. 9.5*c,* so this network is a *feasible* solution (with a value of 24 miles for the total length of the links) for the minimum spanning tree problem. (You soon will see that this solution is not *optimal* because it is possible to construct a spanning tree with only 14 miles of links.)

### Some Applications

Here is a list of some key types of applications of the minimum spanning tree problem.

1. Design of telecommunication networks (fiber-optic networks, computer networks, leased-line telephone networks, cable television networks, etc.)
2. Design of a lightly used transportation network to minimize the total cost of providing the links (rail lines, roads, etc.)
3. Design of a network of high-voltage electrical power transmission lines
4. Design of a network of wiring on electrical equipment (e.g., a digital computer system) to minimize the total length of the wire
5. Design of a network of pipelines to connect a number of locations

In this age of the information superhighway, applications of this first type have become particularly important. In a telecommunication network, it is only necessary to in-

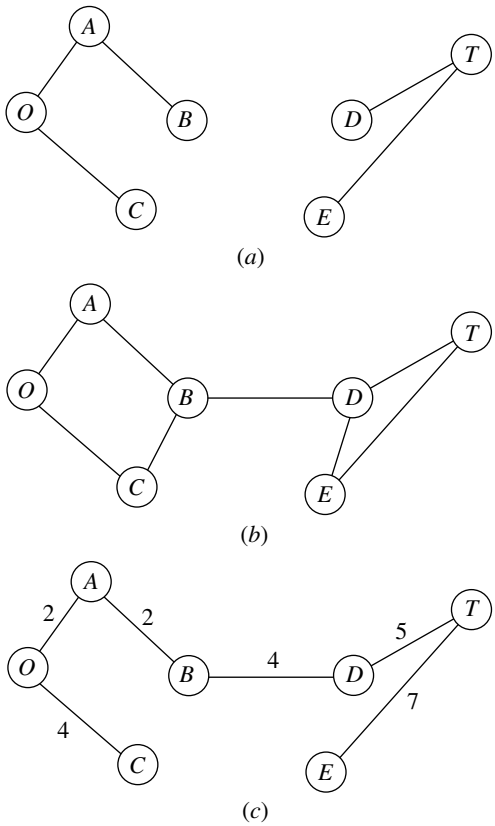



**FIGURE 9.5**
Illustrations of the spanning tree concept for the Seervada Park problem: (*a*) Not a spanning tree; (*b*) not a spanning tree; (*c*) a spanning tree.

sert enough links to provide a path between every pair of nodes, so designing such a network is a classic application of the minimum spanning tree problem. Because some telecommunication networks now cost many millions of dollars, it is very important to optimize their design by finding the minimum spanning tree for each one.

### An Algorithm

The minimum spanning tree problem can be solved in a very straightforward way because it happens to be one of the few OR problems where being *greedy* at each stage of the solution procedure still leads to an overall optimal solution at the end! Thus, beginning with any node, the first stage involves choosing the shortest possible link to another node, without worrying about the effect of this choice on subsequent decisions. The second stage involves identifying the unconnected node that is closest to either of these connected nodes and then adding the corresponding link to the network. This process is repeated, per the following summary, until all the nodes have been connected. (Note that this is the same process already illustrated in Fig. 9.3 for constructing a spanning tree, but now with a specific rule for selecting each new link.) The resulting network is guaranteed to be a minimum spanning tree.

### Algorithm for the Minimum Spanning Tree Problem.

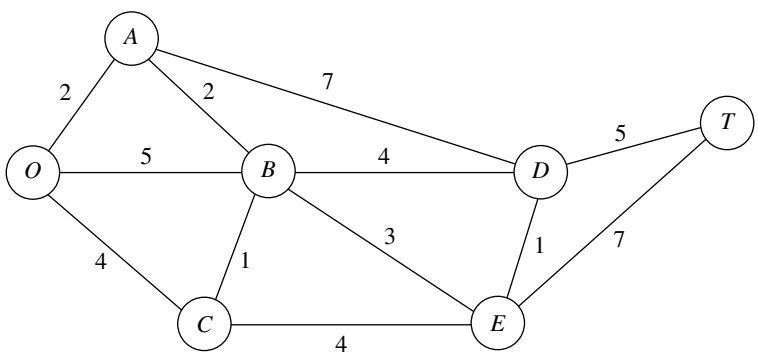
1. Select any node arbitrarily, and then connect it (i.e., add a link) to the nearest distinct node.
2. Identify the unconnected node that is closest to a connected node, and then connect these two nodes (i.e., add a link between them). Repeat this step until all nodes have been connected.
3. Tie breaking: Ties for the nearest distinct node (step 1) or the closest unconnected node (step 2) may be broken arbitrarily, and the algorithm must still yield an optimal solution. However, such ties are a signal that there may be (but need not be) multiple optimal solutions. All such optimal solutions can be identified by pursuing all ways of breaking ties to their conclusion.

The fastest way of executing this algorithm manually is the graphical approach illustrated next.

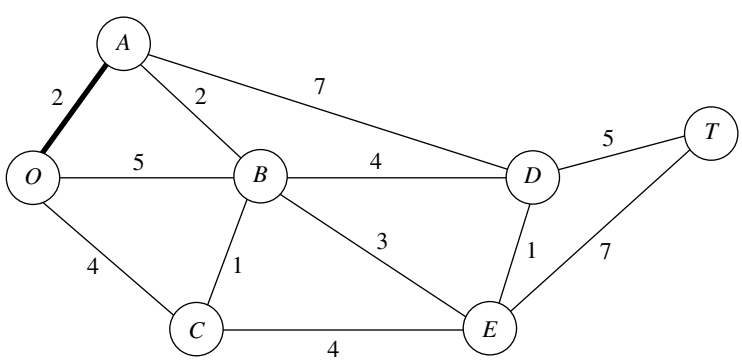
## Applying This Algorithm to the Seervada Park Minimum Spanning Tree Problem

The Seervada Park management (see Sec. 9.1) needs to determine under which roads telephone lines should be installed to connect all stations with a minimum total length of line. Using the data given in Fig. 9.1, we outline the step-by-step solution of this problem.

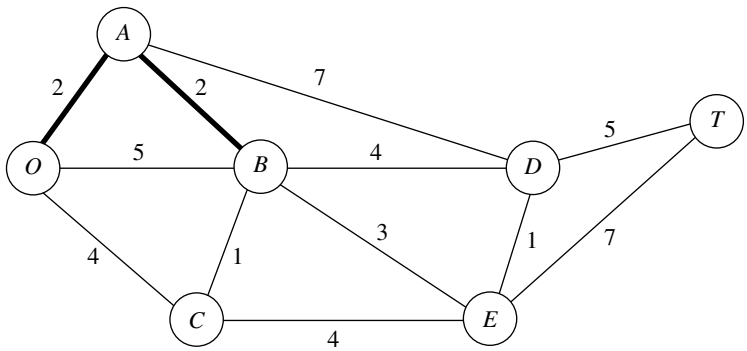
Nodes and distances for the problem are summarized below, where the thin lines now represent *potential* links.



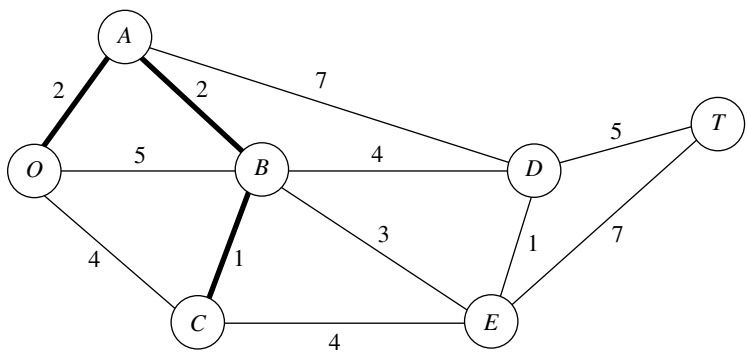

Arbitrarily select node *O* to start. The unconnected node closest to node *O* is node *A*. Connect node *A* to node *O*.



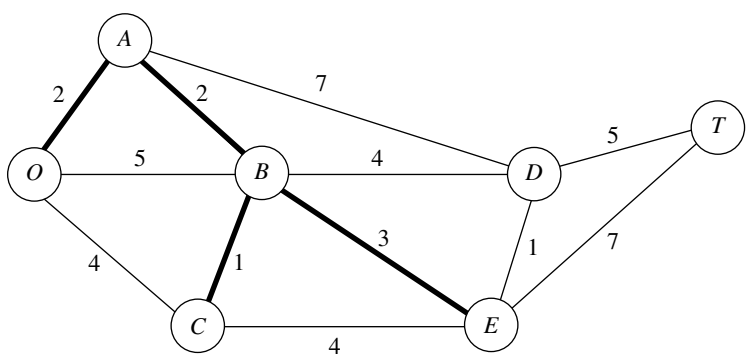
The unconnected node closest to either node *O* or node *A* is node *B* (closest to *A*). Connect node *B* to node *A*.



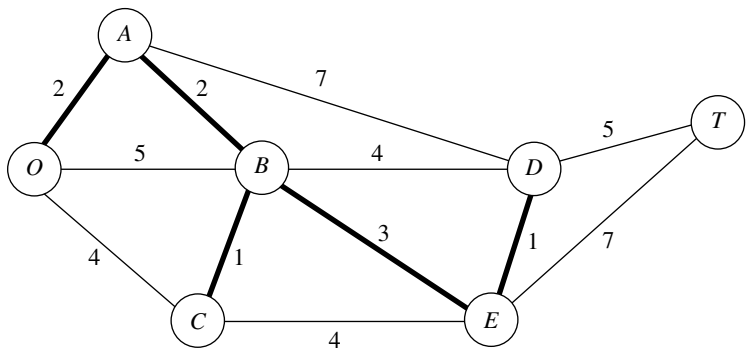

The unconnected node closest to node *O*, *A*, or *B* is node *C* (closest to *B*). Connect node *C* to node *B*.



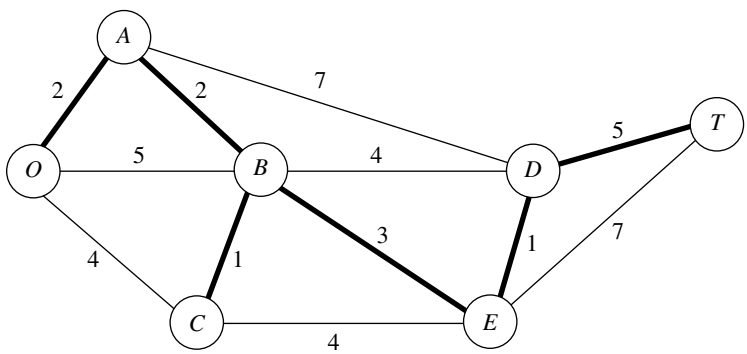

The unconnected node closest to node *O*, *A*, *B*, or *C* is node *E* (closest to *B*). Connect node *E* to node *B*.



The unconnected node closest to node $O$, $A$, $B$, $C$, or $E$ is node $D$ (closest to $E$). Connect node $D$ to node $E$.




The only remaining unconnected node is node $T$. It is closest to node $D$. Connect node $T$ to node $D$.




All nodes are now connected, so this solution to the problem is the desired (optimal) one. The total length of the links is 14 miles.

Although it may appear at first glance that the choice of the initial node will affect the resulting final solution (and its total link length) with this procedure, it really does not. We suggest you verify this fact for the example by reapplying the algorithm, starting with nodes other than node $O$.

The minimum spanning tree problem is the one problem we consider in this chapter that falls into the broad category of *network design.* In this category, the objective is to design the most appropriate network for the given application (frequently involving transportation systems) rather than analyzing an already designed network. Selected Reference 7 provides a survey of this important area.

MPL

## 9.5 THE MAXIMUM FLOW PROBLEM

Now recall that the third problem facing the Seervada Park management (see Sec. 9.1) during the peak season is to determine how to route the various tram trips from the park entrance (station $O$ in Fig. 9.1) to the scenic wonder (station $T$) to maximize the number of trips per day. (Each tram will return by the same route it took on the outgoing trip, so

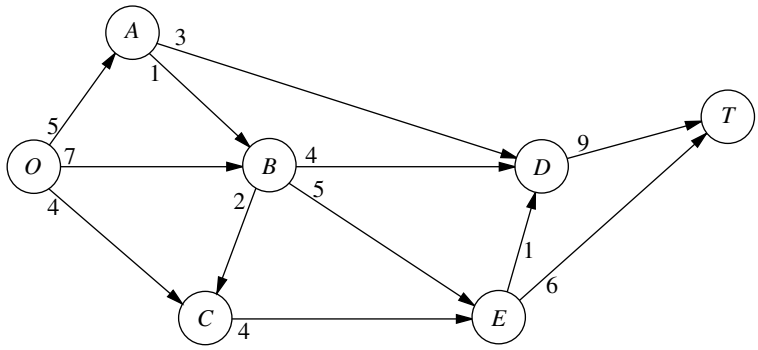



**FIGURE 9.6**
The Seervada Park maximum flow problem.

the analysis focuses on outgoing trips only.) To avoid unduly disturbing the ecology and wildlife of the region, strict upper limits have been imposed on the number of outgoing trips allowed per day in the outbound direction on each individual road. For each road, the direction of travel for outgoing trips is indicated by an arrow in Fig. 9.6. The number at the base of the arrow gives the upper limit on the number of outgoing trips allowed per day. Given the limits, one *feasible solution* is to send 7 trams per day, with 5 using the route $O \rightarrow B \rightarrow E \rightarrow T$, 1 using $O \rightarrow B \rightarrow C \rightarrow E \rightarrow T$, and 1 using $O \rightarrow B \rightarrow C \rightarrow E \rightarrow D \rightarrow T$. However, because this solution blocks the use of any routes starting with $O \rightarrow C$ (because the $E \rightarrow T$ and $E \rightarrow D$ capacities are fully used), it is easy to find better feasible solutions. Many *combinations* of routes (and the number of trips to assign to each one) need to be considered to find the one(s) maximizing the number of trips made per day. This kind of problem is called a *maximum flow problem.*

In general terms, the maximum flow problem can be described as follows.

1. All flow through a directed and connected network originates at one node, called the **source,** and terminates at one other node, called the **sink.** (The source and sink in the Seervada Park problem are the park entrance at node *O* and the scenic wonder at node *T*, respectively.)
2. All the remaining nodes are *transshipment nodes.* (These are nodes *A*, *B*, *C*, *D*, and *E* in the Seervada Park problem.)
3. Flow through an arc is allowed only in the direction indicated by the arrowhead, where the maximum amount of flow is given by the *capacity* of that arc. At the *source,* all arcs point away from the node. At the *sink,* all arcs point into the node.
4. The objective is to maximize the total amount of flow from the source to the sink. This amount is measured in either of two equivalent ways, namely, either the amount *leaving the source* or the amount *entering the sink.*

### Some Applications

Here are some typical kinds of applications of the maximum flow problem.

1. Maximize the flow through a company's distribution network from its factories to its customers.
2. Maximize the flow through a company's supply network from its vendors to its factories.

**3.** Maximize the flow of oil through a system of pipelines.
**4.** Maximize the flow of water through a system of aqueducts.
**5.** Maximize the flow of vehicles through a transportation network.

For some of these applications, the flow through the network may originate at more than one node and may also terminate at more than one node, even though a maximum flow problem is allowed to have only a single source and a single sink. For example, a company's distribution network commonly has multiple factories and multiple customers. A clever reformulation is used to make such a situation fit the maximum flow problem. This reformulation involves expanding the original network to include a *dummy source,* a *dummy sink,* and some new arcs. The dummy source is treated as the node that originates all the flow that, in reality, originates from some of the other nodes. For each of these other nodes, a new arc is inserted that leads from the dummy source to this node, where the capacity of this arc equals the maximum flow that, in reality, can originate from this node. Similarly, the dummy sink is treated as the node that absorbs all the flow that, in reality, terminates at some of the other nodes. Therefore, a new arc is inserted from each of these other nodes to the dummy sink, where the capacity of this arc equals the maximum flow that, in reality, can terminate at this node. Because of all these changes, all the nodes in the original network now are transshipment nodes, so the expanded network has the required single source (the dummy source) and single sink (the dummy sink) to fit the maximum flow problem.

### An Algorithm

Because the maximum flow problem can be formulated as a *linear programming problem* (see Prob. 9.5-2), it can be solved by the simplex method, so any of the linear programming software packages introduced in Chaps. 3 and 4 can be used. However, an even more efficient *augmenting path algorithm* is available for solving this problem. This algorithm is based on two intuitive concepts, a *residual network* and an *augmenting path.*

After some flows have been assigned to the arcs, the **residual network** shows the *remaining* arc capacities (called **residual capacities**) for assigning *additional* flows. For example, consider arc $O \rightarrow B$ in Fig. 9.6, which has an arc capacity of 7. Now suppose that the assigned flows include a flow of 5 through this arc, which leaves a residual capacity of $7 - 5 = 2$ for any additional flow assignment through $O \rightarrow B$. This status is depicted as follows in the residual network.


The number on an arc next to a node gives the residual capacity for flow *from* that node *to* the other node. Therefore, in addition to the residual capacity of 2 for flow from $O$ to $B$, the 5 on the right indicates a residual capacity of 5 for assigning some flow from $B$ to $O$ (that is, for canceling some previously assigned flow from $O$ to $B$).

Initially, before any flows have been assigned, the residual network for the Seervada Park problem has the appearance shown in Fig. 9.7. Every arc in the original network (Fig. 9.6) has been changed from a *directed* arc to an *undirected arc.* However, the arc

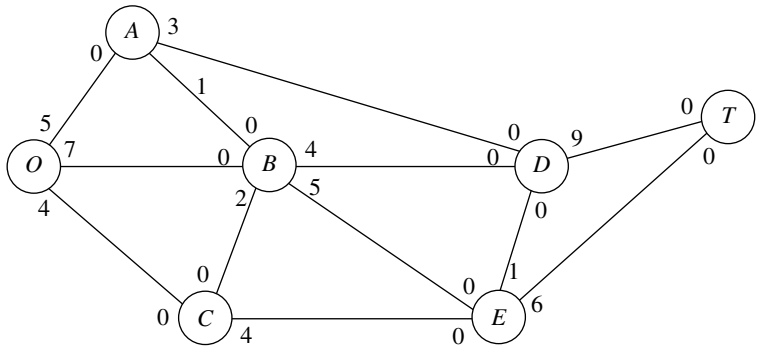



**FIGURE 9.7**
The initial residual network for the Seervada Park maximum flow problem.

capacity in the original direction remains the same and the arc capacity in the opposite direction is *zero,* so the constraints on flows are unchanged.

Subsequently, whenever some amount of flow is assigned to an arc, that amount is *subtracted* from the residual capacity in the same direction and *added* to the residual capacity in the opposite direction.

An **augmenting path** is a directed path from the source to the sink in the residual network such that *every* arc on this path has *strictly positive* residual capacity. The *minimum* of these residual capacities is called the *residual capacity of the augmenting path* because it represents the amount of flow that can feasibly be added to the entire path. Therefore, each augmenting path provides an opportunity to further augment the flow through the original network.

The augmenting path algorithm repeatedly selects some augmenting path and adds a flow equal to its residual capacity to that path in the original network. This process continues until there are no more augmenting paths, so the flow from the source to the sink cannot be increased further. The key to ensuring that the final solution necessarily is optimal is the fact that augmenting paths can cancel some previously assigned flows in the original network, so an indiscriminate selection of paths for assigning flows cannot prevent the use of a better combination of flow assignments.

To summarize, each *iteration* of the algorithm consists of the following three steps.

### The Augmenting Path Algorithm for the Maximum Flow Problem.[1]

1. Identify an augmenting path by finding some directed path from the source to the sink in the residual network such that every arc on this path has strictly positive residual capacity. (If no augmenting path exists, the net flows already assigned constitute an optimal flow pattern.)
2. Identify the residual capacity $c^*$ of this augmenting path by finding the *minimum* of the residual capacities of the arcs on this path. *Increase* the flow in this path by $c^*$.
3. *Decrease* by $c^*$ the residual capacity of each arc on this augmenting path. *Increase* by $c^*$ the residual capacity of each arc in the opposite direction on this augmenting path. Return to step 1.

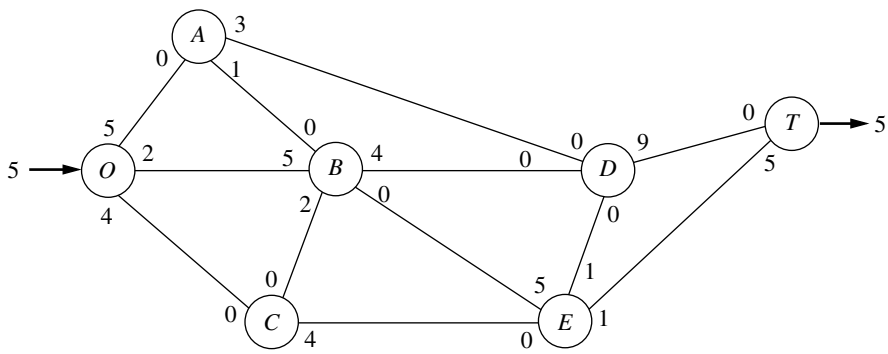
[1]It is assumed that the arc capacities are either integers or rational numbers.

When step 1 is carried out, there often will be a number of alternative augmenting paths from which to choose. Although the algorithmic strategy for making this selection is important for the efficiency of large-scale implementations, we shall not delve into this relatively specialized topic. (Later in the section, we do describe a systematic procedure for finding some augmenting path.) Therefore, for the following example (and the problems at the end of the chapter), the selection is just made arbitrarily.

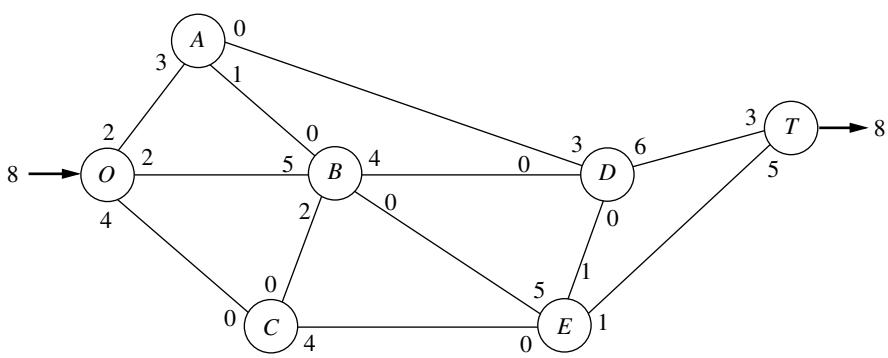
### Applying This Algorithm to the Seervada Park Maximum Flow Problem

Applying this algorithm to the Seervada Park problem (see Fig. 9.6 for the original network) yields the results summarized next. Starting with the initial residual network given in Fig. 9.7, we give the new residual network after each one or two iterations, where the total amount of flow from $O$ to $T$ achieved thus far is shown in **boldface** (next to nodes $O$ and $T$).

*Iteration 1:* In Fig. 9.7, one of several augmenting paths is $O \rightarrow B \rightarrow E \rightarrow T$, which has a residual capacity of $\min\{7, 5, 6\} = 5$. By assigning a flow of 5 to this path, the resulting residual network is

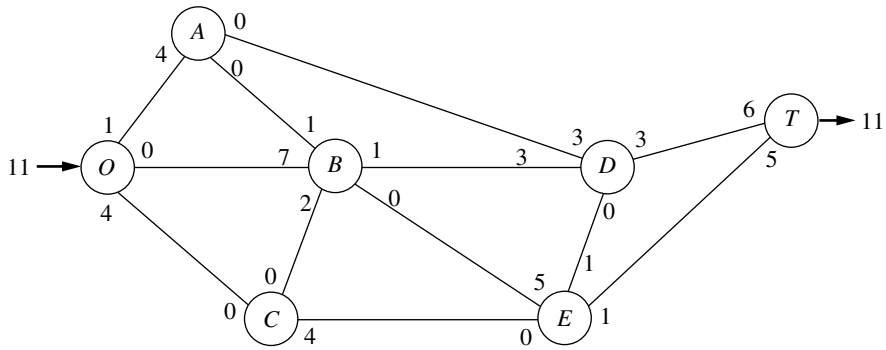



*Iteration 2:* Assign a flow of 3 to the augmenting path $O \rightarrow A \rightarrow D \rightarrow T$. The resulting residual network is



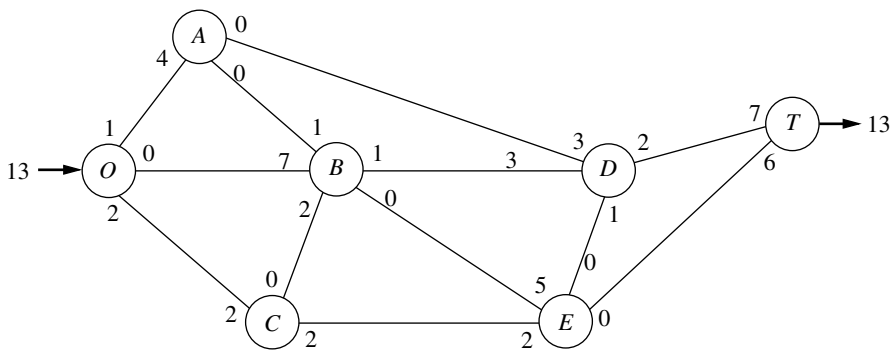

*Iteration 3:* Assign a flow of 1 to the augmenting path $O \rightarrow A \rightarrow B \rightarrow D \rightarrow T$.

*Iteration 4:* Assign a flow of 2 to the augmenting path $O \rightarrow B \rightarrow D \rightarrow T$. The resulting residual network is

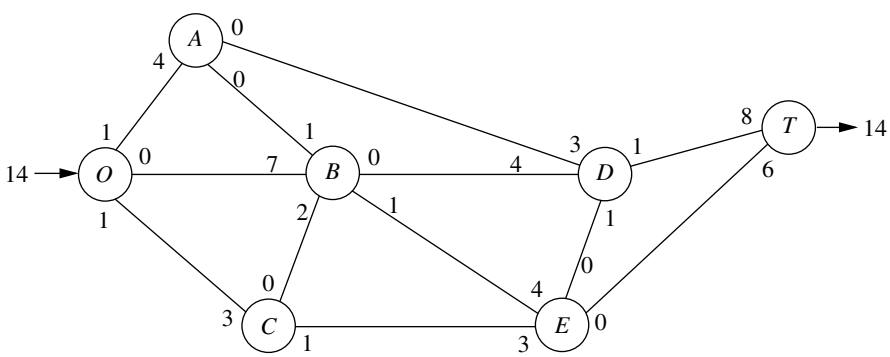



*Iteration 5:* Assign a flow of 1 to the augmenting path $O \rightarrow C \rightarrow E \rightarrow D \rightarrow T$.

*Iteration 6:* Assign a flow of 1 to the augmenting path $O \rightarrow C \rightarrow E \rightarrow T$. The resulting residual network is




*Iteration 7:* Assign a flow of 1 to the augmenting path $O \rightarrow C \rightarrow E \rightarrow B \rightarrow D \rightarrow T$. The resulting residual network is




There are no more augmenting paths, so the current flow pattern is optimal.

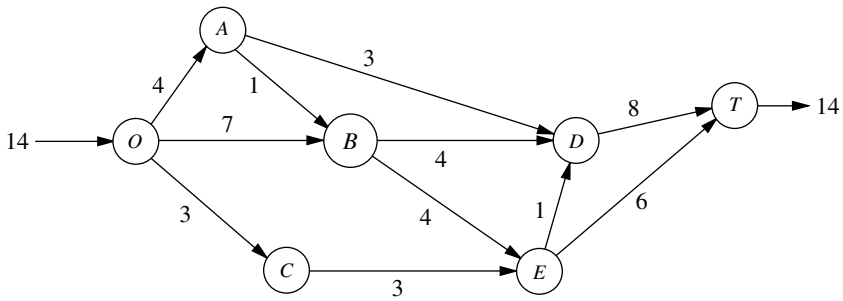



**FIGURE 9.8**
Optimal solution for the Seervada Park maximum flow problem.

The current flow pattern may be identified by either cumulating the flow assignments or comparing the final residual capacities with the original arc capacities. If we use the latter method, there is flow along an arc if the final residual capacity is less than the original capacity. The magnitude of this flow equals the difference in these capacities. Applying this method by comparing the residual network obtained from the last iteration with either Fig. 9.6 or 9.7 yields the optimal flow pattern shown in Fig. 9.8.

This example nicely illustrates the reason for replacing each directed arc $i \rightarrow j$ in the original network by an undirected arc in the residual network and then increasing the residual capacity for $j \rightarrow i$ by $c^*$ when a flow of $c^*$ is assigned to $i \rightarrow j$. Without this refinement, the first six iterations would be unchanged. However, at that point it would appear that no augmenting paths remain (because the real unused arc capacity for $E \rightarrow B$ is zero). Therefore, the refinement permits us to add the flow assignment of 1 for $O \rightarrow C \rightarrow E \rightarrow B \rightarrow D \rightarrow T$ in iteration 7. In effect, this additional flow assignment cancels 1 unit of flow assigned at iteration 1 ($O \rightarrow B \rightarrow E \rightarrow T$) and replaces it by assignments of 1 unit of flow to *both* $O \rightarrow B \rightarrow D \rightarrow T$ and $O \rightarrow C \rightarrow E \rightarrow T$.

### Finding an Augmenting Path

The most difficult part of this algorithm when *large* networks are involved is finding an augmenting path. This task may be simplified by the following systematic procedure. Begin by determining all nodes that can be reached from the source along a single arc with strictly positive residual capacity. Then, for each of these nodes that were reached, determine all *new* nodes (those not yet reached) that can be reached from this node along an arc with strictly positive residual capacity. Repeat this successively with the new nodes as they are reached. The result will be the identification of a tree of all the nodes that can be reached from the source along a path with strictly positive residual flow capacity. Hence, this *fanning-out procedure* will always identify an augmenting path if one exists. The procedure is illustrated in Fig. 9.9 for the residual network that results from *iteration 6* in the preceding example.

Although the procedure illustrated in Fig. 9.9 is a relatively straightforward one, it would be helpful to be able to recognize when optimality has been reached without an exhaustive search for a nonexistent path. It is sometimes possible to recognize this event because of an important theorem of network theory known as the *max-flow min-cut theorem.* A **cut** may be defined as any set of directed arcs containing at least one arc from

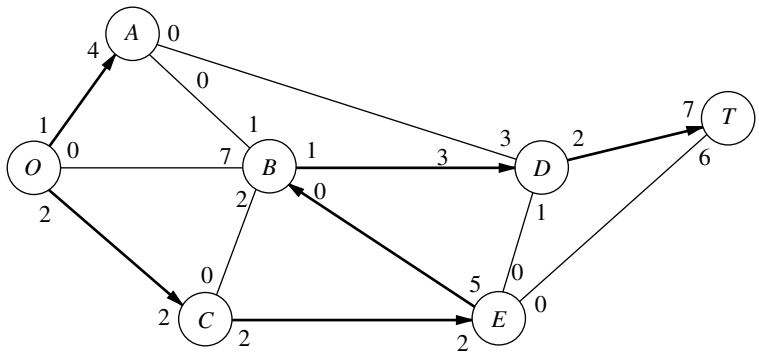



**FIGURE 9.9**
Procedure for finding an augmenting path for iteration 7 of the Seervada Park maximum flow problem.

every directed path from the source to the sink. There normally are many ways to slice through a network to form a cut to help analyze the network. For any particular cut, the **cut value** is the sum of the arc capacities of the arcs (in the specified direction) of the cut. The **max-flow min-cut theorem** states that, for any network with a single source and sink, the *maximum feasible flow* from the source to the sink *equals* the *minimum cut value* for all cuts of the network. Thus, if we let $F$ denote the amount of flow from the source to the sink for any feasible flow pattern, the value of any cut provides an upper bound to $F$, and the smallest of the cut values is equal to the maximum value of $F$. Therefore, if a cut whose value equals the value of $F$ currently attained by the solution procedure can be found in the original network, the current flow pattern must be *optimal.* Eventually, optimality has been attained whenever there exists a cut in the residual network whose value is zero.

To illustrate, consider the network of Fig. 9.7. One interesting cut through this network is shown in Fig. 9.10. Notice that the value of the cut is $3 + 4 + 1 + 6 = 14$, which was found to be the maximum value of $F$, so this cut is a minimum cut. Notice also that, in the residual network resulting from iteration 7, where $F = 14$, the corresponding cut has a value of zero. If this had been noticed, it would not have been necessary to search for additional augmenting paths.

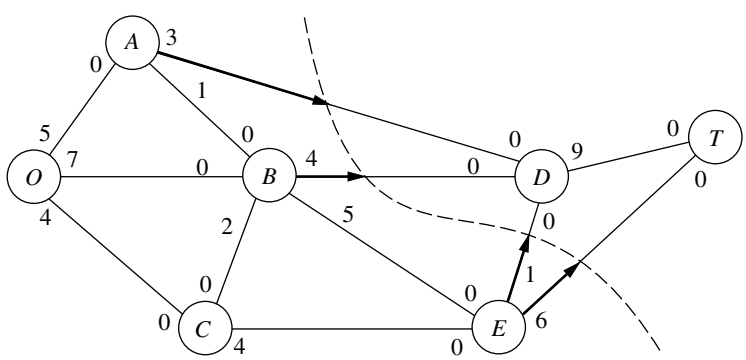



**FIGURE 9.10**
A minimum cut for the Seervada Park maximum flow problem.

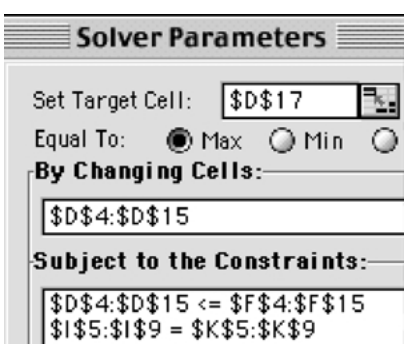
### Using Excel to Formulate and Solve Maximum Flow Problems

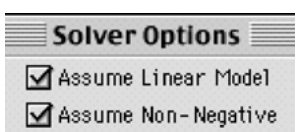
Most maximum flow problems that arise in practice are considerably larger, and occasionally vastly larger, than the Seervada Park problem. Some problems have thousands of nodes and arcs. The augmenting path algorithm just presented is far more efficient than the general simplex method for solving such large problems. However, for problems of modest size, a reasonable and convenient alternative is to use Excel and its Solver based on the general simplex method.

Figure 9.11 shows a spreadsheet formulation for the Seervada Park maximum flow problem. The format is similar to that for the Seervada Park shortest-path problem displayed in Fig. 9.4. The arcs are listed in columns B and C, and the corresponding arc capacities are given in column F. Since the decision variables are the flows through the re-

**FIGURE 9.11**
A spreadsheet formulation for the Seervada Park maximum flow problem, where the changing cells (D4:D15) show the optimal solution obtained by the Excel Solver and the target cell (D17) gives the resulting maximum flow through the network.

| | A | B | C | D | E | F | G | H | I | J | K |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Seervada Park Maximum Flow Problem | | | | | | | | | | |
| 2 | | | | | | | | | | | |
| 3 | | From | To | Flow | | Capacity | | Nodes | Net Flow | | Supply/Demand |
| 4 | | O | A | 4 | ≤ | 5 | | O | 14 | | |
| 5 | | O | B | 7 | ≤ | 7 | | A | 0 | = | 0 |
| 6 | | O | C | 3 | ≤ | 4 | | B | 0 | = | 0 |
| 7 | | A | B | 1 | ≤ | 1 | | C | 0 | = | 0 |
| 8 | | A | D | 3 | ≤ | 3 | | D | 0 | = | 0 |
| 9 | | B | C | 0 | ≤ | 2 | | E | 0 | = | 0 |
| 10 | | B | D | 4 | ≤ | 4 | | T | -14 | | |
| 11 | | B | E | 4 | ≤ | 5 | | | | | |
| 12 | | C | E | 3 | ≤ | 4 | | | | | |
| 13 | | D | T | 8 | ≤ | 9 | | | | | |
| 14 | | E | D | 1 | ≤ | 1 | | | | | |
| 15 | | E | T | 6 | ≤ | 6 | | | | | |
| 16 | | | | | | | | | | | |
| 17 | | Maximum Flow = | | 14 | | | | | | | |




| | D |
|---|---|
| 17 | =I4 |

| | I |
|---|---|
| 4 | =D4+D5+D6 |
| 5 | =-D4+D7+D8 |
| 6 | =-D5-D7+D9+D10+D11 |
| 7 | =-D6-D9+D12 |
| 8 | =-D8-D10+D13-D14 |
| 9 | =-D11-D12+D14+D15 |
| 10 | =-D13-D15 |

spective arcs, these quantities are entered in the changing cells in column D (cells D4:D15). Employing the equations given in the bottom right-hand corner of the figure, these flows then are used to calculate the net flow generated at each of the nodes (see columns H and I). These net flows are required to be 0 for the transshipment nodes (A, B, C, D, and E), as indicated by the second set of constraints (I5:I9 = K5:K9) in the Solver dialogue box. The first set of constraints (D4:D15 ≤ F4:F15) specifies the arc capacity constraints. The total amount of flow from the source (node *O*) to the sink (node *T*) equals the flow generated at the source (cell I4), so the target cell (D17) is set equal to I4. After specifying *maximization* of the target cell in the Solver dialogue box and then clicking on the Solve button, the optimal solution shown in cells D4:D15 is obtained.

MPL

## 9.6 THE MINIMUM COST FLOW PROBLEM

The minimum cost flow problem holds a central position among network optimization models, both because it encompasses such a broad class of applications and because it can be solved extremely efficiently. Like the maximum flow problem, it considers flow through a network with limited arc capacities. Like the shortest-path problem, it considers a cost (or distance) for flow through an arc. Like the transportation problem or assignment problem of Chap. 8, it can consider multiple sources (supply nodes) and multiple destinations (demand nodes) for the flow, again with associated costs. In fact, all four of these previously studied problems are special cases of the minimum cost flow problem, as we will demonstrate shortly.

The reason that the minimum cost flow problem can be solved so efficiently is that it can be formulated as a linear programming problem so it can be solved by a streamlined version of the simplex method called the *network simplex method.* We describe this algorithm in the next section.

The minimum cost flow problem is described below.

**1.** The network is a *directed* and *connected* network.
**2.** *At least one* of the nodes is a *supply node.*
**3.** *At least one* of the other nodes is a *demand node.*
**4.** All the remaining nodes are *transshipment nodes.*
**5.** Flow through an arc is allowed only in the direction indicated by the arrowhead, where the maximum amount of flow is given by the *capacity* of that arc. (If flow can occur in both directions, this would be represented by a pair of arcs pointing in opposite directions.)
**6.** The network has enough arcs with sufficient capacity to enable all the flow generated at the *supply nodes* to reach all the *demand nodes.*
**7.** The cost of the flow through each arc is *proportional* to the amount of that flow, where the cost per unit flow is known.
**8.** The objective is to minimize the total cost of sending the available supply through the network to satisfy the given demand. (An alternative objective is to maximize the total profit from doing this.)

### Some Applications

Probably the most important kind of application of minimum cost flow problems is to the operation of a company's distribution network. As summarized in the first row of Table 9.3, this kind of application always involves determining a plan for shipping goods from

**TABLE 9.3** Typical kinds of applications of minimum cost flow problems

| Kind of Application | Supply Nodes | Transshipment Nodes | Demand Nodes |
|---|---|---|---|
| Operation of a distribution network | Sources of goods | Intermediate storage facilities | Customers |
| Solid waste management | Sources of solid waste | Processing facilities | Landfill locations |
| Operation of a supply network | Vendors | Intermediate warehouses | Processing facilities |
| Coordinating product mixes at plants | Plants | Production of a specific product | Market for a specific product |
| Cash flow management | Sources of cash at a specific time | Short-term investment options | Needs for cash at a specific time |

its *sources* (factories, etc.) to *intermediate storage facilities* (as needed) and then on to the *customers.*

For example, consider the distribution network for the *International Paper Company* (as described in the March–April 1988 issue of *Interfaces*). This company is the world's largest manufacturer of pulp, paper, and paper products, as well as a major producer of lumber and plywood. It also either owns or has rights over about 20 million acres of woodlands. The supply nodes in its distribution network are these woodlands in their various locations. However, before the company's goods can eventually reach the demand nodes (the customers), the wood must pass through a long sequence of transshipment nodes. A typical path through the distribution network is

Woodlands → woodyards → sawmills
→ paper mills → converting plants
→ warehouses → customers.

Another example of a complicated distribution network is the one for the *Citgo Petroleum Corporation* described in Sec. 3.5. Applying a minimum cost flow problem formulation to improve the operation of this distribution network saved Citgo at least $16.5 million annually.

For some applications of minimum cost flow problems, all the transshipment nodes are *processing facilities* rather than intermediate storage facilities. This is the case for *solid waste management,* as indicated in the second row of Table 9.3. Here, the flow of materials through the network begins at the sources of the solid waste, then goes to the facilities for processing these waste materials into a form suitable for landfill, and then sends them on to the various landfill locations. However, the objective still is to determine the flow plan that minimizes the total cost, where the cost now is for both shipping and processing.

In other applications, the *demand nodes* might be processing facilities. For example, in the third row of Table 9.3, the objective is to find the minimum cost plan for obtaining supplies from various possible vendors, storing these goods in warehouses (as needed), and then shipping the supplies to the company's processing facilities (factories, etc.). Since the total amount that could be supplied by all the vendors is more than the company needs,

the network includes a *dummy demand node* that receives (at zero cost) all the unused supply capacity at the vendors.

The July–August 1987 issue of *Interfaces* describes how, even back then, *microcomputers* were being used by *Marshalls, Inc.* (an off-price retail chain) to deal with a minimum cost flow problem this way. In this application, Marshalls was optimizing the flow of freight from vendors to processing centers and then on to retail stores. Some of their networks had over 20,000 arcs.

The next kind of application in Table 9.3 (coordinating product mixes at plants) illustrates that arcs can represent something other than a shipping lane for a physical flow of materials. This application involves a company with several plants (the supply nodes) that can produce the same products but at different costs. Each arc from a supply node represents the production of one of the possible products at that plant, where this arc leads to the transshipment node that corresponds to this product. Thus, this transshipment node has an arc coming in from each plant capable of producing this product, and then the arcs leading out of this node go to the respective customers (the demand nodes) for this product. The objective is to determine how to divide each plant's production capacity among the products so as to minimize the total cost of meeting the demand for the various products.

The last application in Table 9.3 (cash flow management) illustrates that different nodes can represent some event that occurs at different times. In this case, each supply node represents a specific time (or time period) when some cash will become available to the company (through maturing accounts, notes receivable, sales of securities, borrowing, etc.). The supply at each of these nodes is the amount of cash that will become available then. Similarly, each demand node represents a specific time (or time period) when the company will need to draw on its cash reserves. The demand at each such node is the amount of cash that will be needed then. The objective is to maximize the company's income from investing the cash between each time it becomes available and when it will be used. Therefore, each transshipment node represents the choice of a specific short-term investment option (e.g., purchasing a certificate of deposit from a bank) over a specific time interval. The resulting network will have a succession of flows representing a schedule for cash becoming available, being invested, and then being used after the maturing of the investment.

### Formulation of the Model

Consider a directed and connected network where the $n$ nodes include at least one supply node and at least one demand node. The decision variables are

$x_{ij}$ = flow through arc $i \rightarrow j$,

and the given information includes

$c_{ij}$ = cost per unit flow through arc $i \rightarrow j$,
$u_{ij}$ = arc capacity for arc $i \rightarrow j$,
$b_i$ = net flow generated at node $i$.

The value of $b_i$ depends on the nature of node $i$, where

$b_i > 0$ if node $i$ is a supply node,
$b_i < 0$ if node $i$ is a demand node,
$b_i = 0$ if node $i$ is a transshipment node.

The objective is to minimize the total cost of sending the available supply through the network to satisfy the given demand.

By using the convention that summations are taken only over existing arcs, the linear programming formulation of this problem is

$$\text{Minimize} \qquad Z = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = b_i, \qquad \text{for each node } i,$$

and

$$0 \le x_{ij} \le u_{ij}, \qquad \text{for each arc } i \to j.$$

The first summation in the *node constraints* represents the total flow *out* of node $i$, whereas the second summation represents the total flow *into* node $i$, so the difference is the net flow generated at this node.

In some applications, it is necessary to have a lower bound $L_{ij} > 0$ for the flow through each arc $i \to j$. When this occurs, use a translation of variables $x'_{ij} = x_{ij} - L_{ij}$, with $x'_{ij} + L_{ij}$ substituted for $x_{ij}$ throughout the model, to convert the model back to the above format with nonnegativity constraints.

It is not guaranteed that the problem actually will possess *feasible* solutions, depending partially upon which arcs are present in the network and their arc capacities. However, for a reasonably designed network, the main condition needed is the following.

**Feasible solutions property:** A necessary condition for a minimum cost flow problem to have any feasible solutions is that

$$\sum_{i=1}^{n} b_i = 0.$$

That is, the total flow being generated at the supply nodes equals the total flow being absorbed at the demand nodes.

If the values of $b_i$ provided for some application violate this condition, the usual interpretation is that either the supplies or the demands (whichever are in excess) actually represent upper bounds rather than exact amounts. When this situation arose for the transportation problem in Sec. 8.1, either a dummy destination was added to receive the excess supply or a dummy source was added to send the excess demand. The analogous step now is that either a dummy demand node should be added to absorb the excess supply (with $c_{ij} = 0$ arcs added from every supply node to this node) or a dummy supply node should be added to generate the flow for the excess demand (with $c_{ij} = 0$ arcs added from this node to every demand node).

For many applications, $b_i$ and $u_{ij}$ will have *integer* values, and implementation will require that the flow quantities $x_{ij}$ also be integer. Fortunately, just as for the transportation problem, this outcome is guaranteed without explicitly imposing integer constraints on the variables because of the following property.

**Integer solutions property:** For minimum cost flow problems where every $b_i$ and $u_{ij}$ have integer values, all the basic variables in *every* basic feasible (BF) solution (including an optimal one) also have integer values.

## An Example

An example of a minimum cost flow problem is shown in Fig. 9.12. This network actually is the *distribution network* for the Distribution Unlimited Co. problem presented in Sec. 3.4 (see Fig. 3.13). The quantities given in Fig. 3.13 provide the values of the $b_i$, $c_{ij}$, and $u_{ij}$ shown here. The $b_i$ values in Fig. 9.12 are shown in square brackets by the nodes, so the supply nodes ($b_i > 0$) are $A$ and $B$ (the company's two factories), the demand nodes ($b_i < 0$) are $D$ and $E$ (two warehouses), and the one transshipment node ($b_i = 0$) is $C$ (a distribution center). The $c_{ij}$ values are shown next to the arcs. In this example, all but two of the arcs have arc capacities exceeding the total flow generated (90), so $u_{ij} = \infty$ for all practical purposes. The two exceptions are arc $A \rightarrow B$, where $u_{AB} = 10$, and arc $C \rightarrow E$, which has $u_{CE} = 80$.

The linear programming model for this example is

$$\text{Minimize} \quad Z = 2x_{AB} + 4x_{AC} + 9x_{AD} + 3x_{BC} + x_{CE} + 3x_{DE} + 2x_{ED},$$

subject to

$$\begin{aligned}
x_{AB} + x_{AC} + x_{AD} &= 50 \\
-x_{AB} + x_{BC} &= 40 \\
-x_{AC} - x_{BC} + x_{CE} &= 0 \\
-x_{AD} + x_{DE} - x_{ED} &= -30 \\
-x_{CE} - x_{DE} + x_{ED} &= -60
\end{aligned}$$

and

$$x_{AB} \leq 10, \qquad x_{CE} \leq 80, \qquad \text{all } x_{ij} \geq 0.$$

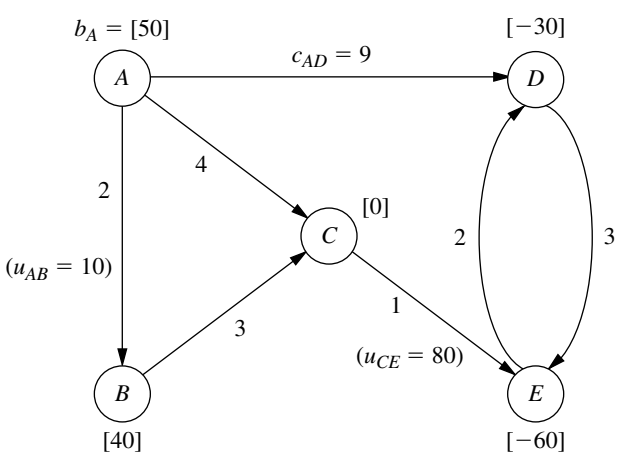



**FIGURE 9.12**
The Distribution Unlimited Co. problem formulated as a minimum cost flow problem.

Now note the pattern of coefficients for each variable in the set of five *node constraints* (the equality constraints). Each variable has exactly *two* nonzero coefficients, where one is $+1$ and the other is $-1$. This pattern recurs in *every* minimum cost flow problem, and it is this special structure that leads to the integer solutions property.

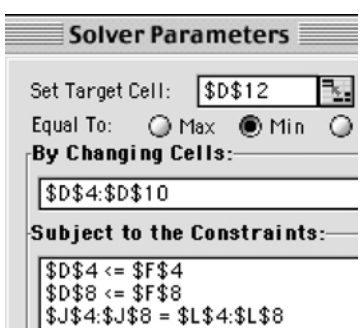
Another implication of this special structure is that (any) one of the node constraints is *redundant.* The reason is that summing all these constraint equations yields nothing but zeros on both sides (assuming feasible solutions exist, so the $b_i$ values sum to zero), so the negative of any one of these equations equals the sum of the rest of the equations. With just $n - 1$ nonredundant node constraints, these equations provide just $n - 1$ basic variables for a BF solution. In the next section, you will see that the network simplex method treats the $x_{ij} \leq u_{ij}$ constraints as mirror images of the nonnegativity constraints, so the *total* number of basic variables is $n - 1$. This leads to a direct correspondence between the $n - 1$ arcs of a *spanning tree* and the $n - 1$ basic variables—but more about that story later.

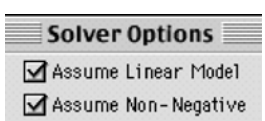
## Using Excel to Formulate and Solve Minimum Cost Flow Problems

Excel provides a convenient way of formulating and solving small minimum cost flow problems like this one, as well as somewhat larger problems. Figure 9.13 shows how this can be done. The format is almost the same as displayed in Fig. 9.11 for a maximum flow problem. One difference is that the unit costs ($c_{ij}$) now need to be included (in column

**FIGURE 9.13**
A spreadsheet formulation for the Distribution Unlimited Co. minimum cost flow problem, where the changing cells (D4:D10) show the optimal solution obtained by the Excel Solver and the target cell (D12) gives the resulting total cost of the flow of shipments through the network.

| | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Distribution Unlimited Co. Minimum Cost Flow Problem | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |
| 3 | | From | To | Ship | | Capacity | Unit Cost | | Nodes | Net Flow | | Supply/Demand |
| 4 | | A | B | 0 | ≤ | 10 | 2 | | A | 50 | = | 50 |
| 5 | | A | C | 40 | | | 4 | | B | 40 | = | 40 |
| 6 | | A | D | 10 | | | 9 | | C | 0 | = | 0 |
| 7 | | B | C | 40 | | | 3 | | D | -30 | = | -30 |
| 8 | | C | E | 80 | ≤ | 80 | 1 | | E | -60 | = | -60 |
| 9 | | D | E | 0 | | | 3 | | | | | |
| 10 | | E | D | 20 | | | 2 | | | | | |
| 11 | | | | | | | | | | | | |
| 12 | | Total Cost = | | 490 | | | | | | | | |




| | J |
|---|---|
| 4 | =D4+D5+D6 |
| 5 | =-D4+D7 |
| 6 | =-D5-D7+D8 |
| 7 | =-D6+D9-D10 |
| 8 | =-D8-D9+D10 |

| | D |
|---|---|
| 12 | =SUMPRODUCT(D4:D10,G4:G10) |

G). Because $b_i$ values are specified for every node, net flow constraints are needed for all the nodes. However, only two of the arcs happen to need arc capacity constraints. The target cell (D12) now gives the total cost of the flow (shipments) through the network (see its equation at the bottom of the figure), so the objective specified in the Solver dialogue box is to *minimize* this quantity. The changing cells (D4:D10) in this spreadsheet show the optimal solution obtained after clicking on the Solve button.

For much larger minimum cost flow problems, the *network simplex method* described in the next section provides a considerably more efficient solution procedure. It also is an attractive option for solving various special cases of the minimum cost flow problem outlined below. This algorithm is commonly included in mathematical programming software packages. For example, it is one of the options with CPLEX.

We shall soon solve this same example by the network simplex method. However, let us first see how some special cases fit into the network format of the minimum cost flow problem.

### Special Cases

**The Transportation Problem.** To formulate the transportation problem presented in Sec. 8.1 as a minimum cost flow problem, a *supply node* is provided for each *source,* as well as a *demand node* for each *destination,* but no transshipment nodes are included in the network. All the arcs are directed from a supply node to a demand node, where distributing $x_{ij}$ units from source $i$ to destination $j$ corresponds to a flow of $x_{ij}$ through arc $i \rightarrow j$. The cost $c_{ij}$ per unit distributed becomes the cost $c_{ij}$ per unit of flow. Since the transportation problem does not impose upper bound constraints on individual $x_{ij}$, all the $u_{ij} = \infty$.

Using this formulation for the P & T Co. transportation problem presented in Table 8.2 yields the network shown in Fig. 8.2. The corresponding network for the general transportation problem is shown in Fig. 8.3.

**The Assignment Problem.** Since the assignment problem discussed in Sec. 8.3 is a special type of transportation problem, its formulation as a minimum cost flow problem fits into the same format. The additional factors are that (1) the number of supply nodes equals the number of demand nodes, (2) $b_i = 1$ for each supply node, and (3) $b_i = -1$ for each demand node.

Figure 8.5 shows this formulation for the general assignment problem.

**The Transshipment Problem.** This special case actually includes all the general features of the minimum cost flow problem except for not having (finite) arc capacities. Thus, any minimum cost flow problem where each arc can carry any desired amount of flow is also called a transshipment problem.

For example, the Distribution Unlimited Co. problem shown in Fig. 9.13 would be a transshipment problem if the upper bounds on the flow through arcs $A \rightarrow B$ and $C \rightarrow E$ were removed.

Transshipment problems frequently arise as generalizations of transportation problems where units being distributed from each source to each destination can first pass through intermediate points. These intermediate points may include other sources and destinations, as well as additional transfer points that would be represented by transshipment nodes in the network representation of the problem. For example, the Distribution Un-

limited Co. problem can be viewed as a generalization of a transportation problem with two sources (the two factories represented by nodes $A$ and $B$ in Fig. 9.13), two destinations (the two warehouses represented by nodes $D$ and $E$), and one additional intermediate transfer point (the distribution center represented by node $C$).

(Chapter 23 on our website includes a further discussion of the transshipment problem.)

**The Shortest-Path Problem.** Now consider the main version of the shortest-path problem presented in Sec. 9.3 (finding the shortest path from one origin to one destination through an *undirected* network). To formulate this problem as a minimum cost flow problem, one supply node with a supply of 1 is provided for the origin, one demand node with a demand of 1 is provided for the destination, and the rest of the nodes are transshipment nodes. Because the network of our shortest-path problem is undirected, whereas the minimum cost flow problem is assumed to have a directed network, we replace each link by a pair of directed arcs in opposite directions (depicted by a single line with arrowheads at both ends). The only exceptions are that there is no need to bother with arcs *into* the supply node or *out of* the demand node. The distance between nodes $i$ and $j$ becomes the unit cost $c_{ij}$ or $c_{ji}$ for flow in either direction between these nodes. As with the preceding special cases, no arc capacities are imposed, so all $u_{ij} = \infty$.

Figure 9.14 depicts this formulation for the Seervada Park shortest-path problem shown in Fig. 9.1, where the numbers next to the lines now represent the unit cost of flow in either direction.

**The Maximum Flow Problem.** The last special case we shall consider is the maximum flow problem described in Sec. 9.5. In this case a network already is provided with one supply node (the source), one demand node (the sink), and various transshipment nodes, as well as the various arcs and arc capacities. Only three adjustments are needed to fit this problem into the format for the minimum cost flow problem. First, set $c_{ij} = 0$ for all existing arcs to reflect the absence of costs in the maximum flow problem. Second, select a quantity $\overline{F}$, which is a safe upper bound on the maximum feasible flow

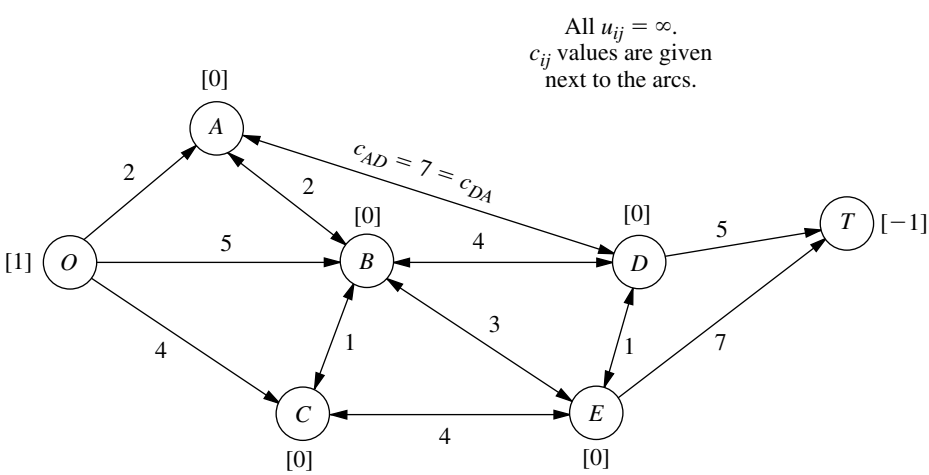



**FIGURE 9.14**
Formulation of the Seervada Park shortest-path problem as a minimum cost flow problem.

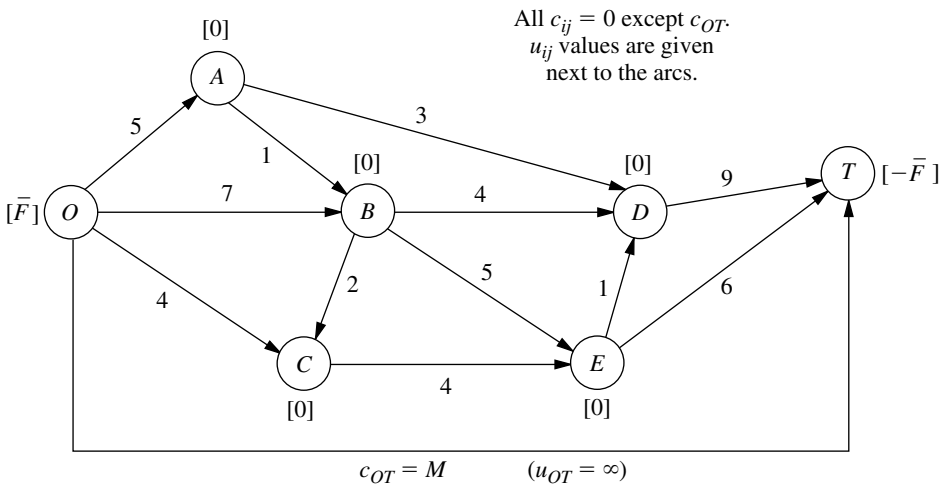



**FIGURE 9.15**
Formulation of the Seervada Park maximum flow problem as a minimum cost flow problem.

through the network, and then assign a supply and a demand of $\bar{F}$ to the supply node and the demand node, respectively. (Because all *other* nodes are transshipment nodes, they automatically have $b_i = 0$.) Third, add an arc going directly from the supply node to the demand node and assign it an arbitrarily large unit cost of $c_{ij} = M$ as well as an unlimited arc capacity ($u_{ij} = \infty$). Because of this positive unit cost for this arc and the zero unit cost for all the *other* arcs, the minimum cost flow problem will send the maximum feasible flow through the *other* arcs, which achieves the objective of the maximum flow problem.

Applying this formulation to the Seervada Park maximum flow problem shown in Fig. 9.6 yields the network given in Fig. 9.15, where the numbers given next to the original arcs are the arc capacities.

**Final Comments.** Except for the transshipment problem, each of these special cases has been the focus of a previous section in either this chapter or Chap. 8. When each was first presented, we talked about a special-purpose algorithm for solving it very efficiently. Therefore, it certainly is not necessary to reformulate these special cases to fit the format of the minimum cost flow problem in order to solve them. However, when a computer code is not readily available for the special-purpose algorithm, it is very reasonable to use the network simplex method instead. In fact, recent implementations of the network simplex method have become so powerful that it now provides an excellent alternative to the special-purpose algorithm.

The fact that these problems are special cases of the minimum cost flow problem is of interest for other reasons as well. One reason is that the underlying theory for the minimum cost flow problem and for the network simplex method provides a unifying theory for all these special cases. Another reason is that some of the many applications of the minimum cost flow problem include features of one or more of the special cases, so it is important to know how to reformulate these features into the broader framework of the general problem.

OR TUTOR

## 9.7 THE NETWORK SIMPLEX METHOD

The network simplex method is a highly streamlined version of the simplex method for solving minimum cost flow problems. As such, it goes through the same basic steps at each iteration—finding the entering basic variable, determining the leaving basic variable, and solving for the new BF solution—in order to move from the current BF solution to a better adjacent one. However, it executes these steps in ways that exploit the special network structure of the problem without ever needing a simplex tableau.

You may note some similarities between the network simplex method and the transportation simplex method presented in Sec. 8.2. In fact, both are streamlined versions of the simplex method that provide alternative algorithms for solving transportation problems in similar ways. The network simplex method extends these ideas to solving other types of minimum cost flow problems as well.

In this section, we provide a somewhat abbreviated description of the network simplex method that focuses just on the main concepts. We omit certain details needed for a full computer implementation, including how to construct an initial BF solution and how to perform certain calculations (such as for finding the entering basic variable) in the most efficient manner. These details are provided in various more specialized textbooks, such as Selected References 1, 2, 3, 5, and 8.

### Incorporating the Upper Bound Technique

The first concept is to incorporate the upper bound technique described in Sec. 7.3 to deal efficiently with the arc capacity constraints $x_{ij} \leq u_{ij}$. Thus, rather than these constraints being treated as *functional* constraints, they are handled just as *nonnegativity* constraints are. Therefore, they are considered only when the leaving basic variable is determined. In particular, as the entering basic variable is increased from zero, the leaving basic variable is the *first* basic variable that reaches either its lower bound (0) or its upper bound ($u_{ij}$). A nonbasic variable at its upper bound $x_{ij} = u_{ij}$ is replaced by $x_{ij} = u_{ij} - y_{ij}$, so $y_{ij} = 0$ becomes the nonbasic variable. See Sec. 7.3 for further details.

In our current context, $y_{ij}$ has an interesting network interpretation. Whenever $y_{ij}$ becomes a basic variable with a strictly positive value ($\leq u_{ij}$), this value can be thought of as flow from node $j$ to node $i$ (so in the "wrong" direction through arc $i \rightarrow j$) that, in actuality, is *canceling* that amount of the previously assigned flow ($x_{ij} = u_{ij}$) from node $i$ to node $j$. Thus, when $x_{ij} = u_{ij}$ is replaced by $x_{ij} = u_{ij} - y_{ij}$, we also replace the *real* arc $i \rightarrow j$ by the **reverse arc** $j \rightarrow i$, where this new arc has arc capacity $u_{ij}$ (the maximum amount of the $x_{ij} = u_{ij}$ flow that can be canceled) and unit cost $-c_{ij}$ (since each unit of flow canceled saves $c_{ij}$). To reflect the flow of $x_{ij} = u_{ij}$ through the deleted arc, we shift this amount of net flow generated from node $i$ to node $j$ by *decreasing* $b_i$ by $u_{ij}$ and *increasing* $b_j$ by $u_{ij}$. Later, if $y_{ij}$ becomes the leaving basic variable by reaching its upper bound, then $y_{ij} = u_{ij}$ is replaced by $y_{ij} = u_{ij} - x_{ij}$ with $x_{ij} = 0$ as the new nonbasic variable, so the above process would be reversed (replace arc $j \rightarrow i$ by arc $i \rightarrow j$, etc.) to the original configuration.

To illustrate this process, consider the minimum cost flow problem shown in Fig. 9.12. While the network simplex method is generating a sequence of BF solutions, suppose that $x_{AB}$ has become the leaving basic variable for some iteration by reaching its upper bound of 10. Consequently, $x_{AB} = 10$ is replaced by $x_{AB} = 10 - y_{AB}$, so $y_{AB} = 0$

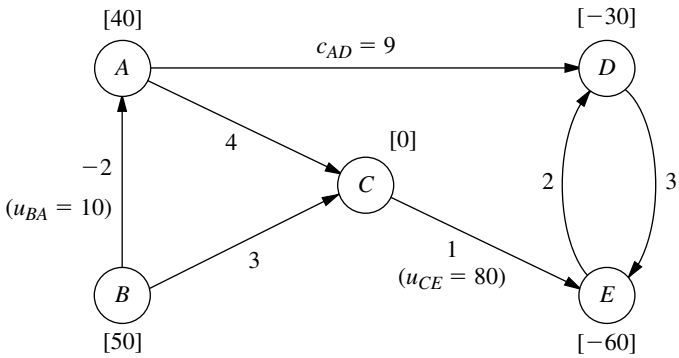



**FIGURE 9.16**
The adjusted network for the example when the upper-bound technique leads to replacing $x_{AB} = 10$ by $x_{AB} = 10 - y_{AB}$.

becomes the new nonbasic variable. At the same time, we replace arc $A \rightarrow B$ by arc $B \rightarrow A$ (with $y_{AB}$ as its flow quantity), and we assign this new arc a capacity of 10 and a unit cost of $-2$. To take $x_{AB} = 10$ into account, we also decrease $b_A$ from 50 to 40 and increase $b_B$ from 40 to 50. The resulting adjusted network is shown in Fig. 9.16.

We shall soon illustrate the entire network simplex method with this same example, starting with $y_{AB} = 0$ ($x_{AB} = 10$) as a nonbasic variable and so using Fig. 9.16. A later iteration will show $x_{CE}$ reaching its upper bound of 80 and so being replaced by $x_{CE} = 80 - y_{CE}$, and so on, and then the next iteration has $y_{AB}$ reaching its upper bound of 10. You will see that all these operations are performed directly on the network, so we will not need to use the $x_{ij}$ or $y_{ij}$ labels for arc flows or even to keep track of which arcs are *real* arcs and which are *reverse* arcs (except when we record the final solution). Using the upper bound technique leaves the *node constraints* (flow out minus flow in $= b_i$) as the only functional constraints. Minimum cost flow problems tend to have far more arcs than nodes, so the resulting number of functional constraints generally is only a small fraction of what it would have been if the arc capacity constraints had been included. The computation time for the simplex method goes up relatively rapidly with the number of functional constraints, but only slowly with the number of variables (or the number of bounding constraints on these variables). Therefore, incorporating the upper bound technique here tends to provide a tremendous saving in computation time.

However, this technique is not needed for *uncapacitated* minimum cost flow problems (including all but the last special case considered in the preceding section), where there are no arc capacity constraints.

### Correspondence between BF Solutions and Feasible Spanning Trees

The most important concept underlying the network simplex method is its network representation of *BF solutions.* Recall from Sec. 9.6 that with $n$ nodes, every BF solution has $(n - 1)$ basic variables, where each basic variable $x_{ij}$ represents the flow through arc $i \rightarrow j$. These $(n - 1)$ arcs are referred to as **basic arcs.** (Similarly, the arcs corresponding to the *nonbasic* variables $x_{ij} = 0$ or $y_{ij} = 0$ are called **nonbasic arcs.**)

A key property of basic arcs is that they never form undirected *cycles.* (This property prevents the resulting solution from being a weighted average of another pair of feasible

solutions, which would violate one of the general properties of BF solutions.) However, *any* set of $n - 1$ arcs that contains no undirected cycles forms a *spanning tree.* Therefore, any complete set of $n - 1$ basic arcs forms a spanning tree.

Thus, BF solutions can be obtained by "solving" spanning trees, as summarized below.

A **spanning tree solution** is obtained as follows:

1. For the arcs *not* in the spanning tree (the nonbasic arcs), set the corresponding variables ($x_{ij}$ or $y_{ij}$) equal to zero.
2. For the arcs that are in the spanning tree (the basic arcs), solve for the corresponding variables ($x_{ij}$ or $y_{ij}$) in the system of linear equations provided by the node constraints.

(The network simplex method actually solves for the new BF solution from the current one much more efficiently, without solving this system of equations from scratch.) Note that this solution process does not consider either the nonnegativity constraints or the arc capacity constraints for the basic variables, so the resulting spanning tree solution may or may not be feasible with respect to these constraints—which leads to our next definition.

A **feasible spanning tree** is a spanning tree whose solution from the node constraints also satisfies all the other constraints ($0 \le x_{ij} \le u_{ij}$ or $0 \le y_{ij} \le u_{ij}$).

With these definitions, we now can summarize our key conclusion as follows:

The **fundamental theorem for the network simplex method** says that basic solutions are *spanning tree solutions* (and conversely) and that BF solutions are solutions for *feasible spanning trees* (and conversely).

To begin illustrating the application of this fundamental theorem, consider the network shown in Fig. 9.16 that results from replacing $x_{AB} = 10$ by $x_{AB} = 10 - y_{AB}$ for our example in Fig. 9.12. One spanning tree for this network is the one shown in Fig. 9.3*e*, where the arcs are $A \to D$, $D \to E$, $C \to E$, and $B \to C$. With these as the *basic arcs,* the process of finding the spanning tree solution is shown below. On the left is the set of node constraints given in Sec. 9.6 after $10 - y_{AB}$ is substituted for $x_{AB}$, where the *basic* variables are shown in **boldface.** On the right, starting at the top and moving down, is the sequence of steps for setting or calculating the values of the variables.

$$
\begin{array}{rrrrrrr|lll}
 & & & & & & & y_{AB} = 0, & x_{AC} = 0, & x_{ED} = 0 \\
\hline
-y_{AB} & + x_{AC} & + \boldsymbol{x_{AD}} & & & & & = 40 & & x_{AD} = 40. \\
y_{AB} & & & + \boldsymbol{x_{BC}} & & & & = 50 & & x_{BC} = 50. \\
 & - x_{AC} & & - \boldsymbol{x_{BC}} & + \boldsymbol{x_{CE}} & & & = 0 & \text{so} & x_{CE} = 50. \\
 & & - \boldsymbol{x_{AD}} & & & + \boldsymbol{x_{DE}} & - x_{ED} & = -30 & \text{so} & x_{DE} = 10. \\
 & & & & - \boldsymbol{x_{CE}} & - \boldsymbol{x_{DE}} & + x_{ED} & = -60 & \text{Redundant.} &
\end{array}
$$

Since the values of all these basic variables satisfy the nonnegativity constraints and the one relevant arc capacity constraint ($x_{CE} \le 80$), the spanning tree is a *feasible spanning tree,* so we have a *BF solution.*

We shall use this solution as the initial BF solution for demonstrating the network simplex method. Figure 9.17 shows its network representation, namely, the feasible spanning tree and its solution. Thus, the numbers given next to the arcs now represent *flows* (values of $x_{ij}$) rather than the unit costs $c_{ij}$ previously given. (To help you distinguish, we shall always put parentheses around flows but not around costs.)

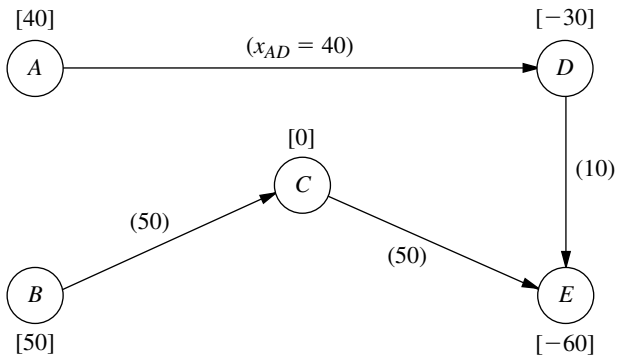



**FIGURE 9.17**
The initial feasible spanning tree and its solution for the example.

### Selecting the Entering Basic Variable

To begin an iteration of the network simplex method, recall that the standard simplex method criterion for selecting the entering basic variable is to choose the nonbasic variable which, when increased from zero, will *improve Z at the fastest rate.* Now let us see how this is done without having a simplex tableau.

To illustrate, consider the nonbasic variable $x_{AC}$ in our initial BF solution, i.e., the nonbasic arc $A \rightarrow C$. Increasing $x_{AC}$ from zero to some value $\theta$ means that the arc $A \rightarrow C$ with flow $\theta$ must be added to the network shown in Fig. 9.17. Adding a nonbasic arc to a spanning tree *always* creates a unique undirected *cycle,* where the cycle in this case is seen in Fig. 9.18 to be *AC–CE–DE–AD*. Figure 9.18 also shows the effect of adding the flow $\theta$ to arc $A \rightarrow C$ on the other flows in the network. Specifically, the flow is thereby *increased* by $\theta$ for other arcs that have the *same* direction as $A \rightarrow C$ in the cycle (arc $C \rightarrow E$), whereas the *net* flow is *decreased* by $\theta$ for other arcs whose direction is *opposite* to $A \rightarrow C$ in the cycle (arcs $D \rightarrow E$ and $A \rightarrow D$). In the latter case, the new flow is, in effect, canceling a flow of $\theta$ in the opposite direction. Arcs not in the cycle (arc $B \rightarrow C$) are unaffected by the new flow. (Check these conclusions by noting the effect of the change in $x_{AC}$ on the values of the other variables in the solution just derived for the initial feasible spanning tree.)

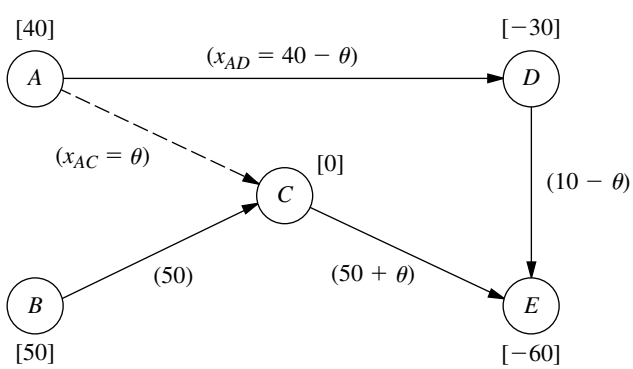



**FIGURE 9.18**
The effect on flows of adding arc $A \rightarrow C$ with flow $\theta$ to the initial feasible spanning tree.

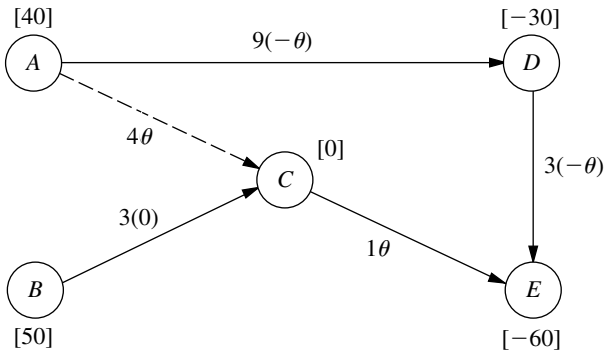



**FIGURE 9.19**
The incremental effect on costs of adding arc $A \rightarrow C$ with flow $\theta$ to the initial feasible spanning tree.

Now what is the incremental effect on $Z$ (total flow cost) from adding the flow $\theta$ to arc $A \rightarrow C$? Figure 9.19 shows most of the answer by giving the unit cost times the change in the flow for each arc of Fig. 9.18. Therefore, the overall increment in $Z$ is

$$\begin{aligned}\Delta Z &= c_{AC}\theta + c_{CE}\theta + c_{DE}(-\theta) + c_{AD}(-\theta) \\ &= 4\theta + \theta - 3\theta - 9\theta \\ &= -7\theta.\end{aligned}$$

Setting $\theta = 1$ then gives the *rate* of change of $Z$ as $x_{AC}$ is increased, namely,

$$\Delta Z = -7, \qquad \text{when } \theta = 1.$$

Because the objective is to *minimize* Z, this large rate of decrease in $Z$ by increasing $x_{AC}$ is very desirable, so $x_{AC}$ becomes a prime candidate to be the entering basic variable.

We now need to perform the same analysis for the other nonbasic variables before we make the final selection of the entering basic variable. The only other nonbasic variables are $y_{AB}$ and $x_{ED}$, corresponding to the two other nonbasic arcs $B \rightarrow A$ and $E \rightarrow D$ in Fig. 9.16.

Figure 9.20 shows the incremental effect on costs of adding arc $B \rightarrow A$ with flow $\theta$ to the initial feasible spanning tree given in Fig. 9.17. Adding this arc creates the undi-

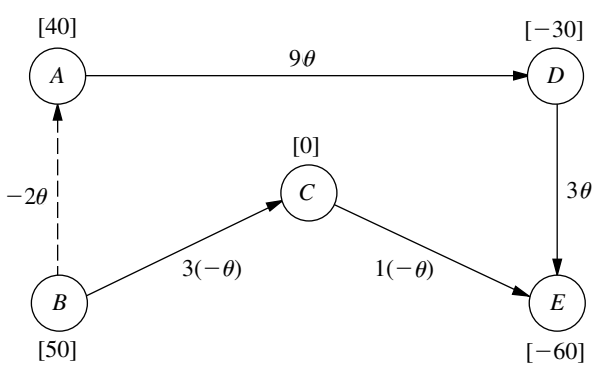



**FIGURE 9.20**
The incremental effect on costs of adding arc $B \rightarrow A$ with flow $\theta$ to the initial feasible spanning tree.

rected cycle *BA–AD–DE–CE–BC*, so the flow increases by $\theta$ for arcs $A \rightarrow D$ and $D \rightarrow E$ but decreases by $\theta$ for the two arcs in the opposite direction on this cycle, $C \rightarrow E$ and $B \rightarrow C$. These flow increments, $\theta$ and $-\theta$, are the multiplicands for the $c_{ij}$ values in the figure. Therefore,

$$\begin{aligned}\Delta Z &= -2\theta + 9\theta + 3\theta + 1(-\theta) + 3(-\theta) = 6\theta \\ &= 6, \qquad \text{when } \theta = 1.\end{aligned}$$

The fact that $Z$ *increases* rather than decreases when $y_{AB}$ (flow through the reverse arc $B \rightarrow A$) is increased from zero rules out this variable as a candidate to be the entering basic variable. (Remember that increasing $y_{AB}$ from zero really means decreasing $x_{AB}$, flow through the real arc $A \rightarrow B$, from its upper bound of 10.)

A similar result is obtained for the last nonbasic arc $E \rightarrow D$. Adding this arc with flow $\theta$ to the initial feasible spanning tree creates the undirected cycle *ED–DE* shown in Fig. 9.21, so the flow also increases by $\theta$ for arc $D \rightarrow E$, but no other arcs are affected. Therefore,

$$\begin{aligned}\Delta Z &= 2\theta + 3\theta = 5\theta \\ &= 5, \qquad \text{when } \theta = 1,\end{aligned}$$

so $x_{ED}$ is ruled out as a candidate to be the entering basic variable.

To summarize,

$$\Delta Z = \begin{cases} -7, & \text{if } \Delta x_{AC} = 1 \\ 6, & \text{if } \Delta y_{AB} = 1 \\ 5, & \text{if } \Delta x_{ED} = 1 \end{cases}$$

so the negative value for $x_{AC}$ implies that $x_{AC}$ becomes the entering basic variable for the first iteration. If there had been more than one nonbasic variable with a *negative* value of $\Delta Z$, then the one having the *largest* absolute value would have been chosen. (If there had been no nonbasic variables with a negative value of $\Delta Z$, the current BF solution would have been optimal.)

Rather than identifying undirected cycles, etc., the network simplex method actually obtains these $\Delta Z$ values by an algebraic procedure that is considerably more efficient (especially for large networks). The procedure is analogous to that used by the transportation simplex method (see Sec. 8.2) to solve for $u_i$ and $v_j$ in order to obtain the value of

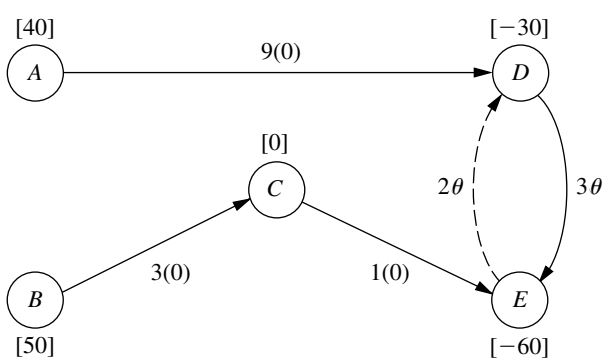



**FIGURE 9.21**
The incremental effect on costs of adding arc $E \rightarrow D$ with flow $\theta$ to the initial feasible spanning tree.

$c_{ij} - u_i - v_j$ for each nonbasic variable $x_{ij}$. We shall not describe this procedure further, so you should just use the undirected cycles method when you are doing problems at the end of the chapter.

### Finding the Leaving Basic Variable and the Next BF Solution

After selection of the entering basic variable, only one more quick step is needed to simultaneously determine the leaving basic variable and solve for the next BF solution. For the first iteration of the example, the key is Fig. 9.18. Since $x_{AC}$ is the entering basic variable, the flow $\theta$ through arc $A \rightarrow C$ is to be increased from zero as far as possible until one of the basic variables reaches *either* its lower bound (0) or its upper bound ($u_{ij}$). For those arcs whose flow *increases* with $\theta$ in Fig. 9.18 (arcs $A \rightarrow C$ and $C \rightarrow E$), only the *upper* bounds ($u_{AC} = \infty$ and $u_{CE} = 80$) need to be considered:

$$x_{AC} = \theta \leq \infty.$$
$$x_{CE} = 50 + \theta \leq 80, \quad \text{so} \quad \theta \leq 30.$$

For those arcs whose flow *decreases* with $\theta$ (arcs $D \rightarrow E$ and $A \rightarrow D$), only the *lower* bound of 0 needs to be considered:

$$x_{DE} = 10 - \theta \geq 0, \quad \text{so} \quad \theta \leq 10.$$
$$x_{AD} = 40 - \theta \geq 0, \quad \text{so} \quad \theta \leq 40.$$

Arcs whose flow is unchanged by $\theta$ (i.e., those not part of the undirected cycle), which is just arc $B \rightarrow C$ in Fig. 9.18, can be ignored since no bound will be reached as $\theta$ is increased.

For the five arcs in Fig. 9.18, the conclusion is that $x_{DE}$ must be the leaving basic variable because it reaches a bound for the smallest value of $\theta$ (10). Setting $\theta = 10$ in this figure thereby yields the flows through the basic arcs in the next BF solution:

$$x_{AC} = \theta = 10,$$
$$x_{CE} = 50 + \theta = 60,$$
$$x_{AD} = 40 - \theta = 30,$$
$$x_{BC} = 50.$$

The corresponding feasible spanning tree is shown in Fig. 9.22.

If the leaving basic variable had reached its upper bound, then the adjustments discussed for the upper bound technique would have been needed at this point (as you will

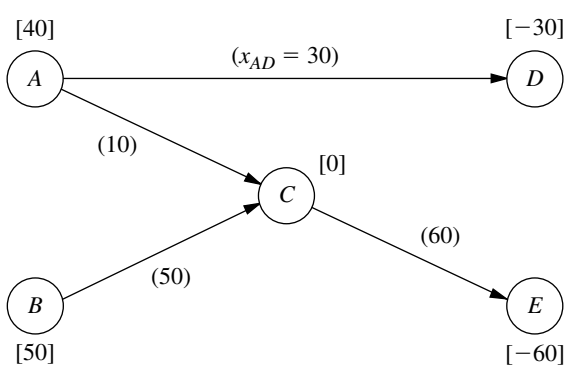



**FIGURE 9.22**
The second feasible spanning tree and its solution for the example.

**TABLE 9.4 Calculations for selecting the entering basic variable for iteration 2**

| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ | |
|---|---|---|---|
| $B \rightarrow A$ | *BA–AC–BC* | $-2 + 4 - 3 = -1$ | |
| $D \rightarrow E$ | *DE–CE–AC–AD* | $3 - 1 - 4 + 9 = 7$ | |
| $E \rightarrow D$ | *ED–AD–AC–CE* | $2 - 9 + 4 + 1 = -2$ | ← Minimum |

see illustrated during the next two iterations). However, because it was the lower bound of 0 that was reached, nothing more needs to be done.

**Completing the Example.** For the two remaining iterations needed to reach the optimal solution, the primary focus will be on some features of the upper bound technique they illustrate. The pattern for finding the entering basic variable, the leaving basic variable, and the next BF solution will be very similar to that described for the first iteration, so we only summarize these steps briefly.

*Iteration 2:* Starting with the feasible spanning tree shown in Fig. 9.22 and referring to Fig. 9.16 for the unit costs $c_{ij}$, we arrive at the calculations for selecting the entering basic variable in Table 9.4. The second column identifies the unique undirected cycle that is created by adding the nonbasic arc in the first column to this spanning tree, and the third column shows the incremental effect on costs because of the changes in flows on this cycle caused by adding a flow of $\theta = 1$ to the nonbasic arc. Arc $E \rightarrow D$ has the largest (in absolute terms) negative value of $\Delta Z$, so $x_{ED}$ is the entering basic variable.

We now make the flow $\theta$ through arc $E \rightarrow D$ as large as possible, while satisfying the following flow bounds:

$$
\begin{array}{llll}
x_{ED} = \theta \leq u_{ED} = \infty, & \text{so} & \theta \leq \infty. & \\
x_{AD} = 30 - \theta \geq 0, & \text{so} & \theta \leq 30. & \\
x_{AC} = 10 + \theta \leq u_{AC} = \infty, & \text{so} & \theta \leq \infty. & \\
x_{CE} = 60 + \theta \leq u_{CE} = 80, & \text{so} & \theta \leq 20. & \leftarrow \text{Minimum}
\end{array}
$$

Because $x_{CE}$ imposes the smallest upper bound (20) on $\theta$, $x_{CE}$ becomes the leaving basic variable. Setting $\theta = 20$ in the above expressions for $x_{ED}$, $x_{AD}$, and $x_{AC}$ then yields the flow through the basic arcs for the next BF solution (with $x_{BC} = 50$ unaffected by $\theta$), as shown in Fig. 9.23.

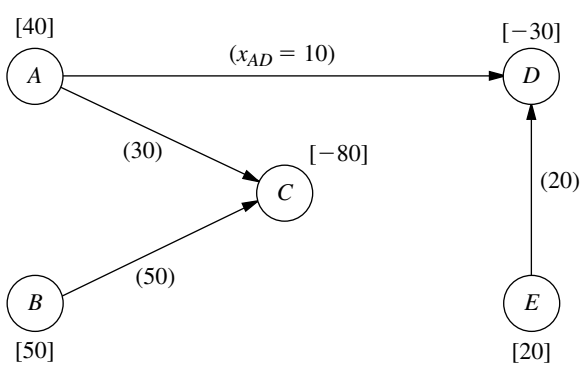



**FIGURE 9.23**
The third feasible spanning tree and its solution for the example.

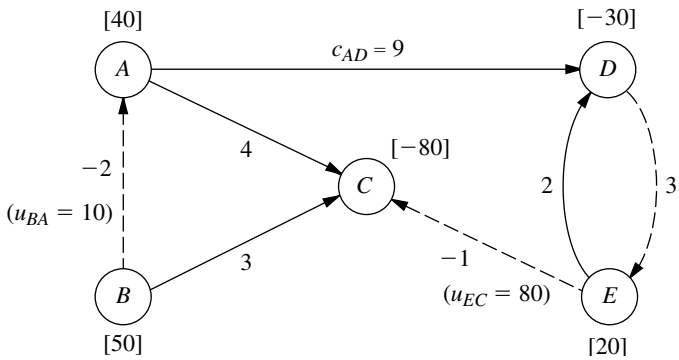



**FIGURE 9.24**
The adjusted network with unit costs at the completion of iteration 2.

What is of special interest here is that the leaving basic variable $x_{CE}$ was obtained by the variable reaching its upper bound (80). Therefore, by using the upper bound technique, $x_{CE}$ is replaced by $80 - y_{CE}$, where $y_{CE} = 0$ is the new nonbasic variable. At the same time, the original arc $C \rightarrow E$ with $c_{CE} = 1$ and $u_{CE} = 80$ is replaced by the reverse arc $E \rightarrow C$ with $c_{EC} = -1$ and $u_{EC} = 80$. The values of $b_E$ and $b_C$ also are adjusted by adding 80 to $b_E$ and subtracting 80 from $b_C$. The resulting adjusted network is shown in Fig. 9.24, where the nonbasic arcs are shown as dashed lines and the numbers by all the arcs are unit costs.

*Iteration 3:* If Figs. 9.23 and 9.24 are used to initiate the next iteration, Table 9.5 shows the calculations that lead to selecting $y_{AB}$ (reverse arc $B \rightarrow A$) as the entering basic variable. We then add as much flow $\theta$ through arc $B \rightarrow A$ as possible while satisfying the flow bounds below:

$$\begin{array}{lll} y_{AB} = \theta \leq u_{BA} = 10, & \text{so} & \theta \leq 10. \quad \leftarrow \text{Minimum} \\ x_{AC} = 30 + \theta \leq u_{AC} = \infty, & \text{so} & \theta \leq \infty. \\ x_{BC} = 50 - \theta \geq 0, & \text{so} & \theta \leq 50. \end{array}$$

The smallest upper bound (10) on $\theta$ is imposed by $y_{AB}$, so this variable becomes the leaving basic variable. Setting $\theta = 10$ in these expressions for $x_{AC}$ and $x_{BC}$ (along with the unchanged values of $x_{AC} = 10$ and $x_{ED} = 20$) then yields the next BF solution, as shown in Fig. 9.25.

As with iteration 2, the leaving basic variable ($y_{AB}$) was obtained here by the variable reaching its upper bound. In addition, there are two other points of special interest concerning this particular choice. One is that the *entering* basic variable $y_{AB}$ also became

**TABLE 9.5 Calculations for selecting the entering basic variable for iteration 3**

| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ | |
|---|---|---|---|
| $B \rightarrow A$ | *BA–AC–BC* | $-2 + 4 - 3 = -1$ | ← Minimum |
| $D \rightarrow E$ | *DE–ED* | $3 + 2 = 5$ | |
| $E \rightarrow C$ | *EC–AC–AD–ED* | $-1 - 4 + 9 - 2 = 2$ | |

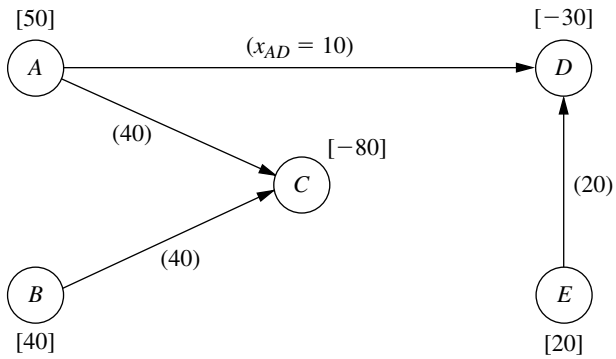



**FIGURE 9.25**
The fourth (and final) feasible spanning tree and its solution for the example.

the *leaving* basic variable on the same iteration! This event occurs occasionally with the upper bound technique whenever increasing the entering basic variable from zero causes *its* upper bound to be reached first before any of the other basic variables reach a bound.

The other interesting point is that the arc $B \rightarrow A$ that now needs to be replaced by a *reverse* arc $A \rightarrow B$ (because of the leaving basic variable reaching an upper bound) already is a reverse arc! This is no problem, because the reverse arc for a reverse arc is simply the original *real* arc. Therefore, the arc $B \rightarrow A$ (with $c_{BA} = -2$ and $u_{BA} = 10$) in Fig. 9.24 now is replaced by arc $A \rightarrow B$ (with $c_{AB} = 2$ and $u_{AB} = 10$), which is the arc between nodes $A$ and $B$ in the original network shown in Fig. 9.12, and a generated net flow of 10 is shifted from node $B$ ($b_B = 50 \rightarrow 40$) to node $A$ ($b_A = 40 \rightarrow 50$). Simultaneously, the variable $y_{AB} = 10$ is replaced by $10 - x_{AB}$, with $x_{AB} = 0$ as the new nonbasic variable. The resulting adjusted network is shown in Fig. 9.26.

*Passing the Optimality Test:* At this point, the algorithm would attempt to use Figs. 9.25 and 9.26 to find the next entering basic variable with the usual calculations shown in Table 9.6. However, *none* of the nonbasic arcs gives a *negative* value of $\Delta Z$, so an improvement in $Z$ *cannot* be achieved by introducing flow through any of them. This means that the current BF solution shown in Fig. 9.25 has *passed* the optimality test, so the algorithm stops.

To identify the flows through real arcs rather than reverse arcs for this optimal solution, the current adjusted network (Fig. 9.26) should be compared with the original network (Fig. 9.12). Note that each of the arcs has the same direction in the two networks with the one exception of the arc between nodes $C$ and $E$. This means that the only re-

**TABLE 9.6 Calculations for the optimality test at the end of iteration 3**

| Nonbasic Arc | Cycle Created | $\Delta Z$ When $\theta = 1$ |
|---|---|---|
| $A \rightarrow B$ | *AB–BC–AC* | $2 + 3 - 4 = 1$ |
| $D \rightarrow E$ | *DE–EC–AC–AD* | $3 - 1 - 4 + 9 = 7$ |
| $E \rightarrow C$ | *EC–AC–AD–ED* | $-1 - 4 + 9 - 2 = 2$ |

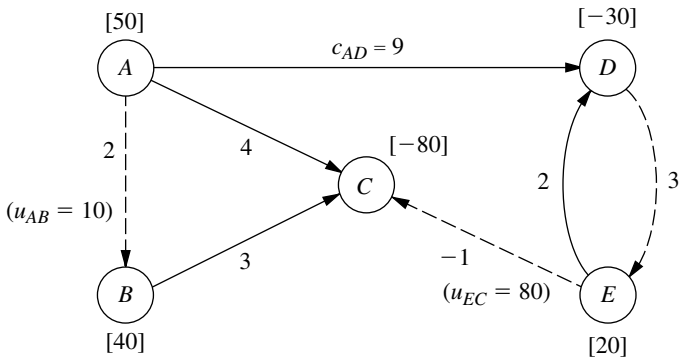



**FIGURE 9.26**
The adjusted network with unit costs at the completion of iteration 3.

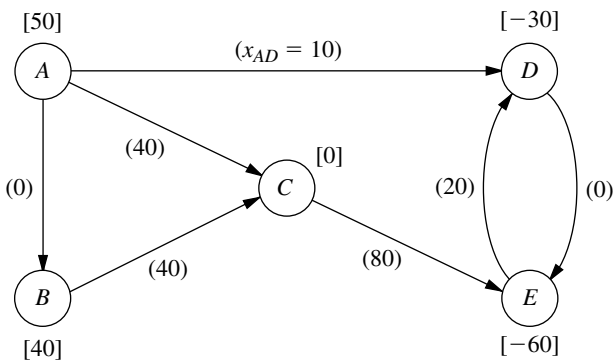



**FIGURE 9.27**
The optimal flow pattern in the original network for the Distribution Unlimited Co. example.

verse arc in Fig. 9.26 is arc $E \rightarrow C$, where its flow is given by the variable $y_{CE}$. Therefore, calculate $x_{CE} = u_{CE} - y_{CE} = 80 - y_{CE}$. Arc $E \rightarrow C$ happens to be a nonbasic arc, so $y_{CE} = 0$ and $x_{CE} = 80$ is the flow through the real arc $C \rightarrow E$. All the other flows through real arcs are the flows given in Fig. 9.25. Therefore, the optimal solution is the one shown in Fig. 9.27.

Another complete example of applying the network simplex method is provided by the demonstration in the *Network Analysis Area* of your OR Tutor. Also included in your OR Courseware is an interactive routine for the network simplex method.

## 9.8 CONCLUSIONS

Networks of some type arise in a wide variety of contexts. Network representations are very useful for portraying the relationships and connections between the components of systems. Frequently, flow of some type must be sent through a network, so a decision needs to be made about the best way to do this. The kinds of network optimization models and algorithms introduced in this chapter provide a powerful tool for making such decisions.

The minimum cost flow problem plays a central role among these network optimization models, both because it is so broadly applicable and because it can be solved extremely efficiently by the network simplex method. Two of its special cases included in this chapter, the shortest-path problem and the maximum flow problem, also are basic network optimization models, as are additional special cases discussed in Chap. 8 (the transportation problem and the assignment problem).

Whereas all these models are concerned with optimizing the *operation* of an *existing* network, the minimum spanning tree problem is a prominent example of a model for optimizing the *design* of a *new* network.

This chapter has only scratched the surface of the current state of the art of network methodology. Because of their combinatorial nature, network problems often are extremely difficult to solve. However, great progress is being made in developing powerful modeling techniques and solution methodologies that are opening up new vistas for important applications. In fact, recent algorithmic advances are enabling us to solve successfully some complex network problems of enormous size.

## SELECTED REFERENCES


**1.** Ahuja, R. K., T. L. Magnanti, and J. B. Orlin: *Network Flows: Theory, Algorithms, and Applications,* Prentice-Hall, Englewood Cliffs, NJ, 1993.

**2.** Ball, M., T. L. Magnanti, C. Monma, and G. L. Nemhauser: *Network Models,* Elsevier, New York, 1995.

**3.** Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali: *Linear Programming and Network Flows,* 2d ed., Wiley, New York, 1990.

**4.** Dantzig, G. B., and M. N. Thapa: *Linear Programming 1: Introduction,* Springer, New York, 1997, chap. 9.

**5.** Glover, F., D. Klingman, and N. V. Phillips: *Network Models in Optimization and Their Applications in Practice,* Wiley, New York, 1992.

**6.** Hillier, F. S., M. S. Hillier, and G. J. Lieberman: *Introduction to Management Science: A Modeling and Case Studies Approach with Spreadsheets,* Irwin/McGraw-Hill, Burr Ridge, IL, 2000, chap. 7.

**7.** Magnanti, T. L., and R. T. Wong: "Network Design and Transportation Planning: Models and Algorithms," *Transportation Science,* **18:** 1–55, 1984.

**8.** Murty, K. G.: *Network Programming,* Prentice-Hall, Englewood Cliffs, NJ, 1992.


## LEARNING AIDS FOR THIS CHAPTER IN YOUR OR COURSEWARE

### A Demonstration Example in OR Tutor:

Network Simplex Method

### An Interactive Routine:

Network Simplex Method—Interactive

### An Excel Add-in:

Premium Solver

### "Ch. 9—Network Opt Models" Files for Solving the Examples:

Excel File
LINGO/LINDO File
MPL/CPLEX File

See Appendix 1 for documentation of the software.

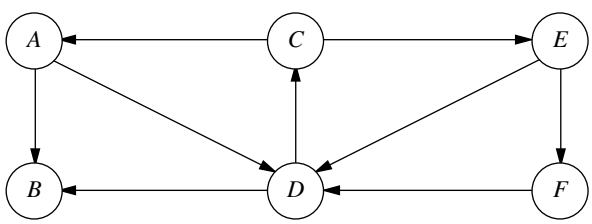
## PROBLEMS

The symbols to the left of some of the problems (or their parts) have the following meaning:

D: The demonstration example listed above may be helpful.
I: We suggest that you use the interactive routine listed above (the printout records your work).
C: Use the computer with any of the software options available to you (or as instructed by your instructor) to solve the problem.

An asterisk on the problem number indicates that at least a partial answer is given in the back of the book.

**9.2-1.** Consider the following directed network.




**(a)** Find a directed path from node *A* to node *F*, and then identify three other undirected paths from node *A* to node *F*.
**(b)** Find three directed cycles. Then identify an undirected cycle that includes every node.
**(c)** Identify a set of arcs that forms a spanning tree.
**(d)** Use the process illustrated in Fig. 9.3 to grow a tree one arc at a time until a spanning tree has been formed. Then repeat this process to obtain another spanning tree. [Do not duplicate the spanning tree identified in part (*c*).]

**9.3-1.** You need to take a trip by car to another town that you have never visited before. Therefore, you are studying a map to determine the shortest route to your destination. Depending on which route you choose, there are five other towns (call them A, B, C, D, E) that you might pass through on the way. The map shows the mileage along each road that directly connects two towns without any intervening towns. These numbers are summarized in the following table, where a dash indicates that there is no road directly connecting these two towns without going through any other towns.

| | Miles between Adjacent Towns | | | | | |
|---|---|---|---|---|---|---|
| **Town** | **A** | **B** | **C** | **D** | **E** | **Destination** |
| Origin | 40 | 60 | 50 | — | — | — |
| A | | 10 | — | 70 | — | — |
| B | | | 20 | 55 | 40 | — |
| C | | | | — | 50 | — |
| D | | | | | 10 | 60 |
| E | | | | | | 80 |

**(a)** Formulate this problem as a shortest-path problem by drawing a network where nodes represent towns, links represent roads, and numbers indicate the length of each link in miles.
**(b)** Use the algorithm described in Sec. 9.3 to solve this shortest-path problem.
C **(c)** Formulate and solve a spreadsheet model for this problem.
**(d)** If each number in the table represented your *cost* (in dollars) for driving your car from one town to the next, would the answer in part (*b*) or (*c*) now give your minimum cost route?
**(e)** If each number in the table represented your *time* (in minutes) for driving your car from one town to the next, would the answer in part (*b*) or (*c*) now give your minimum time route?

**9.3-2.** At a small but growing airport, the local airline company is purchasing a new tractor for a tractor-trailer train to bring luggage to and from the airplanes. A new mechanized luggage system will be installed in 3 years, so the tractor will not be needed after that. However, because it will receive heavy use, so that the running and

maintenance costs will increase rapidly as the tractor ages, it may still be more economical to replace the tractor after 1 or 2 years. The following table gives the total net discounted cost associated with purchasing a tractor (purchase price minus trade-in allowance, plus running and maintenance costs) at the end of year $i$ and trading it in at the end of year $j$ (where year 0 is now).

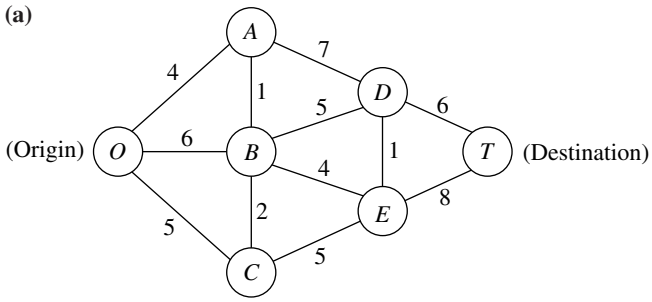
| | | $j$ | | |
|---|---|---|---|---|
| | | **1** | **2** | **3** |
| $i$ | 0 | \$8,000 | \$18,000 | \$31,000 |
| | 1 | | \$10,000 | \$21,000 |
| | 2 | | | \$12,000 |

The problem is to determine at what times (if any) the tractor should be replaced to minimize the total cost for the tractors over 3 years. (Continue at the top of the next column.)

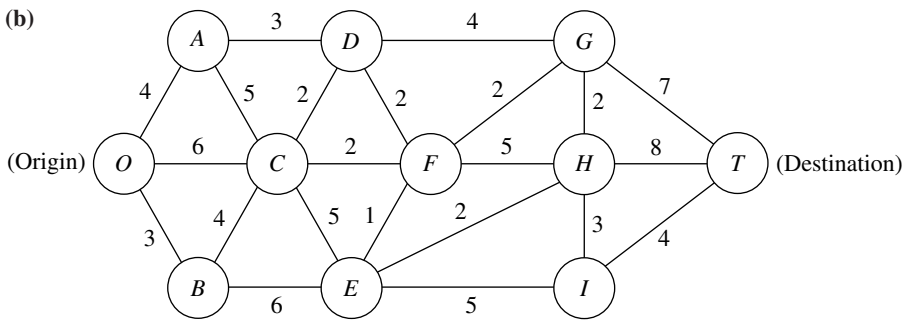
**(a)** Formulate this problem as a shortest-path problem.
**(b)** Use the algorithm described in Sec. 9.3 to solve this shortest-path problem.
C **(c)** Formulate and solve a spreadsheet model for this problem.

**9.3-3.*** Use the algorithm described in Sec. 9.3 to find the *shortest path* through each of the following networks, where the numbers represent actual distances between the corresponding nodes.

**(a)**

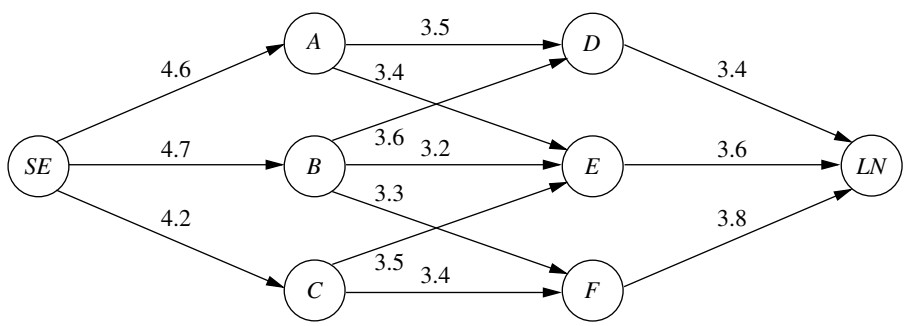



**(b)**




**9.3-4.** Formulate the shortest-path problem as a linear programming problem.

**9.3-5.** One of Speedy Airlines' flights is about to take off from Seattle for a nonstop flight to London. There is some flexibility in choosing the precise route to be taken, depending upon weather conditions. The following network depicts the possible routes under consideration, where SE and LN are Seattle and London, respectively, and the other nodes represent various intermediate locations. The winds along each arc greatly affect the flying time (and so the fuel consumption). Based on current meteorological reports, the flying times (in hours) for this particular flight are shown next to the arcs. Because the fuel consumed is so expensive, the management of Speedy Airlines has established a policy of choosing the route that minimizes the total flight time.



**(a)** What plays the role of "distances" in interpreting this problem to be a shortest-path problem?
**(b)** Use the algorithm described in Sec. 9.3 to solve this shortest-path problem.
C **(c)** Formulate and solve a spreadsheet model for this problem.

**9.3-6.** The Quick Company has learned that a competitor is planning to come out with a new kind of product with a great sales potential. Quick has been working on a similar product that had been scheduled to come to market in 20 months. However, research is nearly complete and Quick's management now wishes to rush the product out to meet the competition.

There are four nonoverlapping phases left to be accomplished, including the remaining research that currently is being conducted at a normal pace. However, each phase can instead be conducted at a priority or crash level to expedite completion, and these are the only levels that will be considered for the last three phases. The times required at these levels are given in the following table. (The times in parentheses at the normal level have been ruled out as too long.)

| | Time | | | |
|---|---|---|---|---|
| Level | Remaining Research | Development | Design of Manufacturing System | Initiate Production and Distribution |
| Normal | 5 months | (4 months) | (7 months) | (4 months) |
| Priority | 4 months | 3 months | 5 months | 2 months |
| Crash | 2 months | 2 months | 3 months | 1 month |

Management has allocated $30 million for these four phases. The cost of each phase at the different levels under consideration is as follows:

| | Cost | | | |
|---|---|---|---|---|
| Level | Remaining Research | Development | Design of Manufacturing System | Initiate Production and Distribution |
| Normal | $3 million | — | — | — |
| Priority | $6 million | $6 million | $9 million | $3 million |
| Crash | $9 million | $9 million | $12 million | $6 million |

Management wishes to determine at which level to conduct each of the four phases to minimize the total time until the product can be marketed subject to the budget restriction of $30 million.
**(a)** Formulate this problem as a shortest-path problem.
**(b)** Use the algorithm described in Sec. 9.3 to solve this shortest-path problem.

**9.4-1.*** Reconsider the networks shown in Prob. 9.3-3. Use the algorithm described in Sec. 9.4 to find the *minimum spanning tree* for each of these networks.

**9.4-2.** The Wirehouse Lumber Company will soon begin logging eight groves of trees in the same general area. Therefore, it must develop a system of dirt roads that makes each grove accessible from every other grove. The distance (in miles) between every pair of groves is as follows:

| | | Distance between Pairs of Groves | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Grove | 1 | — | 1.3 | 2.1 | 0.9 | 0.7 | 1.8 | 2.0 | 1.5 |
| | 2 | 1.3 | — | 0.9 | 1.8 | 1.2 | 2.6 | 2.3 | 1.1 |
| | 3 | 2.1 | 0.9 | — | 2.6 | 1.7 | 2.5 | 1.9 | 1.0 |
| | 4 | 0.9 | 1.8 | 2.6 | — | 0.7 | 1.6 | 1.5 | 0.9 |
| | 5 | 0.7 | 1.2 | 1.7 | 0.7 | — | 0.9 | 1.1 | 0.8 |
| | 6 | 1.8 | 2.6 | 2.5 | 1.6 | 0.9 | — | 0.6 | 1.0 |
| | 7 | 2.0 | 2.3 | 1.9 | 1.5 | 1.1 | 0.6 | — | 0.5 |
| | 8 | 1.5 | 1.1 | 1.0 | 0.9 | 0.8 | 1.0 | 0.5 | — |

Management now wishes to determine between which pairs of groves the roads should be constructed to connect all groves with a minimum total length of road.
**(a)** Describe how this problem fits the network description of the minimum spanning tree problem.
**(b)** Use the algorithm described in Sec. 9.4 to solve the problem.

**9.4-3.** The Premiere Bank soon will be hooking up computer terminals at each of its branch offices to the computer at its main office using special phone lines with telecommunications devices. The phone line from a branch office need not be connected directly to the main office. It can be connected indirectly by being connected to another branch office that is connected (directly or indirectly) to the main office. The only requirement is that every branch office be connected by some route to the main office.

The charge for the special phone lines is $100 times the number of miles involved, where the distance (in miles) between every pair of offices is as follows:

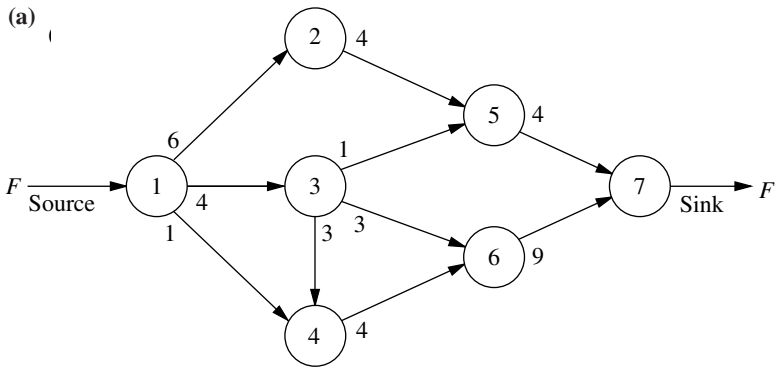
| | Distance between Pairs of Offices | | | | | |
|---|---|---|---|---|---|---|
| | Main | B.1 | B.2 | B.3 | B.4 | B.5 |
| Main office | — | 190 | 70 | 115 | 270 | 160 |
| Branch 1 | 190 | — | 100 | 110 | 215 | 50 |
| Branch 2 | 70 | 100 | — | 140 | 120 | 220 |
| Branch 3 | 115 | 110 | 140 | — | 175 | 80 |
| Branch 4 | 270 | 215 | 120 | 175 | — | 310 |
| Branch 5 | 160 | 50 | 220 | 80 | 310 | — |

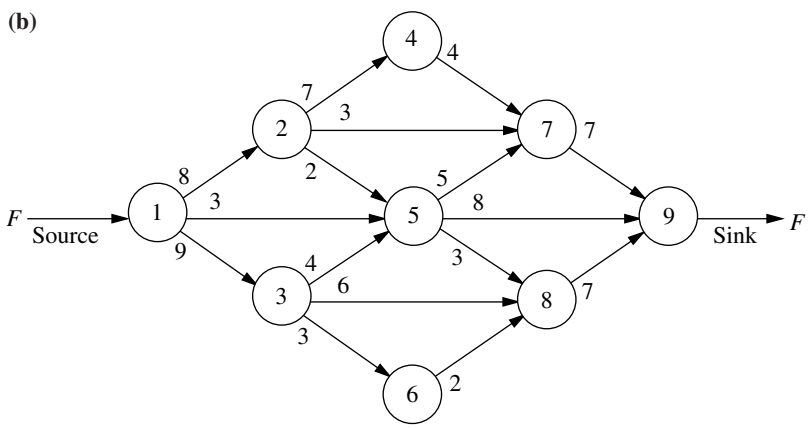
Management wishes to determine which pairs of offices should be directly connected by special phone lines in order to connect every branch office (directly or indirectly) to the main office at a minimum total cost.

**(a)** Describe how this problem fits the network description of the minimum spanning tree problem.

**(b)** Use the algorithm described in Sec. 9.4 to solve the problem.

**9.5-1.*** For networks (*a*) and (*b*), use the augmenting path algorithm described in Sec. 9.5 to find the flow pattern giving the *maximum flow* from the source to the sink, given that the arc capacity from node $i$ to node $j$ is the number nearest node $i$ along the arc between these nodes.

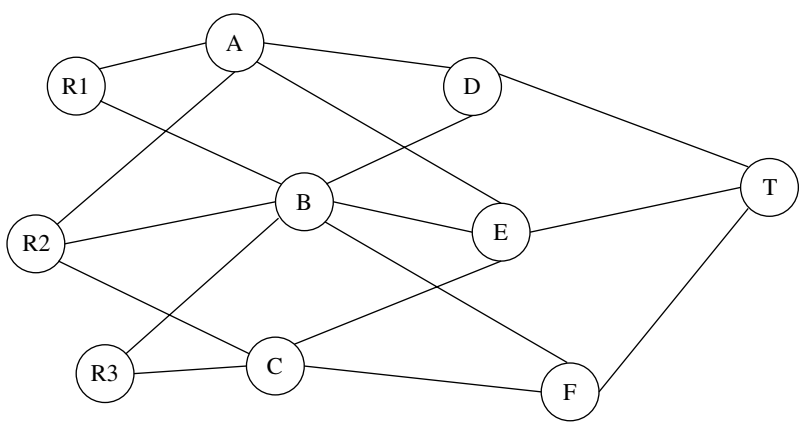



**9.5-2.** Formulate the maximum flow problem as a linear programming problem.

**9.5-3.** The diagram to the right depicts a system of aqueducts that originate at three rivers (nodes R1, R2, and R3) and terminate at a major city (node T), where the other nodes are junction points in the system.

Using units of thousands of acre feet, the following tables show the maximum amount of water that can be pumped through each aqueduct per day.



| From \ To | A | B | C |
|---|---|---|---|
| R1 | 75 | 65 | — |
| R2 | 40 | 50 | 60 |
| R3 | — | 80 | 70 |

| From \ To | D | E | F |
|---|---|---|---|
| A | 60 | 45 | — |
| B | 70 | 55 | 45 |
| C | — | 70 | 90 |

| From \ To | T |
|---|---|
| D | 120 |
| E | 190 |
| F | 130 |

The city water manager wants to determine a flow plan that will maximize the flow of water to the city.

**(a)** Formulate this problem as a maximum flow problem by identifying a source, a sink, and the transshipment nodes, and then drawing the complete network that shows the capacity of each arc.

**(b)** Use the augmenting path algorithm described in Sec. 9.5 to solve this problem.

C **(c)** Formulate and solve a spreadsheet model for this problem.

**9.5-4.** The Texago Corporation has four oil fields, four refineries, and four distribution centers. A major strike involving the transportation industries now has sharply curtailed Texago's capacity to ship oil from the oil fields to the refineries and to ship petroleum products from the refineries to the distribution centers. Using units of thousands of barrels of crude oil (and its equivalent in refined products), the following tables show the maximum number of units that can be shipped per day from each oil field to each refinery, and from each refinery to each distribution center.

| | Refinery | | | |
|---|---|---|---|---|
| **Oil Field** | **New Orleans** | **Charleston** | **Seattle** | **St. Louis** |
| Texas | 11 | 7 | 2 | 8 |
| California | 5 | 4 | 8 | 7 |
| Alaska | 7 | 3 | 12 | 6 |
| Middle East | 8 | 9 | 4 | 15 |

| | Distribution Center | | | |
|---|---|---|---|---|
| **Refinery** | **Pittsburgh** | **Atlanta** | **Kansas City** | **San Francisco** |
| New Orleans | 5 | 9 | 6 | 4 |
| Charleston | 8 | 7 | 9 | 5 |
| Seattle | 4 | 6 | 7 | 8 |
| St. Louis | 12 | 11 | 9 | 7 |

The Texago management now wants to determine a plan for how many units to ship from each oil field to each refinery and from each refinery to each distribution center that will maximize the total number of units reaching the distribution centers.

**(a)** Draw a rough map that shows the location of Texago's oil fields, refineries, and distribution centers. Add arrows to show the flow of crude oil and then petroleum products through this distribution network.

**(b)** Redraw this distribution network by lining up all the nodes representing oil fields in one column, all the nodes representing refineries in a second column, and all the nodes representing distribution centers in a third column. Then add arcs to show the possible flow.

**(c)** Modify the network in part (*b*) as needed to formulate this problem as a maximum flow problem with a single source, a single sink, and a capacity for each arc.

**(d)** Use the augmenting path algorithm described in Sec. 9.5 to solve this maximum flow problem.

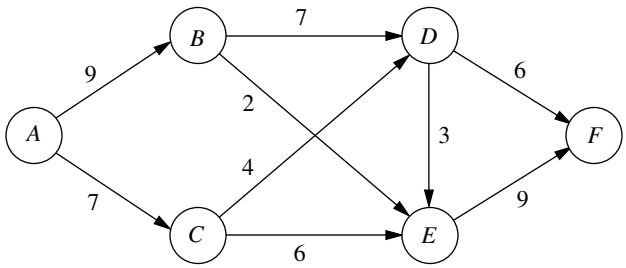
C **(e)** Formulate and solve a spreadsheet model for this problem.

**9.5-5.** One track of the Eura Railroad system runs from the major industrial city of Faireparc to the major port city of Portstown. This track is heavily used by both express passenger and freight trains. The passenger trains are carefully scheduled and have priority over the slow freight trains (this is a European railroad), so that the freight trains must pull over onto a siding whenever a passenger train is scheduled to pass them soon. It is now necessary to increase the freight service, so the problem is to schedule the freight trains so as to maximize the number that can be sent each day without interfering with the fixed schedule for passenger trains.

Consecutive freight trains must maintain a schedule differential of at least 0.1 hour, and this is the time unit used for scheduling them (so that the daily schedule indicates the status of each freight train at times 0.0, 0.1, 0.2, . . . , 23.9). There are $S$ sidings between Faireparc and Portstown, where siding $i$ is long enough to hold $n_i$ freight trains ($i = 1, \ldots, S$). It requires $t_i$ time units (rounded up to an integer) for a freight train to travel from siding $i$ to siding $i + 1$ (where $t_0$ is the time from the Faireparc station to siding 1 and $t_s$ is the time from siding $S$ to the Portstown station). A freight train is allowed to pass or leave siding $i$ ($i = 0, 1, \ldots, S$) at time $j$ ($j = 0.0, 0.1, \ldots, 23.9$) only if it would not be overtaken by a scheduled passenger train before reaching siding $i + 1$ (let $\delta_{ij} = 1$ if it would not be overtaken, and let $\delta_{ij} = 0$ if it would be). A freight train also is required to stop at a siding if there will not be room for it at all subsequent sidings that it would reach before being overtaken by a passenger train.

Formulate this problem as a maximum flow problem by identifying each node (including the supply node and the demand node) as well as each arc and its arc capacity for the network representation of the problem. (*Hint:* Use a different set of nodes for each of the 240 times.)

**9.5-6.** Consider the maximum flow problem shown next, where the source is node $A$, the sink is node $F$, and the arc capacities are the numbers shown next to these directed arcs.




**(a)** Use the augmenting path algorithm described in Sec. 9.5 to solve this problem.

C **(b)** Formulate and solve a spreadsheet model for this problem.

**9.6-1.** Reconsider the maximum flow problem shown in Prob. 9.5-6. Formulate this problem as a minimum cost flow problem, including adding the arc $A \rightarrow F$. Use $\overline{F} = 20$.

**9.6-2.** A company will be producing the same new product at two different factories, and then the product must be shipped to two warehouses. Factory 1 can send an unlimited amount by rail to warehouse 1 only, whereas factory 2 can send an unlimited amount by rail to warehouse 2 only. However, independent truckers can be used to ship up to 50 units from each factory to a distribution center, from which up to 50 units can be shipped to each warehouse. The shipping cost per unit for each alternative is shown in the following table, along with the amounts to be produced at the factories and the amounts needed at the warehouses.

| From \ To | Unit Shipping Cost: Distribution Center | Unit Shipping Cost: Warehouse 1 | Unit Shipping Cost: Warehouse 2 | Output |
|---|---|---|---|---|
| Factory 1 | 3 | 7 | — | 80 |
| Factory 2 | 4 | — | 9 | 70 |
| Distribution center | | 2 | 4 | |
| Allocation | | 60 | 90 | |

**(a)** Formulate the network representation of this problem as a minimum cost flow problem.

**(b)** Formulate the linear programming model for this problem.

**9.6-3.** Reconsider Prob. 9.3-1. Now formulate this problem as a minimum cost flow problem by showing the appropriate network representation.

**9.6-4.** The Makonsel Company is a fully integrated company that both produces goods and sells them at its retail outlets. After production, the goods are stored in the company's two warehouses until needed by the retail outlets. Trucks are used to transport the goods from the two plants to the warehouses, and then from the warehouses to the three retail outlets.

Using units of full truckloads, the following table shows each plant's monthly output, its shipping cost per truckload sent to each warehouse, and the maximum amount that it can ship per month to each warehouse.

| From \ To | Unit Shipping Cost: Warehouse 1 | Unit Shipping Cost: Warehouse 2 | Shipping Capacity: Warehouse 1 | Shipping Capacity: Warehouse 2 | Output |
|---|---|---|---|---|---|
| Plant 1 | $425 | $560 | 125 | 150 | 200 |
| Plant 2 | $510 | $600 | 175 | 200 | 300 |

For each retail outlet (RO), the next table shows its monthly demand, its shipping cost per truckload from each warehouse, and the maximum amount that can be shipped per month from each warehouse.

| From \ To | Unit Shipping Cost: RO1 | Unit Shipping Cost: RO2 | Unit Shipping Cost: RO3 | Shipping Capacity: RO1 | Shipping Capacity: RO2 | Shipping Capacity: RO3 |
|---|---|---|---|---|---|---|
| Warehouse 1 | $470 | $505 | $490 | 100 | 150 | 100 |
| Warehouse 2 | $390 | $410 | $440 | 125 | 150 | 75 |
| Demand | 150 | 200 | 150 | 150 | 200 | 150 |

Management now wants to determine a distribution plan (number of truckloads shipped per month from each plant to each warehouse and from each warehouse to each retail outlet) that will minimize the total shipping cost.

**(a)** Draw a network that depicts the company's distribution network. Identify the supply nodes, transshipment nodes, and demand nodes in this network.

**(b)** Formulate this problem as a minimum cost flow problem by inserting all the necessary data into this network.

C **(c)** Formulate and solve a spreadsheet model for this problem.

C **(d)** Use the computer to solve this problem without using Excel.

**9.6-5.** The Audiofile Company produces boomboxes. However, management has decided to subcontract out the production of the speakers needed for the boomboxes. Three vendors are available to supply the speakers. Their price for each shipment of 1,000 speakers is shown on the next page.

| Vendor | Price |
|---|---|
| 1 | $22,500 |
| 2 | $22,700 |
| 3 | $22,300 |

In addition, each vendor would charge a shipping cost. Each shipment would go to one of the company's two warehouses. Each vendor has its own formula for calculating this shipping cost based on the mileage to the warehouse. These formulas and the mileage data are shown below.

| Vendor | Charge per Shipment |
|---|---|
| 1 | $300 + 40¢/mile |
| 2 | $200 + 50¢/mile |
| 3 | $500 + 20¢/mile |

| Vendor | Warehouse 1 | Warehouse 2 |
|---|---|---|
| 1 | 1,600 miles | 400 miles |
| 2 | 500 miles | 600 miles |
| 3 | 2,000 miles | 1,000 miles |

Whenever one of the company's two factories needs a shipment of speakers to assemble into the boomboxes, the company hires a trucker to bring the shipment in from one of the warehouses. The cost per shipment is given in the next column, along with the number of shipments needed per month at each factory.

| | Unit Shipping Cost | |
|---|---|---|
| | Factory 1 | Factory 2 |
| Warehouse 1 | $200 | $700 |
| Warehouse 2 | $400 | $500 |
| Monthly demand | 10 | 6 |

Each vendor is able to supply as many as 10 shipments per month. However, because of shipping limitations, each vendor is able to send a maximum of only 6 shipments per month to each warehouse. Similarly, each warehouse is able to send a maximum of only 6 shipments per month to each factory.

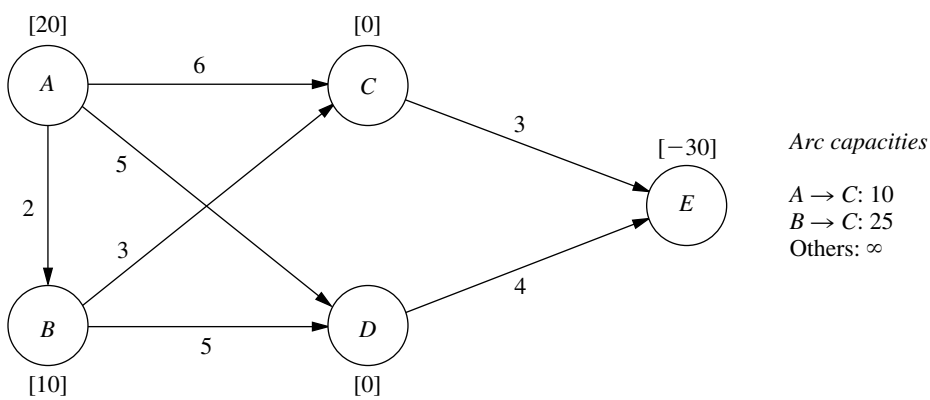
Management now wants to develop a plan for each month regarding how many shipments (if any) to order from each vendor, how many of those shipments should go to each warehouse, and then how many shipments each warehouse should send to each factory. The objective is to minimize the sum of the purchase costs (including the shipping charge) and the shipping costs from the warehouses to the factories.

**(a)** Draw a network that depicts the company's supply network. Identify the supply nodes, transshipment nodes, and demand nodes in this network.

**(b)** Formulate this problem as a minimum cost flow problem by inserting all the necessary data into this network. Also include a dummy demand node that receives (at zero cost) all the unused supply capacity at the vendors.

C **(c)** Formulate and solve a spreadsheet model for this problem.

C **(d)** Use the computer to solve this problem without using Excel.




D **9.7-1.** Consider the minimum cost flow problem shown above, where the $b_i$ values (net flows generated) are given by the nodes, the $c_{ij}$ values (costs per unit flow) are given by the arcs, and the $u_{ij}$ values (arc capacities) are given to the right of the network. Do the following work manually.

**(a)** Obtain an initial BF solution by solving the feasible spanning tree with basic arcs $A \rightarrow B$, $C \rightarrow E$, $D \rightarrow E$, and $C \rightarrow A$ (a reverse arc), where one of the nonbasic arcs ($C \rightarrow B$) also is a reverse arc. Show the resulting network (including $b_i$, $c_{ij}$, and $u_{ij}$) in the same format as the above one (except use dashed

lines to draw the nonbasic arcs), and add the flows in parentheses next to the basic arcs.

**(b)** Use the optimality test to verify that this initial BF solution is optimal and that there are multiple optimal solutions. Apply one iteration of the network simplex method to find the other optimal BF solution, and then use these results to identify the other optimal solutions that are not BF solutions.

**(c)** Now consider the following BF solution.

| Basic Arc | Flow | Nonbasic Arc |
|---|---|---|
| $A \rightarrow D$ | 20 | $A \rightarrow B$ |
| $B \rightarrow C$ | 10 | $A \rightarrow C$ |
| $C \rightarrow E$ | 10 | $B \rightarrow D$ |
| $D \rightarrow E$ | 20 | |

Starting from this BF solution, apply *one* iteration of the network simplex method. Identify the entering basic arc, the leaving basic arc, and the next BF solution, but do not proceed further.

**9.7-2.** Reconsider the minimum cost flow problem formulated in Prob. 9.6-1.

**(a)** Obtain an initial BF solution by solving the feasible spanning tree with basic arcs $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow F$, $B \rightarrow D$, and $E \rightarrow F$, where two of the nonbasic arcs ($E \rightarrow C$ and $F \rightarrow D$) are *reverse* arcs.

D,I **(b)** Use the network simplex method yourself (without an automatic computer routine) to solve this problem.

**9.7-3.** Reconsider the minimum cost flow problem formulated in Prob. 9.6-2.

**(a)** Obtain an initial BF solution by solving the feasible spanning tree that corresponds to using just the two rail lines plus factory 1 shipping to warehouse 2 via the distribution center.

D,I **(b)** Use the network simplex method yourself (without an automatic computer routine) to solve this problem.

D,I **9.7-4.** Reconsider the minimum cost flow problem formulated in Prob. 9.6-3. Starting with the initial BF solution that corresponds to replacing the tractor every year, use the network simplex method yourself (without an automatic computer routine) to solve this problem.

D,I **9.7-5.** For the P & T Co. transportation problem given in Table 8.2, consider its network representation as a minimum cost flow problem presented in Fig. 8.2. Use the northwest corner rule to obtain an initial BF solution from Table 8.2. Then use the network simplex method yourself (without an automatic computer routine) to solve this problem (and verify the optimal solution given in Sec. 8.1).

**9.7-6.** Consider the Metro Water District transportation problem presented in Table 8.12.

**(a)** Formulate the network representation of this problem as a minimum cost flow problem. (*Hint:* Arcs where flow is prohibited should be deleted.)

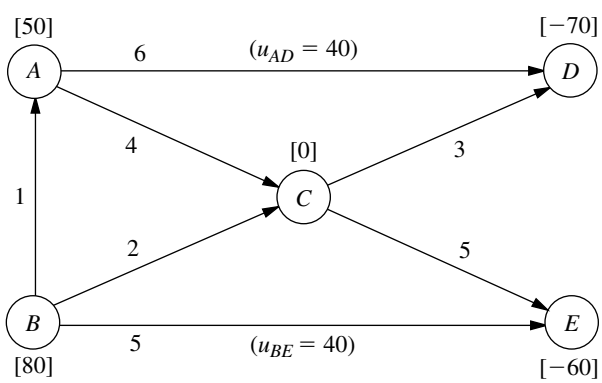
D,I **(b)** Starting with the initial BF solution given in Table 8.19, use the network simplex method yourself (without an automatic computer routine) to solve this problem. Compare the sequence of BF solutions obtained with the sequence obtained by the transportation simplex method in Table 8.23.

D,I **9.7-7.** Consider the transportation problem having the following parameter table:

| | | Destination | | | |
|---|---|---|---|---|---|
| | | 1 | 2 | 3 | Supply |
| Source | 1 | 6 | 7 | 4 | 40 |
| | 2 | 5 | 8 | 6 | 60 |
| Demand | | 30 | 40 | 30 | |

Formulate the network representation of this problem as a minimum cost flow problem. Use the northwest corner rule to obtain an initial BF solution. Then use the network simplex method yourself (without an automatic computer routine) to solve the problem.

D,I **9.7-8.** Consider the minimum cost flow problem shown below, where the $b_i$ values are given by the nodes, the $c_{ij}$ values are given by the arcs, and the *finite* $u_{ij}$ values are given in parentheses by the arcs. Obtain an initial BF solution by solving the feasible spanning tree with basic arcs $A \rightarrow C$, $B \rightarrow A$, $C \rightarrow D$, and $C \rightarrow E$, where one of the nonbasic arcs ($D \rightarrow A$) is a *reverse* arc. Then use the network simplex method yourself (without an automatic computer routine) to solve this problem.



## CASE 9.1 AIDING ALLIES

Commander Votachev steps into the cold October night and deeply inhales the smoke from his cigarette, savoring its warmth. He surveys the destruction surrounding him—shattered windows, burning buildings, torn roads—and smiles. His two years of work training revolutionaries east of the Ural Mountains has proved successful; his troops now occupy seven strategically important cities in the Russian Federation: Kazan, Perm, Yekaterinburg, Ufa, Samara, Saratov, and Orenburg. His siege is not yet over, however. He looks to the west. Given the political and economic confusion in the Russian Federation at this time, he knows that his troops will be able to conquer Saint Petersburg and Moscow shortly. Commander Votachev will then be able to rule with the wisdom and control exhibited by his communist predecessors Lenin and Stalin.

Across the Pacific Ocean, a meeting of the top security and foreign policy advisers of the United States is in progress at the White House. The President has recently been briefed about the communist revolution masterminded by Commander Votachev and is determining a plan of action. The President reflects upon a similar October long ago in 1917, and he fears the possibility of a new age of radical Communist rule accompanied by chaos, bloodshed, escalating tensions, and possibly nuclear war. He therefore decides that the United States needs to respond and to respond quickly. Moscow has requested assistance from the United States military, and the President plans to send troops and supplies immediately.

The President turns to General Lankletter and asks him to describe the preparations being taken in the United States to send the necessary troops and supplies to the Russian Federation.

General Lankletter informs the President that along with troops, weapons, ammunition, fuel, and supplies, aircraft, ships, and vehicles are being assembled at two port cities with airfields: Boston and Jacksonville. The aircraft and ships will transfer all troops and cargo across the Atlantic Ocean to the Eurasian continent. The general hands the President a list of the types of aircraft, ships, and vehicles being assembled along with a description of each type. The list is shown below.

| Transportation Type | Name | Capacity | Speed |
|---|---|---|---|
| Aircraft | C-141 Starlifter | 150 tons | 400 miles per hour |
| Ship | Transport | 240 tons | 35 miles per hour |
| Vehicle | Palletized Load System Truck | 16,000 kilograms | 60 miles per hour |

All aircraft, ships, and vehicles are able to carry both troops and cargo. Once an aircraft or ship arrives in Europe, it stays there to support the armed forces.

The President then turns to Tabitha Neal, who has been negotiating with the NATO countries for the last several hours to use their ports and airfields as stops to refuel and resupply before heading to the Russian Federation. She informs the President that the following ports and airfields in the NATO countries will be made available to the United States military.

| Ports | Airfields |
|---|---|
| Napoli | London |
| Hamburg | Berlin |
| Rotterdam | Istanbul |

The President stands and walks to the map of the world projected on a large screen in the middle of the room. He maps the progress of troops and cargo from the United States to three strategic cities in the Russian Federation that have not yet been seized by Commander Votachev. The three cities are Saint Petersburg, Moscow, and Rostov. He explains that the troops and cargo will be used both to defend the Russian cities and to launch a counterattack against Votachev to recapture the cities he currently occupies. (The map is shown at the end of the case.)

The President also explains that all Starlifters and transports leave Boston or Jacksonville. All transports that have traveled across the Atlantic must dock at one of the NATO ports to unload. Palletized load system trucks brought over in the transports will then carry all troops and materials unloaded from the ships at the NATO ports to the three strategic Russian cities not yet seized by Votachev. All Starlifters that have traveled across the Atlantic must land at one of the NATO airfields for refueling. The planes will then carry all troops and cargo from the NATO airfields to the three Russian cities.

(a) Draw a network showing the different routes troops and supplies may take to reach the Russian Federation from the United States.

(b) Moscow and Washington do not know when Commander Votachev will launch his next attack. Leaders from the two countries have therefore agreed that troops should reach each of the three strategic Russian cities as quickly as possible. The President has determined that the situation is so dire that cost is no object—as many Starlifters, transports, and trucks as are necessary will be used to transfer troops and cargo from the United States to Saint Petersburg, Moscow, and Rostov. Therefore, no limitations exist on the number of troops and amount of cargo that can be transferred between any cities.

The President has been given the following information about the length of the available routes between cities:

| From | To | Length of route in kilometers |
|---|---|---|
| Boston | Berlin | 7,250 km |
| Boston | Hamburg | 8,250 km |
| Boston | Istanbul | 8,300 km |
| Boston | London | 6,200 km |
| Boston | Rotterdam | 6,900 km |
| Boston | Napoli | 7,950 km |
| Jacksonville | Berlin | 9,200 km |
| Jacksonville | Hamburg | 9,800 km |
| Jacksonville | Istanbul | 10,100 km |
| Jacksonville | London | 7,900 km |
| Jacksonville | Rotterdam | 8,900 km |
| Jacksonville | Napoli | 9,400 km |

| From | To | Length of route in kilometers |
|---|---|---|
| Berlin | Saint Petersburg | 1,280 km |
| Hamburg | Saint Petersburg | 1,880 km |
| Istanbul | Saint Petersburg | 2,040 km |
| London | Saint Petersburg | 1,980 km |
| Rotterdam | Saint Petersburg | 2,200 km |
| Napoli | Saint Petersburg | 2,970 km |
| Berlin | Moscow | 1,600 km |
| Hamburg | Moscow | 2,120 km |
| Istanbul | Moscow | 1,700 km |
| London | Moscow | 2,300 km |
| Rotterdam | Moscow | 2,450 km |
| Napoli | Moscow | 2,890 km |
| Berlin | Rostov | 1,730 km |
| Hamburg | Rostov | 2,470 km |
| Istanbul | Rostov | 990 km |
| London | Rostov | 2,860 km |
| Rotterdam | Rostov | 2,760 km |
| Napoli | Rostov | 2,800 km |

Given the distance and the speed of the transportation used between each pair of cities, how can the President most quickly move troops from the United States to each of the three strategic Russian cities? Highlight the path(s) on the network. How long will it take troops and supplies to reach Saint Petersburg? Moscow? Rostov?

(c) The President encounters only one problem with his first plan: he has to sell the military deployment to Congress. Under the War Powers Act, the President is required to consult with Congress before introducing troops into hostilities or situations where hostilities will occur. If Congress does not give authorization to the President for such use of troops, the President must withdraw troops after 60 days. Congress also has the power to decrease the 60-day time period by passing a concurrent resolution.

The President knows that Congress will not authorize significant spending for another country's war, especially when voters have paid so much attention to decreasing the national debt. He therefore decides that he needs to find a way to get the needed troops and supplies to Saint Petersburg, Moscow, and Rostov at the minimum cost.

Each Russian city has contacted Washington to communicate the number of troops and supplies the city needs at a minimum for reinforcement. After analyzing the requests, General Lankletter has converted the requests from numbers of troops, gallons of gasoline, etc., to tons of cargo for easier planning. The requirements are listed below.

| City | Requirements |
|---|---|
| Saint Petersburg | 320,000 tons |
| Moscow | 440,000 tons |
| Rostov | 240,000 tons |

Both in Boston and Jacksonville there are 500,000 tons of the necessary cargo available. When the United States decides to send a plane, ship, or truck between two cities, several costs occur—fuel costs, labor costs, maintenance costs, and appropriate port or airfield taxes and tariffs. These costs are listed below.

| From | To | Cost |
|---|---|---|
| Boston | Berlin | $50,000 per Starlifter |
| Boston | Hamburg | $30,000 per transport |
| Boston | Istanbul | $55,000 per Starlifter |
| Boston | London | $45,000 per Starlifter |
| Boston | Rotterdam | $30,000 per transport |
| Boston | Napoli | $32,000 per transport |
| Jacksonville | Berlin | $57,000 per Starlifter |
| Jacksonville | Hamburg | $48,000 per transport |
| Jacksonville | Istanbul | $61,000 per Starlifter |
| Jacksonville | London | $49,000 per Starlifter |
| Jacksonville | Rotterdam | $44,000 per transport |
| Jacksonville | Napoli | $56,000 per transport |
| Berlin | Saint Petersburg | $24,000 per Starlifter |
| Hamburg | Saint Petersburg | $ 3,000 per truck |
| Istanbul | Saint Petersburg | $28,000 per Starlifter |
| London | Saint Petersburg | $22,000 per Starlifter |
| Rotterdam | Saint Petersburg | $ 3,000 per truck |
| Napoli | Saint Petersburg | $ 5,000 per truck |
| Berlin | Moscow | $22,000 per Starlifter |
| Hamburg | Moscow | $ 4,000 per truck |
| Istanbul | Moscow | $25,000 per Starlifter |
| London | Moscow | $19,000 per Starlifter |
| Rotterdam | Moscow | $ 5,000 per truck |
| Napoli | Moscow | $ 5,000 per truck |
| Berlin | Rostov | $23,000 per Starlifter |
| Hamburg | Rostov | $ 7,000 per truck |
| Istanbul | Rostov | $ 2,000 per Starlifter |
| London | Rostov | $ 4,000 per Starlifter |
| Rotterdam | Rostov | $ 8,000 per truck |
| Napoli | Rostov | $ 9,000 per truck |

The President faces a number of restrictions when trying to satisfy the requirements. Early winter weather in northern Russia has brought a deep freeze with much snow. Therefore, General Lankletter is opposed to sending truck convoys in the area. He convinces the President to supply Saint Petersburg only through the air. Moreover, the truck routes into Rostov are quite limited, so that from each port at most 2,500 trucks can be sent to Rostov. The Ukrainian government is very sensitive about American airplanes flying through their air space. It restricts the U.S. military to at most 200 flights from Berlin to Rostov and to at most 200 flights from London to Rostov. (The U.S. military does not want to fly around the Ukraine and is thus restricted by the Ukrainian limitations.)

How does the President satisfy each Russian city's military requirements at minimum cost? Highlight the path to be used between the United States and Russian Federation on the network.

(d) Once the President releases the number of planes, ships, and trucks that will travel between the United States and the Russian Federation, Tabitha Neal contacts each of the American cities and NATO countries to indicate the number of planes to expect at the airfields, the number of ships to expect at the docks, and the number of trucks to expect traveling across the roads. Unfortunately, Tabitha learns that several additional restrictions exist which cannot be immediately eliminated. Because of airfield congestion and unalterable flight schedules, only a limited number of planes may be sent between any two cities. These plane limitations are given below.

| From | To | Maximum |
|---|---|---|
| Boston | Berlin | 300 airplanes |
| Boston | Istanbul | 500 airplanes |
| Boston | London | 500 airplanes |
| Jacksonville | Berlin | 500 airplanes |
| Jacksonville | Istanbul | 700 airplanes |
| Jacksonville | London | 600 airplanes |
| Berlin | Saint Petersburg | 500 airplanes |
| Istanbul | Saint Petersburg | 0 airplanes |
| London | Saint Petersburg | 1,000 airplanes |
| Berlin | Moscow | 300 airplanes |
| Istanbul | Moscow | 100 airplanes |
| London | Moscow | 200 airplanes |
| Berlin | Rostov | 0 airplanes |
| Istanbul | Rostov | 900 airplanes |
| London | Rostov | 100 airplanes |

In addition, because some countries fear that citizens will become alarmed if too many military trucks travel the public highways, they object to a large number of trucks traveling through their countries. These objections mean that a limited number of trucks are able to travel between certain ports and Russian cities. These limitations are listed below.

| From | To | Maximum |
|---|---|---|
| Rotterdam | Moscow | 600 trucks |
| Rotterdam | Rostov | 750 trucks |
| Hamburg | Moscow | 700 trucks |
| Hamburg | Rostov | 500 trucks |
| Napoli | Moscow | 1,500 trucks |
| Napoli | Rostov | 1,400 trucks |

Tabitha learns that all shipping lanes have no capacity limits, owing to the American control of the Atlantic Ocean.

The President realizes that because of all the restrictions he will not be able to satisfy all the reinforcement requirements of the three Russian cities. He decides to disregard the

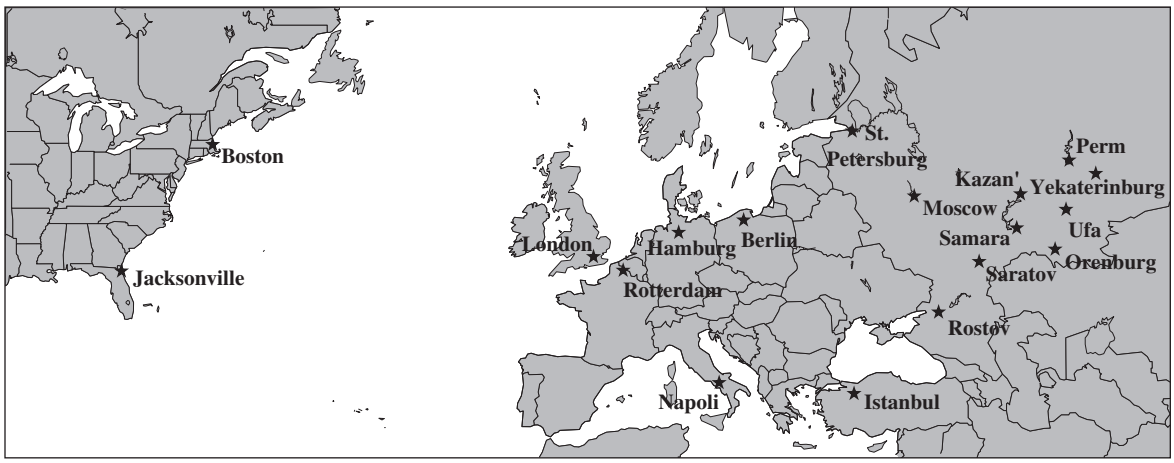
cost issue and instead to maximize the total amount of cargo he can get to the Russian cities. How does the President maximize the total amount of cargo that reaches the Russian Federation? Highlight the path(s) used between the United States and the Russian Federation on the network.

(e) Even before all American troops and supplies had reached Saint Petersburg, Moscow, and Rostov, infighting among Commander Votachev's troops about whether to make the next attack against Saint Petersburg or against Moscow split the revolutionaries. Troops from Moscow easily overcame the vulnerable revolutionaries. Commander Votachev was imprisoned, and the next step became rebuilding the seven cities razed by his armies.

The President's top priority is to help the Russian government to reestablish communications between the seven Russian cities and Moscow at minimum cost. The price of installing communication lines between any two Russian cities varies given the cost of shipping wire to the area, the level of destruction in the area, and the roughness of the terrain. Luckily, a city is able to communicate with all others if it is connected only indirectly to every other city. Saint Petersburg and Rostov are already connected to Moscow, so if any of the seven cities is connected to Saint Petersburg or Rostov, it will also be connected to Moscow. The cost of replacing communication lines between two given cities for which this is possible is shown below.

| Between | Cost to Reestablish Communication Lines |
|---|---|
| Saint Petersburg and Kazan | $210,000 |
| Saint Petersburg and Perm | $185,000 |
| Saint Petersburg and Ufa | $225,000 |
| Moscow and Ufa | $310,000 |
| Moscow and Samara | $195,000 |
| Moscow and Orenburg | $440,000 |
| Moscow and Saratov | $140,000 |
| Rostov and Saratov | $200,000 |
| Rostov and Orenburg | $120,000 |
| Kazan and Perm | $150,000 |
| Kazan and Ufa | $105,000 |
| Kazan and Samara | $ 95,000 |
| Perm and Yekaterinburg | $ 85,000 |
| Perm and Ufa | $125,000 |
| Yekaterinburg and Ufa | $125,000 |
| Ufa and Samara | $100,000 |
| Ufa and Orenburg | $ 75,000 |
| Saratov and Samara | $100,000 |
| Saratov and Orenburg | $ 95,000 |

Where should communication lines be installed to minimize the total cost of reestablishing communications between Moscow and all seven Russian cities?




## CASE 9.2 MONEY IN MOTION

Jake Nguyen runs a nervous hand through his once finely combed hair. He loosens his once perfectly knotted silk tie. And he rubs his sweaty hands across his once immaculately pressed trousers.

Today has certainly not been a good day.

Over the past few months, Jake had heard whispers circulating from Wall Street—whispers from the lips of investment bankers and stockbrokers famous for their outspokenness. They had whispered about a coming Japanese economic collapse—whispered because they had believed that publicly vocalizing their fears would hasten the collapse.

And today, their very fears have come true. Jake and his colleagues gather round a small television dedicated exclusively to the Bloomberg channel. Jake stares in disbelief as he listens to the horrors taking place in the Japanese market. And the Japanese market is taking the financial markets in all other East Asian countries with it on its tailspin. He goes numb. As manager of Asian foreign investment for Grant Hill Associates, a small West Coast investment boutique specializing in currency trading, Jake bears personal responsibility for any negative impacts of the collapse.

And Grant Hill Associates will experience negative impacts.

Jake had not heeded the whispered warnings of a Japanese collapse. Instead, he had greatly increased the stake Grant Hill Associates held in the Japanese market. Because the Japanese market had performed better than expected over the past year, Jake had increased investments in Japan from 2.5 million to 15 million dollars only 1 month ago. At that time, 1 dollar was worth 80 yen.

No longer. Jake realizes that today's devaluation of the yen means that 1 dollar is worth 125 yen. He will be able to liquidate these investments without any loss in yen,

but now the dollar loss when converting back into U.S. currency would be huge. He takes a deep breath, closes his eyes, and mentally prepares himself for serious damage control.

Jake's meditation is interrupted by a booming voice calling for him from a large corner office. Grant Hill, the president of Grant Hill Associates, yells, "Nguyen, get the hell in here!"

Jake jumps and looks reluctantly toward the corner office hiding the furious Grant Hill. He smooths his hair, tightens his tie, and walks briskly into the office.

Grant Hill meets Jake's eyes upon his entrance and continues yelling, "I don't want one word out of you, Nguyen! No excuses; just fix this debacle! Get all of our money out of Japan! My gut tells me this is only the beginning! Get the money into safe U.S. bonds! NOW! And don't forget to get our cash positions out of Indonesia and Malaysia ASAP with it!"

Jake has enough common sense to say nothing. He nods his head, turns on his heel, and practically runs out of the office.

Safely back at his desk, Jake begins formulating a plan to move the investments out of Japan, Indonesia, and Malaysia. His experiences investing in foreign markets have taught him that when playing with millions of dollars, *how* he gets money out of a foreign market is almost as important as *when* he gets money out of the market. The banking partners of Grant Hill Associates charge different transaction fees for converting one currency into another one and wiring large sums of money around the globe.

And now, to make matters worse, the governments in East Asia have imposed very tight limits on the amount of money an individual or a company can exchange from the domestic currency into a particular foreign currency and withdraw it from the country. The goal of this dramatic measure is to reduce the outflow of foreign investments out of those countries to prevent a complete collapse of the economies in the region. Because of Grant Hill Associates' cash holdings of 10.5 billion Indonesian rupiahs and 28 million Malaysian ringgits, along with the holdings in yen, it is not clear how these holdings should be converted back into dollars.

Jake wants to find the most cost-effective method to convert these holdings into dollars. On his company's website he always can find on-the-minute exchange rates for most currencies in the world (Table 1).

The table states that, for example, 1 Japanese yen equals 0.008 U.S. dollars. By making a few phone calls he discovers the transaction costs his company must pay for large currency transactions during these critical times (Table 2).

Jake notes that exchanging one currency for another one results in the same transaction cost as a reverse conversion. Finally, Jake finds out the maximum amounts of domestic currencies his company is allowed to convert into other currencies in Japan, Indonesia, and Malaysia (Table 3).

(a) Formulate Jake's problem as a minimum cost flow problem, and draw the network for his problem. Identify the supply and demand nodes for the network.

(b) Which currency transactions must Jake perform in order to convert the investments from yen, rupiah, and ringgit into U.S. dollars to ensure that Grant Hill Associates has the maximum dollar amount after all transactions have occurred? How much money does Jake have to invest in U.S. bonds?

**TABLE 1** Currency exchange rates

| From \ To | Yen | Rupiah | Ringgit | U.S. Dollar | Canadian Dollar | Euro | Pound | Peso |
|---|---|---|---|---|---|---|---|---|
| Japanese yen | 1 | 50 | 0.04 | 0.008 | 0.01 | 0.0064 | 0.0048 | 0.0768 |
| Indonesian rupiah | | 1 | 0.0008 | 0.00016 | 0.0002 | 0.000128 | 0.000096 | 0.001536 |
| Malaysian ringgit | | | 1 | 0.2 | 0.25 | 0.16 | 0.12 | 1.92 |
| U.S. dollar | | | | 1 | 1.25 | 0.8 | 0.6 | 9.6 |
| Canadian dollar | | | | | 1 | 0.64 | 0.48 | 7.68 |
| European euro | | | | | | 1 | 0.75 | 12 |
| English pound | | | | | | | 1 | 16 |
| Mexican peso | | | | | | | | 1 |

(c) The World Trade Organization forbids transaction limits because they promote protectionism. If no transaction limits exist, what method should Jake use to convert the Asian holdings from the respective currencies into dollars?

(d) In response to the World Trade Organization's mandate forbidding transaction limits, the Indonesian government introduces a new tax that leads to an increase of transaction costs for transaction of rupiah by 500 percent to protect their currency. Given these new transaction costs but no transaction limits, what currency transactions should Jake perform in order to convert the Asian holdings from the respective currencies into dollars?

**TABLE 2** Transaction cost, percent

| From \ To | Yen | Rupiah | Ringgit | U.S. Dollar | Canadian Dollar | Euro | Pound | Peso |
|---|---|---|---|---|---|---|---|---|
| Yen | — | 0.5 | 0.5 | 0.4 | 0.4 | 0.4 | 0.25 | 0.5 |
| Rupiah | | — | 0.7 | 0.5 | 0.3 | 0.3 | 0.75 | 0.75 |
| Ringgit | | | — | 0.7 | 0.7 | 0.4 | 0.45 | 0.5 |
| U.S. dollar | | | | — | 0.05 | 0.1 | 0.1 | 0.1 |
| Canadian dollar | | | | | — | 0.2 | 0.1 | 0.1 |
| Euro | | | | | | — | 0.05 | 0.5 |
| Pound | | | | | | | — | 0.5 |
| Peso | | | | | | | | — |

**TABLE 3** Transaction limits in equivalent of 1,000 dollars

| From \ To | Yen | Rupiah | Ringgit | U.S. Dollar | Canadian Dollar | Euro | Pound | Peso |
|---|---|---|---|---|---|---|---|---|
| Yen | — | 5,000 | 5,000 | 2,000 | 2,000 | 2,000 | 2,000 | 4,000 |
| Rupiah | 5,000 | — | 2,000 | 200 | 200 | 1,000 | 500 | 200 |
| Ringgit | 3,000 | 4,500 | — | 1,500 | 1,500 | 2,500 | 1,000 | 1,000 |

(e) Jake realizes that his analysis is incomplete because he has not included all aspects that might influence his planned currency exchanges. Describe other factors that Jake should examine before he makes his final decision.