C H A P T E R 6

Network Models

Chapter Guide. The network models in this chapter include the traditional applications of finding the most efficient way to link a number of locations directly or indirectly, finding the shortest route between two cities, determining the maximum flow in a pipeline network, determining the minimum-cost flow in a network that satisfies supply and demand requirements at different locations, and scheduling the activities of a project.

The minimum-cost capacitated algorithm is a generalized network that subsumes the shortest-route and the maximal-flow models presented in this chapter. Its details can be found in Section 20.1 on the CD.

As you study the material in this chapter, you should pay special attention to the nontraditional applications of these models. For example, the shortest-route model can be used to determine the optimal equipment replacement policy and the maximum-flow model can be used to determine the optimum number of ships that meet a specific shipping schedule. These situations are included in the chapter as solved examples, problems, or cases.

Throughout the chapter, the formulation and solution of a network model as a linear program is emphasized. It is recommended that you study these relationships, because most commercial codes solve network problems as mere linear programs. Additionally, some formulations require imposing side constraints, which can be implemented only if the problem is solved as an LP.

To understand the computational details, you are encouraged to use TORA's interactive modules that create the steps of the solution in the exact manner presented in the book. For large-scale problems, the chapter offers both Excel Solver and AMPL models for the different algorithms.

This chapter includes a summary of 1 real-life application, 17 solved examples, 2 Solver models, 3 AMPL models, 69 end-of-section problems, and 5 cases. The cases are in Appendix E on the CD. The AMPL/Excel/Solver/TORA programs are in folder ch6Files.

Real-Life Application—Saving Federal Travel Dollars

U.S. Federal Government employees are required to attend development conferences and training courses in different locations around the country. Because the federal employees are located in offices scattered around the United States, the selection of the host city impacts travel cost. Currently, the selection of the city hosting conferences /training events is done without consideration of incurred travel cost. The problem seeks the determination of the optimal location of the host city. For Fiscal Year 1997, the developed model was estimated to save at least $400,000. Case 4 in Chapter 24 on the CD provides the details of the study.

6.1 SCOPE AND DEFINITION OF NETWORK MODELS

A multitude of operations research situations can be modeled and solved as networks (nodes connected by branches):

1. Design of an offshore natural-gas pipeline network connecting well heads in the Gulf of Mexico to an inshore delivery point. The objective of the model is to minimize the cost of constructing the pipeline.
2. Determination of the shortest route between two cities in an existing network of roads.
3. Determination of the maximum capacity (in tons per year) of a coal slurry pipeline network joining coal mines in Wyoming with power plants in Houston. (Slurry pipelines transport coal by pumping water through specially designed pipes.)
4. Determination of the time schedule (start and completion dates) for the activities of a construction project.
5. Determination of the minimum-cost flow schedule from oil fields to refineries through a pipeline network.

The solution of these situations, and others like it, is accomplished through a variety of network optimization algorithms. This chapter presents four of these algorithms.

1. Minimal spanning tree (situation 1)
2. Shortest-route algorithm (situation 2)
3. Maximal-flow algorithm (situation 3)
4. Critical path (CPM) algorithm (situation 4)

For the fifth situation, the minimum-cost capacitated network algorithm is presented in Section 20.1 on the CD.

Network Definitions. A network consists of a set of **nodes** linked by **arcs** (or **branches**). The notation for describing a network is (N, A), where N is the set of nodes and A is the set of arcs. As an illustration, the network in Figure 6.1 is described as

$$N = \{1, 2, 3, 4, 5\}$$
$$A = \{(1,2), (1,3), (2,3), (2,5), (3,4), (3,5), (4,2), (4,5)\}$$

Associated with each network is a **flow** (e.g., oil products flow in a pipeline and automobile traffic flows in highways). In general, the flow in a network is limited by the capacity of its arcs, which may be finite or infinite.

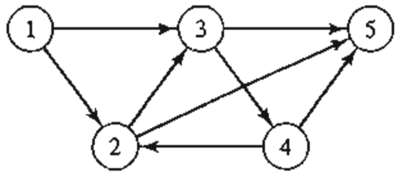


FIGURE 6.1
Example of (N, A) Network

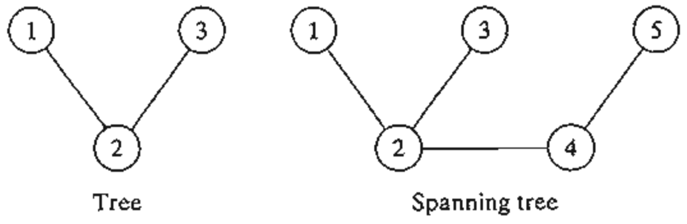


FIGURE 6.2
Examples of a tree and a spanning tree

An arc is said to be **directed** or **oriented** if it allows positive flow in one direction and zero flow in the opposite direction. A **directed network** has all directed arcs.

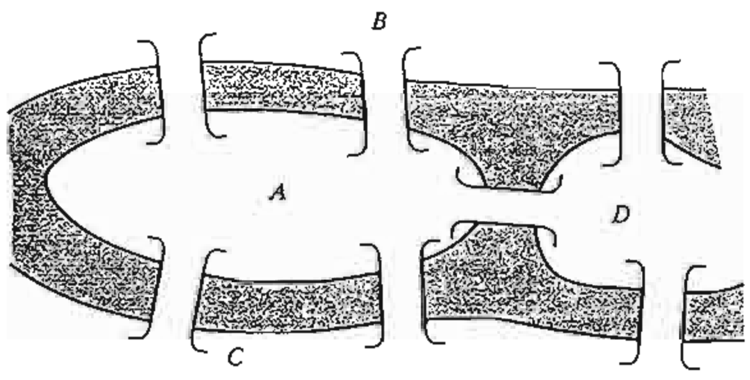
A **path** is a sequence of distinct arcs that join two nodes through other nodes regardless of the direction of flow in each arc. A path forms a **cycle** or a **loop** if it connects a node to itself through other nodes. For example, in Figure 6.1, the arcs $(2,3)$, $(3,4)$, and $(4,2)$ form a cycle.

A **connected network** is such that every two distinct nodes are linked by at least one path. The network in Figure 6.1 demonstrates this type of network. A **tree** is a *cycle-free* connected network comprised of a *subset* of all the nodes, and a **spanning tree** is a tree that links *all* the nodes of the network. Figure 6.2 provides examples of a tree and a spanning tree from the network in Figure 6.1.

Example 6.1-1 (Bridges of Königsberg)

The Prussian city of Königsberg (now Kalingrad in Russia) was founded in 1254 on the banks of river Pergel with seven bridges connecting its four sections (labeled *A, B, C,* and *D*) as shown in Figure 6.3. A problem circulating among the inhabitants of the city was to find out if a *round trip*

FIGURE 6.3
Bridges of Königsberg



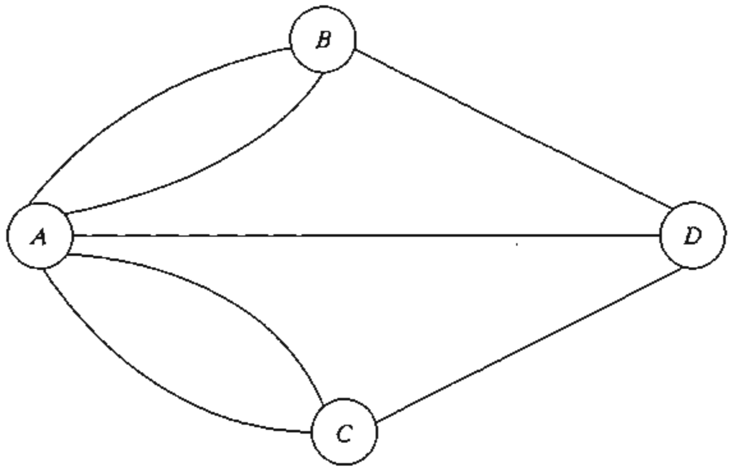


FIGURE 6.4
Network representation of Königsberg problem

of the four sections could be made with each bridge being crossed exactly once. No limits were set on the number of times any of the four sections could be visited.

In the mid-eighteenth century, the famed mathematician Leonhard Euler developed a special "path construction" argument to prove that it was impossible to make such a trip. Later, in the early nineteenth century the same problem was solved by representing the situation as a network in which each of the four sections (*A*, *B*, *C*, and *D*) is a node and each bridge is an arc joining applicable nodes, as shown in Figure 6.4.

The network-based solution is that the desired round trip (starting and ending in one section of the city) is impossible, because there are four nodes and each is associated with an *odd* number of arcs, which does not allow distinct entrance and exit (and hence distinct use of the bridges) to each section of the city.[1] The example demonstrates how the solution of the problem is facilitated by using network representation.

PROBLEM SET 6.1A

*1. For each network in Figure 6.5 determine (a) a path, (b) a cycle, (c) a tree, and (d) a spanning tree.
2. Determine the sets N and A for the networks in Figure 6.5.
3. Draw the network defined by

$$N = \{1, 2, 3, 4, 5, 6\}$$
$$A = \{(1,2), (1,5), (2,3), (2,4), (3,4), (3,5), (4,3), (4,6), (5,2), (5,6)\}$$

*4. Consider eight equal squares arranged in three rows, with two squares in the first row, four in the second, and two in the third. The squares of each row are arranged symmetrically about the vertical axis. It is desired to fill the squares with distinct numbers in the range 1 through 8

[1]General solution: A tour exists if all nodes have an even number of branches or if exactly two nodes have an odd number of branches. Else no tour exists. See B. Hopkins and R. Wilson, "The Truth about Königsberg," *College Math Journal*, Vol. 35, No. 3, pp. 198–207, 2004.

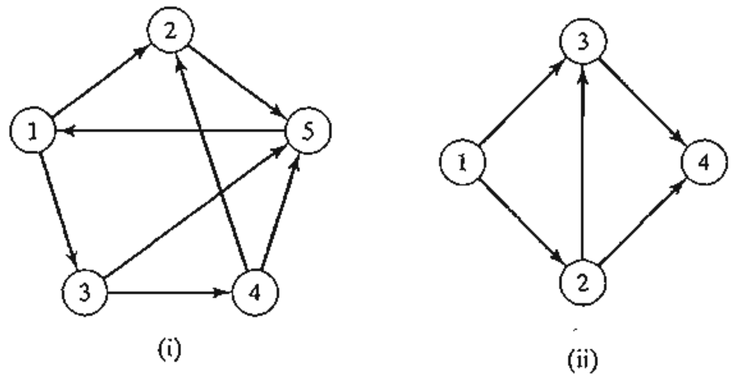


FIGURE 6.5
Networks for Problem 1, Set 6.1a

so that no two *adjacent* vertical, horizontal, or diagonal squares hold consecutive numbers. Use some form of a network representation to find the solution in a systematic way.

5. Three inmates escorted by 3 guards must be transported by boat from the mainland to a penitentiary island to serve their sentences. The boat cannot transfer more than two persons in either direction. The inmates are certain to overpower the guards if they outnumber them at any time. Develop a network model that designs the boat trips in a manner that ensures a smooth transfer of the inmates.

6.2 MINIMAL SPANNING TREE ALGORITHM

The minimal spanning tree algorithm deals with linking the nodes of a network, directly or indirectly, using the shortest total length of connecting branches. A typical application occurs in the construction of paved roads that link several rural towns. The road between two towns may pass through one or more other towns. The most economical design of the road system calls for minimizing the total miles of paved roads, a result that is achieved by implementing the minimal spanning tree algorithm.

The steps of the procedure are given as follows. Let $N = \{1, 2, \ldots, n\}$ be the set of nodes of the network and define

$$C_k = \text{Set of nodes that have been permanently connected at iteration } k$$
$$\overline{C}_k = \text{Set of nodes as yet to be connected permanently after iteration } k$$

Step 0. Set $C_0 = \emptyset$ and $\overline{C}_0 = N$.

Step 1. Start with *any* node i in the unconnected set $\overline{C}_0$ and set $C_1 = \{i\}$, which renders $\overline{C}_1 = N - \{i\}$. Set $k = 2$.

General step *k*. Select a node, j^*, in the unconnected set $\overline{C}_{k-1}$ that yields the shortest arc to a node in the connected set C_{k-1}. Link j^* permanently to C_{k-1} and remove it from $\overline{C}_{k-1}$; that is,

$$C_k = C_{k-1} + \{j^*\}, \overline{C}_k = \overline{C}_{k-1} - \{j^*\}$$

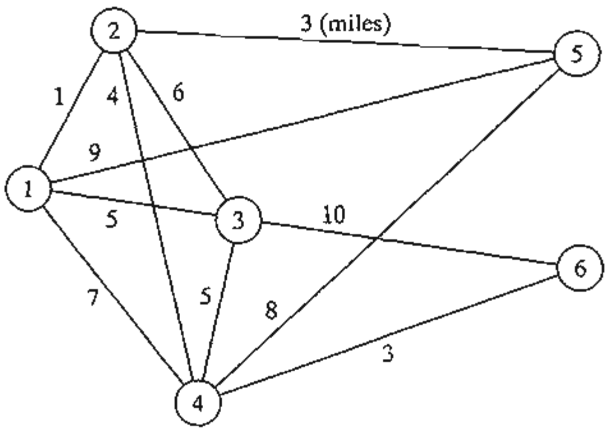


FIGURE 6.6
Cable connections for Midwest TV Company

If the set of unconnected nodes, $\overline{C}_k$, is empty, stop. Otherwise, set $k = k + 1$ and repeat the step.

Example 6.2-1

Midwest TV Cable Company is in the process of providing cable service to five new housing development areas. Figure 6.6 depicts possible TV linkages among the five areas. The cable miles are shown on each arc. Determine the most economical cable network.

The algorithm starts at node 1 (any other node will do as well), which gives

$$C_1 = \{1\}, \overline{C}_1 = \{2, 3, 4, 5, 6\}$$

The iterations of the algorithm are summarized in Figure 6.7. The thin arcs provide all the candidate links between C and $\overline{C}$. The thick branches represent the permanent links between the nodes of the connected set C, and the dashed branch represents the new (permanent) link added at each iteration. For example, in iteration 1, branch $(1, 2)$ is the shortest link ($= 1$ mile) among all the candidate branches from node 1 to nodes 2, 3, 4, 5, and 6 of the unconnected set $\overline{C}_1$. Hence, link $(1, 2)$ is made permanent and $j^* = 2$, which yields

$$C_2 = \{1, 2\}, \overline{C}_2 = \{3, 4, 5, 6\}$$

The solution is given by the minimal spanning tree shown in iteration 6 of Figure 6.7. The resulting minimum cable miles needed to provide the desired cable service are $1 + 3 + 4 + 3 + 5 = 16$ miles.

TORA Moment

You can use TORA to generate the iterations of the minimal spanning tree. From Main menu, select Network models ⇒ Minimal spanning tree. Next, from SOLVE/MODIFY menu, select Solve problem ⇒ Go to output screen. In the output screen, select a Starting node then use Next iteration or All iterations to generate the successive iterations. You can restart the iterations by selecting a new Starting Node. File toraEx6.2-1.txt gives TORA's data for Example 6.2-1.

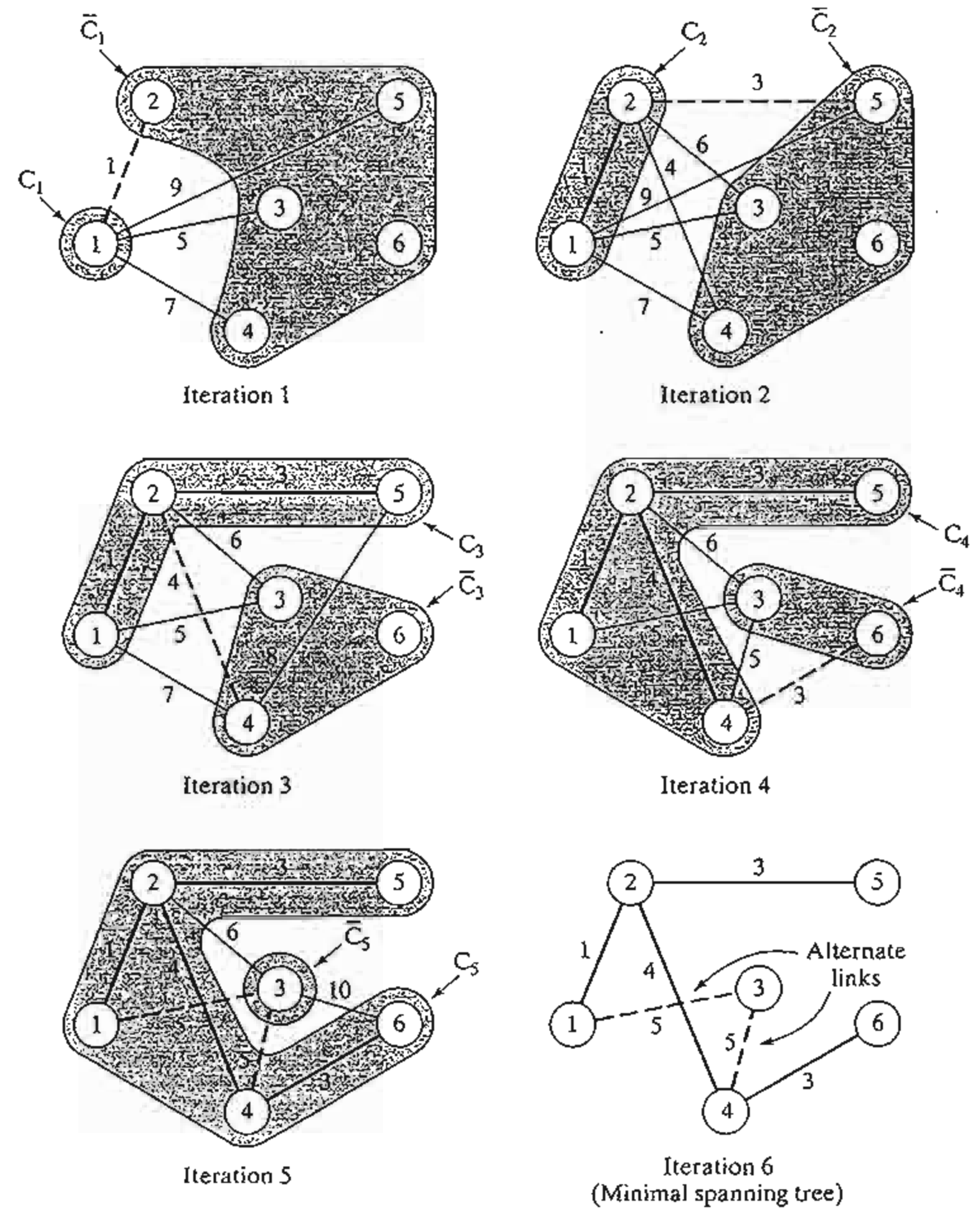


FIGURE 6.7
Solution iterations for Midwest TV Company

PROBLEM SET 6.2A

1. Solve Example 6.2-1 starting at node 5 (instead of node 1), and show that the algorithm produces the same solution.
2. Determine the minimal spanning tree of the network of Example 6.2-1 under each of the following separate conditions:
   *(a) Nodes 5 and 6 are linked by a 2-mile cable.
   (b) Nodes 2 and 5 cannot be linked.
   (c) Nodes 2 and 6 are linked by a 4-mile cable.

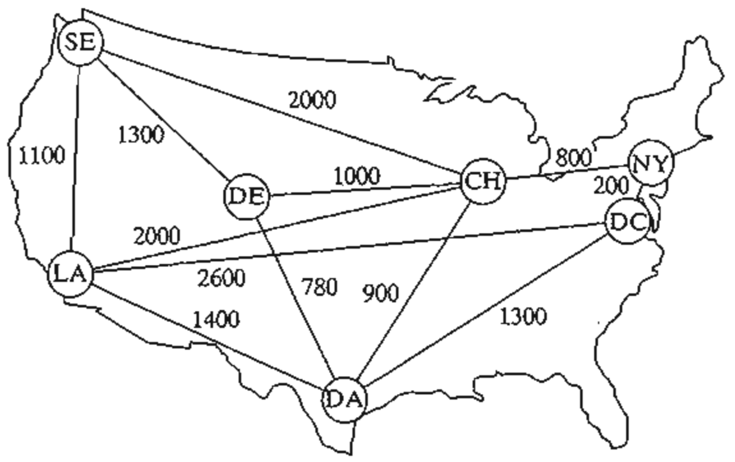


FIGURE 6.8
Network for Problem 3, Set 6.2a

(d) The cable between nodes 1 and 2 is 8 miles long.
(e) Nodes 3 and 5 are linked by a 2-mile cable.
(f) Node 2 cannot be linked directly to nodes 3 and 5.

3. In intermodal transportation, loaded truck trailers are shipped between railroad terminals on special flatbed carts. Figure 6.8 shows the location of the main railroad terminals in the United States and the existing railroad tracks. The objective is to decide which tracks should be "revitalized" to handle the intermodal traffic. In particular, the Los Angeles (LA) terminal must be linked directly to Chicago (CH) to accommodate expected heavy traffic. Other than that, all the remaining terminals can be linked, directly or indirectly, such that the total length (in miles) of the selected tracks is minimized. Determine the segments of the railroad tracks that must be included in the revitalization program.
4. Figure 6.9 gives the mileage of the feasible links connecting nine offshore natural gas wellheads with an inshore delivery point. Because wellhead 1 is the closest to shore, it is equipped with sufficient pumping and storage capacity to pump the output of the remaining eight wells to the delivery point. Determine the minimum pipeline network that links the wellheads to the delivery point.
*5. In Figure 6.9 of Problem 4, suppose that the wellheads can be divided into two groups depending on gas pressure: a high-pressure group that includes wells 2, 3, 4, and 6, and a low-pressure group that includes wells 5, 7, 8, and 9. Because of pressure difference, it is not possible to link the wellheads from the two groups. At the same time, both groups must be connected to the delivery point through wellhead 1. Determine the minimum pipeline network for this situation.
6. Electro produces 15 electronic parts on 10 machines. The company wants to group the machines into cells designed to minimize the "dissimilarities" among the parts processed in each cell. A measure of "dissimilarity," d_{ij}, among the parts processed on machines i and j can be expressed as

$$d_{ij} = 1 - \frac{n_{ij}}{n_{ij} + m_{ij}}$$

where n_{ij} is the number of parts shared between machines i and j, and m_{ij} is the number of parts that are used by either machine i or machine j only.

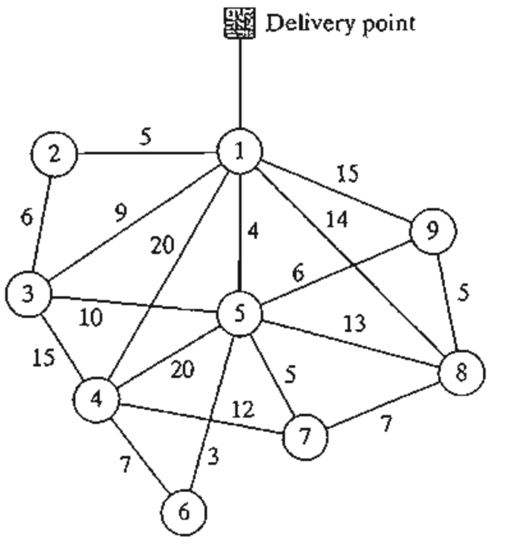


FIGURE 6.9
Network for Problem 4, Set 6.2a

The following table assigns the parts to machines:

Machine	Assigned parts
1	1, 6
2	2, 3, 7, 8, 9, 12, 13, 15
3	3, 5, 10, 14
4	2, 7, 8, 11, 12, 13
5	3, 5, 10, 11, 14
6	1, 4, 5, 9, 10
7	2, 5, 7, 8, 9, 10
8	3, 4, 15
9	4, 10
10	3, 8, 10, 14, 15

(a) Express the problem as a network model.
(b) Show that the determination of the cells can be based on the minimal spanning tree solution.
(c) For the data given in the preceding table, construct the two- and three-cell solutions.

6.3 SHORTEST-ROUTE PROBLEM

The shortest-route problem determines the shortest route between a source and destination in a transportation network. Other situations can be represented by the same model, as illustrated by the following examples.

6.3.1 Examples of the Shortest-Route Applications

Example 6.3-1 (Equipment Replacement)

RentCar is developing a replacement policy for its car fleet for a 4-year planning horizon. At the start of each year, a decision is made as to whether a car should be kept in operation or replaced.

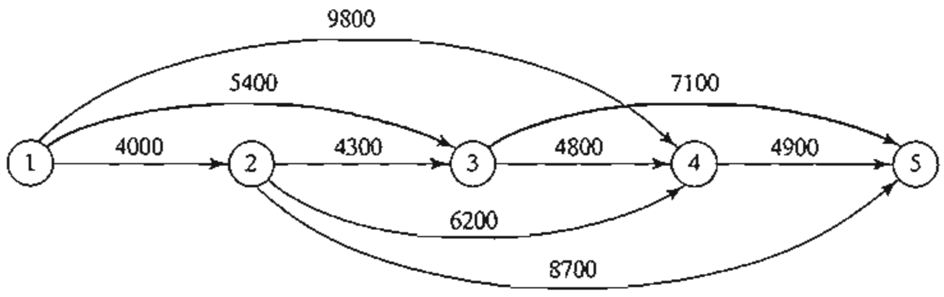


FIGURE 6.10
Equipment replacement problem as a shortest route model

A car must be in service a minimum of 1 year and a maximum of 3 years. The following table provides the replacement cost as a function of the year a car is acquired and the number of years in operation.

Equipment acquired at start of year	Replacement cost ($) for given years in operation *1*	*2*	*3*
1	4000	5400	9800
2	4300	6200	8700
3	4800	7100	—
4	4900	—	—

The problem can be formulated as a network in which nodes 1 to 5 represent the start of years 1 to 5. Arcs from node 1 (year 1) can reach only nodes 2, 3, and 4 because a car must be in operation between 1 and 3 years. The arcs from the other nodes can be interpreted similarly. The length of each arc equals the replacement cost. The solution of the problem is equivalent to finding the shortest route between nodes 1 and 5.

Figure 6.10 shows the resulting network. Using TORA,[2] the shortest route (shown by the thick path) is $1 \rightarrow 3 \rightarrow 5$. The solution means that a car acquired at the start of year 1 (node 1) must be replaced after 2 years at the start of year 3 (node 3). The replacement car will then be kept in service until the end of year 4. The total cost of this replacement policy is $12,500 (= $5400 + $7100).

Example 6.3-2 (Most Reliable Route)

I. Q. Smart drives daily to work. Having just completed a course in network analysis, Smart is able to determine the shortest route to work. Unfortunately, the selected route is heavily patrolled by police, and with all the fines paid for speeding, the shortest route may not be the best choice. Smart has thus decided to choose a route that maximizes the probability of *not* being stopped by police.

The network in Figure 6.11 shows the possible routes between home and work, and the associated probabilities of not being stopped on each segment. The probability of not being

[2]From Main menu, select Network models ⇒ Shortest route. From SOLVE/MODIFY menu, select Solve problem ⇒ Shortest routes.

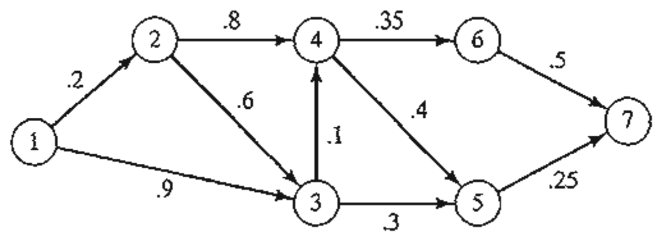


FIGURE 6.11
Most-reliable-route network model

stopped on a route is the product of the probabilities associated with its segments. For example, the probability of not receiving a fine on the route $1 \rightarrow 3 \rightarrow 5 \rightarrow 7$ is $.9 \times .3 \times .25 = .0675$. Smart's objective is to select the route that *maximizes* the probability of not being fined.

The problem can be formulated as a shortest-route model by using a logarithmic transformation that converts the product probability into the sum of the logarithms of probabilities—that is, if $p_{1k} = p_1 \times p_2 \times \cdots \times p_k$ is the probability of not being stopped, then $\log p_{1k} = \log p_1 + \log p_2 + \cdots + \log p_k$.

Mathematically, the maximization of $\log p_{1k}$ is equivalent to the maximization of $\log p_{1k}$. Because $\log p_{1k} \leq 0$, the maximization of $\log p_{1k}$ is equivalent to the minimization of $-\log p_{1k}$. Using this transformation, the individual probabilities p_j in Figure 6.11 are replaced with $-\log p_j$ for all j in the network, thus yielding the shortest-route network in Figure 6.12.

Using TORA, the shortest route in Figure 6.12 is defined by the nodes 1,3,5, and 7 with a corresponding "length" of 1.1707 ($= -\log p_{17}$). Thus, the maximum probability of not being stopped is $p_{17} = .0675$ only, not very encouraging news for Smart!

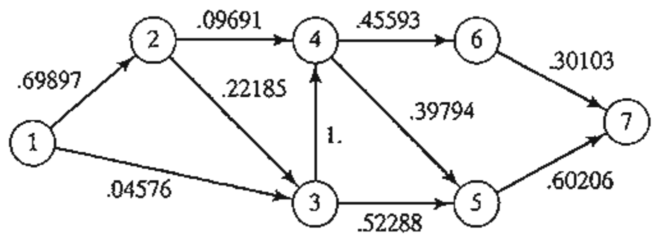
Example 6.3-3 (Three-Jug Puzzle)

An 8-gallon jug is filled with fluid. Given two empty 5- and 3-gallon jugs, we want to divide the 8 gallons of fluid into two equal parts using the three jugs. No other measuring devices are allowed. What is the smallest number of transfers (decantations) needed to achieve this result?

You probably can guess the solution to this puzzle. Nevertheless, the solution process can be systematized by representing the problem as a shortest-route problem.

A node is defined to represent the amount of fluid in the 8-, 5-, and 3-gallon jugs, respectively. This means that the network starts with node (8, 0, 0) and terminates with the desired

FIGURE 6.12
Most-reliable-route representation as a shortest-route model



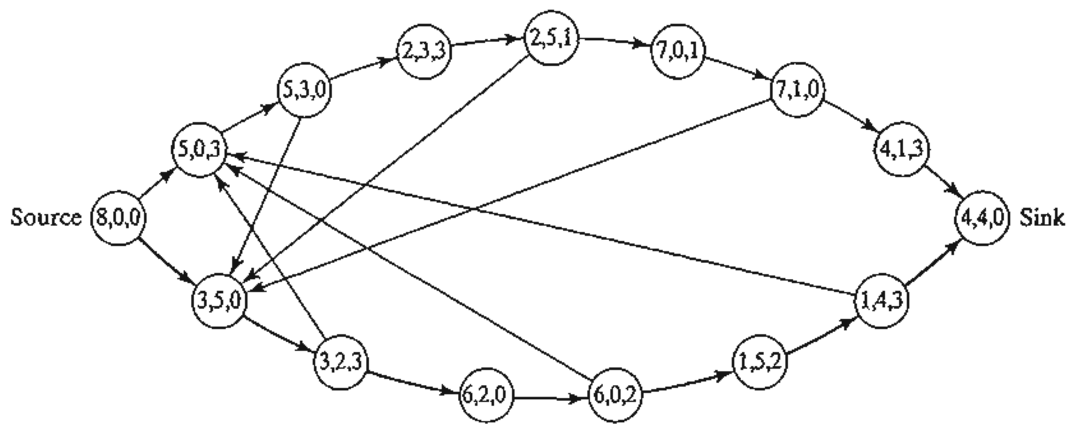


FIGURE 6.13
Three-jug puzzle representation as a shortest-route model

solution node (4, 4, 0). A new node is generated from the current node by decanting fluid from one jug into another.

Figure 6.13 shows different routes that lead from the start node (8, 0, 0) to the end node (4, 4, 0). The arc between two successive nodes represents a single transfer, and hence can be assumed to have a length of 1 unit. The problem thus reduces to determining the shortest route between node (8, 0, 0) and node (4, 4, 0).

The optimal solution, given by the bottom path in Figure 6.13, requires 7 transfers.

PROBLEM SET 6.3A

*1. Reconstruct the equipment replacement model of Example 6.3-1, assuming that a car must be kept in service at least 2 years, with a maximum service life of 4 years. The planning horizon is from the start of year 1 to the end of year 5. The following table provides the necessary data.

	Replacement cost ($) for given years in operation		
Year acquired	2	3	4
1	3800	4100	6800
2	4000	4800	7000
3	4200	5300	7200
4	4800	5700	—
5	5300	—	—

2. Figure 6.14 provides the communication network between two stations, 1 and 7. The probability that a link in the network will operate without failure is shown on each arc. Messages are sent from station 1 to station 7, and the objective is to determine the route that will maximize the probability of a successful transmission. Formulate the situation as a shortest-route model and determine the optimum solution.

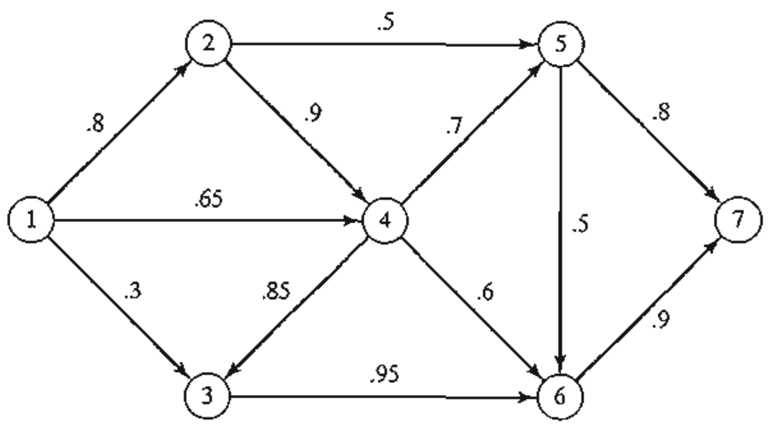


FIGURE 6.14
Network for Problem 2, Set 6.3a

3. *Production Planning.* DirectCo sells an item whose demands over the next 4 months are 100, 140, 210, and 180 units, respectively. The company can stock just enough supply to meet each month's demand, or it can overstock to meet the demand for two or more successive and consecutive months. In the latter case, a holding cost of $1.20 is charged per overstocked unit per month. DirectCo estimates the unit purchase prices for the next 4 months to be $15, $12, $10, and $14, respectively. A setup cost of $200 is incurred each time a purchase order is placed. The company wants to develop a purchasing plan that will minimize the total costs of ordering, purchasing, and holding the item in stock. Formulate the problem as a shortest-route model, and use TORA to find the optimum solution.

*4. *Knapsack Problem.* A hiker has a 5-ft^3 backpack and needs to decide on the most valuable items to take on the hiking trip. There are three items from which to choose. Their volumes are 2, 3, and 4 ft^3, and the hiker estimates their associated values on a scale from 0 to 100 as 30, 50, and 70, respectively. Express the problem as longest-route network, and find the optimal solution. (*Hint:* A node in the network may be defined as $[i, v]$, where i is the item number considered for packing, and v is the volume remaining immediately before a decision is made on i.)

5. An old-fashioned electric toaster has two spring-loaded base-hinged doors. The two doors open outward in opposite directions away from the heating element. A slice of bread is toasted one side at a time by pushing open one of the doors with one hand and placing the slice with the other hand. After one side is toasted, the slice is turned over to get the other side toasted. The goal is to determine the sequence of operations (placing, toasting, turning, and removing) needed to toast three slices of bread in the shortest possible time. Formulate the problem as a shortest-route model, using the following elemental times for the different operations:

Operation	Time (seconds)
Place one slice in either side	3
Toast one side	30
Turn slice already in toaster	1
Remove slice from either side	3

6.3.2 Shortest-Route Algorithms

This section presents two algorithms for solving both cyclic (i.e., containing loops) and acyclic networks:

1. Dijkstra's algorithm
2. Floyd's algorithm

Dijkstra's algorithm is designed to determine the shortest routes between the source node and every other node in the network. Floyd's algorithm is more general because it allows the determination of the shortest route between *any* two nodes in the network.

Dijkstra's Algorithm. Let u_i be the shortest distance from source node 1 to node i, and define d_{ij} (≥ 0) as the length of arc (i, j). Then the algorithm defines the label for an immediately succeeding node j as

$$[u_j, i] = [u_i + d_{ij}, i], d_{ij} \geq 0$$

The label for the starting node is [0, —], indicating that the node has no predecessor.

Node labels in Dijkstra's algorithm are of two types: *temporary* and *permanent*. A temporary label is modified if a shorter route to a node can be found. If no better route can be found, the status of the temporary label is changed to permanent.

Step 0. Label the source node (node 1) with the *permanent* label [0, —]. Set $i = 1$.

Step i. **(a)** Compute the *temporary* labels $[u_i + d_{ij}, i]$ for each node j that can be reached from node i, *provided j is not permanently labeled.* If node j is already labeled with $[u_j, k]$ through another node k and if $u_i + d_{ij} < u_j$, replace $[u_j, k]$ with $[u_i + d_{ij}, i]$.

(b) If *all* the nodes have *permanent* labels, stop. Otherwise, select the label $[u_r, s]$ having the shortest distance ($= u_r$) among all the *temporary* labels (break ties arbitrarily). Set $i = r$ and repeat step i.

Example 6.3-4

The network in Figure 6.15 gives the permissible routes and their lengths in miles between city 1 (node 1) and four other cities (nodes 2 to 5). Determine the shortest routes between city 1 and each of the remaining four cities.

Iteration 0. Assign the *permanent* label [0, —] to node 1.

Iteration 1. Nodes 2 and 3 can be reached from (the last permanently labeled) node 1. Thus, the list of labeled nodes (temporary and permanent) becomes

Node	Label	Status
1	[0, —]	**Permanent**
2	[0 + 100, 1] = [100, 1]	Temporary
3	[0 + 30, 1] = [30, 1]	Temporary

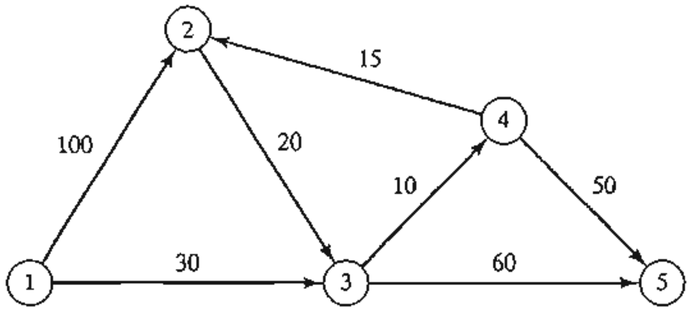


FIGURE 6.15

Network example for Dijkstra's shortest-route algorithm

For the two temporary labels [100, 1] and [30, 1], node 3 yields the smaller distance ($u_3 = 30$). Thus, the status of node 3 is changed to permanent.

Iteration 2. Nodes 4 and 5 can be reached from node 3, and the list of labeled nodes becomes

Node	Label	Status
1	[0, —]	Permanent
2	[100, 1]	Temporary
3	**[30, 1]**	**Permanent**
4	[30 + 10, 3] = [40, 3]	Temporary
5	[30 + 60, 3] = [90, 3]	Temporary

The status of the temporary label [40, 3] at node 4 is changed to permanent ($u_4 = 40$).

Iteration 3. Nodes 2 and 5 can be reached from node 4. Thus, the list of labeled nodes is updated as

Node	Label	Status
1	[0, —]	Permanent
2	[40 + 15, 4] = [55, 4]	Temporary
3	[30, 1]	Permanent
4	**[40, 3]**	**Permanent**
5	[90, 3] or [40 + 50, 4] = [90, 4]	Temporary

Node 2's temporary label [100, 1] obtained in iteration 1 is changed to [55, 4] in iteration 3 to indicate that a shorter route has been found through node 4. Also, in iteration 3, node 5 has two alternative labels with the same distance $u_5 = 90$.

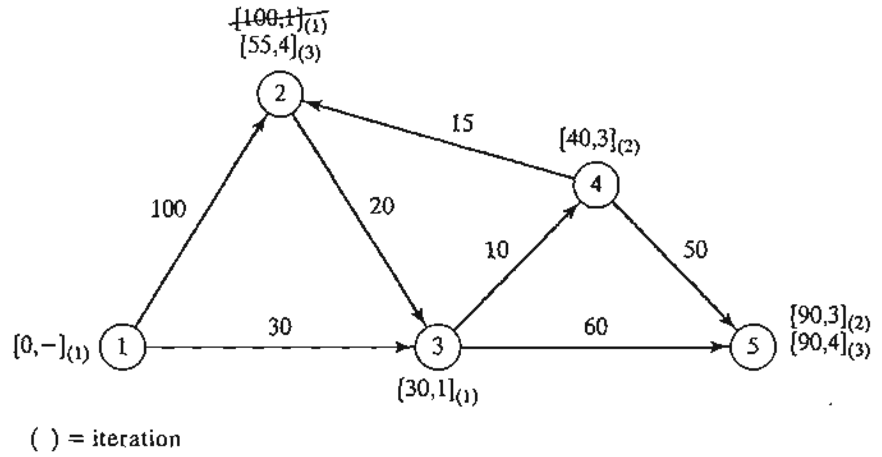


FIGURE 6.16
Dijkstra's labeling procedure

The list for iteration 3 shows that the label for node 2 is now permanent.

Iteration 4. Only node 3 can be reached from node 2. However, node 3 has a permanent label and cannot be relabeled. The new list of labels remains the same as in iteration 3 except that the label at node 2 is now permanent. This leaves node 5 as the only temporary label. Because node 5 does not lead to other nodes, its status is converted to permanent, and the process ends.

The computations of the algorithm can be carried out more easily on the network, as Figure 6.16 demonstrates.

The shortest route between nodes 1 and any other node in the network is determined by starting at the desired destination node and backtracking through the nodes using the information given by the permanent labels. For example, the following sequence determines the shortest route from node 1 to node 2:

$$(2) \to [55, 4] \to (4) \to [40, 3] \to (3) \to [30, 1] \to (1)$$

Thus, the desired route is $1 \to 3 \to 4 \to 2$ with a total length of 55 miles.

TORA Moment

TORA can be used to generate Dijkstra's iterations. From SOLVE/MODIFY menu, select Solve problem ⇒ Iterations ⇒ Dijkstra's algorithm. File toraEx6.3-4.txt provides TORA's data for Example 6.3-4.

PROBLEM SET 6.3B

1. The network in Figure 6.17 gives the distances in miles between pairs of cities 1, 2, ..., and 8. Use Dijkstra's algorithm to find the shortest route between the following cities:
   (a) Cities 1 and 8
   (b) Cities 1 and 6

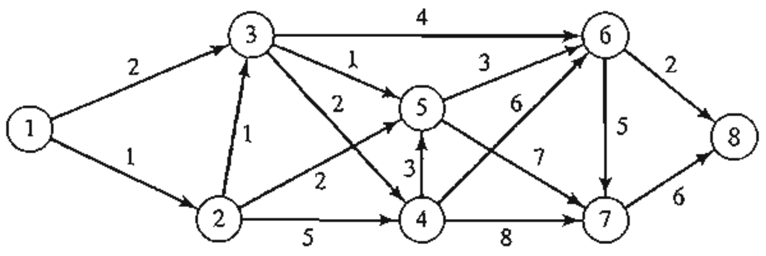


FIGURE 6.17
Network for Problem 1, Set 6.3b

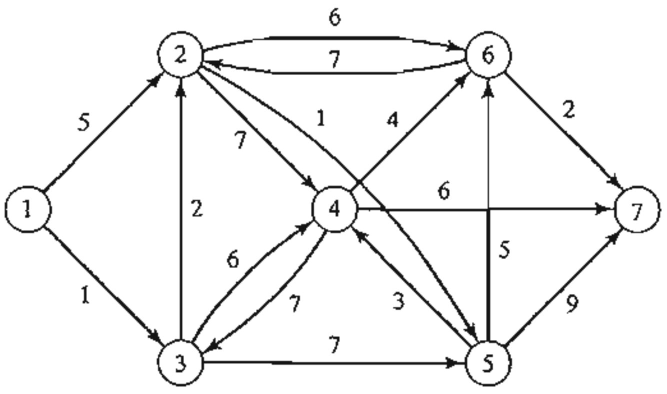


FIGURE 6.18
Network for Problem 2, Set 6.3b

- ***(c)** Cities 4 and 8
- **(d)** Cities 2 and 6

2. Use Dijkstra's algorithm to find the shortest route between node 1 and every other node in the network of Figure 6.18.
3. Use Dijkstr'a algorithm to determine the optimal solution of each of the following problems:
   - **(a)** Problem 1, Set 6.3a.
   - **(b)** Problem 2, Set 6.3a.
   - **(c)** Problem 4, Set 6.3a.

Floyd's Algorithm. Floyd's algorithm is more general than Dijkstra's because it determines the shortest route between *any* two nodes in the network. The algorithm represents an n-node network as a square matrix with n rows and n columns. Entry (i, j) of the matrix gives the distance d_{ij} from node i to node j, which is finite if i is linked directly to j, and infinite otherwise.

The idea of Floyd's algorithm is straightforward. Given three nodes i, j, and k in Figure 6.19 with the connecting distances shown on the three arcs, it is shorter to reach j from i passing through k if

$$d_{ik} + d_{kj} < d_{ij}$$

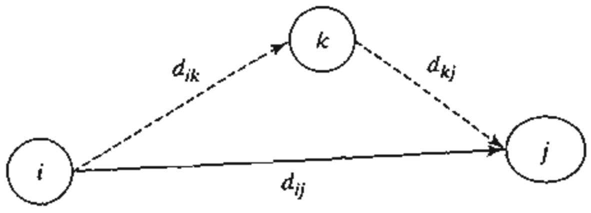


FIGURE 6.19
Floyd's triple operation

In this case, it is optimal to replace the direct route from $i \rightarrow j$ with the indirect route $i \rightarrow k \rightarrow j$. This **triple operation** exchange is applied systematically to the network using the following steps:

Step 0. Define the starting distance matrix D_0 and node sequence matrix S_0 as given below. The diagonal elements are marked with (—) to indicate that they are blocked. Set $k = 1$.

$$D_0 = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & n \\ \hline 1 & — & d_{12} & \cdots & d_{ij} & \cdots & d_{1n} \\ 2 & d_{21} & — & \cdots & d_{2j} & \cdots & d_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i & d_{i1} & d_{i2} & \cdots & d_{ij} & \cdots & d_{in} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & D_{n1} & d_{n2} & \cdots & d_{nj} & \cdots & — \\ \hline \end{array}$$

$$S_0 = \begin{array}{c|cccccc|} & 1 & 2 & \cdots & j & \cdots & n \\ \hline 1 & — & 2 & \cdots & j & \cdots & n \\ 2 & 1 & — & \cdots & j & \cdots & n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ i & 1 & 2 & \cdots & j & \cdots & n \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ n & 1 & 2 & \cdots & j & \cdots & — \\ \hline \end{array}$$

General step *k*. Define row k and column k as *pivot row* and *pivot column*. Apply the *triple operation* to each element d_{ij} in D_{k-1}, for all i and j. If the condition

$$d_{ik} + d_{kj} < d_{ij}, (i \neq k, j \neq k, \text{and } i \neq j)$$

is satisfied, make the following changes:

(a) Create D_k by replacing d_{ij} in D_{k-1} with $d_{ik} + d_{kj}$

(b) Create S_k by replacing s_{ij} in S_{k-1} with k. Set $k = k + 1$. If $k = n + 1$, stop; else repeat step k.

Step k of the algorithm can be explained by representing D_{k-1} as shown in Figure 6.20. Here, row k and column k define the current pivot row and column. Row i

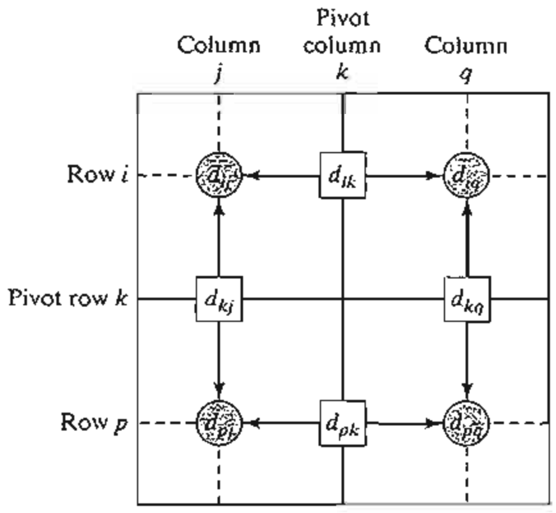


FIGURE 6.20
Implementation of triple operation in matrix form

represents any of the rows $1, 2, \ldots$, and $k - 1$, and row p represents any of the rows $k + 1, k + 2, \ldots$, and n. Similarly, column j represents any of the columns $1, 2, \ldots$, and $k - 1$, and column q represents any of the columns $k + 1$, $k + 2, \ldots$, and n. The *triple operation* can be applied as follows. If the sum of the elements on the pivot row and the pivot column (shown by squares) is smaller than the associated intersection element (shown by a circle), then it is optimal to replace the intersection distance by the sum of the pivot distances.

After n steps, we can determine the shortest route between nodes i and j from the matrices D_n and S_n using the following rules:

1. From D_n, d_{ij} gives the shortest distance between nodes i and j.
2. From S_n, determine the intermediate node $k = s_{ij}$ that yields the route $i \rightarrow k \rightarrow j$. If $s_{ik} = k$ and $s_{kj} = j$, stop; all the intermediate nodes of the route have been found. Otherwise, repeat the procedure between nodes i and k, and between nodes k and j.

Example 6.3-5

For the network in Figure 6.21, find the shortest routes between every two nodes. The distances (in miles) are given on the arcs. Arc (3,5) is directional, so that no traffic is allowed from node 5 to node 3. All the other arcs allow two-way traffic.

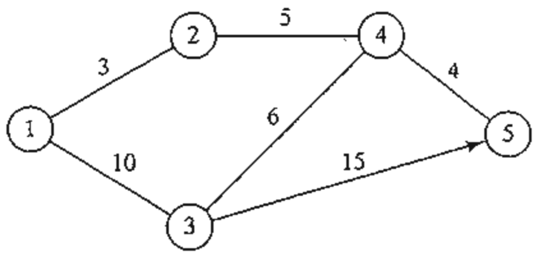


FIGURE 6.21
Network for Example 6.3-5

Iteration 0. The matrices D_0 and S_0 give the initial representation of the network. D_0 is symmetrical, except that $d_{53} = \infty$ because no traffic is allowed from node 5 to node 3.

D_0

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	∞	5	∞
3	10	∞	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_0

	1	2	3	4	5
1	—	2	3	4	5
2	1	—	3	4	5
3	1	2	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

Iteration 1. Set $k = 1$. The pivot row and column are shown by the lightly shaded first row and first column in the D_0-matrix. The darker cells, d_{23} and d_{32}, are the only ones that can be improved by the *triple operation*. Thus, D_1 and S_1 are obtained from D_0 and S_0 in the following manner:

1. Replace d_{23} with $d_{21} + d_{13} = 3 + 10 = 13$ and set $s_{23} = 1$.
2. Replace d_{32} with $d_{31} + d_{12} = 10 + 3 = 13$ and set $s_{32} = 1$.

D_1

	1	2	3	4	5
1	—	3	10	∞	∞
2	3	—	**13**	5	∞
3	10	**13**	—	6	15
4	∞	5	6	—	4
5	∞	∞	∞	4	—

S_1

	1	2	3	4	5
1	—	2	3	3	5
2	1	—	**1**	4	5
3	1	**1**	—	4	5
4	1	2	3	—	5
5	1	2	3	4	—

Iteration 2. Set $k = 2$, as shown by the lightly shaded row and column in D_1. The *triple operation* is applied to the darker cells in D_1 and S_1. The resulting changes are shown in bold in D_2 and S_2.

D_2

	1	2	3	4	5
1	—	3	10	**8**	∞
2	3	—	13	5	∞
3	10	13	—	6	15
4	**8**	5	6	—	4
5	∞	∞	∞	4	—

S_2

	1	2	3	4	5
1	—	2	3	**2**	5
2	1	—	1	4	5
3	1	1	—	4	5
4	**2**	2	3	—	5
5	1	2	3	4	—

Iteration 3. Set $k = 3$, as shown by the shaded row and column in D_2. The new matrices are given by D_3 and S_3.

D_3

	1	2	3	4	5
1	—	3	10	8	25
2	3	—	13	5	28
3	10	13	—	6	15
4	8	5	6	—	4
5	∞	∞	∞	4	—

S_3

	1	2	3	4	5
1	—	2	3	2	3
2	1	—	1	4	3
3	1	1	—	4	5
4	2	2	3	—	5
5	1	2	3	4	—

Iteration 4. Set $k = 4$, as shown by the shaded row and column in D_3. The new matrices are given by D_4 and S_4.

D_4

	1	2	3	4	5
1	—	3	10	8	**12**
2	3	—	**11**	5	**9**
3	10	**11**	—	6	**10**
4	8	5	6	—	4
5	**12**	**9**	**10**	4	—

S_4

	1	2	3	4	5
1	—	2	3	2	**4**
2	1	—	**4**	4	**4**
3	1	**4**	—	4	**4**
4	2	2	3	—	5
5	**4**	**4**	**4**	4	—

Iteration 5. Set $k = 5$, as shown by the shaded row and column in D_4. No further improvements are possible in this iteration.

The final matrices D_4 and S_4 contain all the information needed to determine the shortest route between any two nodes in the network. For example, from D_4, the shortest distance from node 1 to node 5 is $d_{15} = 12$ miles. To determine the associated route, recall that a segment (i, j) represents a direct link only if $s_{ij} = j$. Otherwise, i and j are linked through at least one other intermediate node. Because $s_{15} = 4 \neq 5$, the route is initially given as $1 \rightarrow 4 \rightarrow 5$. Now, because $s_{14} = 2 \neq 4$, the segment $(1, 4)$ is not a *direct* link, and $1 \rightarrow 4$ is replaced with $1 \rightarrow 2 \rightarrow 4$, and the route $1 \rightarrow 4 \rightarrow 5$ now becomes $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$. Next, because $s_{12} = 2$, $s_{24} = 4$, and $s_{45} = 5$, no further "dissecting" is needed, and $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ defines the shortest route.

TORA Moment

As in Dijkstra's algorithm, TORA can be used to generate Floyd's iterations. From SOLVE/MODIFY menu, select Solve problem ⇒ Iterations ⇒ Floyd's algorithm. File toraEx6.3-5.txt provides TORA's data for Example 6.3-5.

PROBLEM SET 6.3C

1. In Example 6.3-5, use Floyd's algorithm to determine the shortest routes between each of the following pairs of nodes:
   *(a) From node 5 to node 1.
   (b) From node 3 to node 5.

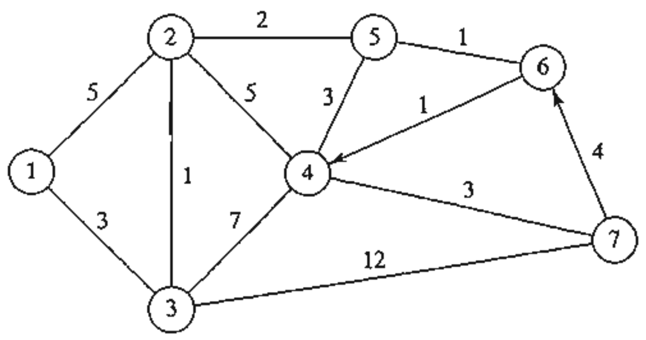


FIGURE 6.22
Network for Problem 2, Set 6.3c

(c) From node 5 to node 3.
(d) From node 5 to node 2.

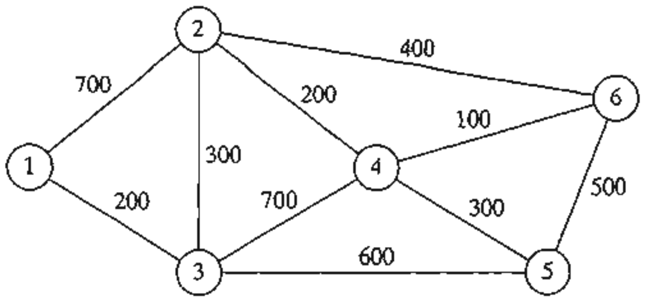
2. Apply Floyd's algorithm to the network in Figure 6.22. Arcs (7, 6) and (6, 4) are unidirectional, and all the distances are in miles. Determine the shortest route between the following pairs of nodes:
   (a) From node 1 to node 7.
   (b) From node 7 to node 1.
   (c) From node 6 to node 7.

3. The Tell-All mobile-phone company services six geographical areas. The satellite distances (in miles) among the six areas are given in Figure 6.23. Tell-All needs to determine the most efficient message routes that should be established between each two areas in the network.

*4. Six kids, Joe, Kay, Jim, Bob, Rae, and Kim, play a variation of *hide and seek*. The hiding place of a child is known only to a select few of the other children. A child is then paired with another with the objective of finding the partner's hiding place. This may be achieved through a chain of other kids who eventually will lead to discovering where the designated child is hiding. For example, suppose that Joe needs to find Kim and that Joe knows where Jim is hiding, who in turn knows where Kim is. Thus, Joe can find Kim by first finding Jim, who in turn will lead Joe to Kim. The following list provides the whereabouts of the children:

   Joe knows the hiding places of Bob and Kim.
   Kay knows the hiding places of Bob, Jim, and Rae.
   Jim and Bob each know the hiding place of Kay only.
   Rae knows where Kim is hiding.
   Kim knows where Joe and Bob are hiding.

FIGURE 6.23
Network for Problem 3, Set 6.3c



6.3.

Devise a plan for each child to find every other child through the smallest number of contacts. What is the largest number of contacts?

6.3.3 Linear Programming Formulation of the Shortest-Route Problem

This section provides an LP model for the shortest-route problem. The model is general in the sense that it can be used to find the shortest route between any two nodes in the network. In this regard, it is equivalent to Floyd's algorithm.

Suppose that the shortest-route network includes n nodes and that we desire to determine the shortest route between any two nodes s and t in the network. The LP assumes that one unit of flow enters the network at node s and leaves at node t.

Define

$$\begin{aligned} x_{ij} &= \text{amount of flow in arc } (i, j) \\ &= \begin{cases} 1, \text{ if arc } (i, j) \text{ is on the shortest route} \\ 0, \text{ otherwise} \end{cases} \\ c_{ij} &= \text{length of arc } (i, j) \end{aligned}$$

Thus, the objective function of the linear program becomes

$$\text{Minimize } z = \sum_{\substack{\text{all defined} \\ \text{arcs } (i, j)}} c_{ij}x_{ij}$$

The constraints represent the *conservation-of-flow equation* at each node:

$$\text{Total input flow} = \text{Total output flow}$$

Mathematically, this translates for node j to

$$\begin{pmatrix}\text{External input} \\ \text{into node } j\end{pmatrix} + \sum_{\substack{i \\ \text{all defined} \\ \text{arcs } (i, j)}} x_{ij} = \begin{pmatrix}\text{External output} \\ \text{from node } j\end{pmatrix} + \sum_{\substack{k \\ \text{all defined} \\ \text{arcs } (j, k)}} x_{jk}$$

Example 6.3-6

Consider the shortest-route network of Example 6.3-4. Suppose that we want to determine the shortest route from node 1 to node 2—that is, $s = 1$ and $t = 2$. Figure 6.24 shows how the unit of flow enters at node 1 and leaves at node 2.

We can see from the network that the flow-conservation equation yields

$$\begin{aligned} &\text{Node 1:} & 1 &= x_{12} + x_{13} \\ &\text{Node 2: } x_{12} + x_{42} & &= x_{23} + 1 \\ &\text{Node 3: } x_{13} + x_{23} & &= x_{34} + x_{35} \\ &\text{Node 4:} & x_{34} &= x_{42} + x_{45} \\ &\text{Node 5: } x_{35} + x_{45} & &= 0 \end{aligned}$$

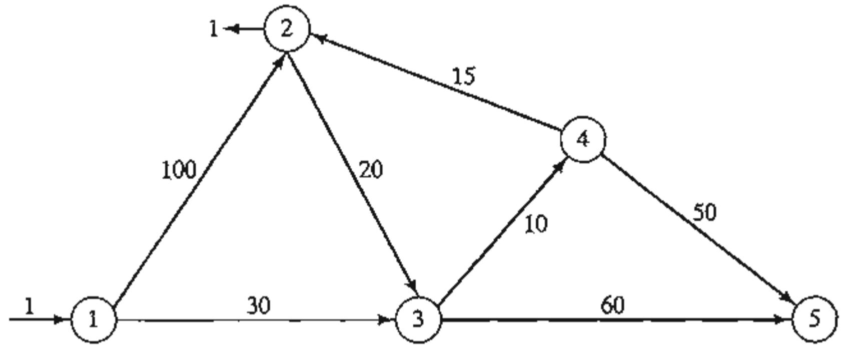


FIGURE 6.24
Insertion of unit flow to determine shortest route between node $s = 1$ and node $t = 2$

The complete LP can be expressed as

	x_{12}	x_{13}	x_{23}	x_{34}	x_{35}	x_{42}	x_{45}	
Minimize $z =$	100	30	20	10	60	15	50	
Node 1	1	1						$= 1$
Node 2	-1		1			-1		$= -1$
Node 3		-1	-1	1	1			$= 0$
Node 4				-1		1	1	$= 0$
Node 5					-1		-1	$= 0$

Notice that column x_{ij} has exactly one "1" entry in row i and one "-1" entry in row j, a typical property of a network LP.

The optimal solution (obtained by TORA, file toraEx6.3-6.txt) is

$$z = 55, x_{13} = 1, x_{34} = 1, x_{42} = 1$$

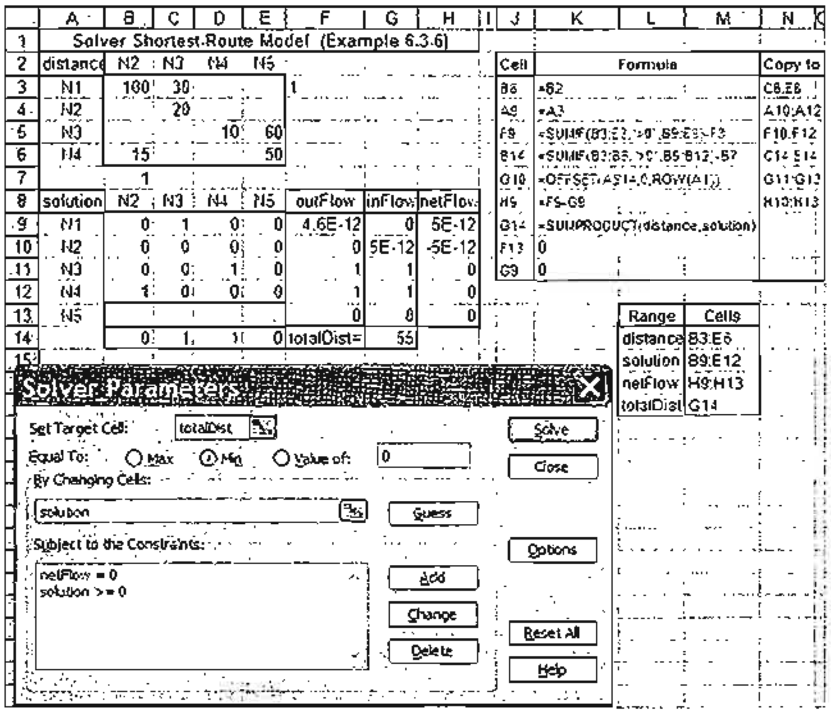
This solution gives the shortest route from node 1 to node 2 as $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$, and the associated distance is $z = 55$ (miles).

PROBLEM 6.3D

1. In Example 6.3-6, use LP to determine the shortest routes between the following pairs of nodes:
   *(a) Node 1 to node 5.
   (b) Node 2 to node 5.

Solver Moment

Figure 6.25 provides the Excel Solver spreadsheet for finding the shortest route between *start* node N1 and *end* node N2 of Example 6.3-6 (file solverEx6.3-6.xls). The input data of the model is the distance matrix in cells B3:E6. Node N1 has no column because it has no incoming arcs, and node N5 has no row because it has no outgoing arcs. A blank entry means that the corresponding arc does not exist. Nodes N1 and N2 are designated as the *start* and *end* nodes by entering 1 in F3 and B7, respectively. These designations can be changed simply by moving the entry 1 to new cells. For example, to find the shortest route from node N2 to node N4, we enter 1 in each of F4 and D7.



	A	B	C	D	E	F	G	H
1	Solver Shortest-Route Model (Example 6.3-6)							
2	distance	N2	N3	N4	N5			
3	N1	100	30			1		
4	N2		20					
5	N3			10	60			
6	N4	15			50			
7		1						
8	solution	N2	N3	N4	N5	outFlow	inFlow	netFlow
9	N1	0	1	0	0	4.6E-12	0	5E-12
10	N2	0	0	0	0	0	5E-12	-5E-12
11	N3	0	0	1	0	1	1	0
12	N4	1	0	0	0	1	1	0
13	N5					0	0	0
14		0	1	1	0	totalDist=	55	

Cell	Formula	Copy to
B8	=B2	C8:E8
A9	=A3	A10:A12
F9	=SUMIF(B3:E3,">0",B9:E9)+F3	F10:F12
B14	=SUMIF(B3:B6,">0",B9:B12)+B7	C14:E14
G10	=OFFSET(A14,0,ROW(A1))	G11:G13
H9	=F9-G9	H10:H13
G14	=SUMPRODUCT(distance,solution)	
F13	0	
G9	0	

Range	Cells
distance	B3:E6
solution	B9:E12
netFlow	H9:H13
totalDist	G14

Solver Parameters

Set Target Cell: totalDist

Equal To: ○ Max [✓ ◉ Min] ○ Value of: 0

By Changing Cells:

solution

Subject to the Constraints:

netFlow = 0
solution >= 0

FIGURE 6.25

Excel Solver solution of the shortest route between nodes 1 and 2 in Example 6.3-6 (file solverEx6.3-6.xls)

The solution of the model is given cells B9:E12. A cell defines a leg connecting its designated nodes. For example, cell C10 defines the leg (N2, N3), and its associated variable is x_{23}. A cell variable $x_{ij} = 1$ if its leg (N*i*, N*j*) is on the route. Otherwise, its value is zero.

With the distance matrix given by the range B3:E6 (named *distance*) and the solution matrix given by the range B9:E12 (named *solution*), the objective function is computed in cell G14 as =SUMPRODUCT(B3:E6,B9:E12) or, equivalently, =SUMPRODUCT (distance,solution). You may wonder about the significance of the blank entries (which default to zero by Excel) in the distance matrix and their impact on the definition of the objective function. This point will be addressed shortly, after we have shown how the corresponding variables are totally excluded from the constraints of the problem.

As explained in the LP of Example 6.3-6, the constraints of the problem are of the general form:

$$(\text{Output flow}) - (\text{Input flow}) = 0$$

This definition is adapted to the spreadsheet layout by incorporating the external unit flow, if any, directly in either *Output flow* or *Input flow* of the equation. For example, in Example 6.3-6, an external flow unit enters at N1 and leaves at N2. Thus, the associated constraints are given as

$$\left.\begin{array}{ll}(\text{Output flow at N1}) & = x_{12} + x_{13} - 1 \\ (\text{Input flow at N1}) & = 0\end{array}\right\} \Rightarrow x_{12} + x_{13} - 1 = 0$$

$$\left.\begin{array}{ll}(\text{Output flow at N2}) & = x_{23} \\ (\text{Input flow at N2}) & = x_{12} + x_{42} - 1\end{array}\right\} \Rightarrow x_{23} - x_{12} - x_{42} - 1 = 0$$

Looking at the spreadsheet in Figure 6.25, the two constraints are expressed in terms of the cells as

(Output flow at N1)`=B9+C9-F3`

(Input flow at N1) `=0`

(Output flow at N2)`=C10`

(Input flow at N2) `=B9+B12-B7`

To identify the solution cells in the range B9:E12 that apply to each constraint, we note that a solution cell forms part of a constraint only if it has a positive entry in the distance matrix.[3] Thus, we use the following formulas to identify the output and input flows for each node:

1. Output flow: Enter`=SUMIF(B3:E3,">0",B9:E9)-F3` in cell F9 and copy it in cells F10:F12.
2. Input Flow: Enter`=SUMIF(B3:B6,">0",B9:B12)-B7` in cell B14 and copy it in cells C14:E14.
3. Enter `=OFFSET(A$14,0,ROW(A1))` in cell G10 and copy it in cells G11:G13 to transpose the input flow to column G.
4. Enter 0 in each of G9 and F13 to indicate that N1 has no input flow and N5 has no output flow (per spreadsheet definitions).
5. Enter `=F9-G9` in cell H9 and copy it in cells H10:H13 to compute the net flow.

The spreadsheet is now ready for the application of Solver as shown in Figure 6.25. There is one curious occurrence though: When you define the constraints within the **Solver Parameters** dialogue box as $outFlow = inFlow$, Solver does not locate a feasible solution, even after making adjustments in *precision* in the **Solver Option** box. (To reproduce this experience, *solution* cells B9:E12 must be reset to zero or blank.) More curious yet, if the constraints are replaced with $inFlow = outFlow$, the optimum solution is found. In file solverEx6.3-6.xls, we use the *netFlow* range in cells H9:H12 and express the constraint as $netFlow = 0$ with no problem. It is not clear why this peculiarity occurs, but the problem could be related to roundoff error.

The output in Figure 6.25 yields the solution (N1-N3 = 1, N3-N4 = 1, N4-N2 = 1) with a total distance of 55 miles. This means that the optimal route is $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$.

Remarks. In most textbooks, the network is defined by its explicit arcs (node *i*, node *j*, distance), a lengthy and inconvenient process that may not be practical when the number of arcs is large. Our model is driven primarily by the compact distance matrix, which is all we need to develop the flow constraints. It may be argued that our model deals with $(n - 1 \times n - 1)x_{ij}$-variables, which could be much larger than the number of variables associated with the arcs of the model (for instance, Example 6.3-6 has 7 arcs

[3]If a problem happens to have a zero distance between two nodes, the zero distance can be replaced with a very small positive value.

and hence 7 x_{ij}-variables, as opposed to $4 \times 4 = 16$ in our formulation). Keep in mind that these additional variables appear only in the objective function and with zero coefficients (blank entries) and that the flow constraints are *exactly the same* as in other presentations (per the SUMIF function). As a result, *pre-solvers* in commercial software will spot this "oddity" and automatically exclude the additional variables prior to applying the simplex method, with no appreciable computational overhead.

AMPL Moment

Figure 6.26 provides the AMPL model for solving Example 6.3-6 (file amplEx6.3-6a.txt). The variable `x[i,j]` assumes the value 1 if arc `[i,j]` is on the shortest route and 0 otherwise. The model is general in the sense that it can be used to find the shortest route between any two nodes in a problem of any size.

As explained in Example 6.3-6, AMPL treats the problem as a network in which an external flow unit enters and exits at specified `start` and `end` nodes. The main input data of the model is an $n \times n$ matrix representing the distance `d[i,j]` of the arc joining nodes `i` and `j`. Per AMPL syntax, a dot entry in `d[i,j]` is a placeholder that signifies

FIGURE 6.26

AMPL shortest route model (file amplEx6.3-6a.txt)

```
#------ shortest route model (Example 6.3-6)------
param n;
param start;
param end;
param M=999999; #infinity
param d{i in 1..n, j in 1..n} default M;
param rhs{i in 1..n}=if i=start then 1
                    else (if i=end then -1 else 0);
var x{i in 1..n,j in 1..n}>=0;
var outFlow{i in 1..n}=sum{j in 1..n}x[i,j];
var inFlow{j in 1..n}=sum{i in 1..n}x[i,j];

minimize z: sum{i in 1..n, j in 1..n}d[i,j]*x[i,j];
subject to limit{i in 1..n}:outFlow[i]-inFlow[i]=rhs[i];

data;
param n:=5;
param start:=1;
param end:=2;
param d:
   1  2   3  4  5:=
1 .  100 30 .  .
2 .  .   20 .  .
3 .  .   .  10 60
4 .  15  .  .  50
5 .  .   .  .  .;
solve;
print "Shortest length from",start,"to",end,"=",z;
printf "Associated route: %2i",start;
for {i in 1..n-1} for {j in 2..n}
     {if x[i,j]=1 then printf" - %2i",j;} print;
```

that no distance is specified for the corresponding arc. In the model, the dot entry is overridden by the infinite distance `M` (= 999999) in `param d{i in 1...n, j in 1...n} default M;` which will convert it into an infinite-distance route. The same result could be achieved by replacing the dot entry `(.)` with `999999` in the data section, which, in addition to "cluttering" the data, is inconvenient.

The constraints represent flow conservation through each node:

$$(\text{Input flow}) - (\text{Output flow}) = (\text{External flow})$$

From `x[i, j]`, we can define the input and output flow for node i using the statements

```
var inFlow{j in 1..n}=sum{i in 1..n}x[i,j];
var outFlow{i in 1..n}=sum{j in 1..n}x[i,j];
```

The left-hand side of the constraint i is thus given as `outFlow[i]-inFlow[i]`.

The right-hand side of constraint i (external flow at node i) is defined as

```
param rhs{i in 1..n}=if i=start then 1 else(if i=end then -1 else 0);
```

(See Section A.3 for details of `if then else`.) With this statement, specifying `start` and `end` nodes automatically assigns `1, -1, or 0` to `rhs`, the right-hand side of the constraints.

The objective function seeks the minimization of the sum of `d[i,j]*x[i,j]` over all `i` and `j`.

In the present example, `start=1` and `end=2`, meaning that we want to determine the shortest route from node 1 to node 2. The associated output is

```
Shortest length from 1 to 2 = 55
Associated route:  1 -  3 -  4 -  2
```

Remarks. The AMPL model as given in Figure 6.26 has one flaw: The number of *active* variables x_{ij} is n^2, which could be significantly much larger than the actual number of (positive-distance) arcs in the network, thus resulting in a much larger problem. The reason is that the model accounts for the nonexisting arcs by assigning them an infinite distance `M` (= 999999) to guarantee that they will be zero in the optimum solution. This situation can be remedied by using a subset of `{i in 1..n,j in 1..n}` that excludes nonexisiting arcs, as the following statement shows:

```
var x{i in 1..n,j in 1..n:d[i,j]<M}>=0;
```

(See Section A.4 for the use of conditions to define subsets.) The same logic must be applied to the constraints as well by using the following statements:

```
var inFlow{j in 1..n}=sum{i in 1..n:d[i,j]<M}x[i,j];
var outFlow{i in 1..n}=sum{j in 1..n:d[i,j]<M}x[i,j];
```

File amplEx6.36b.txt gives the complete model.

PROBLEM 6.3E

1. Modify solverEx6.3-6.xls to find the shortest route between the following pairs of nodes:
   (a) Node 1 to node 5.
   (b) Node 4 to node 3.
2. Adapt amplEx6.3-6b.txt for Problem 2, Set 6.3a, to find the shortest route between node 1 and node 7. The input data must be the raw probabilities. Use AMPL programming facilities to print/display the optimum transmission route and its success probability.

6.4 MAXIMAL FLOW MODEL

Consider a network of pipelines that transports crude oil from oil wells to refineries. Intermediate booster and pumping stations are installed at appropriate design distances to move the crude in the network. Each pipe segment has a finite maximum discharge rate of crude flow (or capacity). A pipe segment may be uni- or bidirectional, depending on its design. Figure 6.27 demonstrates a typical pipeline network. How can we determine the maximum capacity of the network between the wells and the refineries?

The solution of the proposed problem requires equipping the network with a single source and a single sink by using unidirectional infinite capacity arcs as shown by dashed arcs in Figure 6.27.

Given arc (i, j) with $i < j$, we use the notation $(\overline{C}_{ij}, \overline{C}_{ji})$ to represent the flow capacities in the two directions $i \rightarrow j$ and $j \rightarrow i$, respectively. To eliminate ambiguity, we place $\overline{C}_{ij}$ on the arc next to node i with $\overline{C}_{ji}$ placed next to node j, as shown in Figure 6.28.

6.4.1 Enumeration of Cuts

A **cut** defines a set of arcs which when deleted from the network will cause a total disruption of flow between the source and sink nodes. The **cut capacity** equals the sum of the capacities of its arcs. Among *all* possible cuts in the network, the cut with the *smallest capacity* gives the maximum flow in the network.

Example 6.4-1

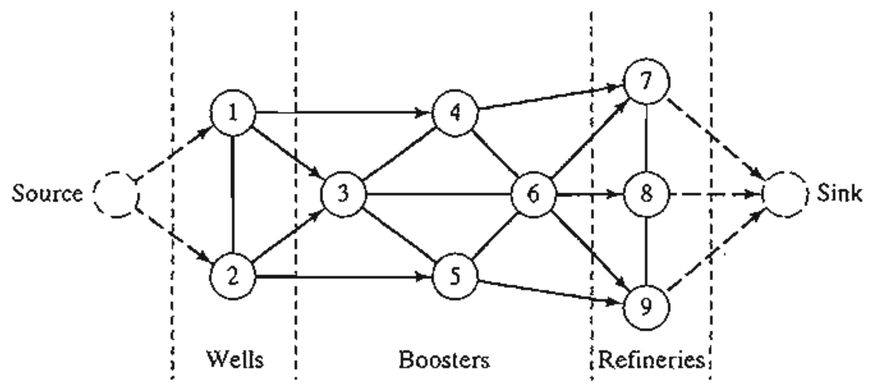
Consider the network in Figure 6.29. The bidirectional capacities are shown on the respective arcs using the convention in Figure 6.28. For example, for arc (3, 4), the flow limit is 10 units from 3 to 4 and 5 units from 4 to 3.

Figure 6.29 illustrates three cuts whose capacities are computed in the following table.

Cut	Associated arcs	Capacity
1	(1,2), (1,3), (1,4)	$20 + 30 + 10 = 60$
2	(1,3), (1,4), (2,3), (2,5)	$30 + 10 + 40 + 30 = 110$
3	(2,5), (3,5), (4,5)	$30 + 20 + 20 = 70$

FIGURE 6.27

Capacitated network connecting wells and refineries through booster stations



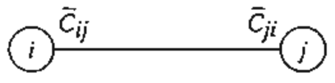


FIGURE 6.28
Arc flows $\bar{C}_{ij}$ from $i \to j$ and $\bar{C}_{ji}$ from $j \to i$

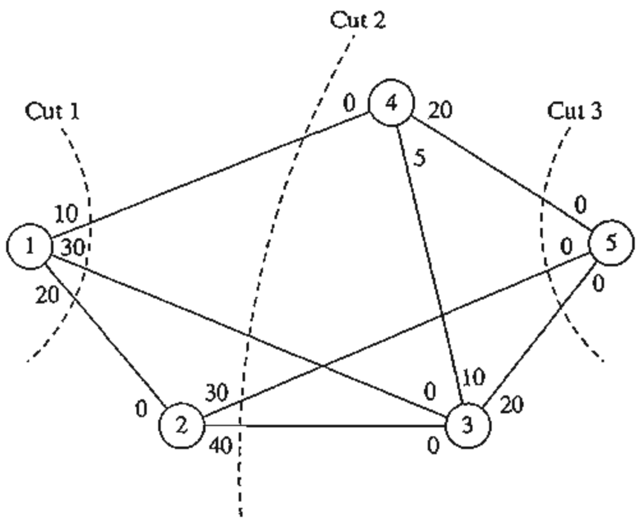


FIGURE 6.29
Examples of cuts in flow networks

The only information we can glean from the three cuts is that the maximum flow in the network cannot exceed 60 units. To determine the maximum flow, it is necessary to enumerate *all* the cuts, a difficult task for the general network. Thus, the need for an efficient algorithm is imperative.

PROBLEM SET 6.4A

*1. For the network in Figure 6.29, determine two additional cuts, and find their capacities.

6.4.2 Maximal Flow Algorithm

The maximal flow algorithm is based on finding **breakthrough paths** with net *positive* flow between the source and sink nodes. Each path commits part or all of the capacities of its arcs to the total flow in the network.

Consider arc (i, j) with (initial) capacities $(\bar{C}_{ij}, \bar{C}_{ji})$. As portions of these capacities are committed to the flow in the arc, the **residuals** (or remaining capacities) of the arc are updated. We use the notation (c_{ij}, c_{ji}) to represent these residuals.

For a node j that receives flow from node i, we attach a label $[a_j, i]$, where a_j is the flow from node i to node j. The steps of the algorithm are thus summarized as follows.

Step 1. For all arcs (i, j), set the residual capacity equal to the initial capacity— that is $(c_{ij}, c_{ji}) = (\bar{C}_{ij}, \bar{C}_{ji})$. Let $a_1 = \infty$ and label source node 1 with $[\infty, -]$. Set $i = 1$, and go to step 2.

Step 2. Determine S_i, the set of unlabeled nodes j that can be reached directly from node i by arcs with *positive* residuals (that is, $c_{ij} > 0$ for all $j \in S_i$). If $S_i \neq \emptyset$, go to step 3. Otherwise, go to step 4.

Step 3. Determine $k \in S_i$ such that

$$c_{ik} = \max_{j \in S_i}\{c_{ij}\}$$

Set $a_k = c_{ik}$ and label node k with $[a_k, i]$. If $k = n$, the sink node has been labeled, and a *breakthrough path* is found, go to step 5. Otherwise, set $i = k$, and go to step 2.

Step 4. **(Backtracking).** If $i = 1$, no breakthrough is possible; go to step 6. Otherwise, let r be the node that has been labeled *immediately* before current node i and remove i from the set of nodes adjacent to r. Set $i = r$, and go to step 2.

Step 5. **(Determination of residuals).** Let $N_p = (1, k_1, k_2, \ldots, n)$ define the nodes of the pth breakthrough path from source node 1 to sink node n. Then the maximum flow along the path is computed as

$$f_p = \min\{a_1, a_{k_1}, a_{k_2}, \ldots, a_n\}$$

The residual capacity of each arc along the breakthrough path is *decreased* by f_p in the direction of the flow and *increased* by f_p in the reverse direction—that is, for nodes i and j on the path, the residual flow is changed from the current (c_{ij}, c_{ji}) to

(a) $(c_{ij} - f_p, c_{ji} + f_p)$ if the flow is from i to j

(b) $(c_{ij} + f_p, c_{ji} - f_p)$ if the flow is from j to i

Reinstate any nodes that were removed in step 4. Set $i = 1$, and return to step 2 to attempt a new breakthrough path.

Step 6. **(Solution).**

(a) Given that m breakthrough paths have been determined, the maximal flow in the network is

$$F = f_1 + f_2 + \cdots + f_m$$

(b) Using the *initial* and *final* residuals of arc (i, j), $(\overline{C}_{ij}, \overline{C}_{ji})$ and (c_{ij}, c_{ji}), respectively, the optimal flow in arc (i, j) is computed as follows: Let $(\alpha, \beta) = (\overline{C}_{ij} - c_{ij}, \overline{C}_{ji} - c_{ji})$. If $\alpha > 0$, the optimal flow from i to j is α. Otherwise, if $\beta > 0$, the optimal flow from j to i is β. (It is impossible to have both α and β positive.)

The backtracking process of step 4 is invoked when the algorithm becomes "dead-ended" at an intermediate node. The flow adjustment in step 5 can be explained via the simple flow network in Figure 6.30. Network (a) gives the first breakthrough path $N_1 = \{1, 2, 3, 4\}$ with its maximum flow $f_1 = 5$. Thus, the residuals of each of arcs (1, 2), (2, 3), and (3, 4) are changed from (5, 0) to (0, 5), per step 5. Network (b) now gives the second breakthrough path $N_2 = \{1, 3, 2, 4\}$ with $f_2 = 5$. After making the necessary flow adjustments, we get network (c), where no further breakthroughs are possible. What happened in the transition from (b) to (c) is nothing but a cancellation of a previously committed flow in the direction $2 \rightarrow 3$. The algorithm is able to "remember" that

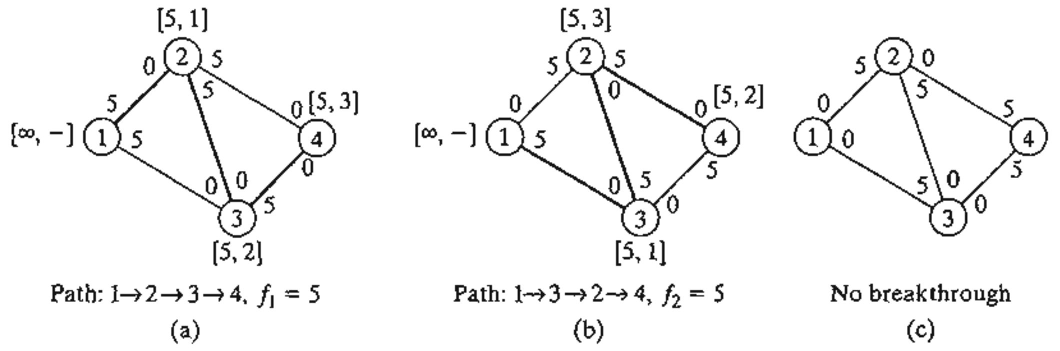


FIGURE 6.30
Use of residuals to calculate maximum flow

a flow from 2 to 3 has been committed previously only because we have increased the capacity in the reverse direction from 0 to 5 (per step 5).

Example 6.4-2

Determine the maximal flow in the network of Example 6.4-1 (Figure 6.29). Figure 6.31 provides a graphical summary of the iterations of the algorithm. You will find it helpful to compare the description of the iterations with the graphical summary.

Iteration 1. Set the initial residuals (c_{ij}, c_{ji}) equal to the initial capacities $(\overline{C}_{ij}, \overline{C}_{ji})$.

Step 1. Set $a_1 = \infty$ and label node 1 with $[\infty, —]$. Set $i = 1$.

Step 2. $S_1 = \{2, 3, 4\}$ $(\neq \emptyset)$.

Step 3. $k = 3$, because $c_{13} = \max\{c_{12}, c_{13}, c_{14}\} = \max\{20, 30, 10\} = 30$. Set $a_3 = c_{13} = 30$, and label node 3 with [30,1]. Set $i = 3$, and repeat step 2.

Step 2. $S_3 = (4, 5)$.

Step 3. $k = 5$ and $a_5 = c_{35} = \max\{10, 20\} = 20$. Label node 5 with [20, 3]. Breakthrough is achieved. Go to step 5.

Step 5. The breakthrough path is determined from the labels starting at node 5 and moving backward to node 1—that is, $(5) \rightarrow [20, 3] \rightarrow (3) \rightarrow [30, 1] \rightarrow (1)$. Thus, $N_1 = \{1, 3, 5\}$ and $f_1 = \min\{a_1, a_3, a_5\} = \{\infty, 30, 20\} = 20$. The residual capacities along path N_1 are

$$(c_{13}, c_{31}) = (30 - 20, 0 + 20) = (10, 20)$$

$$(c_{35}, c_{53}) = (20 - 20, 0 + 20) = (0, 20)$$

Iteration 2

Step 1. Set $a_1 = \infty$, and label node 1 with $[\infty, —]$. Set $i = 1$.

Step 2. $S_1 = \{2, 3, 4\}$.

Step 3. $k = 2$ and $a_2 = c_{12} = \max\{20, 10, 10\} = 20$. Set $i = 2$, and repeat step 2.

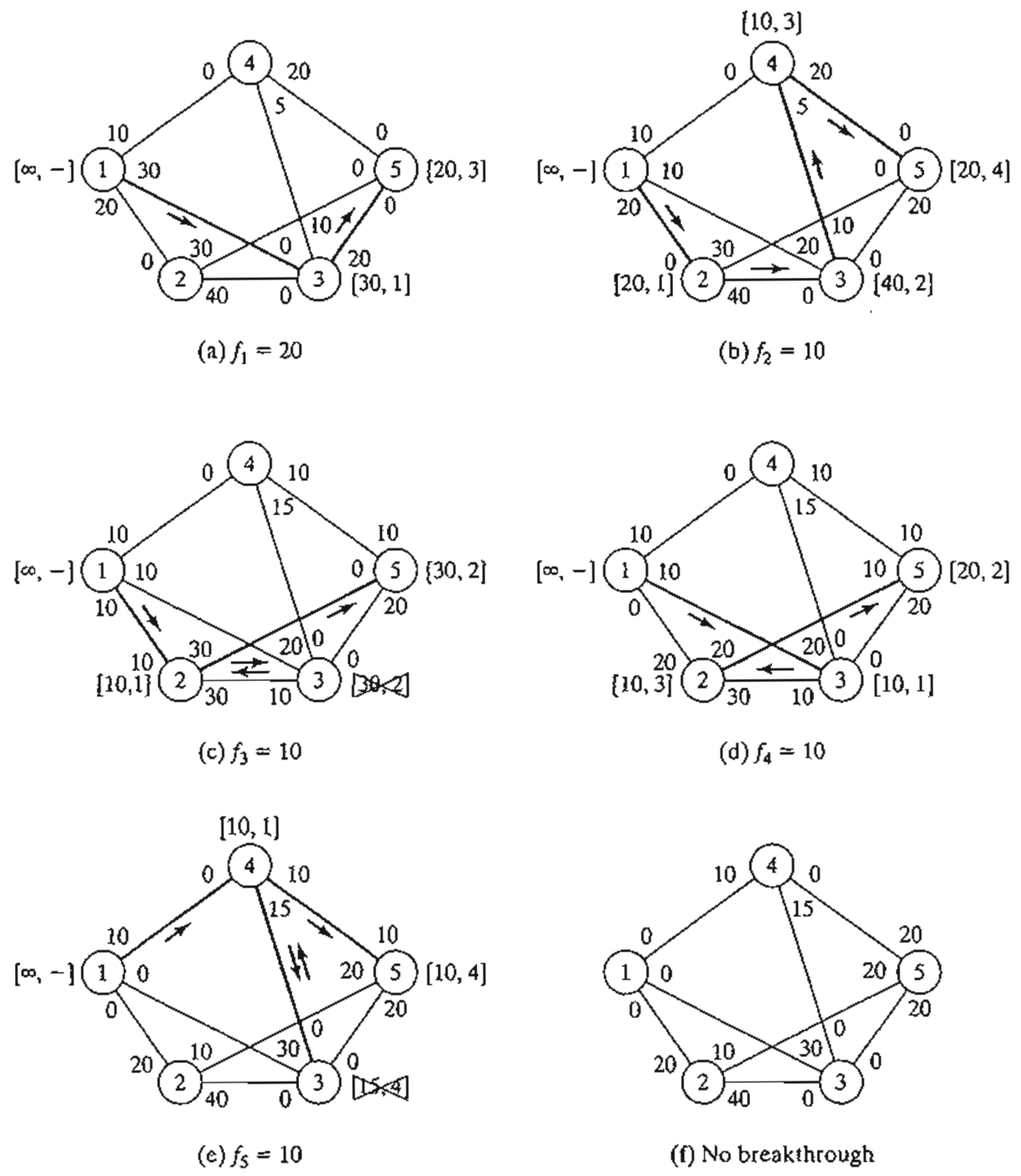


FIGURE 6.31
Iterations of the maximum flow algorithm of Example 6.4-2

Step 2. $S_2 = \{3, 5\}$.

Step 3. $k = 3$ and $a_3 = c_{23} = 40$. Label node 3 with $[40, 2]$. Set $i = 3$, and repeat step 2.

Step 2. $S_3 = \{4\}$ (note that $c_{35} = 0$—hence, node 5 cannot be included in S_3).

Step 3. $k = 4$ and $a_4 = c_{34} = 10$. Label node 4 with $[10, 3]$. Set $i = 4$, and repeat step 2.

Step 2. $S_4 = \{5\}$ (note that nodes 1 and 3 are already labeled—hence, they cannot be included in S_4).

Step 3. $k = 5$ and $a_5 = c_{45} = 20$. Label node 5 with $[20, 4]$. Breakthrough has been achieved. Go to step 5.

Step 5. $N_2 = \{1, 2, 3, 4, 5\}$ and $f_2 = \min\{\infty, 20, 40, 10, 20\} = 10$. The residuals along the path of N_2 are

$$(c_{12}, c_{21}) = (20 - 10, 0 + 10) = (10, 10)$$
$$(c_{23}, c_{32}) = (40 - 10, 0 + 10) = (30, 10)$$
$$(c_{34}, c_{43}) = (10 - 10, 5 + 10) = (0, 15)$$
$$(c_{45}, c_{54}) = (20 - 10, 0 + 10) = (10, 10)$$

Iteration 3

Step 1. Set $a_1 = \infty$ and label node 1 with $[\infty, —]$. Set $i = 1$.

Step 2. $S_1 = \{2, 3, 4\}$.

Step 3. $k = 2$ and $a_2 = c_{12} = \max\{10, 10, 10\} = 10$. (Though ties are broken arbitrarily, TORA always selects the tied node with the smallest index. We will use this convention throughout the example.) Label node 2 with [10,1]. Set $i = 2$, and repeat step 2.

Step 2. $S_2 = \{3, 5\}$.

Step 3. $k = 3$ and $a_3 = c_{23} = 30$. Label node 3 with $[30, 2]$. Set $i = 3$, and repeat step 2.

Step 2. $S_3 = \emptyset$ (because $c_{34} = c_{35} = 0$). Go to step 4 to backtrack.

Step 3. *Backtracking.* The label $[30, 2]$ at node 3 gives the immediately preceding node $r = 2$. Remove node 3 from further consideration *in this iteration* by crossing it out. Set $i = r = 2$, and repeat step 2.

Step 2. $S_2 = \{5\}$ (note that node 3 has been removed in the backtracking step).

Step 3. $k = 5$ and $a_5 = c_{25} = 30$. Label node 5 with $[30, 2]$. Breakthrough has been achieved; go to step 5.

Step 5. $N_3 = \{1, 2, 5\}$ and $c_5 = \min\{\infty, 10, 30\} = 10$. The residuals along the path of N_3 are

$$(c_{12}, c_{21}) = (10 - 10, 10 + 10) = (0, 20)$$
$$(c_{25}, c_{52}) = (30 - 10, 0 + 10) = (20, 10)$$

Iteration 4. This iteration yields $N_4 = \{1, 3, 2, 5\}$ with $f_4 = 10$ (verify!).

Iteration 5. This iteration yields $N_5 = \{1, 4, 5\}$ with $f_5 = 10$ (verify!).

Iteration 6. All the arcs out of node 1 have zero residuals. Hence, no further breakthroughs are possible. We turn to step 6 to determine the solution.

Step 6. Maximal flow in the network is $F = f_1 + f_2 + \cdots + f_5 = 20 + 10 + 10 + 10 + 10 = 60$ units. The flow in the different arcs is computed by subtracting the last residuals (c_{ij}, c_{ji}) in iterations 6 from the initial capacities $(\overline{C}_{ij}, \overline{C}_{ji})$, as the following table shows.

Arc	$(\overline{C}_{ij}, \overline{C}_{ji}) - (c_{ij}, c_{ji})_6$	Flow amount	Direction
(1,2)	(20, 0) − (0, 20) = (20, −20)	20	$1 \rightarrow 2$
(1,3)	(30, 0) − (0, 30) = (30, −30)	30	$1 \rightarrow 3$
(1,4)	(10, 0) − (0, 10) = (10, −10)	10	$1 \rightarrow 4$
(2,3)	(40, 0) − (40, 0) = (0, 0)	0	—
(2,5)	(30, 0) − (10, 20) = (20, −20)	20	$2 \rightarrow 5$
(3,4)	(10, 5) − (0, 15) = (10, −10)	10	$3 \rightarrow 4$
(3,5)	(20, 0) − (0, 20) = (20, −20)	20	$3 \rightarrow 5$
(4,5)	(20, 0) − (0, 20) = (20, −20)	20	$4 \rightarrow 5$

TORA Moment

You can use TORA to solve the maximal flow model in an automated mode or to produce the iterations outlined above. From the SOLVE/MODIFY menu, select Solve Problem. After specifying the output format, go to the output screen and select either Maximum Flows or Iterations. File toraEx6.4-2.txt provides TORA's data for Example 6.4-2.

PROBLEM SET 6.4B

*1. In Example 6.4-2,
   - **(a)** Determine the surplus capacities for all the arcs.
   - **(b)** Determine the amount of flow through nodes 2, 3, and 4.
   - **(c)** Can the network flow be increased by increasing the capacities in the directions $3 \rightarrow 5$ and $4 \rightarrow 5$?

2. Determine the maximal flow and the optimum flow in each arc for the network in Figure 6.32.

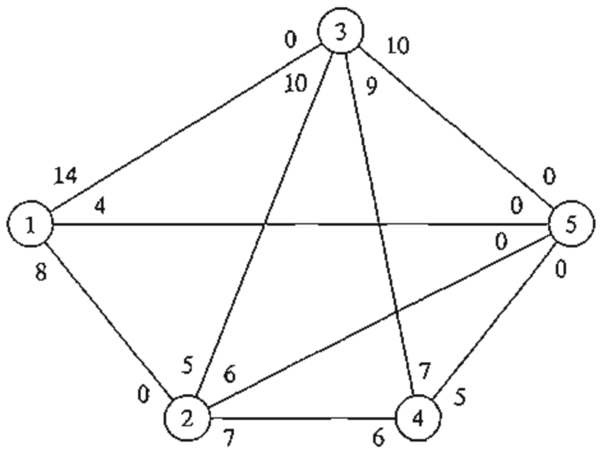


FIGURE 6.32
Network for Problem 2, Set 6.4b

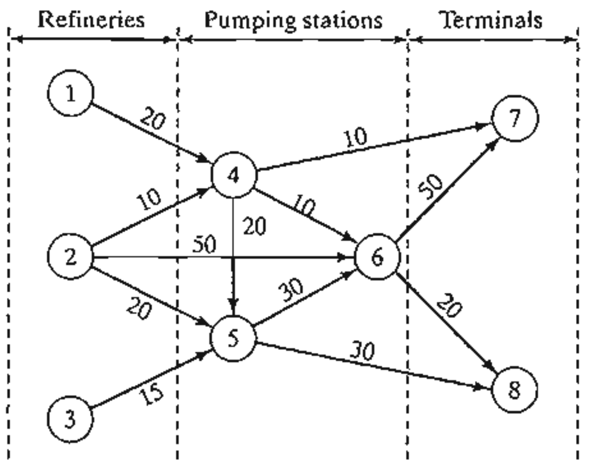


FIGURE 6.33
Network for Problem 3, Set 6.4b

3. Three refineries send a gasoline product to two distribution terminals through a pipeline network. Any demand that cannot be satisfied through the network is acquired from other sources. The pipeline network is served by three pumping stations, as shown in Figure 6.33. The product flows in the network in the direction shown by the arrows. The capacity of each pipe segment (shown directly on the arcs) is in million bbl per day. Determine the following:
   (a) The daily production at each refinery that matches the maximum capacity of the network.
   (b) The daily demand at each terminal that matches the maximum capacity of the network.
   (c) The daily capacity of each pump that matches the maximum capacity of the network.
4. Suppose that the maximum daily capacity of pump 6 in the network of Figure 6.33 is limited to 50 million bbl per day. Remodel the network to include this restriction. Then determine the maximum capacity of the network.
5. Chicken feed is transported by trucks from three silos to four farms. Some of the silos cannot ship directly to some of the farms. The capacities of the other routes are limited by the number of trucks available and the number of trips made daily. The following table shows the daily amounts of supply at the silos and demand at the farms (in thousands of pounds). The cell entries of the table specify the daily capacities of the associated routes.

		Farm				
		1	2	3	4	
	1	30	5	0	40	**20**
Silo	2	0	0	5	90	**20**
	3	100	40	30	40	**200**
		200	**10**	**60**	**20**	

   (a) Determine the schedule that satisfies the most demand.
   (b) Will the proposed schedule satisfy all the demand at the farms?

6. In Problem 5, suppose that transshipping is allowed between silos 1 and 2 and silos 2 and 3. Suppose also that transshipping is allowed between farms 1 and 2, 2 and 3, and 3 and 4. The maximum two-way daily capacity on the proposed transshipping routes is 50 (thousand) lb. What is the effect of transshipping on the unsatisfied demands at the farms?

*7. A parent has five (teenage) children and five household chores to assign to them. Past experience has shown that forcing chores on a child is counterproductive. With this in mind, the children are asked to list their preferences among the five chores, as the following table shows:

Child	Preferred chore
Rif	3, 4, or 5
Mai	1
Ben	1 or 2
Kim	1, 2, or 5
Ken	2

The parent's modest goal now is to finish as many chores as possible while abiding by the children's preferences. Determine the maximum number of chores that can be completed and the assignment of chores to children.

8. Four factories are engaged in the production of four types of toys. The following table lists the toys that can be produced by each factory.

Factory	Toy productions mix
1	1, 2, 3
2	2, 3
3	1, 4
4	3, 4

All toys require approximately the same per-unit labor and material. The daily capacities of the four factories are 250, 180, 300, and 100 toys, respectively. The daily demands for the four toys are 200, 150, 350, and 100 units, respectively. Determine the factories' production schedules that will most satisfy the demands for the four toys.

9. The academic council at the U of A is seeking representation from among six students who are affiliated with four honor societies. The academic council representation includes three areas: mathematics, art, and engineering. At most two students in each area can be on the council. The following table shows the membership of the six students in the four honor societies:

Society	Affiliated students
1	1, 2, 3
2	1, 3, 5
3	3, 4, 5
4	1, 2, 4, 6

The students who are skilled in the areas of mathematics, art, and engineering are shown in the following table:

Area	Skilled students
Mathematics	1, 2, 4
Art	3, 4
Engineering	4, 5, 6

A student who is skilled in more than one area must be assigned exclusively to one area only. Can all four honor societies be represented on the council?

10. *Maximal/minimal flow in networks with lower bounds.* The maximal flow algorithm given in this section assumes that all the arcs have zero lower bounds. In some models, the lower bounds may be strictly positive, and we may be interested in finding the maximal or minimal flow in the network (see case 6-3 in Appendix E). The presence of the lower bound poses difficulty because the network may not have a feasible flow at all. The objective of this exercise is to show that any maximal and minimal flow model with positive lower bounds can be solved using two steps.

Step 1. Find an initial feasible solution for the network with positive lower bounds.

Step 2. Using the feasible solution in step 1, find the maximal or minimal flow in the original network.

(a) Show that an arc (i, j) with flow limited by $l_{ij} \leq x_{ij} \leq u_{ij}$ can be represented equivalently by a *sink* with demand l_{ij} at node i and a *source* with supply l_{ij} at node j with flow limited by $0 \leq x_{ij} \leq u_{ij} - l_{ij}$.

(b) Show that finding a feasible solution for the original network is equivalent to finding the maximal flow x'_{ij} in the network after (1) modifying the bounds on x_{ij} to $0 \leq x'_{ij} \leq u_{ij} - l_{ij}$, (2) "lumping" all the resulting sources into one supersource with outgoing arc capacities l_{ij}, (3) "lumping" all the resulting sinks into one supersink with incoming arc capacities l_{ij}, and (4) connecting the terminal node t to the source node s in the original network by a return infinite-capacity arc. A feasible solution exists if the maximal flow in the new network equals the sum of the lower bounds in the original network. Apply the procedure to the following network and find a feasible flow solution:

Arc (i, j)	(l_{ij}, u_{ij})
(1, 2)	(5, 20)
(1, 3)	(0, 15)
(2, 3)	(4, 10)
(2, 4)	(3, 15)
(3, 4)	(0, 20)

(c) Use the feasible solution for the network in (b) together with the maximal flow algorithm to determine the *minimal* flow in the original network. (*Hint:* First compute the residue network given the initial feasible solution. Next, determine the maximum flow *from the end node to the start node.* This is equivalent to finding the maximum flow that should be canceled from the start node to the end node. Now, combining the feasible and maximal flow solutions yields the minimal flow in the original network.)

(d) Use the feasible solution for the network in (b) together with the *maximal* flow model to determine the maximal flow in the original network. (*Hint:* As in part (c), start with the residue network. Next apply the breakthrough algorithm to the resulting residue network exactly as in the regular maximal flow model.)

6.4.3 Linear Programming Formulation of Maximal Flow Mode

Define x_{ij} as the amount of flow in arc (i, j) with capacity C_{ij}. The objective is to determine x_{ij} for all i and j that will maximize the flow between start node s and terminal node t subject to flow restrictions (input flow = output flow) at all but nodes s and t.

Example 6.4-3

In the maximal flow model of Figure 6.29 (Example 6.4-2), $s = 1$ and $t = 5$. The following table summarizes the associated LP with two different, but equivalent, objective functions depending on whether we maximize the output from start node 1 ($= z_1$) or the input to terminal node 5 ($= z_2$).

	x_{12}	x_{13}	x_{14}	x_{23}	x_{25}	x_{34}	x_{35}	x_{43}	x_{45}	
Maximize z_1 =	1	1	1							
Maximize z_2 =					1		1		1	
Node 2	1			−1	−1					= 0
Node 3		1		1		−1	−1	1		= 0
Node 4			1			1		−1	−1	= 0
Capacity	20	30	10	40	30	10	20	5	20	

The optimal solution using either objective function is

$$x_{12} = 20, x_{13} = 30, x_{14} = 10, x_{25} = 20, x_{34} = 10, x_{35} = 20, x_{45} = 20$$

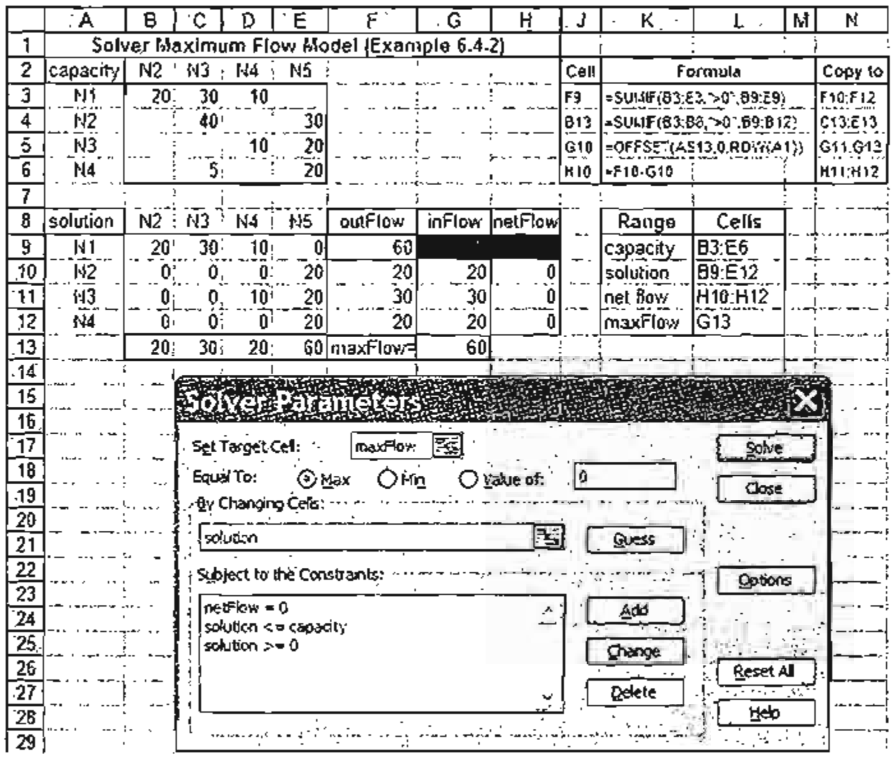
The associated maximum flow is $z_1 = z_2 = 60$.

Solver Moment

Figure 6.34 gives the Excel Solver model for the maximum flow model of Example 6.4-2 (file solverEx6.4-2.xls). The general idea of the model is similar to that used with the shortest-route model, which was detailed following Example 6.3-6. The main differences are: (1) there are no flow equations for the start node 1 and end node 5, and (2) the objective is to maximize the total outflow at start node 1 (F9) or, equivalently, the total inflow at terminal node 5 (G13). File solverEx6.4-2.xls uses G13 as the target cell. You are encouraged to execute the model with F9 as the target cell.

AMPL Moment

Figure 6.35 provides the AMPL model for the maximal flow problem. The data applies to Example 6.4-2 (file amplEx6.4-2.txt). The overall idea of determining the input and output flows at a node is similar to the one detailed following Example 6.3-6 of the shortest-route model (you will find it helpful to review files amplEx6.3-6a.txt and amplEx6.3-6b.txt first). However, because the model is designed to find the maximum flow between *any* two nodes, `start` and `end`, two additional constraints are needed to ensure that no flow *enters* `start` and no flow *leaves* `end`. Constraints `inStart` and



	A	B	C	D	E	F	G	H
1	Solver Maximum Flow Model (Example 6.4-2)							
2	capacity	N2	N3	N4	N5			
3	N1	20	30	10				
4	N2		40		30			
5	N3			10	20			
6	N4		5		20			
7								
8	solution	N2	N3	N4	N5	outFlow	inFlow	netFlow
9	N1	20	30	10	0	60		
10	N2	0	0	0	20	20	20	0
11	N3	0	0	10	20	30	30	0
12	N4	0	0	0	20	20	20	0
13		20	30	20	60	maxFlow=	60	

Cell	Formula	Copy to
F9	=SUMIF(B3:E3,">0",B9:E9)	F10:F12
B13	=SUMIF(B3:B6,">0",B9:B12)	C13:E13
G10	=OFFSET(A$13,0,ROW(A1))	G11:G12
H10	=F10-G10	H11:H12

Range	Cells
capacity	B3:E6
solution	B9:E12
net flow	H10:H12
maxFlow	G13

Solver Parameters

Set Target Cell: maxFlow

Equal To: [✓ Max] Min Value of: 0

By Changing Cells: solution

Subject to the Constraints:

netFlow = 0
solution <= capacity
solution >= 0

FIGURE 6.34

Excel Solver solution of the maximal flow model of 6.4-2 (file solverEx6.4-2.xls)

`outEnd` in the model ensure this result. These two constraints are not needed when `start=1` and `end=5` because the nature of the data guarantees the desired result. However, for `start=3`, node 3 allows both input and output flow (arcs 4-3 and 3-4) and, hence, constraint `inStart` is needed (try the model without `inStart`!).

The objective function maximizes the sum of the output flow at node `start`. Equivalently, we can choose to maximize the sum of the input flow at node `end`. The model can find the maximum flow between any two designated `start` and `end` nodes in the network.

PROBLEM SET 6.4C

1. Model each of the following problems as a linear program, then solve using Solver and AMPL.

(a) Problem 2, Set 6.4b.

(b) Problem 5, Set 6.4b

(c) Problem 9, Set 6.4b.

2. Jim lives in Denver, Colorado, and likes to spend his annual vacation in Yellowstone National Park in Wyoming. Being a nature lover, Jim tries to drive a different scenic route each year. After consulting the appropriate maps, Jim has represented his preferred routes between Denver (D) and Yellowstone (Y) by the network in Figure 6.36. Nodes 1 through 14 represent intermediate cities. Although driving distance is not an issue, Jim's stipulation is that selected routes between D and Y do not include any common cities.

```
#-------- Maximal Flow model (Example 6.4-2)---------
param n;
param start;
param end;
param c{i in 1..n, j in 1..n} default 0;

var x{i in 1..n,j in 1..n:c[i,j]>0}>=0,<=c[i,j];
var outFlow{i in 1..n}=sum{j in 1..n:c[i,j]>0}x[i,j];
var inFlow{i in 1..n}=sum{j in 1..n:c[j,i]>0}x[j,i];

maximize z: sum {j in 1..n:c[start,j]>0}x[start,j];
subject to
limit{i in 1..n:
     i<>start and i<>end}:outFlow[i]-inFlow[i]=0;
inStart:sum{i in 1..n:c[i,start]>0}x[i,start]=0;
outEnd:sum{j in 1..n:c[end,j]>0}x[end,j]=0;

data;
param n:=5;
param start:=1;
param end:=5;
param c:
  1  2  3  4  5 :=
1 .  20 30 10 0
2 .  .  40 0  30
3 .  .  0  10 20
4 .  .  5  .  20
5 .  .  .  .  .;

solve;
print "MaxFlow between nodes",start,"and",end, "=",z;
printf "Associated flows:\n";
for {i in 1..n-1} for {j in 2..n:c[i,j]>0}
     {if x[i,j]>0 then
      printf"(%2i-%2i)= %5.2f\n",i,j,x[i,j];} print;
```

FIGURE 6.35
AMPL model of the maximal flow problem of Example 6.4-2 (file amplEx6.4-2.txt)

Determine (using AMPL and Solver) all the distinct routes available to Jim. (*Hint:* Modify the maximal flow LP model to determine the maximum number of unique paths between *D* and *Y*.)

3. (Guéret and Associates, 2002, Section 12.1) A military telecommunication system connecting 9 sites is given in Figure 6.37. Sites 4 and 7 must continue to communicate even if as many as three other sites are destroyed by enemy actions. Does the present communication network meet this requirement? Use AMPL and Solver to work out the problem.

6.5 CPM AND PERT

CPM (Critical Path Method) and PERT (Program Evaluation and Review Technique) are network-based methods designed to assist in the planning, scheduling, and control of projects. A project is defined as a collection of interrelated activities with each activity

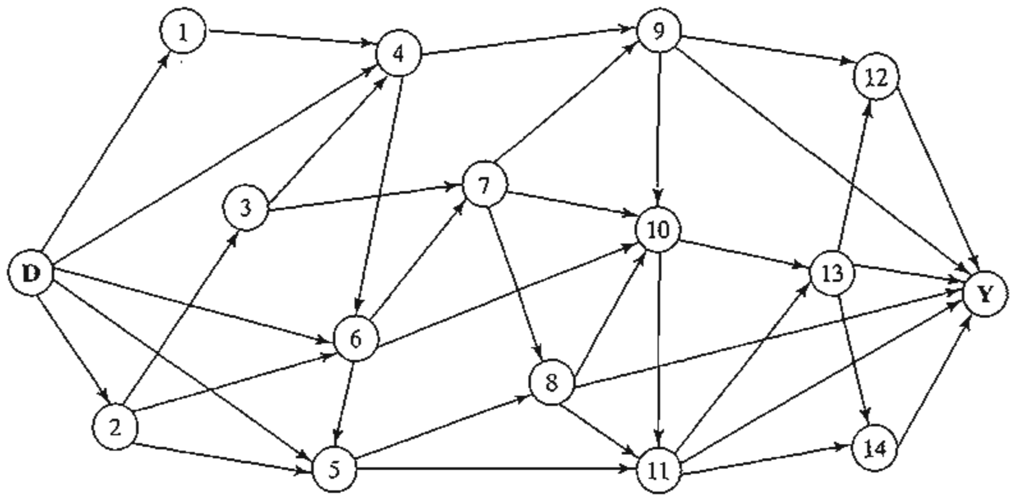


FIGURE 6.36
Network for Problem 2, Set 6.4c

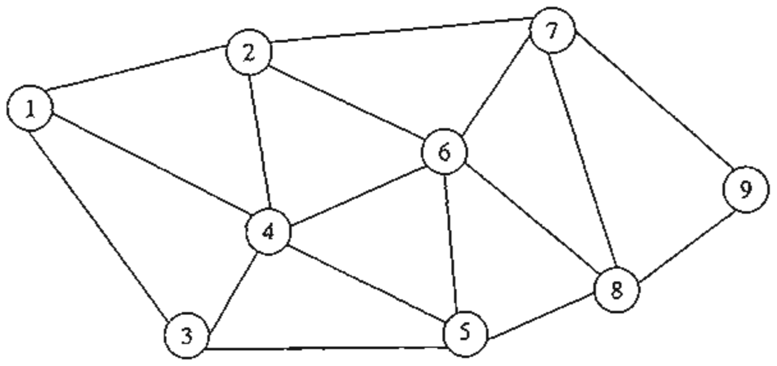


FIGURE 6.37
Network for Problem 3, Set 6.4c

consuming time and resources. The objective of CPM and PERT is to provide analytic means for scheduling the activities. Figure 6.38 summarizes the steps of the techniques. First, we define the activities of the project, their precedence relationships, and their time requirements. Next, the precedence relationships among the activities are represented by a network. The third step involves specific computations to develop the time schedule for the project. During the actual execution of the project things may not proceed as planned, as some of the activities may be expedited or delayed. When this happens, the schedule must be revised to reflect the realities on the ground. This is the reason for including a feedback loop between the time schedule phase and the network phase, as shown in Figure 6.38.

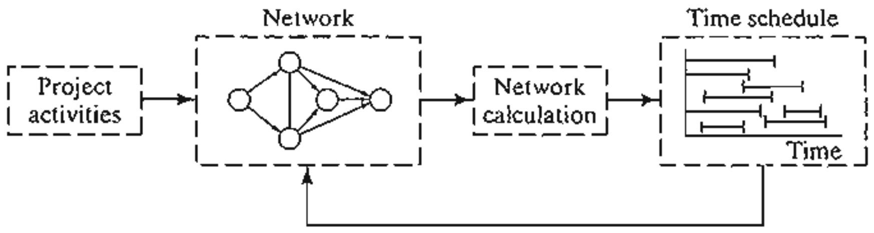


FIGURE 6.38
Phases for project planning with CPM-PERT

The two techniques, CPM and PERT, which were developed independently, differ in that CPM assumes deterministic activity durations and PERT assumes probabilistic durations. This presentation will start with CPM and then proceed with the details of PERT.

6.5.1 Network Representation

Each activity of the project is represented by an arc pointing in the direction of progress in the project. The nodes of the network establish the precedence relationships among the different activities.

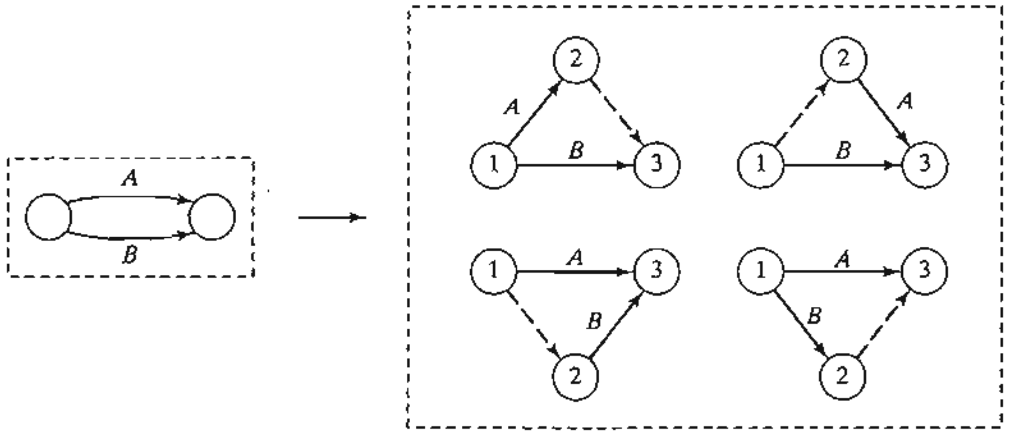
Three rules are available for constructing the network.

Rule 1. *Each activity is represented by one, and only one, arc.*

Rule 2. *Each activity must be identified by two distinct end nodes.*

Figure 6.39 shows how a dummy activity can be used to represent two concurrent activities, A and B. By definition, a dummy activity, which normally is depicted by a

FIGURE 6.39
Use of dummy activity to produce unique representation of concurrent activities



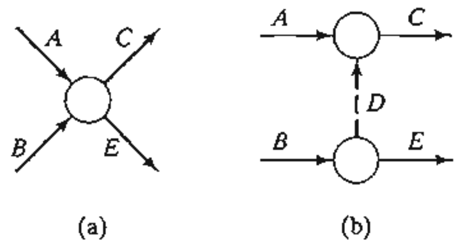


FIGURE 6.40
Use of dummy activity to ensure correct precedence relationship

dashed arc, consumes no time or resources. Inserting a dummy activity in one of the four ways shown in Figure 6.39, we maintain the concurrence of A and B, and provide unique end nodes for the two activities (to satisfy rule 2).

Rule 3. *To maintain the correct precedence relationships, the following questions must be answered as each activity is added to the network:*

(a) *What activities must immediately precede the current activity?*

(b) *What activities must follow the current activity?*

(c) *What activities must occur concurrently with the current activity?*

The answers to these questions may require the use of dummy activities to ensure correct precedences among the activities. For example, consider the following segment of a project:

1. Activity C starts immediately after A and B have been completed.
2. Activity E starts only after B has been completed.

Part (a) of Figure 6.40 shows the incorrect representation of the precedence relationship because it requires both A and B to be completed before E can start. In part (b), the use of a dummy activity rectifies the situation.

Example 6.5-1

A publisher has a contract with an author to publish a textbook. The (simplified) activities associated with the production of the textbook are given below. The author is required to submit to the publisher a hard copy and a computer file of the manuscript. Develop the associated network for the project.

Activity	Predecessor(s)	Duration (weeks)
A: Manuscript proofreading by editor	—	3
B: Sample pages preparation	—	2
C: Book cover design	—	4
D: Artwork preparation	—	3
E: Author's approval of edited manuscript and sample pages	A, B	2
F: Book formatting	E	4
G: Author's review of formatted pages	F	2
H: Author's review of artwork	D	1
I: Production of printing plates	G, H	2
J: Book production and binding	C, I	4

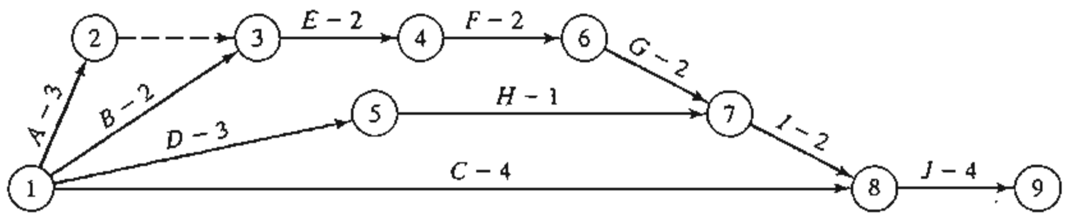


FIGURE 6.41
Project network for Example 6.5-1

Figure 6.41 provides the network describing the precedence relationships among the different activities. Dummy activity (2, 3) produces unique end nodes for concurrent activities *A* and *B*. It is convenient to number the nodes in ascending order in the direction of progress in the project.

PROBLEM SET 6.5A

1. Construct the project network comprised of activities *A* to *L* with the following precedence relationships:
   (a) A, B, and C, the first activities of the project, can be executed concurrently.
   (b) A and *B* precede *D*.
   (c) B precedes *E*, *F*, and *H*.
   (d) F and *C* precede *G*.
   (e) E and *H* precede *I* and *J*.
   (f) C, D, F, and *J* precede *K*.
   (g) K precedes *L*.
   (h) I, G, and *L* are the terminal activities of the project.
2. Construct the project network comprised of activities *A* to *P* that satisfies the following precedence relationships:
   (a) A, B, and C, the first activities of the project, can be executed concurrently.
   (b) D, E, and *F* follow *A*.
   (c) I and *G* follow both *B* and *D*.
   (d) H follows both *C* and *G*.
   (e) K and *L* follow *I*.
   (f) J succeeds both *E* and *H*.
   (g) M and *N* succeed *F*, but cannot start until both *E* and *H* are completed.
   (h) O succeeds *M* and *I*.
   (i) P succeeds *J*, *L*, and *O*.
   (j) K, N, and *P* are the terminal activities of the project.

*3. The footings of a building can be completed in four consecutive sections. The activities for each section include (1) digging, (2) placing steel, and (3) pouring concrete. The digging of one section cannot start until that of the preceding section has been completed. The same restriction applies to pouring concrete. Develop the project network.

4. In Problem 3, suppose that 10% of the plumbing work can be started simultaneously with the digging of the first section but before any concrete is poured. After each section of the

footings is completed, an additional 5% of the plumbing can be started provided that the preceding 5% portion is complete. The remaining plumbing can be completed at the end of the project. Construct the project network.

5. An opinion survey involves designing and printing questionnaires, hiring and training personnel, selecting participants, mailing questionnaires, and analyzing the data. Construct the project network, stating all assumptions.

6. The activities in the following table describe the construction of a new house. Construct the associated project network.

	Activity	Predecessor(s)	Duration (days)
A:	Clear site	—	1
B:	Bring utilities to site	—	2
C:	Excavate	*A*	1
D:	Pour foundation	*C*	2
E:	Outside plumbing	*B*, *C*	6
F:	Frame house	*D*	10
G:	Do electric wiring	*F*	3
H:	Lay floor	*G*	1
I:	Lay roof	*F*	1
J:	Inside plumbing	*E*, *H*	5
K:	Shingling	*I*	2
L:	Outside sheathing insulation	*F*, *J*	1
M:	Install windows and outside doors	*F*	2
N:	Do brick work	*L*, *M*	4
O:	Insulate walls and ceiling	*G*, *J*	2
P:	Cover walls and ceiling	*O*	2
Q:	Insulate roof	*I*, *P*	1
R:	Finish interior	*P*	7
S:	Finish exterior	*I*, *N*	7
T:	Landscape	*S*	3

7. A company is in the process of preparing a budget for launching a new product. The following table provides the associated activities and their durations. Construct the project network.

	Activity	Predecessor(s)	Duration (days)
A:	Forecast sales volume	—	10
B:	Study competitive market	—	7
C:	Design item and facilities	*A*	5
D:	Prepare production schedule	*C*	3
E:	Estimate cost of production	*D*	2
F:	Set sales price	*B*, *E*	1
G:	Prepare budget	*E*, *F*	14

8. The activities involved in a candlelight choir service are listed in the following table. Construct the project network.

	Activity	Predecessor(s)	Duration (days)
A:	Select music	—	2
B:	Learn music	*A*	14

C:	Make copies and buy books	*A*	14
D:	Tryouts	*B*, *C*	3
E:	Rehearsals	*D*	70
F:	Rent candelabra	*D*	14
G:	Decorate candelabra	*F*	1
H:	Set up decorations	*D*	1
I:	Order choir robe stoles	*D*	7
J:	Check out public address system	*D*	7
K:	Select music tracks	*J*	14
L:	Set up public address system	*K*	1
M:	Final rehearsal	*E*, *G*, *L*	1
N:	Choir party	*H*, *L*, *M*	1
O:	Final program	*I*, *N*	1

9. The widening of a road section requires relocating ("reconductoring") 1700 feet of 13.8-kV overhead primary line. The following table summarizes the activities of the project. Construct the associated project network.

	Activity	Predecessor(s)	Duration (days)
A:	Job review	—	1
B:	Advise customers of temporary outage	*A*	$\frac{1}{2}$
C:	Requisition stores	*A*	1
D:	Scout job	*A*	$\frac{1}{2}$
E:	Secure poles and material	*C*, *D*	3
F:	Distribute poles	*E*	$3\frac{1}{2}$
G:	Pole location coordination	*D*	$\frac{1}{2}$
H:	Re-stake	*G*	$\frac{1}{2}$
I:	Dig holes	*H*	3
J:	Frame and set poles	*F*, *I*	4
K:	Cover old conductors	*F*, *I*	1
L:	Pull new conductors	*J*, *K*	2
M:	Install remaining material	*L*	2
N:	Sag conductor	*L*	2
O:	Trim trees	*D*	2
P:	De-energize and switch lines	*B*, *M*, *N*, *O*	$\frac{1}{10}$
Q:	Energize and switch new line	*P*	$\frac{1}{2}$
R:	Clean up	*Q*	1
S:	Remove old conductor	*Q*	1
T:	Remove old poles	*S*	2
U:	Return material to stores	*R*, *T*	2

10. The following table gives the activities for buying a new car. Construct the project network.

	Activity	Predecessor(s)	Duration (days)
A:	Conduct feasibility study	—	3
B:	Find potential buyer for present car	*A*	14
C:	List possible models	*A*	1
D:	Research all possible models	*C*	3
E:	Conduct interview with mechanic	*C*	1
F:	Collect dealer propaganda	*C*	2
G:	Compile pertinent data	*D*, *E*, *F*	1
H:	Choose top three models	*G*	1

I:	Test-drive all three choices	*H*	3
J:	Gather warranty and financing data	*H*	2
K:	Choose one car	*I, J*	2
L:	Choose dealer	*K*	2
M:	Search for desired color and options	*L*	4
N:	Test-drive chosen model once again	*L*	1
O:	Purchase new car	*B, M, N*	3

6.5.2 Critical Path (CPM) Computations

The end result in CPM is the construction of the time schedule for the project (see Figure 6.38). To achieve this objective conveniently, we carry out special computations that produce the following information:

1. Total duration needed to complete the project.
2. Classification of the activities of the project as *critical* and *noncritical*.

An activity is said to be **critical** if there is no "leeway" in determining its start and finish times. A **noncritical** activity allows some scheduling slack, so that the start time of the activity can be advanced or delayed within limits without affecting the completion date of the entire project.

To carry out the necessary computations, we define an **event** as a point in time at which activities are terminated and others are started. In terms of the network, an event corresponds to a node. Define

$\square_j$ = Earliest occurrence time of event j

Δ_j = Latest occurrence time of event j

D_{ij} = Duration of activity (i, j)

The definitions of the *earliest* and *latest* occurrences of event j are specified relative to the start and completion dates of the entire project.

The critical path calculations involve two passes: The **forward pass** determines the *earliest* occurrence times of the events, and the **backward pass** calculates their *latest* occurrence times.

Forward Pass (Earliest Occurrence Times, $\square$). The computations start at node 1 and advance recursively to end node n.

Initial Step. Set $\square_1 = 0$ to indicate that the project starts at time 0.

General Step *j*. Given that nodes $p, q, \ldots,$ and v are linked *directly* to node j by incoming activities $(p, j), (q, j), \ldots,$ and (v, j) and that the earliest occurrence times of events (nodes) $p, q, \ldots,$ and v have already been computed, then the earliest occurrence time of event j is computed as

$$\square_j = \max\{\square_p + D_{pj}, \square_q + D_{qj}, \ldots, \square_v + D_{vj}\}$$

The forward pass is complete when $\square_n$ at node n has been computed. By definition $\square_j$ represents the longest path (duration) to node j.

Backward Pass (Latest Occurrence Times, Δ). Following the completion of the forward pass, the backward pass computations start at node n and end at node 1.

Initial Step. Set $\Delta_n = \square_n$ to indicate that the earliest and latest occurrences of the last node of the project are the same.

General Step *j*. Given that nodes $p, q, \ldots$, and v are linked *directly* to node j by *outgoing* activities $(j, p), (j, q), \ldots$, and (j, v) and that the latest occurrence times of nodes $p, q, \ldots$, and v have already been computed, the latest occurrence time of node j is computed as

$$\Delta_j = \min\{\Delta_p - D_{jp}, \Delta_q - D_{jq}, \ldots, \Delta_v - D_{jv}\}$$

The backward pass is complete when Δ_1 at node 1 is computed. At this point, $\Delta_1 = \square_1 (= 0)$.

Based on the preceding calculations, an activity (i, j) will be *critical* if it satisfies three conditions.

1. $\Delta_i = \square_i$
2. $\Delta_j = \square_j$
3. $\Delta_j - \Delta_i = \square_j - \square_i = D_{ij}$

The three conditions state that the earliest and latest occurrence times of end nodes i and j are equal and the duration D_{ij} fits "tightly" in the specified time span. An activity that does not satisfy all three conditions is thus *noncritical.*

By definition, the critical activities of a network must constitute an uninterrupted path that spans the entire network from start to finish.

Example 6.5-2

Determine the critical path for the project network in Figure 6.42. All the durations are in days.

Forward Pass

Node 1. Set $\square_1 = 0$

Node 2. $\square_2 = \square_1 + D_{12} = 0 + 5 = 5$

Node 3. $\square_3 = \max\{\square_1 + D_{13}, \square_2 + D_{23}\} = \max\{0 + 6, 5 + 3\} = 8$

Node 4. $\square_4 = \square_2 + D_{24} = 5 + 8 = 13$

Node 5. $\square_5 = \max\{\square_3 + D_{35}, \square_4 + D_{45}\} = \max\{8 + 2, 13 + 0\} = 13$

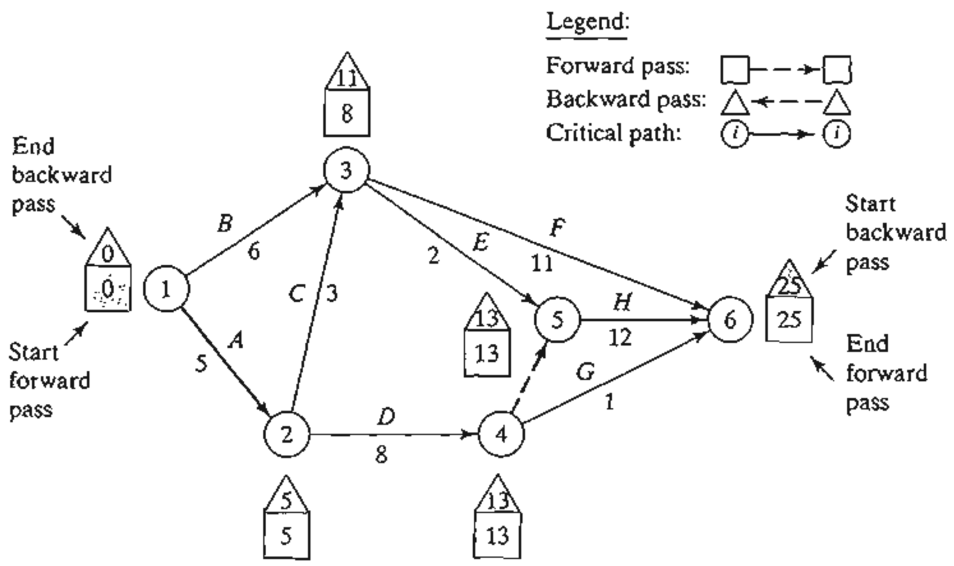


FIGURE 6.42
Forward and backward pass calculations for the project of Example 6.5-2

Node 6. $\square_6 = \max\{\square_3 + D_{36}, \square_4 + D_{46}, \square_5 + D_{56}\}$
$= \max\{8 + 11, 13 + 1, 13 + 12\} = 25$

The computations show that the project can be completed in 25 days.

Backward Pass

Node 6. Set $\Delta_6 = \square_6 = 25$
Node 5. $\Delta_5 = \Delta_6 - D_{56} = 25 - 12 = 13$
Node 4. $\Delta_4 = \min\{\Delta_6 - D_{46}, \Delta_5 - D_{45}\} = \min\{25 - 1, 13 - 0\} = 13$
Node 3. $\Delta_3 = \min\{\Delta_6 - D_{36}, \Delta_5 - D_{35}\} = \min\{25 - 11, 13 - 2\} = 11$
Node 2. $\Delta_2 = \min\{\Delta_4 - D_{24}, \Delta_3 - D_{23}\} = \min\{13 - 8, 11 - 3\} = 5$
Node 1. $\Delta_1 = \min\{\Delta_3 - D_{13}, \Delta_2 - D_2\} = \min\{11 - 6, 5 - 5\} = 0$

Correct computations will always end with $\Delta_1 = 0$.

The forward and backward pass computations can be made directly on the network as shown in Figure 6.42. Applying the rules for determining the critical activities, the critical path is $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$, which, as should be expected, spans the network from start (node 1) to finish (node 6). The sum of the durations of the critical activities [(1, 2), (2, 4), (4, 5), and (5, 6)] equals the duration of the project (= 25 days). Observe that activity (4, 6) satisfies the first two conditions for a critical activity ($\Delta_4 = \square_4 = 13$ and $\Delta_5 = \square_5 = 25$) but not the third ($\square_6 - \square_4 \neq D_{46}$). Hence, the activity is noncritical.

PROBLEM SET 6.5B

*1. Determine the critical path for the project network in Figure 6.43.
2. Determine the critical path for the project networks in Figure 6.44.

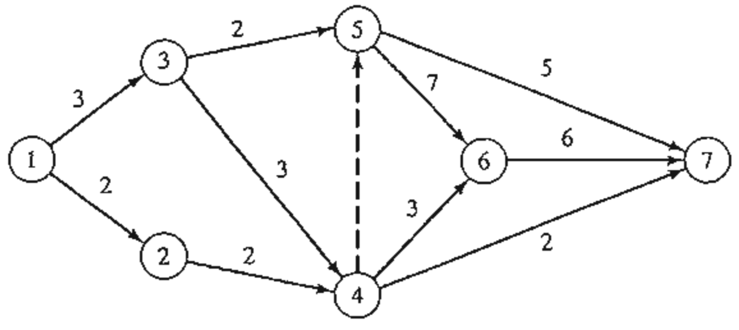
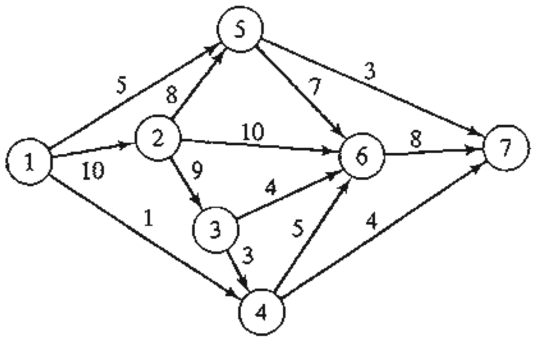


FIGURE 6.43
Project networks for Problem 1, Set 6.5b



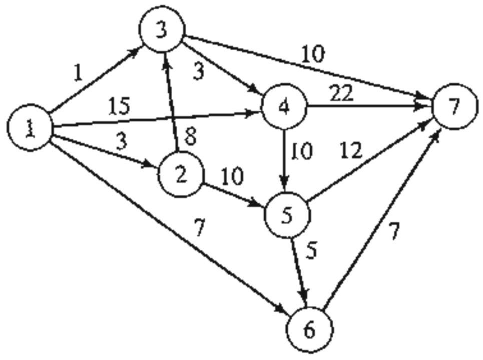


FIGURE 6.44
Project network for Problem 2, Set 6.5b

3. Determine the critical path for the project in Problem 6, Set6.5a.
4. Determine the critical path for the project in Problem 8, Set 6.5a.
5. Determine the critical path for the project in Problem 9, Set 6.5a.
6. Determine the critical path for the project in Problem 10, Set 6.5a.

6.5.3 Construction of the Time Schedule

This section shows how the information obtained from the calculations in Section 6.5.2 can be used to develop the time schedule. We recognize that for an activity (i, j), $\square_i$ represents the *earliest start time*, and Δ_j represents the *latest completion time*. This means that the interval $(\square_i, \Delta_j)$ delineates the (maximum) span during which activity (i, j) may be scheduled without delaying the entire project.

Construction of Preliminary Schedule. The method for constructing a preliminary schedule is illustrated by an example.

Example 6.5-3

Determine the time schedule for the project of Example 6.5-2 (Figure 6.42).

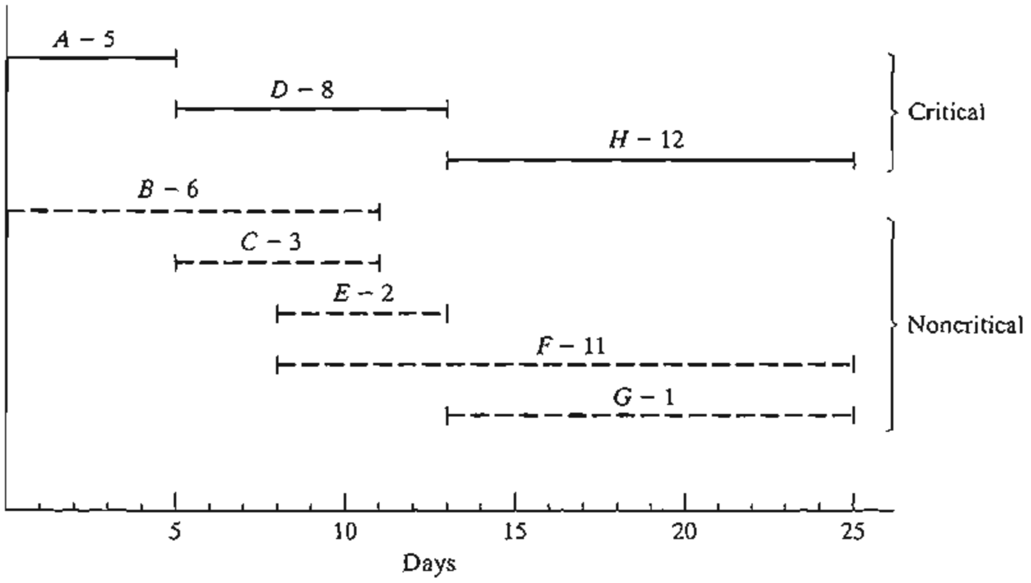


FIGURE 6.45
Preliminary schedule for the project of Example 6.5-2

We can get a preliminary time schedule for the different activities of the project by delineating their respective time spans as shown in Figure 6.45. Two observations are in order.

1. The critical activities (shown by solid lines) must be stacked one right after the other to ensure that the project is completed within its specified 25-day duration.
2. The noncritical activities (shown by dashed lines) have time spans that are larger than their respective durations, thus allowing slack (or "leeway") in scheduling them within their allotted time intervals.

How should we schedule the noncritical activities within their respective spans? Normally, it is preferable to start each noncritical activity as early as possible. In this manner, slack periods will remain opportunely available at the end of the allotted span where they can be used to absorb unexpected delays in the execution of the activity. It may be necessary, however, to delay the start of a noncritical activity past its earliest start time. For example, in Figure 6.45, suppose that each of the noncritical activities E and F requires the use of a bulldozer, and that only one is available. Scheduling both E and F as early as possible requires two bulldozers between times 8 and 10. We can remove the overlap by starting E at time 8 and pushing the start time of F to somewhere between times 10 and 14.

If all the noncritical activities can be scheduled as early as possible, the resulting schedule automatically is feasible. Otherwise, some precedence relationships may be violated if noncritical activities are delayed past their earliest time. Take for example activities C and E in Figure 6.45. In the project network (Figure 6.42), though C must be completed before E, the spans of C and E in Figure 6.45 allow us to schedule C between times 6 and 9, and E between times 8 and 10, which violates the requirement that C precede E. The need for a "red flag" that automatically reveals schedule conflict is thus evident. Such information is provided by computing the *floats* for the noncritical activities.

Determination of the Floats. Floats are the slack times available within the allotted span of the noncritical activity. The most common are the **total float** and the **free float.**

Figure 6.46 gives a convenient summary for computing the total float (TF_{ij}) and the free float (FF_{ij}) for an activity (i, j). The total float is the excess of the time span defined from the *earliest* occurrence of event i to the *latest* occurrence of event j over the duration of (i, j)—that is,

$$TF_{ij} = \Delta_j - \square_i - D_{ij}$$

The free float is the excess of the time span defined from the *earliest* occurrence of event i to the *earliest* occurrence of event j over the duration of (i, j)—that is,

$$FF_{ij} = \square_j - \square_i - D_{ij}$$

By definition, $FF_{ij} \leq TF_{ij}$.

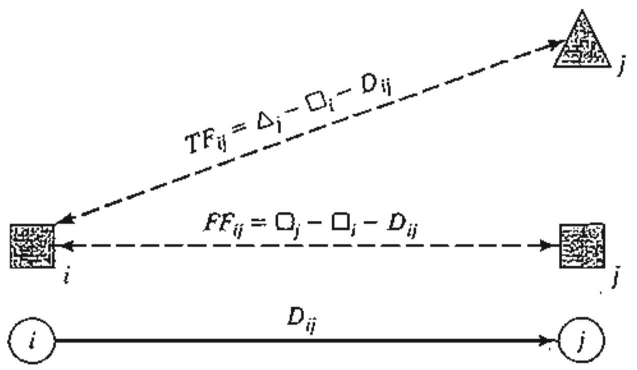
Red-Flagging Rule. *For a noncritical activity (i, j)*

- **(a)** *If $FF_{ij} = TF_{ij}$, then the activity can be scheduled anywhere within its $(\square_i, \Delta_j)$ span without causing schedule conflict.*
- **(b)** *If $FF_{ij} < TF_{ij}$, then the start of the activity can be delayed by at most FF_{ij} relative to its earliest start time ($\square_i$) without causing schedule conflict. Any delay larger than FF_{ij} (but not more than TF_{ij}) must be coupled with an equal delay relative to $\square_j$ in the start time of all the activities leaving node j.*

The implication of the rule is that a noncritical activity (i, j) will be red-flagged if its $FF_{ij} < TF_{ij}$. This red flag is important only if we decide to delay the start of the activity past its earliest start time, $\square_i$, in which case we must pay attention to the start times of the activities leaving node j to avoid schedule conflicts.

FIGURE 6.46

Computation of total and free floats


$TF_{ij} = \Delta_j - \square_i - D_{ij}$
$FF_{ij} = \square_j - \square_i - D_{ij}$
D_{ij}
i
j
i
j
j

Example 6.5-4

Compute the floats for the noncritical activities of the network in Example 6.5-2, and discuss their use in finalizing a schedule for the project.

The following table summarizes the computations of the total and free floats. It is more convenient to do the calculations directly on the network using the procedure in Figure 6.42.

Noncritical activity	Duration	Total float (TF)	Free float (FF)
$B(1,3)$	6	$11 - 0 - 6 = 5$	$8 - 0 - 6 = 2$
$C(2,3)$	3	$11 - 5 - 3 = 3$	$8 - 5 - 3 = 0$
$E(3,5)$	2	$13 - 8 - 2 = 3$	$13 - 8 - 2 = 3$
$F(3,6)$	11	$25 - 8 - 11 = 6$	$25 - 8 - 11 = 6$
$G(4,6)$	1	$25 - 13 - 1 = 11$	$25 - 13 - 1 = 11$

The computations red-flag activities B and C because their $FF < TF$. The remaining activities (E, F, and G) have $FF = TF$, and hence may be scheduled anywhere between their earliest start and latest completion times.

To investigate the significance of the red-flagged activities, consider activity B. Because its $TF = 5$ days, this activity can start as early as time 0 or as late as time 5 (see Figure 6.45). However, because its $FF = 2$ days, starting B anywhere between time 0 and time 2 will have no effect on the succeeding activities E and F. If, however, activity B must start at time $2 + d$ (≤ 5), then the start times of the immediately succeeding activities E and F must be pushed forward past their earliest start time ($= 8$) by at least d. In this manner, the precedence relationship between B and its successors E and F is preserved.

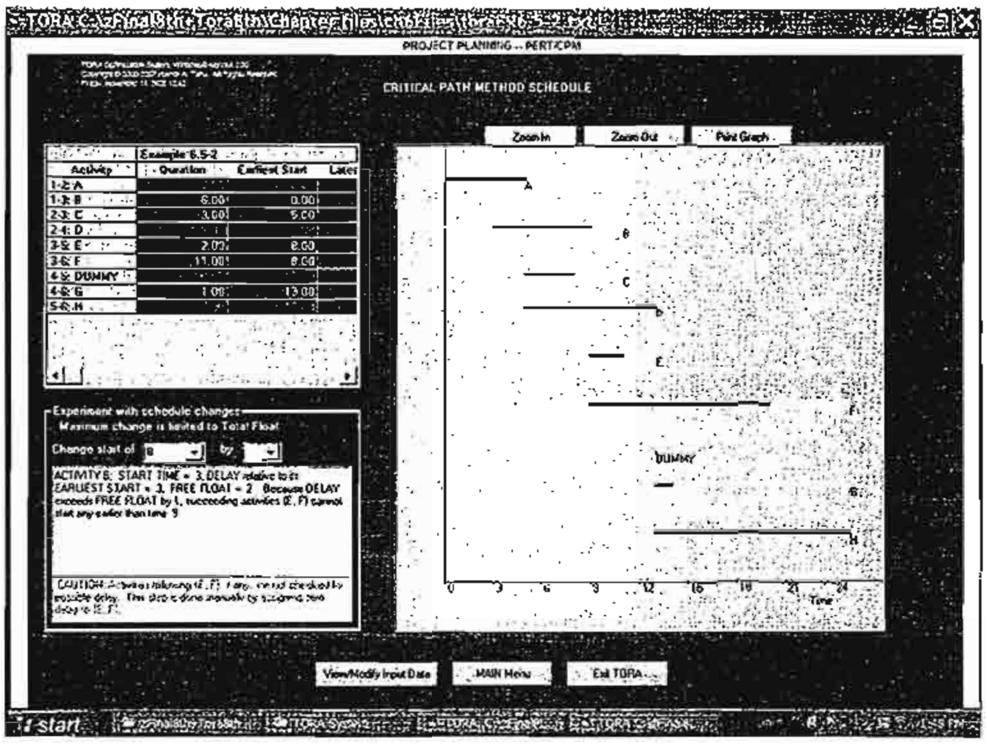
Turning to red-flagged activity C, we note that its $FF = 0$. This means that *any* delay in starting C past its earliest start time ($= 5$) must be coupled with at least an equal delay in the start of its successor activities E and F.

TORA Moment

TORA provides useful tutorial tools for CPM calculations and for constructing the time schedule. To use these tools, select Project Planning $\Rightarrow$ CPM-Critical Path Method from Main Menu. In the output screen, you have the option to select CPM Calculations to produce step-by-step computations of the forward pass, backward pass, and the floats or CPM Bar Chart to construct and experiment with the time schedule.

File toraEx6.5-2.txt provides TORA's data Example 6.5-2. If you elect to generate the output using the Next Step option, TORA will guide you through the details of the forward and backward pass calculations.

Figure 6.47 provides the TORA schedule produced by CPM Bar Chart option for the project of Example 6.5-2. The default bar chart automatically schedules all noncritical activities as early as possible. You can study the impact of delaying the start time of a noncritical activity by using the self-explanatory drop-down lists inside the bottom left frame of the screen. The impact of a delay of a noncritical activity will be shown directly on the bar chart together with an explanation. For example, if you delay the start of activity B by more than 2 time units, the succeeding activities E and F will be



Example 6.5-2		
Activity	Duration	Earliest Start
1-2: A	[illegible]	[illegible]
1-3: B	6.00	0.00
2-3: C	3.00	5.00
2-4: D	[illegible]	[illegible]
3-5: E	2.00	8.00
3-6: F	11.00	8.00
4-5: DUMMY	[illegible]	[illegible]
4-6: G	1.00	13.00
5-6: H	[illegible]	[illegible]

Experiment with schedule changes
Maximum change is limited to Total Float
Change start of B by

ACTIVITY B: START TIME = 3, DELAY relative to its EARLIEST START = 3, FREE FLOAT = 2. Because DELAY exceeds FREE FLOAT by 1, succeeding activities (E, F) cannot start any earlier than time 9

CAUTION: Activities following (E, F), if any, are not checked for possible delay. This can be done manually by assigning zero delay to (E, F).

FIGURE 6.47

TORA bar chart output for Example 6.5-2 (file toraEx6.5-2.txt)

delayed by an amount equal to the difference between the delay over the free float of activity B. Specifically, given that the free float for B is 2 time units, if B is delayed by 3 time units, then E and F must be delayed by at least $3 - 2 = 1$ time unit. This situation is demonstrated in Figure 6.47.

AMPL Moment

Figure 6.48 provides the AMPL model for the CPM (file amplEx6.52.txt). The model is driven by the data of Example 6.5-2. This AMPL model is a unique application because it uses *indexed sets* (see Section A.4) and requires no optimization. In essence, no `solve` command is needed, and AMPL is implemented as a pure programming language similar to Basic or C.

The nature of the computations in CPM requires representing the network by associating two *indexed* sets with each node: `into` and `from`. For node `i`, the set `into[i]` defines all the nodes that feed into node `i`, and the set `from[i]` defines all the nodes that are reached from node `i`. For example, in Example 6.5-2, `from[1] = {2,3}` and `into[1]` is empty.

The determination of subsets `from` and `into` is achieved in the model as follows: Because `D[i,j]` can be zero when a CPM network uses dummy activities, the default value for `D[i,j]` is -1 for all nonexisting arcs. Thus, the set `from[i]` represents all the nodes `j` in the set `{1..n}` that can be reached from node `i`, which can happen only if `D[i,j]>=0`. This says that `from[i]` is defined by the subset `{j in 1..n:D[i,j]> =0}`.

```
#--------- CPM (Example 6.5-2)-----------
param n;
param D{1..n,1..n} default -1;

set into{1..n};
set from{1..n};

var x{i in 1..n,j in from[i]}>=0;
var ET{i in 1..n};
var LT{i in 1..n};
var TF{i in 1..n, j in from[i]};
var FF{i in 1..n, j in from[i]};

data;
param n:=6;
param D: 1 2 3 4 5 6:=
1 . 5 6 . . .
2 . . 3 8 . .
3 . . . . 2 11
4 . . . . 0 1
5 . . . . . 12
6 . . . . . .;

for {i in 1..n} {let from[i]:={j in 1..n:D[i,j]>=0}};
for {j in 1..n} {let into[j]:={i in 1..n:D[i,j]>=0}};

#------nodes earliest and latest times and floats
let ET[1]:=0;           #earliest node time
for {i in 2..n}let ET[i]:=max{j in into[i]}(ET[j]+D[j,i]);

let LT[n]:=ET[n];       #latest node time
for{i in n-1..1 by -1}let LT[i]:=min{j in from[i]}(LT[j]-D[i,j]);

printf "%1s-%1s %5s %5s %5s %5s %5s %5s %5s \n\n",
     "i","j","D","ES","EC","LS","LC","TF","FF" >Ex6.6-2out.txt;
for {i in 1..n, j in from[i]}
{
let TF[i,j]:=LT[j]-ET[i]-D[i,j];
let FF[i,j]:=ET[j]-ET[i]-D[i,j];
printf "%1i-%1i %5i %5i %5i %5i %5i %5i %5i %3s\n",
  i,j,D[i,j],ET[i],ET[i]+D[i,j],LT[j]-D[i,j],LT[j],TF[i,j],FF[i,j],
  if TF[i,j]=0 then "c" else "" >Ex6.6-2out.txt;
}
```

FIGURE 6.48
AMPL model for Example 6.5-2 (file amplEx6.5-2.txt)

Similar reasoning applies to the determination of subsets `into[i]`. The following AMPL statements automate the determination of these sets and must follow the `D[i,j]` data, as shown in Figure 6.48:

```
for {i in 1..n} {let from[i]:={j in 1..n:D[i,j]>=0}};
for {j in 1..n} {let into[j]:={i in 1..n:D[i,j]>=0}};
```

Once the sets `from` and `into` have been determined, the model goes through the forward pass to compute the earliest time, `ET[i]`. With the completion of this pass, we can initiate the backward pass by using

```
let LT[n]:=ET[n];
```

The rest of the model is needed to obtain the output shown in Figure 6.49. This output determines all the data needed to construct the CPM chart. The logic of this segment is based on the computations given in Examples 6.5-2 and 6.5-4.

PROBLEM SET 6.5C

1. Given an activity (i, j) with duration D_{ij} and its earliest start time $\square_i$ and its latest completion time Δ_j, determine the earliest completion and the latest start times of (i, j).
2. What are the total and free floats of a critical activity? Explain.

*3. For each of the following activities, determine the maximum delay in the starting time relative to its earliest start time that will allow all the immediately succeeding activities to be scheduled anywhere between their earliest and latest completion times.
   (a) $TF = 10, FF = 10, D = 4$
   (b) $TF = 10, FF = 5, D = 4$
   (c) $TF = 10, FF = 0, D = 4$

4. In Example 6.5-4, use the floats to answer the following:
   (a) If activity B is started at time 1, and activity C is started at time 5, determine the earliest start times for E and F.
   (b) If activity B is started at time 3, and activity C is started at time 7, determine the earliest start times for E and F.
   (c) How is the scheduling of other activities impacted if activity B starts at time 6?

*5. In the project of Example 6.5-2 (Figure 6.42), assume that the durations of activities B and F are changed from 6 and 11 days to 20 and 25 days, respectively.
   (a) Determine the critical path.
   (b) Determine the total and free floats for the network, and identify the red-flagged activities.

FIGURE 6.49
Output of AMPL model for Example 6.5-2 (file amplEx6.5-2.txt)

```
i-j      D     ES     EC     LS     LC     TF     FF

1-2      5      0      5      0      5      0      0    c
1-3      6      0      6      5     11      5      2
2-3      3      5      8      8     11      3      0
2-4      8      5     13      5     13      0      0    c
3-5      2      8     10     11     13      3      3
3-6     11      8     19     14     25      6      6
4-5      0     13     13     13     13      0      0    c
4-6      1     13     14     24     25     11     11
5-6     12     13     25     13     25      0      0    c
```

(c) If activity A is started at time 5, determine the earliest possible start times for activities C, D, E, and G.

(d) If activities F, G, and H require the same equipment, determine the minimum number of units needed of this equipment.

6. Compute the floats and identify the red-flagged activities for the projects (a) and (b) in Figure 6.44, then develop the time schedules under the following conditions:

Project (a)

(i) Activity $(1,5)$ cannot start any earlier than time 14.

(ii) Activities $(5,6)$ and $(5,7)$ use the same equipment, of which only one unit is available.

(iii) All other activities start as early as possible.

Project (b)

(i) Activity $(1,3)$ must be scheduled at its earliest start time while accounting for the requirement that $(1,2)$, $(1,3)$, and $(1,6)$ use a special piece of equipment, of which 1 unit only is available.

(ii) All other activities start as early as possible.

6.5.4 Linear Programming Formulation of CPM

A CPM problem can thought of as the opposite of the shortest-route problem (Section 6.3), in the sense that we are interested in finding the *longest* route of a unit flow entering at the start node and terminating at the finish node. We can thus apply the shortest route LP formulation in Section 6.3.3 to CPM in the following manner. Define

$$x_{ij} = \text{Amount of flow in activity } (i, j), \text{ for all defined } i \text{ and } j$$
$$D_{ij} = \text{Duration of activity } (i, j), \text{ for all defined } i \text{ and } j$$

6

Thus, the objective function of the linear program becomes

$$\text{Maximize } z = \sum_{\substack{\text{all defined} \\ \text{activities } (i,j)}} D_{ij}x_{ij}$$

(Compare with the shortest route LP formulation in Section 6.3.3 where the objective function is minimized.) For each node, there is one constraint that represents the conservation of flow:

$$\text{Total input flow} = \text{Total output flow}$$

All the variables, x_{ij}, are nonnegative.

Example 6.5-5

The LP formulation of the project of Example 6.5-2 (Figure 6.42) is given below. Note that nodes 1 and 6 are the start and finish nodes, respectively.

	A	B	C	D	E	F	Dummy	G	H	
	x_{12}	x_{13}	x_{23}	x_{24}	x_{35}	x_{36}	x_{45}	x_{46}	x_{56}	
Maximize z =	6	6	3	8	2	11	0	1	12	
Node 1	−1	−1								= −1
Node 2	1		−1	−1						= 0
Node 3		1	1		−1	−1				= 0
Node 4				1			−1	−1		= 0
Node 5					1		1		−1	= 0
Node 6						1		1	1	= 1

The optimum solution is

$$z = 25, x_{12}(A) = 1, x_{24}(D) = 1, x_{45}(\text{Dummy}) = 1, x_{56}(H) = 1, \text{all others} = 0$$

The solution defines the critical path as $A \rightarrow D \rightarrow \text{Dummy} \rightarrow H$, and the duration of the project is 25 days. The LP solution is not complete, because it determines the critical path, but does not provide the data needed to construct the CPM chart. We have seen in Figure 6.48, however, that AMPL can be used to provide all the needed information without the LP optimization.

PROBLEM SET 6.5D

1. Use LP to determine the critical path for the project network in Figure 6.43.
2. Use LP to determine the critical path for the project networks in Figure 6.44.

6.5.5 PERT Networks

PERT differs from CPM in that it bases the duration of an activity on three estimates:

1. **Optimistic time,** a, which occurs when execution goes extremely well.
2. **Most likely time,** m, which occurs when execution is done under normal conditions.
3. **Pessimistic time,** b, which occurs when execution goes extremely poorly.

The range (a, b) encloses all possible estimates of the duration of an activity. The estimate m lies somewhere in the range (a, b). Based on the estimates, the average duration time, $\overline{D}$, and variance, v, are approximated as:

$$\overline{D} = \frac{a + 4m + b}{6}$$

$$v = \left(\frac{b - a}{6}\right)^2$$

CPM calculations given in Sections 6.5.2 and 6.5.3 may be applied directly, with $\overline{D}$ replacing the single estimate D.

It is possible now to estimate the probability that a node j in the network will occur by a prespecified scheduled time, S_j. Let e_j be the earliest occurrence time of node j. Because the durations of the activities leading from the start node to node j are random variables, e_j also must be a random variable. Assuming that all the activities in the network are statistically independent, we can determine the mean, $E\{e_j\}$, and variance, $\text{var}\{e_j\}$, in the following manner. If there is only one path from the start node to node j, then the mean is the sum of expected durations, $\overline{D}$, for all the activities along this path and the variance is the sum of the variances, v, of the same activities. On the other hand, if more than one path leads to node j, then it is necessary first to determine the statistical distribution of the duration of the longest path. This problem is rather difficult because it is equivalent to determining the distribution of the maximum of two or more random variables. A simplifying assumption thus calls for computing the mean and variance, $E\{e_j\}$ and $\text{var}\{e_j\}$, as those of the path to node j that has the largest sum of *expected* activity durations. If two or more paths have the same mean, the one with the largest variance is selected because it reflects the most uncertainty and, hence, leads to a more conservative estimate of probabilities.

Once the mean and variance of the path to node j, $E\{e_j\}$ and $\text{var}\{e_j\}$, have been computed, the probability that node j will be realized by a preset time S_j is calculated using the following formula:

$$P\{e_j \le S_j\} = P\left\{\frac{e_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}} \le \frac{S_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}}\right\} = P\{z \le K_j\}$$

where

$$z = \text{Standard normal random variable}$$

$$K_j = \frac{S_j - E\{e_j\}}{\sqrt{\text{var}\{e_j\}}}$$

The standard normal variable z has mean 0 and standard deviation 1 (see Section 12.4.4). Justification for the use of the normal distribution is that e_j is the sum of independent random variables. According to the *central limit theorem* (see Section 12.4.4), e_j is approximately normally distributed.

Example 6.5-6

Consider the project of Example 6.5-2. To avoid repeating critical path calculations, the values of $a, m,$ and b in the table below are selected such that $\overline{D_{ij}} = D_{ij}$ for all i and j in Example 6.5-2.

Activity	i–j	(a, m, b)	Activity	i–j	(a, m, b)
A	1–2	(3, 5, 7)	E	3–5	(1, 2, 3)
B	1–3	(4, 6, 8)	F	3–6	(9, 11, 13)
C	2–3	(1, 3, 5)	G	4–6	(1, 1, 1)
D	2–4	(5, 8, 11)	H	5–6	(10, 12, 14)

The mean $\overline{D_{ij}}$ and variance v_{ij} for the different activities are given in the following table. Note that for a dummy activity $(a, m, b) = (0, 0, 0)$, hence its mean and variance also equal zero.

Activity	i–j	$\overline{D}_{ij}$	V_{ij}	Activity	i–j	$\overline{D}_{ij}$	V_{ij}
A	1–2	5	.444	E	3–5	2	.111
B	1–3	6	.444	F	3–6	11	.444
C	2–3	3	.444	G	4–6	1	.000
D	2–4	8	1.000	H	5–6	12	.444

The next table gives the longest path from node 1 to the different nodes, together with their associated mean and standard deviation.

Node	Longest path based on mean durations	Path mean	Path standard deviation
2	1–2	5.00	0.67
3	1–2–3	8.00	0.94
4	1–2–4	13.00	1.20
5	1–2–4–5	13.00	1.20
6	1–2–4–5–6	25.00	1.37

Finally, the following table computes the probability that each node is realized by time S_j specified by the analyst.

Node j	Longest path	Path mean	Path standard deviation	S_j	K_j	$P\{z \leq K_j\}$
2	1–2	5.00	0.67	5.00	0	.5000
3	1–2–3	8.00	0.94	11.00	3.19	.9993
4	1–2–4	13.00	1.20	12.00	−.83	.2033
5	1–2–4–5	13.00	1.20	14.00	.83	.7967
6	1–2–4–5–6	25.00	1.37	26.00	.73	.7673

TORA Moment.

TORA provides a module for carrying out PERT calculations. To use this module, select Project Planning ⇒ PERT-Program Evaluation and Review Technique from Main Menu. In the output screen, you have the option to select Activity Mean/Var to compute the mean and variance for each activity or PERT Calculations to compute the mean and variance of the longest path to each node in the network. File toraEx6.5-6.txt provides TORA's data for Example 6.5-6.

PROBLEM SET 6.5E

1. Consider Problem 2, Set 6.5b. The estimates (a, m, b) are listed below. Determine the probabilities that the different nodes of the project will be realized without delay.

Project (a)				Project (b)			
Activity	*(a, m, b)*	*Activity*	*(a, m, b)*	Activity	*(a, m, b)*	*Activity*	*(a, m, b)*
1-2	(5, 6, 8)	3-6	(3, 4, 5)	1-2	(1, 3, 4)	3-7	(12, 13, 14)
1-4	(1, 3, 4)	4-6	(4, 8, 10)	1-3	(5, 7, 8)	4-5	(10, 12, 15)
1-5	(2, 4, 5)	4-7	(5, 6, 8)	1-4	(6, 7, 9)	4-7	(8, 10, 12)
2-3	(4, 5, 6)	5-6	(9, 10, 15)	1-6	(1, 2, 3)	5-6	(7, 8, 11)
2-5	(7, 8, 10)	5-7	(4, 6, 8)	2-3	(3, 4, 5)	5-7	(2, 4, 8)
2-6	(8, 9, 13)	6-7	(3, 4, 5)	2-5	(7, 8, 9)	6-7	(5, 6, 7)
3-4	(5, 9, 19)			3-4	(10, 15, 20)		

REFERENCES

Ahuja, R., T. Magnati, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications,* Prentice Hall, Upper Saddle River, NJ, 1993.

Bazaraa, M., J. Jarvis, and H. Sherali, *Linear Programming and Network Flow,* 2nd ed., Wiley, New York, 1990.

Bersetkas, D., *Network Optimization: Continuous and Discrete Models*, Athena Scientific, Nashua, NH, 1998.

Charnes, A. and W. Cooper, "Some Network Characterization for Mathematical Programming and Accounting Applications to Planning and Control," *The Accounting Review*, Vol. 42, No. 3, pp. 24–52, 1967.

Evans, J.R., and E. Minieka, *Optimization Algorithms for Networks and Graphs,* 2nd ed., Marcel Dekker, New York, 1992.

Guéret, C., C. Prins, and M. Sevaux, *Applications of Optimization with Xpress-MP*, translated and revised by Susanne Heipke, Dash Optimization Ltd., London, 2002.

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

Glover, F., and M. Laguna, *Tabu Search*, Kulwer Academic Publishers, Boston, 1997.

Murty, K., *Network Programming,* Prentice Hall, Upper Saddle River, NJ, 1992.

Robinson, E.W, L. Gao, and S. Muggenborg, "Designing an Integrated Distribution System at DowBrands, Inc,"*Interfaces*, Vol. 23, No. 3, pp. 107–117, 1993.