CHAPTER 6

Network Models

There is a multitude of operations research situations that can be modeled and solved as networks (nodes connected by branches). Some recent surveys report that as much as 70% of the real-world mathematical programming problems can be represented by network-related models. The following list illustrates possible applications of networks.

1. Design of an offshore natural gas pipeline network connecting wellheads 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 a network of roads.
3. Determination of the maximum capacity (in tons per year) of a coal slurry pipeline network joining the coal mines in Wyoming with the power plants in Houston. (Slurry pipelines transport coal by pumping water through specially designed pipes.)
4. Determination of the minimum-cost flow schedule from oil fields to refineries through a pipeline network.
5. Determination of the time schedule (start and completion dates) for the activities of a construction project.

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

1. Minimal spanning tree (situation 1)
2. Shortest-route algorithm (situation 2)
3. Maximum flow algorithm (situation 3)
4. Minimum-cost capacitated network algorithm (situation 4)
5. Critical path (CPM) algorithm (situation 5)

The situations for which these algorithms apply can also be formulated and solved as explicit linear programs. However, the proposed network-based algorithms are more efficient than the simplex method.

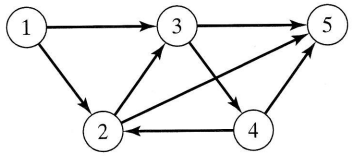
6.1 NETWORK DEFINITIONS

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

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

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

FIGURE 6.1
Example of (N, A) network



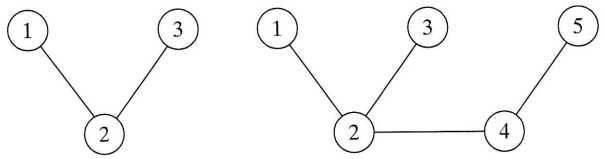
Associated with each network is some type of flow (e.g., oil products flow in a pipeline and automobile traffic flows on highways). In general, the flow in a network is limited by the capacity of its arcs, which may be finite or infinite.

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** if it connects a node to itself through other nodes. For example, in Figure 6.1, arcs (2,3), (3,5), and (5,2) form a loop. A cycle is *directed* if it consists of a directed path; e.g., (2, 3), (3, 4), and (4, 2) in Figure 6.1.

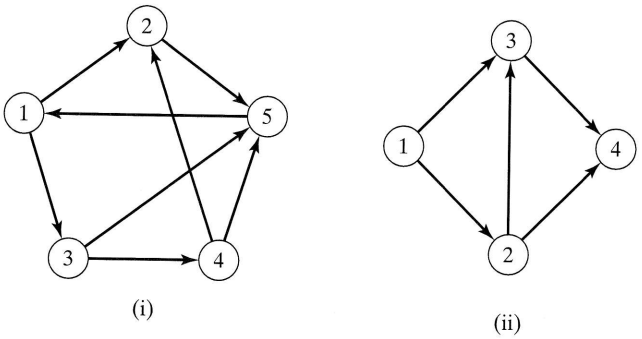
A **connected network** is such that every two distinct nodes are linked by at least one path. The network in Figure 6.1 demonstrates this type of network. A **tree** is a connected network that may involve only a *subset* of all the nodes of the network with no cycles allowed, and a **spanning tree** is a tree that links *all* the nodes of the network, also with no cycles allowed. Figure 6.2 provides examples of a tree and a spanning tree for the network in Figure 6.1.

FIGURE 6.2
Examples of a tree and a spanning tree given the network in Figure 6.1



PROBLEM SET 6.1A

1. For each network in Figure 6.3 determine (a) a path, (b) a cycle, (c) a directed cycle, (d) a tree, and (e) a spanning tree.



(i) (ii)

FIGURE 6.3
Networks for Problem 1, Set 6.1a

2. Determine the sets N and A for the networks in Figure 6.3.
3. Draw the network defined by

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

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

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, 2, ..., and 8 so that no two *adjacent* vertical, horizontal, or diagonal squares hold consecutive numbers. Use network representation as a vehicle to find the solution in a systematic way.
5. Three inmates escorted by 3 guards must be transported by boat from San Francisco to the Alcatraz 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 safe transfer of the inmates. Assume that the inmates will not flee if given a chance.

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 length of connecting branches. A typical application occurs in the construction of paved roads that link several 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 = Set of nodes that have been permanently connected at iteration k

$\overline{C}_k$ = Set of nodes as yet to be connected permanently

Step 0. Set $C_0 = \varnothing$ 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,

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

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 6.4 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

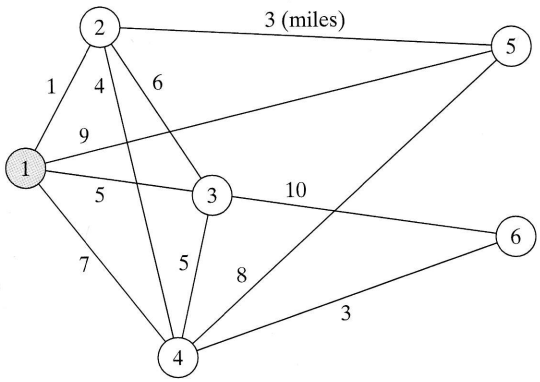
$$C_1 = \{1\}, \overline{C}_1 = \{2,3,4,5,6\}$$

The iterations of the algorithm are summarized in Figure 6.5. The thin arcs provide all the candidate links between C and $\overline{C}$. The thick branches represent the permanent links among 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, and 5 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 6.5. The resulting minimum cable miles needed to provide the desired cable service are $1 + 3 + 4 + 3 + 5 = 16$ miles.

FIGURE 6.4
Cable connections for Midwest TV Cable Company



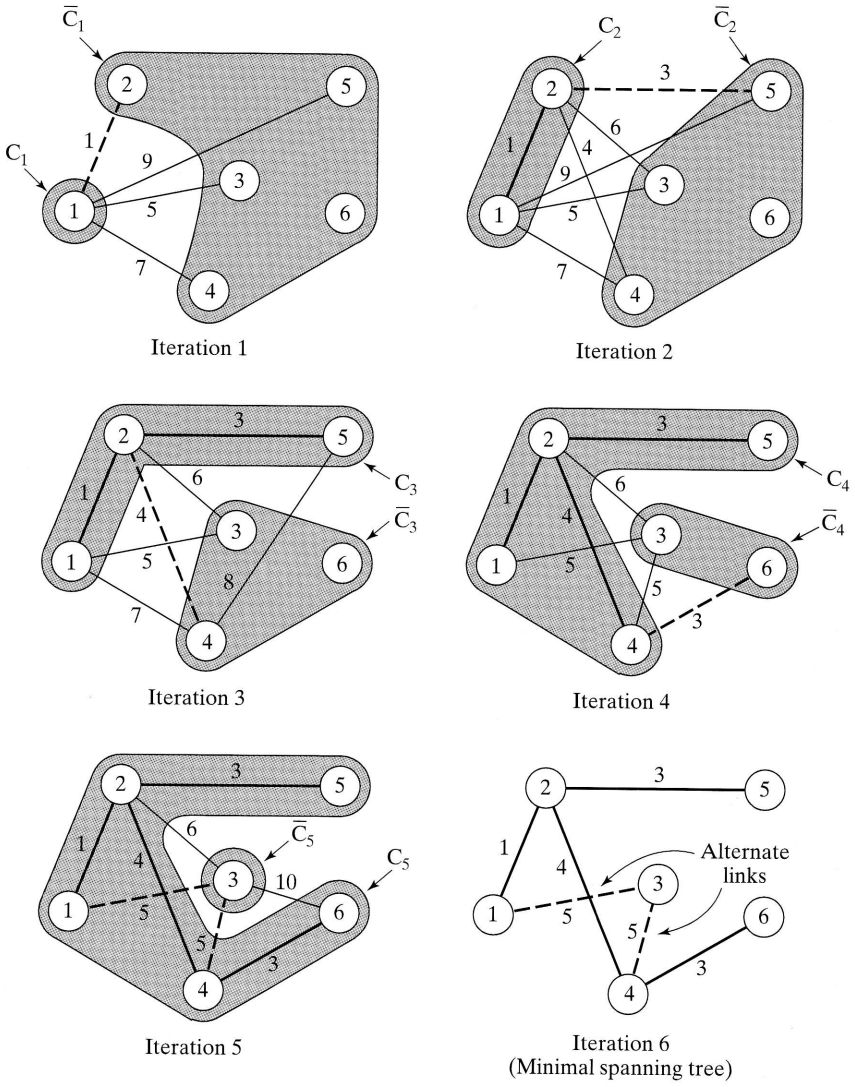


FIGURE 6.5
Solution iterations for Midwest TV Cable Company

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` and then use `Next iteration` or `All iterations` to generate the successive iterations. You can restart the iterations by selecting a new `Starting node`. Figure 6.6 gives TORA output for Example 6.2-1 (file ch6ToraMinSpanEx6-2-1.txt).

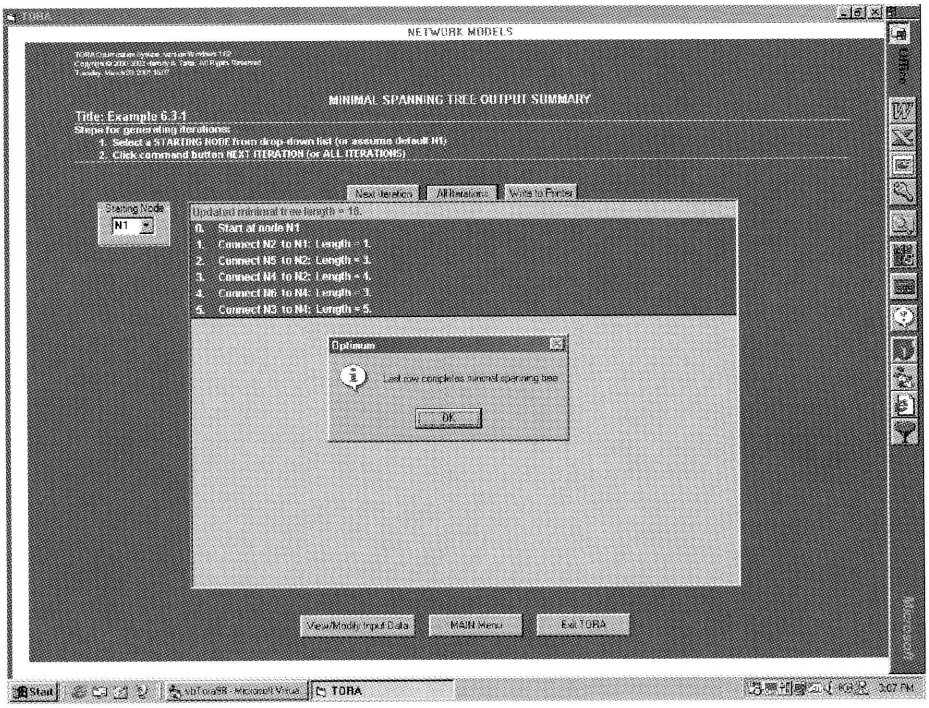


FIGURE 6.6
Output of the minimal spanning tree of Example 6.2-1

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.
   (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 by placing the trailer on special flatbed carts. Figure 6.7 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.

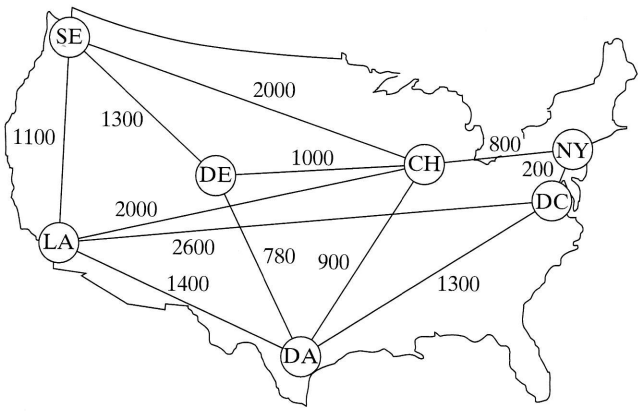


FIGURE 6.7
Network for Problem 3, Set 6.2a

4. Figure 6.8 gives the mileage of the feasible links connecting nine offshore natural gas wellheads with an inshore delivery point. Because the location of 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.

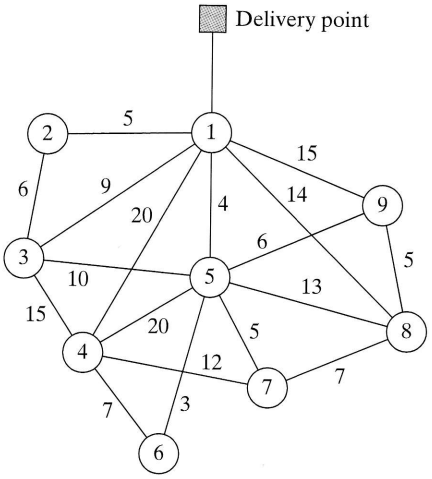


FIGURE 6.8
Network for Problem 4, Set 6.2a

5. In Figure 6.8 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, wellheads from the two groups cannot be linked. 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 j only.

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 plan for its car fleet for a 4-year planning horizon that starts January 1, 2001, and terminates December 31, 2004. At the start of each year, a decision is made as to whether a car should be kept in operation or replaced. 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	Replacement cost ($) for given years in operation		
	1	*2*	*3*
2001	4000	5400	9800
2002	4300	6200	8700
2003	4800	7100	—
2004	4900	—	—

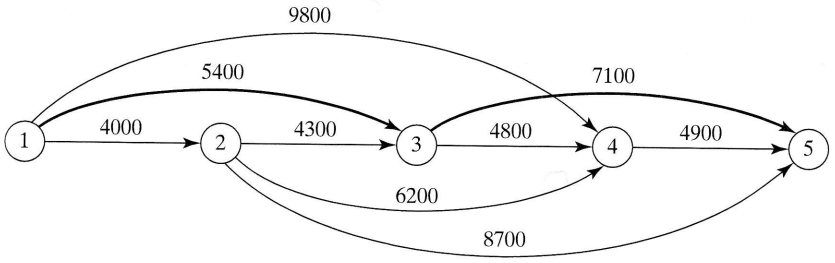


FIGURE 6.9
Equipment replacement problem as a shortest-route model

The problem can be formulated as a network in which nodes 1 to 5 represent the start of years 2001 to 2005. Arcs from node 1 (year 2001) 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 6.9 shows the resulting network. Using TORA,[1] 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 2001 (node 1) must be replaced after 2 years at the start of 2003 (node 3). The replacement car will then be kept in service until the end of 2004. 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 6.10 shows the possible routes between home and work, and the associated probabilities of not being stopped on each segment. The probability of not being stopped on the way to work is the product of the probabilities associated with the successive segments of the selected route. For example, the probability of not

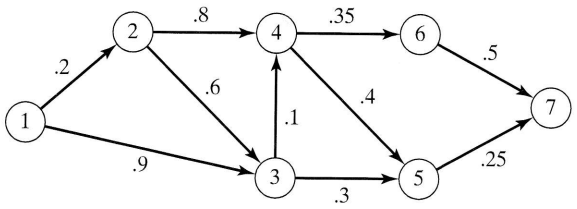


FIGURE 6.10
Most-reliable-route network model

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

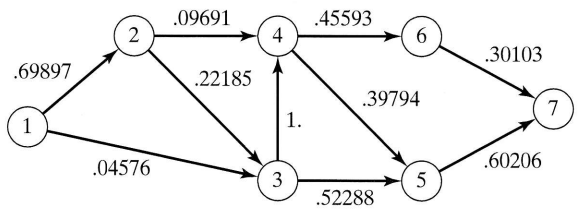


FIGURE 6.11
Most-reliable-route representation as a shortest-route model

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 $p_{1k} = p_1 \times p_2 \times \ldots \times p_k$ is the probability of not being stopped, then $\log p_{1k} = \log p_1 + \log p_2 + \ldots + \log p_k$.

Mathematically, the maximization of p_{1k} is equivalent to the maximization of $\log p_{1k}$. Because $\log p_{1k} \leq 0$, the maximization of $\log p_{1k}$ is, in turn, equivalent to the minimization of $-\log p_{1k}$. Using this transformation, the individual probabilities p_j in Figure 6.10 are replaced with $-\log p_j$ for all j in the network, thus yielding the shortest-route network in Figure 6.11.

Using TORA, nodes 1, 3, 5, and 7 define the shortest route in Figure 6.11 with a corresponding "length" of 1.1707 $(= -\log p_{17})$. Thus, the maximum probability of not being stopped is $p_{17} = .0675$.

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 pourings needed to achieve this result?

You probably can guess the solution of 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 solution node (4, 4, 0). A new node is generated from the current node by pouring fluid from one jug into another.

Figure 6.12 shows different routes that lead from start node (8, 0, 0) to end node (4, 4, 0). The arc between two successive nodes represents a single pouring, and hence can be assumed to have a length of 1 unit. The problem 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 6.12, requires 7 pourings.

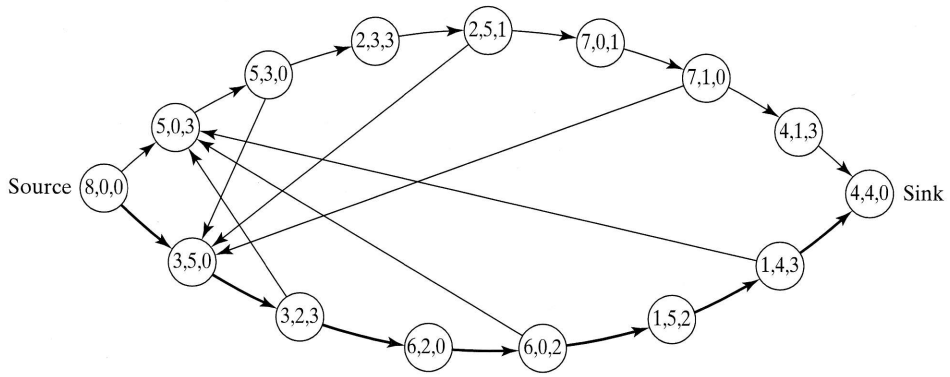


FIGURE 6.12
Three-jug puzzle representation as a shortest-route model

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 2001 to the end of 2005. The following table provides the necessary data.

	Replacement cost ($) for given years in operation		
Year acquired	*2*	*3*	*4*
2001	3800	4100	6800
2002	4000	4800	7000
2003	4200	5300	7200
2004	4800	5700	—
2005	5300	—	—

2. Figure 6.13 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 solve with TORA.

3. 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. It is desired to determine the sequence of operations (placing, toasting, turning, and removing) needed to toast three slices of bread in the shortest possible

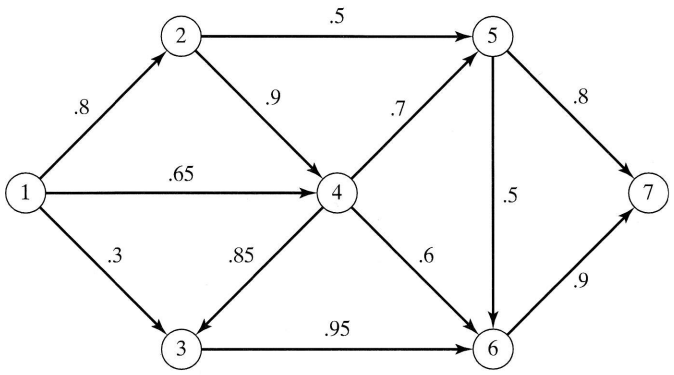


FIGURE 6.13
Network for Problem 2, Set 6.3a

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

4. *Production Planning.* DirectCo sells an item whose demand over the next 4 months is 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.
5. *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 a 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 the decision is made on i.)

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 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. At the point when no better routes 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 6.14 gives the 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

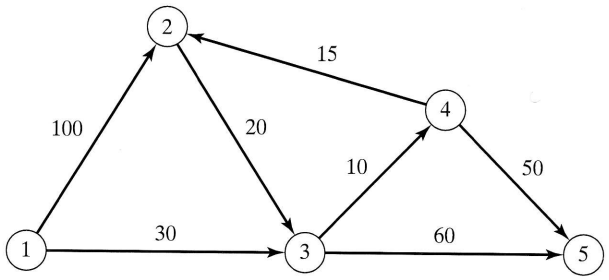


FIGURE 6.14
Network example for Dijkstra's shortest-route algorithm

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] in iteration 2 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 $u_5 = 90$.

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

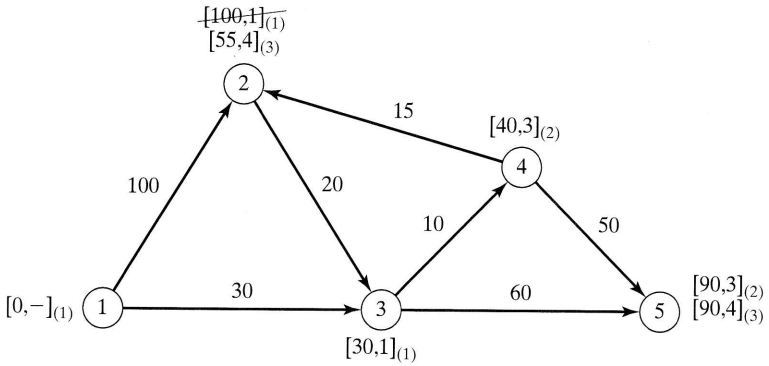
Iteration 4. Only node 3 can be reached from node 2. However, node 3 has a permanent label and cannot be relabeled. The new list of labels remains the same as in iteration 3 except that the label at node 2 is now permanent. This leaves node 5 as the only temporary label. Because node 5 does not lead 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 6.15 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:

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

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



( ) = iteration

FIGURE 6.15
Dijkstra's labeling procedure

TORA can be used to generate Dijkstra's iterations. From the `SOLVE/MODIFY` menu, select `Solve problem` ⇒ `Iterations` ⇒ `Dijkstra's algorithm`. Figure 6.16 provides TORA's iterations output for Example 6.3-4 (file ch6ToraDijkstraEx6-3-4.txt).

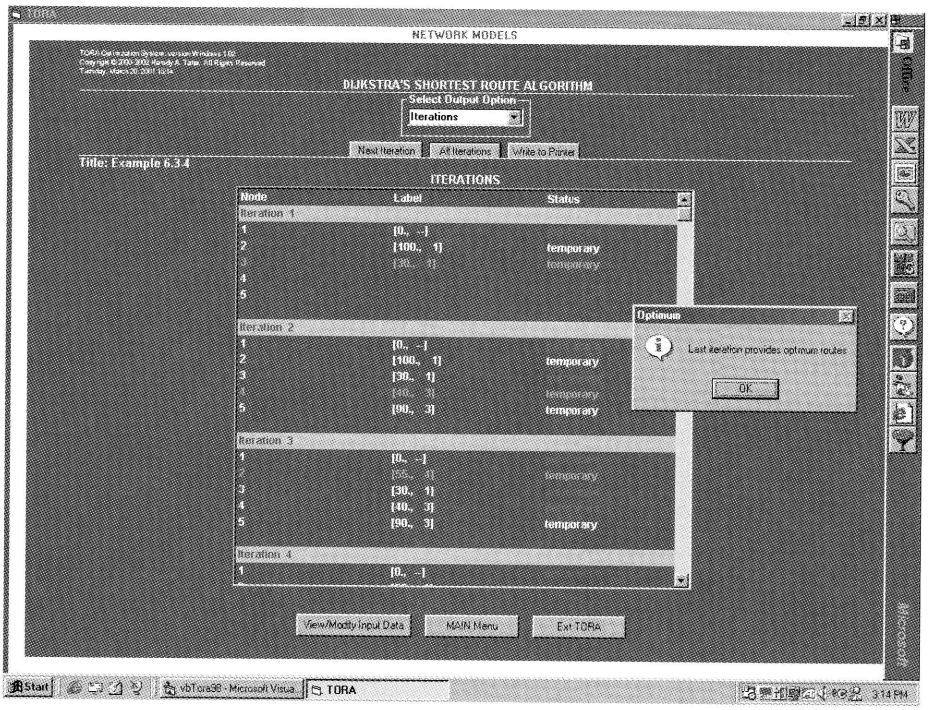
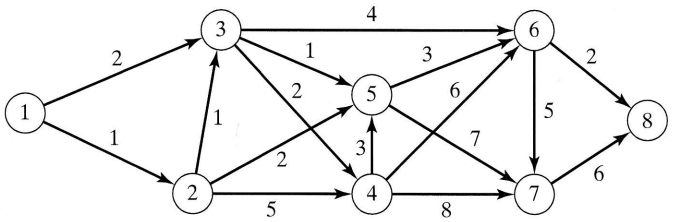


FIGURE 6.16
TORA Dijkstra iterations for Example 6.3-4

PROBLEM SET 6.3B

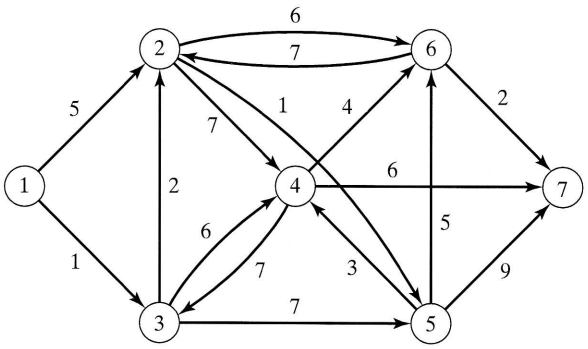
1. The network in Figure 6.17 gives the distances in miles between pairs of cities 1, 2, ..., and 8. Use Dijkstra's algorithm to find the shortest route between the following cities:
   (a) Cities 1 and 8
   (b) Cities 1 and 6
   (c) Cities 4 and 8
   (d) Cities 2 and 6

FIGURE 6.17
Network for Problem 1, Set 6.3b



2. Use Dijkstra's algorithm to find the shortest route between node 1 and every other node in the network of Figure 6.18.

FIGURE 6.18
Network for Problem 2, Set 6.3b



3. Use Dijkstra's 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.

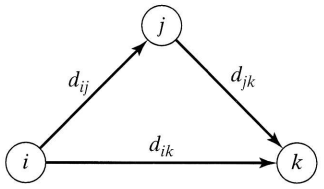


FIGURE 6.19
Floyd's triple operation

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 k from i passing through j if

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

In this case, it is optimal to replace the direct route from $i \rightarrow k$ with the indirect route $i \rightarrow j \rightarrow k$. 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 indicates that they are blocked. Set $k = 1$.

$D_0 =$

	1	2	…	j	…	n
1	—	d_{12}	…	d_{ij}	…	d_{1n}
2	d_{21}	—	…	d_{2j}	…	d_{2n}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	d_{i1}	d_{i2}	…	d_{ij}	…	d_{in}
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	D_{n1}	d_{n2}	…	d_{nj}	…	—

$S_0 =$

	1	2	…	j	…	n
1	—	2	…	j	…	n
2	1	—	…	j	…	n
⋮	⋮	⋮	⋮	⋮	⋮	⋮
i	1	2	…	j	…	n
⋮	⋮	⋮	⋮	⋮	⋮	⋮
n	1	2	…	j	…	—

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 $k = k + 1$, and repeat step k.

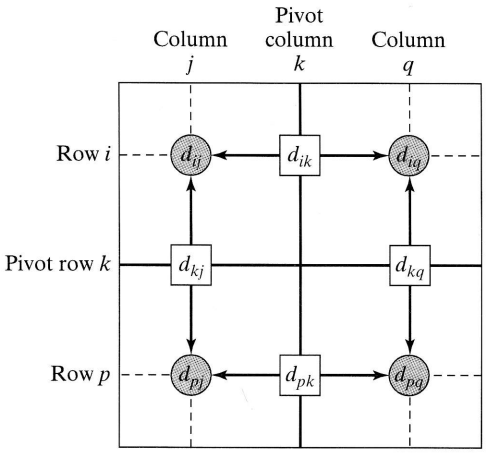


FIGURE 6.20
Implementation of triple operation in matrix form

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 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. With the *triple operation,* 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.

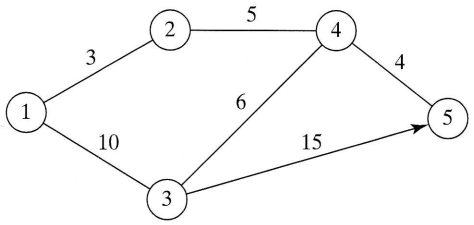
After n steps, we can determine the shortest route between nodes i and j from the matrices D_n and S_n using the following rules:

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 traffic in both directions.

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 $s_{32} = 1$.

These changes are shown in bold in matrices D_1 and S_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	4	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 lightly-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. Hence, D_5 and S_5 are the same as D_4 and S_4.

The final matrices D_5 and S_5 contain all the information needed to determine the shortest route between any two nodes in the network. For example, consider determining the shortest route from node 1 to node 5. First, the associated shortest distance is given by $d_{15} = 12$ 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 $s_{15} = 4$, 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 $1 \rightarrow 4$ must be 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 $s_{45} = 5$, the route $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ needs no further "dissecting" and the process ends.

As in Dijkstra's algorithm, TORA can be used to generate Floyd's iterations. From the `SOLVE/MODIFY` menu, select `Solve problem` ⇒ `Iterations` ⇒ `Floyd's algorithm`. Figure 6.22 illustrates TORA's output for Floyd's Example 6.3-5 (file ch6ToraFloydEx6-3-5.txt).

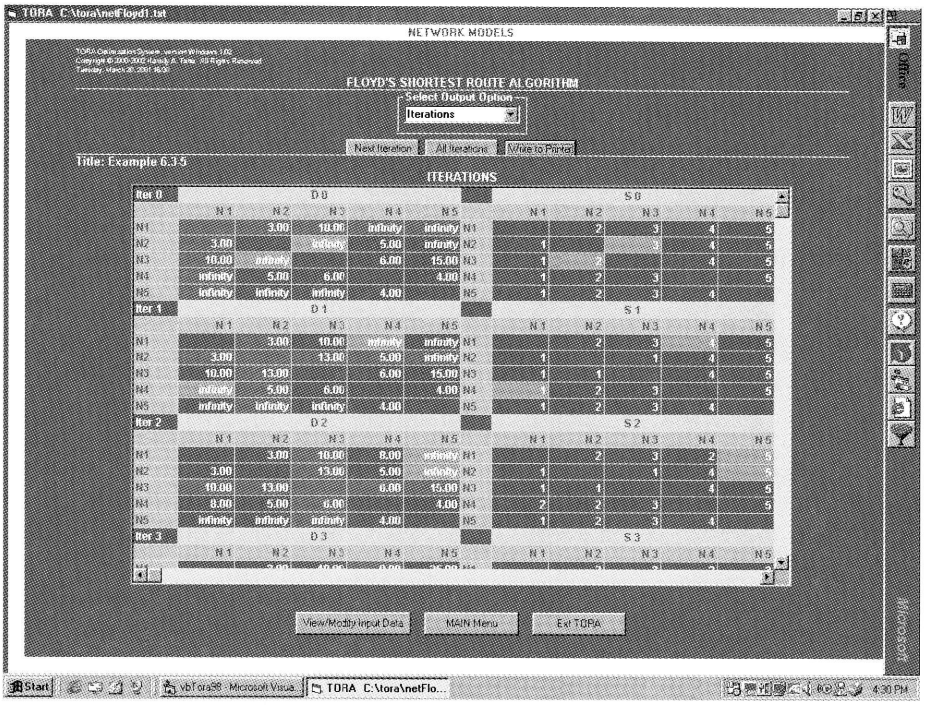


FIGURE 6.22
TORA Floyd iterations for Example 6.3-5

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
   (c) From node 5 to node 3
   (d) From node 5 to node 2
2. Apply Floyd's algorithm to the network in Figure 6.23. 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 6.24. 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 the game of hide and seek. The hiding place of a child is known only to a select few of the other children. A

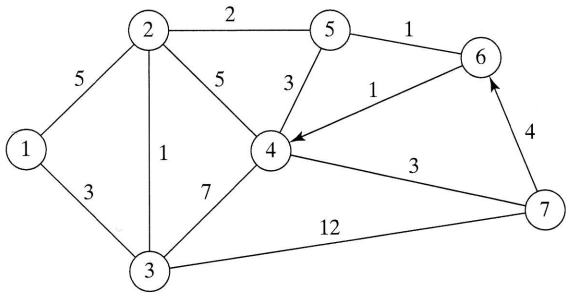


FIGURE 6.23
Network for Problem 2, Set 6.3c

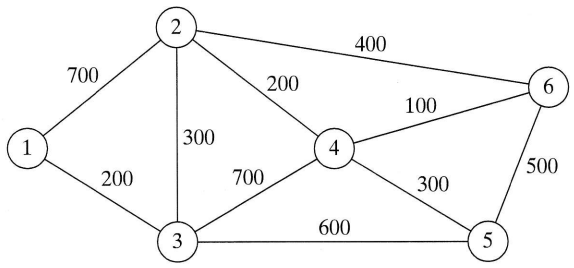


FIGURE 6.24
Network for Problem 3, Set 6.3c

child is then paired with another with the objective of finding his or her 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 know the hiding place of Kay only.

Rae knows where Kim is hiding.

Kim knows where Joe and Bob are hiding.

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 two LP formulations for the shortest-route problem. The formulations are general in the sense that they can be used to find the shortest route between any two nodes in the network. In this regard, the LP formulations are 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.

Formulation 1: This formulation assumes that an external one unit of flow enters the network at node s and leaves it at node t, where s and t are the two target nodes between which we seek to determine the shortest route.

Define

x_{ij} = amount of flow in arc (i, j), for all feasible i and j

c_{ij} = length of arc (i, j), for all feasible i and j

Because only *one* unit of flow can be in any arc at any one time, the variable x_{ij} must assume binary values (0 or 1) only. Thus, the objective function of the linear program becomes

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

There is one constraint that represents the conservation of flow at each node—that is, for any node j,

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

Formulation 2: The second formulation is actually the dual problem of the LP in Formulation 1. Because the number of constraints in Formulation 1 equals the number of nodes, the dual problem will have as many variables as the number of nodes in the network. Also, all the dual variables must be unrestricted because all the constraints in Formulation 1 are equations.

Let

y_j = dual constraint associated with node j

Given s and t are the start and terminal nodes of the network, the dual problem is defined as

$$\text{Maximize } z = y_t - y_s$$

subject to

$$y_j - y_i \leq c_{ij}, \text{ for all feasible } i \text{ and } j$$

$$\text{all } y_i \text{ and } y_j \text{ unrestricted in sign}$$

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 6.25 shows how the unit of flow enters at node 1 and leaves at node 2.

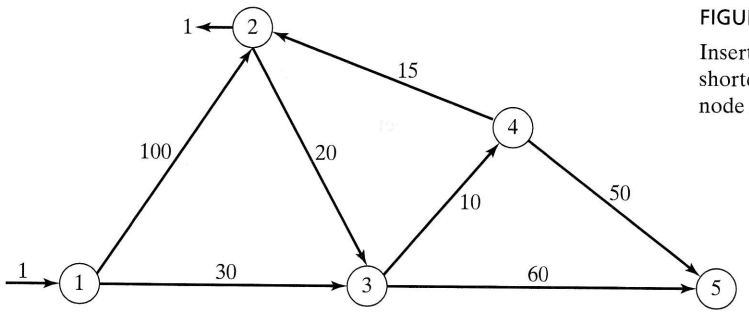


FIGURE 6.25
Insertion of unit flow to determine shortest route between node $s = 1$ and node $t = 2$

Using Formulation 1, the associated LP is listed below.

	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

The constraints represent flow conservation at each node. For example, at node 2, "input flow = output flow" yields $x_{12} + x_{42} = 1 + x_{23}$. Note that one of the constraints is always redundant. For example, adding the last four constraints simultaneously yields $x_{12} + x_{13} = 1$, which is the same as constraint 1.

The optimal solution (obtained by TORA)[2] 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).

To use Formulation 2, the dual problem associated with the LP above is given as

$$\text{Maximize } z = y_2 - y_1$$

subject to

$$y_2 - y_1 \leq 100 \text{ (Route 1-2)}$$
$$y_3 - y_1 \leq 30 \text{ (Route 1-3)}$$
$$y_3 - y_2 \leq 20 \text{ (Route 2-3)}$$
$$y_4 - y_3 \leq 10 \text{ (Route 3-4)}$$
$$y_5 - y_3 \leq 60 \text{ (Route 3-5)}$$
$$y_2 - y_4 \leq 15 \text{ (Route 4-2)}$$
$$y_5 - y_4 \leq 50 \text{ (Route 4-5)}$$
$$y_1, y_2, \ldots, y_5 \text{ unrestricted}$$

Although the dual problem given above is a pure mathematical definition derived from the primal problem, we actually can interpret the problem in a logical manner. Define

$$y_i = \text{Distance to node } i$$

[2]TORA does not accept a negative right-hand side. You can get around this inconvenience by selecting the redundant constraint as the one having the negative right-hand side, then make it redundant by changing = to ≤ and setting the right-hand side to a very large value. Another trick is to add a new variable whose upper and lower bounds equal 1, effectively forcing it to equal 1 in any solution. The constraint coefficients of the new variable equal those of the current right-hand side, but with opposite sign. The right-hand side of the "new" problem must be changed to zero for all the constraints (see file ch6ToraLpShortRoute Ex6-3-6.txt).

With this definition, the shortest distance from the start node 1 to the terminal node 2 is determined by maximizing $y_2 - y_1$. The constraint associated with route (i, j) says that the distance from node i to node j cannot exceed the direct length of that route. It can be less if node j can be reached from node i through other nodes that provide a shorter path. For example, the distance from node 1 to node 2 is at most 100. With the definition of y_i as the distance to node i, we can assume that all the variables are nonnegative (instead of being unrestricted). We can also assume that $y_1 = 0$ as the distance to node 1.

Based on the discussion above, and assuming that all the variables are nonnegative, the optimum solution is given as

$$z = 55,\ y_1 = 0,\ y_2 = 55,\ y_3 = 30,\ y_4 = 40,\ y_5 = 0$$

The value of $z = 55$ gives the shortest distance from node 1 to node 2, which also equals $y_2 - y_1 = 55 - 0 = 55$.

The determination of the route itself from this solution is somewhat tricky. We note that the solution satisfies *in equation form* the constraints of routes 1-3, 3-4, and 4-2 because their slacks equal zero—that is, $y_3 - y_1 = 30$, $y_4 - y_3 = 10$, and $y_2 - y_4 = 15$. This result identifies the shortest route as $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$.

Another way for identifying the constraints that are satisfied in equation form is to consult the dual solution of the LP of Formulation 2. Any constraint that has a nonzero dual value must be satisfied in equation form (see Section 4.2.4). The following table pairs the routes (constraints) with their associated dual values.

Route (constraint)	1-2	1-3	2-3	3-4	3-5	4-2	4-5
Associated dual value	0	1	0	1	0	1	0

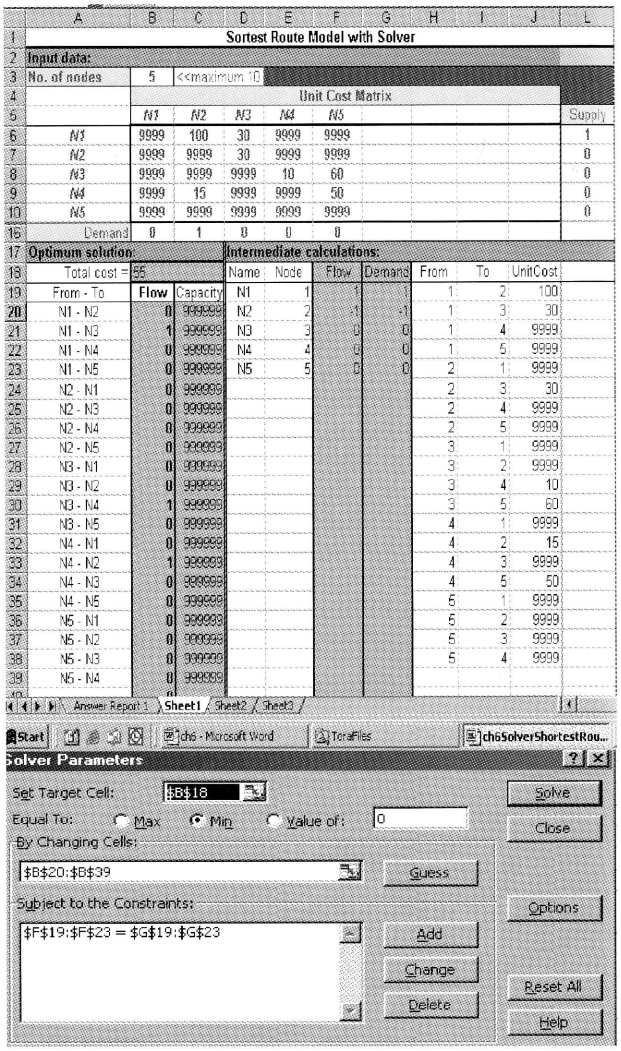
PROBLEM SET 6.3D

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

6.3.4 Excel Spreadsheet Solution of the Shortest-Route Problem

The Excel spreadsheet developed for the general transportation model (Section 5.3.3) can be modified readily to find the shortest route between two nodes. The spreadsheet is based on Formulation 1, Section 6.3.3, and is designed for problems with a maximum of 10 nodes. Figure 6.26 shows the application of the spreadsheet to Example 6.3-4 (file ch6SolverShortestRoute.xls). The distance matrix resides in cells B6:K15.[3] An infinite distance (= 9999, or any relatively large value) is entered for nonexisting arcs. Because we are seeking the shortest route between nodes 1 and 2, the supply amount for node 1 and the demand amount for node 2 is 1 unit. A zero amount is entered for the remaining supply and demand entries.

[3]In Figure 6.26, rows 11 through 15 and column K are hidden to conserve space.



Sortest Route Model with Solver

Input data:

No. of nodes: 5 <=maximum 10

Unit Cost Matrix

	N1	N2	N3	N4	N5	Supply
N1	9999	100	30	9999	9999	1
N2	9999	9999	30	9999	9999	0
N3	9999	9999	9999	10	60	0
N4	9999	15	9999	9999	50	0
N5	9999	9999	9999	9999	9999	0
Demand	0	1	0	0	0	

Optimum solution:

Total cost = 85

From - To	Flow	Capacity
N1 - N2	0	999999
N1 - N3	1	999999
N1 - N4	0	999999
N1 - N5	0	999999
N2 - N1	0	999999
N2 - N3	0	999999
N2 - N4	0	999999
N2 - N5	0	999999
N3 - N1	0	999999
N3 - N2	0	999999
N3 - N4	1	999999
N3 - N5	0	999999
N4 - N1	0	999999
N4 - N2	1	999999
N4 - N3	0	999999
N4 - N5	0	999999
N5 - N1	0	999999
N5 - N2	0	999999
N5 - N3	0	999999
N5 - N4	0	999999

Intermediate calculations:

Name	Node	Flow	Demand
N1	1	1	1
N2	2	-1	-1
N3	3	0	0
N4	4	0	0
N5	5	0	0

From	To	UnitCost
1	2	100
1	3	30
1	4	9999
1	5	9999
2	1	9999
2	3	30
2	4	9999
2	5	9999
3	1	9999
3	2	9999
3	4	10
3	5	60
4	1	9999
4	2	15
4	3	9999
4	5	50
5	1	9999
5	2	9999
5	3	9999
5	4	9999

Answer Report 1 | Sheet1 | Sheet2 | Sheet3

Start | ch6 - Microsoft Word | ToraFiles | ch6SolverShortestRou...

Solver Parameters

Set Target Cell: $B$18 — Solve

Equal To: Max Min Value of: 0 — Close

By Changing Cells:

$B$20:$B$39 — Guess

Subject to the Constraints:

$F$19:$F$23 = $G$19:$G$23 — Add, Change, Delete — Options, Reset All, Help

FIGURE 6.26

Excel Solver solution of the shortest route between nodes 1 and 2 in Example 6.3-4

Once the unit cost and supply/demand data are entered, the remainder of the spreadsheet (*intermediate calculations* and *optimum solution* sections) is generated automatically. Solver parameters must correspond to the input data of the problem as shown in highlighted columns B, C, F, and G. Column B specifies the changing cells (arcs flow) of the problem (cells B20:B39). Column C specifies the capacities of the arcs of the network (cells C20:C39). In the shortest-route model, these capacities do not play a role in the computations and hence are infinite (=999999). The constraints of the model represent the balance equation for each node. Cells F19:F23 define the left-hand side and cells G19:G23 represent the right-hand side of the flow equations. As explained in Section 5.3.3, SUMIF is used to generate the proper net flow in each node using the information in columns I and J. These calculations are automated by the

spreadsheet. Thus, all you need to do after entering the input data is to update Changing Cells and Constraints specifications of Solver to match the input data. The Target Cell remains the same for all input data. In Example 6.3-4, we have

```
Changing Cells: B20:B39
Constraints: F19:F23=G19:G23
```

The output in Figure 6.26 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 $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$.

PROBLEM SET 6.3E

1. Modify spreadsheet ch6SolverShortestRoute.xls (applied to Example 6.3-4) 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 spreadsheet ch6SolverShortestRoute.xls for Problem 2, Set 6.3a, to find the shortest routes between node 4 and node 7.

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 rate of crude flow (or capacity). A pipe segment may be unidirectional or bidirectional, depending on its design. A unidirectional segment has a finite capacity in one direction and a zero capacity in the opposite direction. 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 converting the network into one with a single source and a single sink. This requirement can be accomplished by using unidirectional infinite capacity arcs as shown by dashed arcs in Figure 6.27.

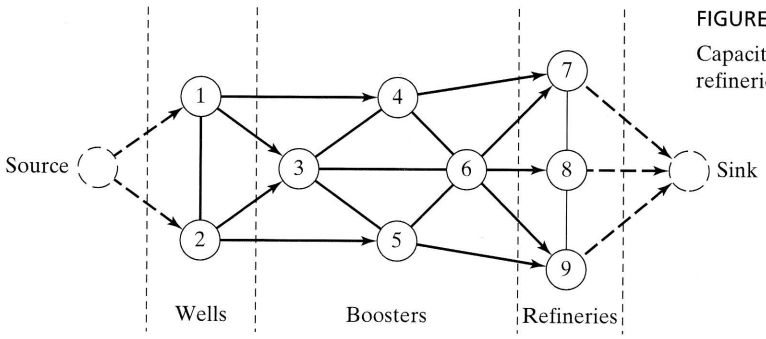
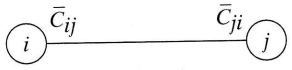


FIGURE 6.27
Capacitated network connecting wells and refineries through booster stations

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.

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



6.4.1 Enumeration of Cuts

A **cut** defines a set of arcs which when deleted from the network will cause a complete disruption of flow between the source and sink nodes. The **cut capacity** equals the sum of the capacities of the associated 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
Examples of cuts in flow networks

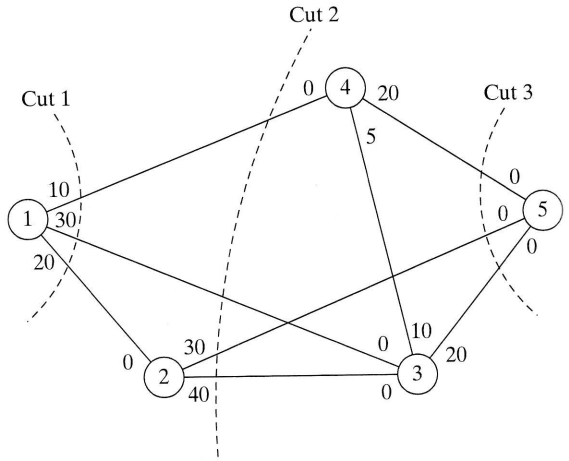


Figure 6.29 illustrates three cuts whose capacities are computed in the following table.

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

We cannot tell what the maximal flow in the network is unless we exhaustively enumerate all possible cuts. The only piece of information we can get from the partial enumeration of three cuts is that the maximum flow in the network cannot exceed 60 units. Unfortunately, exhaustive enumeration of all cuts is not a simple task, thus making it necessary to develop the efficient algorithm in Section 6.4.2.

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 the capacities of its arcs to the total flow in the network.

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

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

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

Step 2. Determine S_i as 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 $S_i \neq \varnothing$, 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 further breakthroughs are possible; go to step 6. Otherwise, let r be the node that has been labeled *immediately* before the current node i and remove i from the set of nodes that are adjacent to r. Set $i = r$, and go to step 2.

Step 5. *(Determination of Residue Network).* 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

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

(b) Given that the *initial* and *final* residuals of arc (i, j) are given by $(\overline{C}_{ij}, \overline{C}_{ji})$ 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 inadvertently "dead-ended" at an intermediate node before a breakthrough can be realized. 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 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).

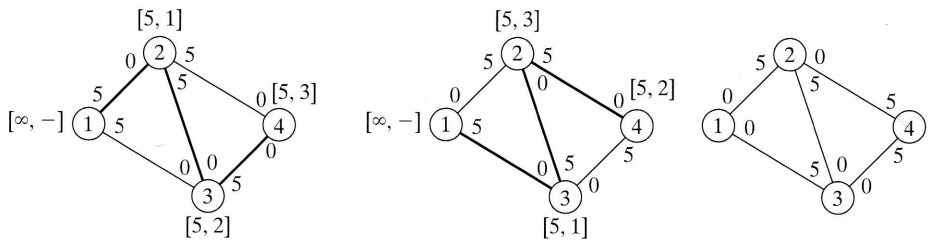
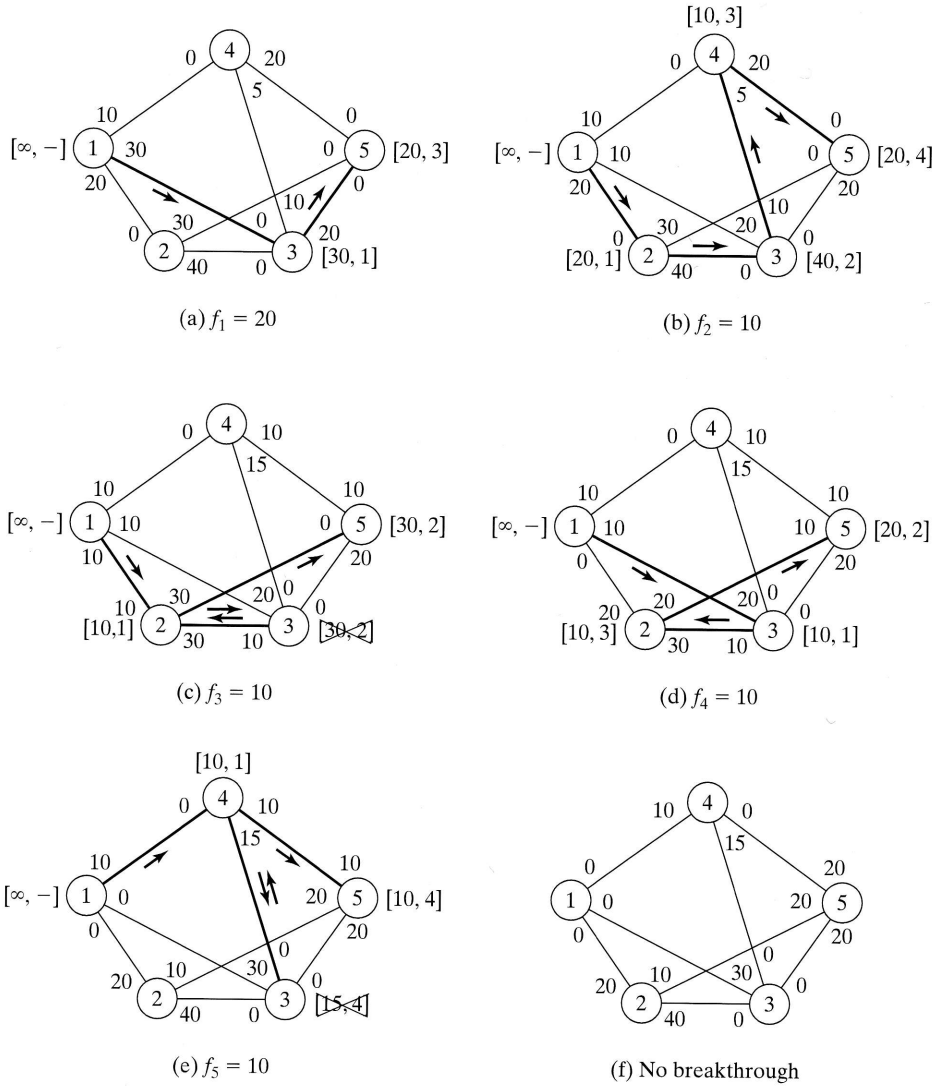


FIGURE 6.30

Use of residual to calculate maximum flow

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.



(a) $f_1 = 20$ (b) $f_2 = 10$

(c) $f_3 = 10$ (d) $f_4 = 10$

(e) $f_5 = 10$ (f) No breakthrough

FIGURE 6.31

Iterations of the maximum flow algorithm of Example 6.4-2

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

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. Breakthrough path is determined from the labels starting at node 5 and ending at node 1—that is, $(\mathbf{5}) \rightarrow [20, \mathbf{3}] \rightarrow (\mathbf{3}) \rightarrow [30, \mathbf{1}] \rightarrow (\mathbf{1})$. 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.

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

Step 4. 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 $F = f_1 + f_2 + \ldots + f_5 = 20 + 10 + 10 + 10 + 10 = 60$ 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 → 2
(1,3)	(30, 0) − (0, 30) = (30, −30)	30	1 → 3
(1,4)	(10, 0) − (0, 10) = (10, −10)	10	1 → 4
(2,3)	(40, 0) − (40, 0) = (0, 0)	0	—
(2,5)	(30, 0) − (10, 20) = (20, −20)	20	2 → 5
(3,4)	(10, 5) − (0, 15) = (10, −10)	10	3 → 4
(3,5)	(20, 0) − (0, 20) = (20, −20)	20	3 → 5
(4,5)	(20, 0) − (0, 20) = (20, −20)	20	4 → 5

You can use TORA to solve the maximum 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 output screen and select either `Maximum Flows` or `Iterations`. Figure 6.32 illustrates the first two iterations of Example 6.4-2 (file ch6ToraMaxFlowEx6-4-2.txt).

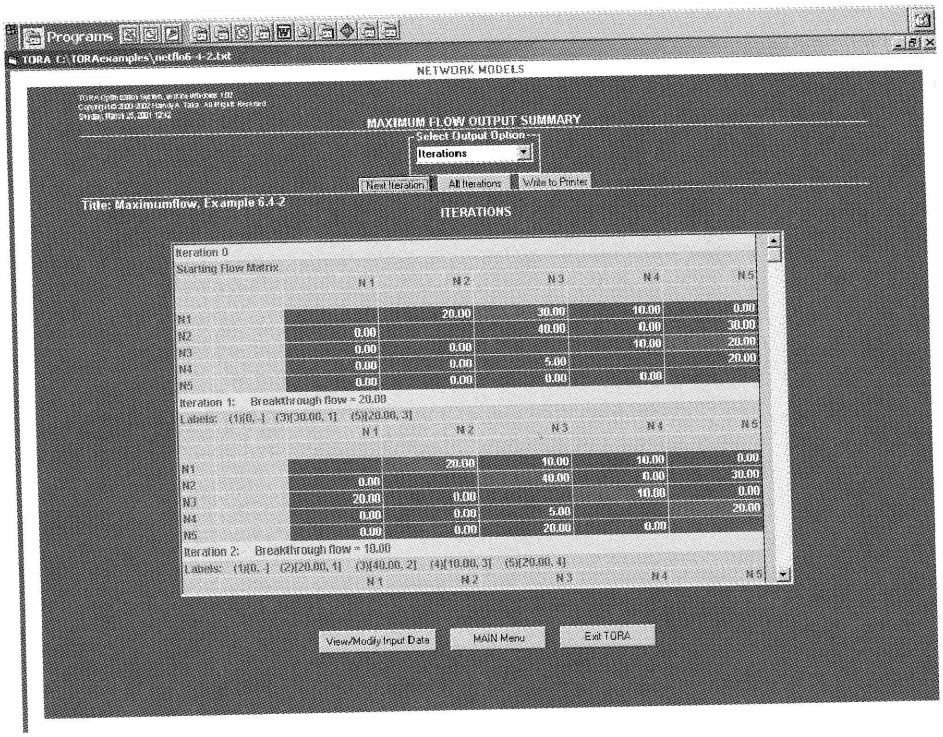


FIGURE 6.32
TORA's maximum flow iterations for Example 6.4-2

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 6.33.
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 6.34. 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.

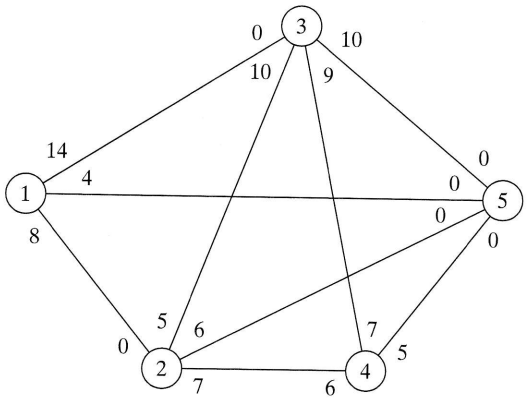


FIGURE 6.33
Network for Problem 2, Set 6.4b

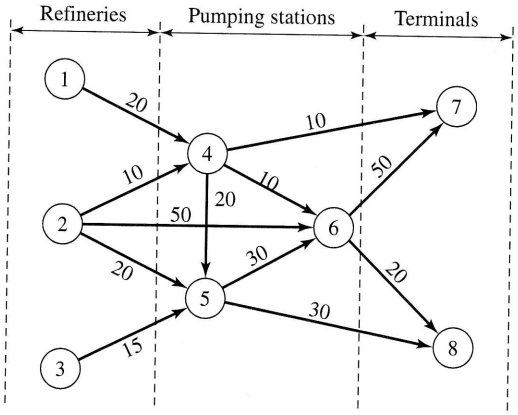


FIGURE 6.34
Network for Problem 3, Set 6.4b

4. Suppose that the maximum daily capacity of pump 6 in the network of Figure 6.34 is limited to 60 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 chicken 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	
Silo 1	30	5	0	40	**20**
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 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 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 Comprehensive Problem 6-3). 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 $l_{ij} \le x_{ij} \le u_{ij}$ 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 $0 \le x'_{ij} \le u_{ij} - l_{ij}$.

(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 $0 \le x'_{ij} \le u_{ij} - l_{ij}$, (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 the Maximum Flow Model

Define x_{ij} as the amount of flow in arc (i, j) and let c_{ij} be the capacity of the same arc. Assume that s and t are the start and terminal nodes between which we need to determine the maximum flow in the capacitated network.

The constraints of the problem preserve the in-out flow at each node, with the exception of start and terminal nodes. The objective function maximizes either the total "out" flow from start node s or the total "in" flow to terminal node t.

Example 6.4-3

In the maximum 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 objective functions depending on whether we are maximizing the output from node 1 $(=z_1)$ or the input to 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 $z_1 = z_2 = 60$.

PROBLEM SET 6.4C

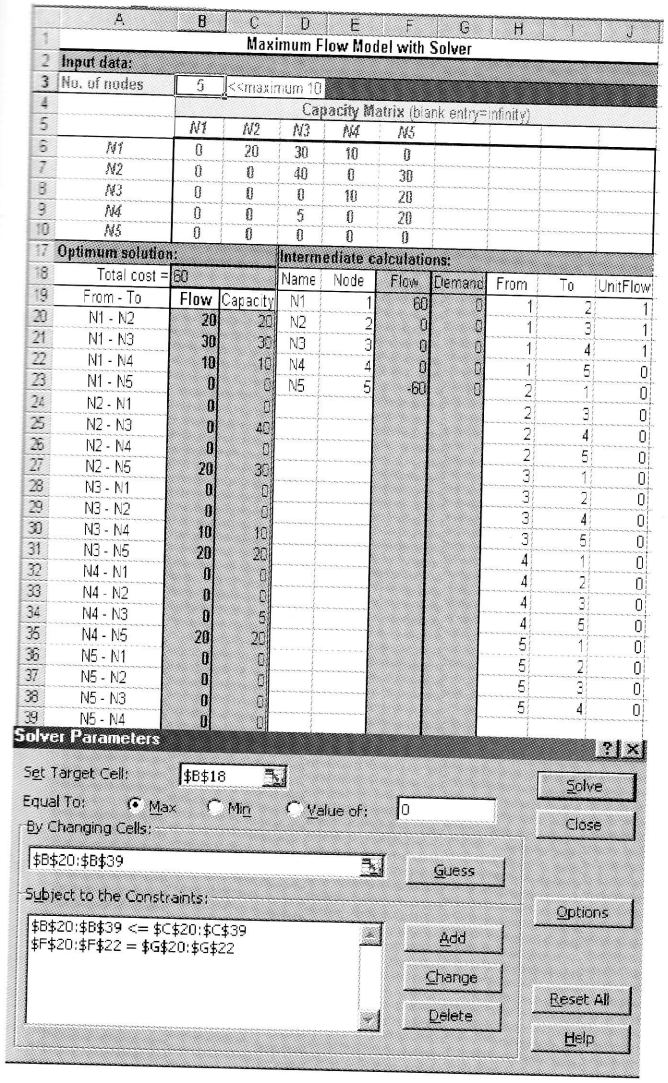
1. Rework Problem 2, Set 6.4b using linear programming.
2. Rework Problem 5, Set 6.4b using linear programming.

6.4.4 Excel Spreadsheet Solution of the Maximum Flow Model

The network-based Excel spreadsheet developed for the transportation model (Section 5.3.3) is modified to determine the maximum flow in a capacitated network. The spreadsheet is designed for problems with a maximum of 10 nodes. Figure 6.35 shows the application of the spreadsheet to Example 6.4-2 (file ch6SolverMax Flow.xls). The capacity flow matrix resides in cells B6:K15.[4] A blank cell in the capacity matrix indicates that the associated arc has infinite capacity. A zero entry corresponds to a nonexisting flow arc. Otherwise, all the remaining arcs must have finite capacities.

Once the flow capacity data have been entered, the remainder of the spreadsheet (*intermediate calculations* and *optimum solution* sections) is created automatically. All

[4]In Figure 6.35, rows 11 through 16 and column K are hidden to conserve space.



Maximum Flow Model with Solver

Input data:

No. of nodes: 5 <=maximum 10

Capacity Matrix (blank entry=infinity)

	N1	N2	N3	N4	N5
N1	0	20	30	10	0
N2	0	0	40	0	30
N3	0	0	0	10	20
N4	0	0	5	0	20
N5	0	0	0	0	0

Optimum solution:

Total cost = 60

From - To	Flow	Capacity
N1 - N2	20	20
N1 - N3	30	30
N1 - N4	10	10
N1 - N5	0	0
N2 - N1	0	0
N2 - N3	0	40
N2 - N4	0	0
N2 - N5	20	30
N3 - N1	0	0
N3 - N2	0	0
N3 - N4	10	10
N3 - N5	20	20
N4 - N1	0	0
N4 - N2	0	0
N4 - N3	0	5
N4 - N5	20	20
N5 - N1	0	0
N5 - N2	0	0
N5 - N3	0	0
N5 - N4	0	0

Intermediate calculations:

Name	Node	Flow	Demand
N1	1	60	0
N2	2	0	0
N3	3	0	0
N4	4	0	0
N5	5	-60	0

From	To	UnitFlow
1	2	1
1	3	1
1	4	1
1	5	0
2	1	0
2	3	0
2	4	0
2	5	0
3	1	0
3	2	0
3	4	0
3	5	0
4	1	0
4	2	0
4	3	0
4	5	0
5	1	0
5	2	0
5	3	0
5	4	0

Solver Parameters

Set Target Cell: $B$18

Equal To: Max Min Value of: 0

By Changing Cells:

$B$20:$B$39

Subject to the Constraints:

$B$20:$B$39 <= $C$20:$C$39
$F$20:$F$22 = $G$20:$G$22

Solve
Close
Guess
Options
Add
Change
Delete
Reset All
Help

FIGURE 6.35

Excel Solver solution of the maximum flow model of Example 6.4-2

that is needed now is to update Solver parameters to match the input data. Column B specifies the changing cells (arcs flow) of the problem. The range for Changing Cells must encompass all the arcs specified in column A (make sure that you give each node a name in the input data matrix, else column A will only show a hyphen in the associated cells). In the present example, cells B20:B39 provide Changing Cells range. Column C specifies the capacities of the arcs of the network (cells C20:C39).

The constraints of the model represent the flow balance equation for each node. The LP formulation in Section 6.4.3 shows that it is not necessary to construct flow equations for the first and last nodes of the network (nodes 1 and 5 in Figure 6.35). Thus, cells F20:F22 define the left-hand side and cells G20:G22 represent the right-hand side of the flow equations.

Based on given information, Solver parameters for the example in Figure 6.26 are entered as

```
Changing Cells: B20:B39
Constraints: B20:B39<=C20:C39     (Arc capacity)
             F20:F22=G20:G22      (Flow equations for nodes 2, 3, and 4)
```

Note that Target Cell is automated and need not be changed. The Equal to parameter is Max because this is a maximization problem.

The output in Figure 6.35 yields the solution (N1-N2 = 20, N1-N3 = 30, N1-N4 = 10, N2-N5 = 20, N3-N4 = 10, N3-N5 = 20, N4-N5 = 20) with a maximum flow of 60 units.

PROBLEM SET 6.4D

1. Solve Problem 2, Set 6.4b using Excel Solver.
2. Solve Problem 5, Set 6.4b using Excel Solver.

6.5 MINIMUM-COST CAPACITATED FLOW PROBLEM

The minimum-cost capacitated flow problem is based on the following assumptions:

1. A (nonnegative) unit flow cost is associated with each arc.
2. Arcs may have positive lower capacity limits.
3. Any node in the network may act as a source or as a sink.

The new model determines the flows in the different arcs that minimize the total cost while satisfying the flow restrictions on the arcs and the supply and demand amounts at the nodes. We first present the capacitated network flow model and its equivalent linear programming formulation. The linear programming formulation is the basis for the development of a special capacitated simplex algorithm for solving the network flow model. The section ends with a presentation of a spreadsheet template of the minimum-cost capacitated network.

6.5.1 Network Representation

Consider a capacitated network $G = (N, A)$, where N is the set of nodes, and A is the set of arcs and define

x_{ij} = amount of flow from node i to node j

u_{ij} (l_{ij}) = upper (lower) capacity of arc (i, j)

c_{ij} = unit flow cost from node i to node j

f_i = net flow at node i

Figure 6.36 depicts these definitions on arc (i, j). The label $[f_i]$ assumes a positive (negative) value when a net supply (demand) is associated with node i.

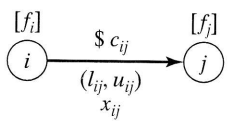


FIGURE 6.36
Capacitated arc with external flow

Example 6.5-1

GrainCo supplies corn from three silos to three poultry farms. The supply amounts at the three silos are 100, 200, and 50 thousand bushels; and the demand at the three farms is 150, 80, and 120 thousand bushels. GrainCo mostly uses railroads to transport the corn to the farms, with the exception of three routes where trucks are used.

Figure 6.37 shows the available route between the silos and the farms. The silos are represented by nodes 1, 2, and 3 whose supply amounts are [100], [200], and [50], respectively. The farms are represented by nodes 4, 5, and 6 whose demand amounts are [−150], [−80], and [−120], respectively. The routes allow transshipping between the silos. Arcs (1, 4), (3, 4), and (4, 6) are truck routes with minimum and maximum capacities. For example, the capacity of route (1, 4) is between 50 and 80 thousand bushels. All other routes use trainloads, whose maximum capacity is practically unlimited. The transportation costs per bushel are indicated on the respective arcs.

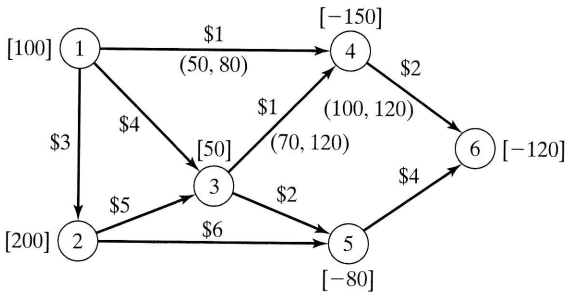


FIGURE 6.37
Capacitated network for Example 6.5-1

PROBLEM SET 6.5A

1. A product is manufactured to satisfy demand over a 4-period planning horizon according to the following data:

Period	Units of demand	Unit production cost ($)	Unit holding cost ($)
1	100	24	1
2	110	26	2
3	95	21	1
4	125	24	2

Given that no back-ordering is allowed, represent the problem as a network model.

2. In Problem 1, suppose that back-ordering is allowed at a penalty of $1.50 per unit per period. Formulate the problem as a network model.

3. In Problem 1, suppose that the production capacities of periods 1 to 4 are 110, 95, 125, and 100 units, respectively, in which case the given demand cannot be satisfied without back-ordering. Assuming that the penalty cost for back-ordering is \$1.50 per unit per period, formulate the problem as a network model.
4. Daw Chemical owns two plants that manufacture a basic chemical compound for two customers at the rate of 660 and 800 tons per month. The monthly production capacity of plant 1 is between 400 and 800 tons and that of plant 2 is between 450 and 900 tons. The production costs per ton in plants 1 and 2 are \$25 and \$28, respectively. Raw material for the plants is provided by two suppliers, who are contracted to ship at least 500 and 700 tons per month for plants 1 and 2 at the costs of \$200 and \$210 per ton, respectively. Daw Chemical also assumes the transportation cost of both the raw material and the final compound. The costs per ton of transporting the raw material from supplier 1 to plants 1 and 2 are \$10 and \$12. Similar costs from supplier 2 are \$9 and \$13, respectively. The transportation costs per ton from plant 1 to clients 1 and 2 are \$3 and \$4, and from plant 2 costs are \$5 and \$2, respectively. Assuming that 1 ton of raw material produces 1 ton of the final compound, formulate the problem as a network model.
5. Two nonintegrated public schools are required to change the racial balance of their enrollments by accepting minority students. Minority enrollment must be between 30% and 40% in both schools. Nonminority students live in two communities, and minority students live in three other communities. Traveled distances, in miles, from the five communities to the two schools are summarized in the following table:

		Round-trip miles from school to				
		Minority areas			*Nonminority areas*	
School	*Maximum enrollment*	*1*	*2*	*3*	*1*	*2*
1	1500	20	12	10	4	5
2	2000	15	18	8	6	5
Student population		500	450	300	1000	1000

Formulate the problem as a network model to determine the number of minority and nonminority students enrolled in each school.

6.5.2 Linear Programming Formulation

The formulation of the capacitated network model as a linear program provides the foundation for the development of the capacitated simplex algorithm, which we will present in the next section. Using the notation introduced in Section 6.5.1, the linear program for the capacitated network is given as

$$\text{Minimize } z = \sum_{(i,j)\in A}\sum c_{ij}x_{ij}$$

subject to

$$\sum_{(j,k)\in A} x_{jk} - \sum_{(i,j)\in A} x_{ij} = f_j,\ j \in N$$

$$l_{ij} \le x_{ij} \le u_{ij}$$

The equation for node j measures the net flow f_j in node j as

$$(\text{Outgoing flow from node } j) - (\text{Incoming flow into node } j) = f_j$$

Node j acts as a source if $f_j > 0$ and as a sink if $f_j < 0$.

We can always remove the lower bound l_{ij} from the constraints by using the substitution

$$x_{ij} = x'_{ij} + l_{ij}$$

The new flow variable, x'_{ij}, has an upper limit of $u_{ij} - l_{ij}$. Additionally, the net flow at node i becomes $f_i - l_{ij}$, and that at node j is $f_j + l_{ij}$. Figure 6.38 shows the transformation of activity (i, j) after the lower bound is substituted out.

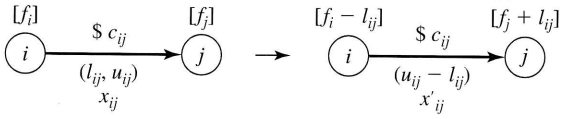


FIGURE 6.38
Removal of the lower bound in arcs

Example 6.5-2

Write the linear program for the network in Figure 6.37, before and after the lower bounds are substituted out.

The main constraints of the linear program relate the input-output flow at each node, which yields the following LP:

	x_{12}	x_{13}	x_{14}	x_{23}	x_{25}	x_{34}	x_{35}	x_{46}	x_{56}	
Minimize	3	4	1	5	6	1	2	2	4	
Node 1	1	1	1							= 100
Node 2	−1			1	1					= 200
Node 3		−1		−1		1	1			= 50
Node 4			−1			−1		1		=−150
Node 5					−1		−1		1	= −80
Node 6								−1	−1	=−120
Lower bounds	0	0	50	0	0	70	0	100	0	
Upper bounds	∞	∞	80	∞	∞	120	∞	120	∞	

Note the arrangement of the constraints coefficients. The column associated with variable x_{ij} has exactly one +1 in row i and one −1 in row j. The rest of the coefficients are 0. This structure is typical of network flow models.

The variables with lower bounds are substituted as

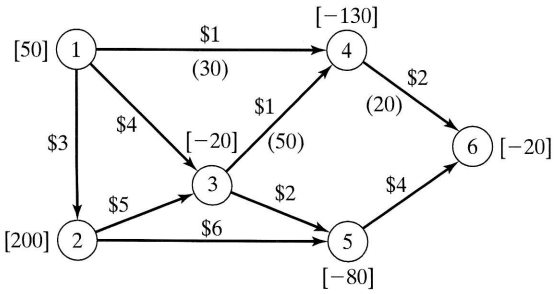
$$x_{14} = x'_{14} + 50$$
$$x_{34} = x'_{34} + 70$$
$$x_{46} = x'_{46} + 100$$

The resulting linear program is

	x_{12}	x_{13}	x'_{14}	x_{23}	x_{25}	x'_{34}	x_{35}	x'_{46}	x_{56}	
Minimize	3	4	1	5	6	1	2	2	4	
Node 1	1	1	1							= 50
Node 2	−1			1	1					= 200
Node 3		−1		−1		1	1			= −20
Node 4			−1			−1		1		=−130
Node 5					−1		−1		1	= −80
Node 6								−1	−1	= −20
Upper bounds	∞	∞	30	∞	∞	50	∞	20	∞	

The corresponding network after substituting out the lower bounds is shown in Figure 6.39. Note that the lower-bound substitution can be effected directly from Figure 6.37 using the substitution in Figure 6.38, and without the need to express the problem as a linear program first.

FIGURE 6.39
Network of Example 6.5-2 after substituting out lower bounds



Example 6.5-3 (Employment Scheduling)

This example illustrates a network model that initially does not satisfy the "node flow" requirement (i.e., node output flow less node input flow equals node net flow), but that can be converted to this form readily through special manipulation of the constraints of the linear program.

Tempo Employment Agency has a contract to provide workers over the next 4 months (January to April) according to the following schedule:

Month	Jan.	Feb.	Mar.	Apr.
No. of workers	100	120	80	170

Because of change in demand, it may be economical to retain more workers than needed in a given month. The cost of recruiting and maintaining a worker is a function of their employment period as the following table shows:

Employment period (months)	1	2	3	4
Cost per worker ($)	100	130	180	220

Let

x_{ij} = number of workers hired at the *start* of month i and terminated at the *start* of month j

For example, x_{12} gives the number of workers hired in January for 1 month only.

To formulate the problem as a linear program for the 4-month period, we add May as a dummy month (month 5), so that x_{45} defines hiring in April for April. The constraints recognize that the demand for period k can be satisfied by all x_{ij} such that $i \le k < j$. Letting $s_i \ge 0$ be the surplus number of workers in month i, the linear program is given as

	x_{12}	x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{34}	x_{35}	x_{45}	s_1	s_2	s_3	s_4	
Minimize	100	130	180	220	100	130	180	100	130	100					
Jan.	1	1	1	1							−1				=100
Feb.		1	1	1	1	1	1					−1			=120
Mar.			1	1		1	1	1	1				−1		= 80
Apr.				1			1		1	1				−1	=170

The preceding LP does not have the (−1, +1) special structure of the network flow model (see Example 6.5-2). Nevertheless, the given linear program can be converted into an equivalent network flow model by using the following arithmetic manipulations:

1. In an n-equation linear program, create a new equation, $n + 1$, by multiplying equation n by −1.
2. Leave equation 1 unchanged.
3. For $i = 2, 3, \ldots, n$, replace each equation i with (equation i) − (equation $i - 1$).

The application of these manipulations to the employment scheduling example yields the following linear program whose structure fits the network flow model:

	x_{12}	x_{13}	x_{14}	x_{15}	x_{23}	x_{24}	x_{25}	x_{34}	x_{35}	x_{45}	s_1	s_2	s_3	s_4	
Minimize	100	130	180	220	100	130	180	100	130	100					
Jan.	1	1	1	1							−1				= 100
Feb.	−1				1	1	1				1	−1			= 20
Mar.		−1			−1			1	1			1	−1		= −40
Apr.			−1			−1		−1		1			1	−1	= 90
May				−1			−1		−1	−1				1	=−170

Using the preceding formulation, the employment scheduling model can be represented equivalently by the minimum-cost flow network shown in Figure 6.40. Actually, because the arcs have no upper bounds, the problem can be solved also as a transshipment model (see Section 5.5).

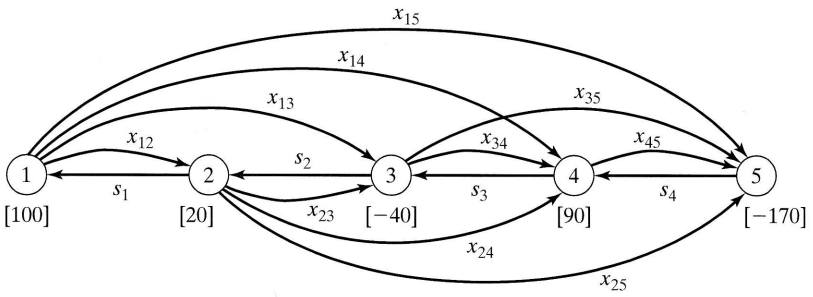


FIGURE 6.40
Network representation of employment scheduling problem

PROBLEM SET 6.5B

1. Write the linear program associated with the minimum-cost flow network in Figure 6.41, before and after the lower bounds are substituted out.

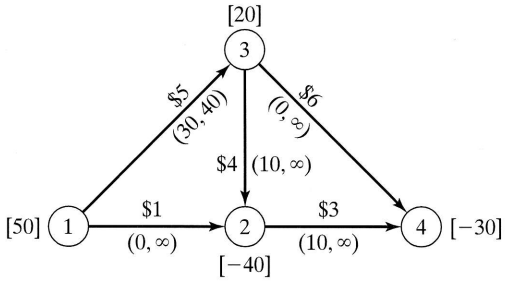


FIGURE 6.41
Network for Problem 1, Set 6.5b

2. Use inspection to find a feasible solution to the minimum-cost network model of the employment scheduling problem in Example 6.5-3 (Figure 6.40). Interpret the solution by showing the pattern of hiring and firing that satisfies the demand for each month, and compute the associated total cost.

3. Reformulate the employment scheduling model of Example 6.5-3, assuming that a worker must be hired for at least 2 months. Write the linear program, and convert it to a minimum-cost flow network.

4. Develop the linear program and the associated minimum-cost flow network for the employment scheduling model of Example 6.5-3 using the following 5-month demand data. The per worker costs of recruiting and maintaining a worker for periods of 1 to 5 months are \$50, \$70, \$85, \$100, and \$130, respectively.

(a)

Month	1	2	3	4	5
No. of workers	300	180	90	170	200

(b)

Month	1	2	3	4	5
No. of workers	200	220	300	50	240

5. *Conversion of a Capacitated Network into an Uncapacitated Network.* Show that an arc $(i \rightarrow j)$ with capacitated flow $x_{ij} \leq u_{ij}$ can be replaced with two *uncapacitated* arcs $(i \rightarrow k)$ and $(j \rightarrow k)$ with a net (output) flow of $[-u_{ij}]$ at node k and an additional (input) flow of $[+u_{ij}]$ at node j. The result is that the *capacitated* network can be converted to an *uncapacitated* transportation cost model (Section 5.1). Apply the resulting transformation to the network in Figure 6.42 and find the optimum solution to the original network by applying TORA to the uncapacitated transportation model.

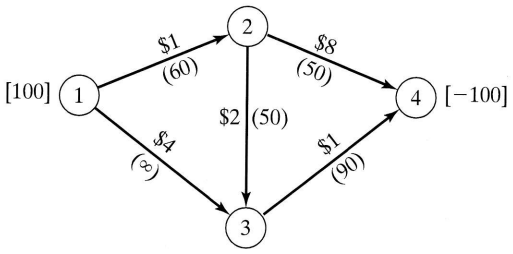


FIGURE 6.42
Network for Problem 5, Set 6.5b

6.5.3 Capacitated Network Simplex Algorithm

The algorithm is based on the exact steps of the regular simplex method, but designed to exploit the special network structure of the minimum-cost flow model.

Given f_i is the net flow at node i as defined in the linear program of Section 6.5.2, the capacitated simplex algorithm stipulates that the network must satisfy

$$\sum_{i=1}^{n} f_i = 0$$

The condition says that the total supply in the network equals the total demand. We can always satisfy this requirement by adding a balancing dummy source or destination, which we connect to all other nodes in the network by zero unit cost and infinite capacity arcs. However, the balancing of the network does not guarantee a feasible solution as this depends on the restricting capacities of the arcs.

We will now present the steps of the capacitated algorithm. Familiarity with the simplex method and duality theory (Chapters 3 and 4) is essential. Also, knowledge of the upper-bounded simplex method (Section 7.3) is helpful.

Step 0. Determine a starting basic feasible solution (set of arcs) for the network. Go to step 1.

Step 1. Determine an entering arc (variable) using the simplex method optimality condition. If the solution is optimal, stop; otherwise, go to step 2.

Step 2. Determine the leaving arc (variable) using the simplex method feasibility condition. Determine the new solution, and then go to step 1.

An n-node network with zero net flow (i.e., $f_1 + f_2 + \ldots + f_n = 0$) consists of $n - 1$ *independent* constraint equations. Thus, an associated basic solution must include $n - 1$ arcs. It can be proved that a basic solution always corresponds to a *spanning tree* of the network (see Section 6.2).

The entering arc (step 1) is determined by computing $z_{ij} - c_{ij}$, the objective coefficients, for all the current nonbasic arcs (i, j). If $z_{ij} - c_{ij} \leq 0$ for all i and j, the current basis is optimum. Otherwise, we select the nonbasic arc with the most positive $z_{ij} - c_{ij}$ to enter the basis.

The computation of objective coefficients is based on duality, exactly as we did with the transportation model (see Section 5.3.4). Using the linear program defined in Section 6.5.2, let w_i be the dual variable associated with the constraint of node i; then the dual problem (excluding the upper bounds) is given as

$$\text{Maximize } z = \sum_{i=1}^{n} f_i w_i$$

subject to

$$w_i - w_j \leq c_{ij}, (i, j) \in A$$

$$w_i \text{ unrestricted in sign, } i = 1, 2, \ldots n$$

From the theory of linear programming, we have

$$w_i - w_j = c_{ij}, \text{ for basic arc } (i, j)$$

Because the original linear program (Section 6.5.2) has one redundant constraint by definition, we can assign an arbitrary value to one of the dual variables (compare with the transportation algorithm, Section 5.3). For convenience, we will set $w_1 = 0$. We then solve the (basic) equations $w_i - w_j = c_{ij}$ to determine the remaining dual values. From Section 4.2.3, Method 2, we know that the objective coefficient of nonbasic x_{ij} is the difference between the left-hand side and the right-hand side of the dual associated dual constraint—that is

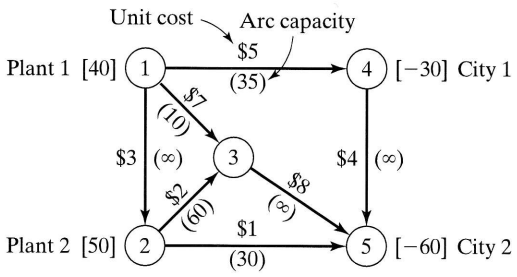
$$z_{ij} - c_{ij} = w_i - w_j - c_{ij}$$

The only remaining detail is to show how the leaving variable is determined. We do so by using a numeric example.

Example 6.5-4

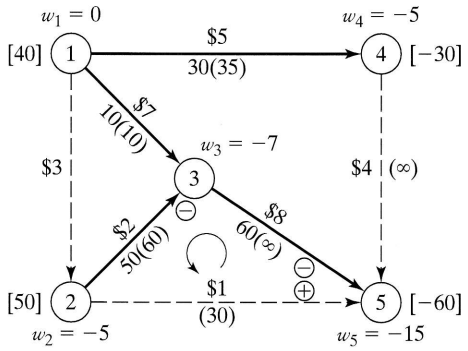
A network of pipelines connects two water desalinization plants to two cities. The daily supply amounts at the two plants are 40 and 50 million gallons, and the daily demand amounts at cities 1 and 2 are 30 and 60 million gallons. Nodes 1 and 2 represent plants 1 and 2, and nodes 4 and 5 represent cities 1 and 2. Node 3 is a booster station between the plants and the cities. The model is already balanced because the supply at nodes 1 and 2 equals the demand at nodes 4 and 5. Figure 6.43 gives the associated network.

FIGURE 6.43
Network for Example 6.5-4



Iteration 0.

Step 0. *Determination of a Starting Basic Feasible Solution:* The starting *feasible* spanning tree in Figure 6.44 (shown with solid arcs) is obtained by inspection. Normally, we use an artificial variable technique to find such a solution (for details, see Bazaraa et al., 1990, pp. 440–46).



$z_{12} - c_{12} = 0 - (-5) - 3 = 2$
$z_{25} - c_{25} = -5 - (-15) - 1 = \mathbf{9}$
$z_{45} - c_{45} = -5 - (-15) - 4 = 6$

Arc (2, 5) reaches upper bound at 30.

Substitute $x_{25} = 30 - x_{52}$.

Reduce x_{23} and x_{35} each by 30.

FIGURE 6.44
Network for iteration 0

In Figure 6.44, the basic feasible solution consists of (solid) arcs (1, 3), (1,4), (2, 3), and (3, 5) with the *feasible* flows of 10, 30, 50, and 60 units, respectively. This leaves (dashed) arcs (1, 2), (2, 5), and (4, 5) to represent the nonbasic variables. The notation $x(c)$ shown on the arcs indicates that a flow of x units is assigned to an arc with capacity c. The default values for x and c are 0 and ∞, respectively.

Iteration 1.

Step 1. *Determination of the Entering Arc:* We obtain the dual values by solving the current basic equations

$$w_1 = 0$$
$$w_i - w_j = c_{ij}, \text{ for basic } (i, j)$$

We thus get,

$$\text{Arc } (1, 3) : w_1 - w_3 = 7, \text{ hence } w_3 = -7$$
$$\text{Arc } (1, 4) : w_1 - w_4 = 5, \text{ hence } w_4 = -5$$
$$\text{Arc } (2, 3) : w_2 - w_3 = 2, \text{ hence } w_2 = -5$$
$$\text{Arc } (3, 5) : w_3 - w_5 = 8, \text{ hence } w_5 = -15$$

Now, we compute $z_{ij} - c_{ij}$ for the nonbasic variables as

$$\text{Arc } (1, 2) : w_1 - w_2 - c_{12} = 0 - (-5) - 3 = 2$$
$$\text{Arc } (2, 5) : w_2 - w_5 - c_{25} = (-5) - (-15) - 1 = 9$$
$$\text{Arc } (4, 5) : w_4 - w_5 - c_{45} = (-5) - (-15) - 4 = 6$$

Thus, arc (2, 5) enters the basic solution.

Step 2. *Determination of the Leaving Arc:* From Figure 6.44, arc (2, 5) forms a loop with basic arcs (2, 3) and (3, 5). *From the definition of the spanning tree, no other loop can be formed.* Because the flow in the new arc (2, 5) must be increased, we adjust the flow in the arcs of the loop by an equal amount to maintain the feasibility of the new solution. To achieve this, we identify the positive (+) flow in the loop by the direction of flow of the entering arc (i.e., from 2 to 5). We then assign (+) or (−) to the remaining arcs of the loop, depending on whether the flow of each arc is *with* or *against* the direction of flow of the entering arc. These sign conventions are shown in Figure 6.44.

Determination of the maximum level of flow in the entering arc (2, 5) is based on two conditions:

1. New flow in current basic arcs of the loop cannot be negative.
2. New flow in the entering arc cannot exceed its capacity.

The application of condition 1 shows that the flows in arcs (2, 3) and (3, 5) cannot be decreased by more than min {50, 60} = 50 units. Condition 2 stipulates that the flow in arc (2, 5) can be increased to at most the arc capacity (=30 units). Thus, the maximum flow change in the loop is min {30, 50} = 30 units. The new flows in the loop are thus 30 units in arc (2, 5), 50 − 30 = 20 units in arc (2, 3), and 60 − 30 = 30 units in arc (3, 5).

Because none of the current basic arcs leave the basis at zero level, the new arc (2, 5) must remain nonbasic at the upper bound. However, to avoid dealing with nonbasic arcs that are at capacity (or upper bound) level, we implement the substitution

$$x_{25} = 30 - x_{52},\ 0 \le x_{52} \le 30$$

This substitution is effected in the flow equations associated with nodes 2 and 5 as follows. Consider

$$\text{Current flow equation at node 2: } 50 + x_{12} = x_{23} + x_{25}$$

$$\text{Current flow equation at node 5: } x_{25} + x_{35} + x_{45} = 60$$

Then, the substitution $x_{25} = 30 - x_{52}$ gives

$$\text{New flow equation at node 2: } 20 + x_{12} + x_{52} = x_{23}$$

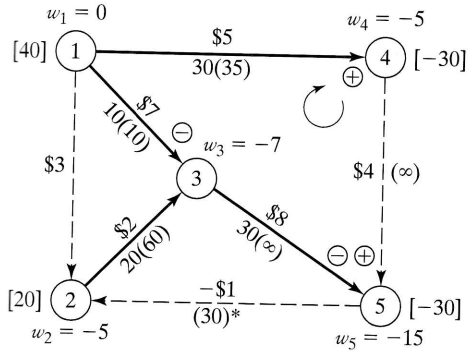
$$\text{New flow equation at node 5: } x_{35} + x_{45} = x_{52} + 30$$

The results of these changes are shown in Figure 6.45. The direction of flow in arc (2, 5) is now reversed to $5 \rightarrow 2$ with $x_{52} = 0$, as desired. The substitution also requires changing the unit cost of arc (5, 2) to −$1. We will indicate this direction reversal on the network by tagging the arc with an asterisk.

Iteration 2. Figure 6.45 summarizes the new values of $z_{ij} - c_{ij}$ (verify!) and shows that arc (4, 5) enters the basic solution. It also defines the loop associated with the new entering arc and assigns the signs to its arcs.

The flow in arc (4, 5) can be increased by the smallest of

1. Maximum allowable *increase* in entering arc (4, 5) = ∞
2. Maximum allowable *increase* in arc (1, 4) = 35 − 30 = 5 units



$z_{12} - c_{12} = 0 - (-5) - 3 = 2$
$z_{52} - c_{52} = -15 - (-5) - (-1) = -9$
$z_{45} - c_{45} = -5 - (-15) - 4 = \mathbf{6}$

Arc (4, 5) enters at level 5.

Arc (1, 4) leaves at upper bound.

Substitute $x_{14} = 35 - x_{41}$.

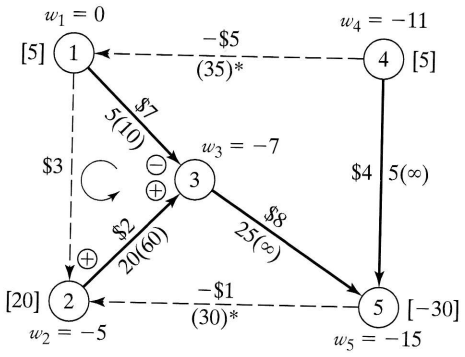
Reduce x_{13} and x_{35} each by 5.

FIGURE 6.45
Network for iteration 1

3. Maximum allowable *decrease* in arc (1, 3) = 10 units
4. Maximum allowable *decrease* in arc (3, 5) = 30 units

Thus, the flow in arc (4, 5) can be increased to 5 units, which will make (4, 5) basic and will force basic arc (1, 4) to be nonbasic at its upper bound (= 35).

Using the substitution $x_{14} = 35 - x_{41}$, the network is changed as shown in Figure 6.46, with arcs (1, 3), (2, 3), (3, 5), and (4, 5) forming the basic (spanning tree) solution. The reversal of flow in arc (1, 4) changes its unit cost to −$5. Also, convince yourself that the substitution in the flow equations of nodes 1 and 4 will net 5 *input* units at each node.



$z_{12} - c_{12} = 0 - (-5) - 3 = \mathbf{2}$
$z_{41} - c_{41} = -11 - 0 - (-5) = -6$
$z_{52} - c_{52} = -15 - (-5) - (-1) = -9$

Arc (1, 2) enters at level 5.

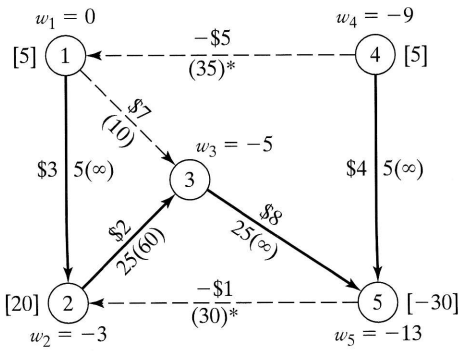
Arc (1, 3) leaves at level 0.

Increase x_{23} by 5.

FIGURE 6.46
Network for iteration 2

Iteration 3. The computations of the new $z_{ij} - c_{ij}$ for the nonbasic arcs (1, 2), (4, 1), and (5, 2) are summarized in Figure 6.46, which shows that arc (1, 2) enters at level 5, and arc (1, 3) becomes nonbasic at level 0. The new solution is depicted in Figure 6.47.

Iteration 4. The new $z_{ij} - c_{ij}$ in Figure 6.47 shows that the solution is optimum. The values of the original variables are obtained by back substitution as shown in Figure 6.47.



$z_{13} - c_{13} = 0 - (-5) - 7 = -2$
$z_{41} - c_{41} = -9 - 0 - (-5) = -4$
$z_{52} - c_{52} = -13 - (-3) - (-1) = -9$

Optimum solution:
$x_{12} = 5, x_{13} = 0$
$x_{14} = 35 - 0 = 35$
$x_{23} = 25$
$x_{25} = 30 - 0 = 30$
$x_{35} = 25, x_{45} = 5$
Total cost = $490

FIGURE 6.47
Network for iteration 3

PROBLEM SET 6.5C

1. Solve Problem 1, Set 6.5a by the capacitated simplex algorithm, and also show that it can be solved by the transshipment model.
2. Solve Problem 2, Set 6.5a by the capacitated simplex algorithm, and also show that it can be solved by the transshipment model.
3. Solve Problem 3, Set 6.5a by the capacitated simplex algorithm.
4. Solve Problem 4, Set 6.5a by the capacitated simplex algorithm.
5. Solve Problem 5, Set 6.5a by the capacitated simplex algorithm.
6. Solve the employment scheduling problem of Example 6.5-3 by the capacitated simplex algorithm.
7. Wyoming Electric uses existing slurry pipes to transport coal (carried by pumped water) from three mining areas (1, 2, and 3) to three power plants (4, 5, and 6). Each pipe can transport at most 10 tons per hour. The transportation costs per ton and the supply and demand per hour are given in the following table.

	4	5	6	Supply
1	$5	$8	$4	**8**
2	$6	$9	$12	**10**
3	$3	$1	$5	**18**
Demand	**16**	**6**	**14**	

   Determine the optimum shipping schedule.
8. The network in Figure 6.48 gives the distances among seven cities. Use the capacitated simplex algorithm to find the shortest distance between nodes 1 and 7. (*Hint:* Assume that nodes 1 and 7 have net flows of [+1] and [−1], respectively. All the other nodes have zero net flow.)
9. Show how the capacitated minimum-cost flow model can be specialized to represent the maximum flow model of Section 6.4. Apply the transformation to the network in Example 6.4-2. For convenience, assume that the flow capacity from 4 to 3 is zero. All the remaining data are unchanged.

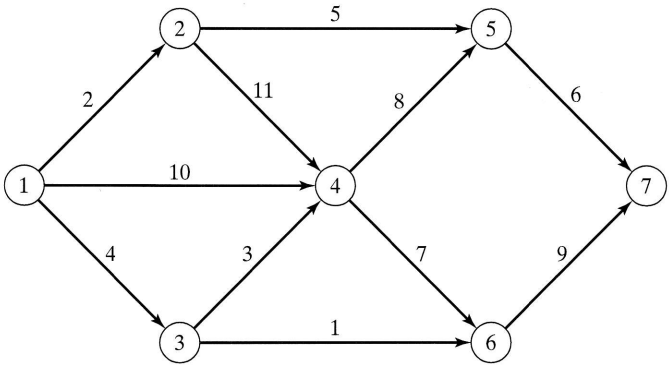


FIGURE 6.48
Network for Problem 8, Set 6.5c

6.5.4 Excel Spreadsheet Solution of the Minimum-Cost Capacitated Flow Model

As in the cases of the shortest-route and maximum flow models, the Excel spreadsheet developed for the general transportation model (Section 5.3.3) applies readily to the capacitated network flow model. Figure 6.49 shows the application of the spreadsheet to Example 6.5-4 (file ch6SolverMinCostCapacitatedNetwork.xls). The spreadsheet is designed for networks with a maximum of 10 nodes. In the capacity matrix (cells N6:W15),[5] a blank entry signifies an infinite capacity arc. A nonexisting arc is represented by a zero-capacity entry. As an illustration, in Example 6.5-4, infinite capacity arc 1-2 is represented by a blank entry in cell O6, and nonexisting arc 3-4 is shown by a zero entry in cell Q8. The unit cost matrix resides in cells B6:K15. We arbitrarily assign zero unit cost to all nonexisting arcs.

Once the unit cost and capacity matrices are created, the remainder of the spreadsheet (*intermediate calculations* and *optimum solution* sections) is created automatically, delineating the cells needed to update Solver parameters for Changing Cells and Constraints. Target Cell is already defined for any network (with 10 nodes or less). Specifically, for Example 6.5-4, we have,

```
Changing cells: B20:B39
Constraints: B20:B39<=C20:C39     (Arc capacity)
             F19:F23=G19:G23      (Node flow equation)
```

Figure 6.49 provides the following solution: N1-N2 = 5, N1-N4 = 35, N2-N3 = 25, N2-N5 = 30, N3-N5 = 25, and N4-N5 = 5. The associated total cost is $490.

PROBLEM SET 6.5D

1. Solve the following problem using the spreadsheet in Section 6.5.4:
   (a) Problem 3, Set 6.5c
   (b) Problem 4, Set 6.5c

[5]In Figure 6.49, rows 11 through 15 and column K are hidden to conserve space.

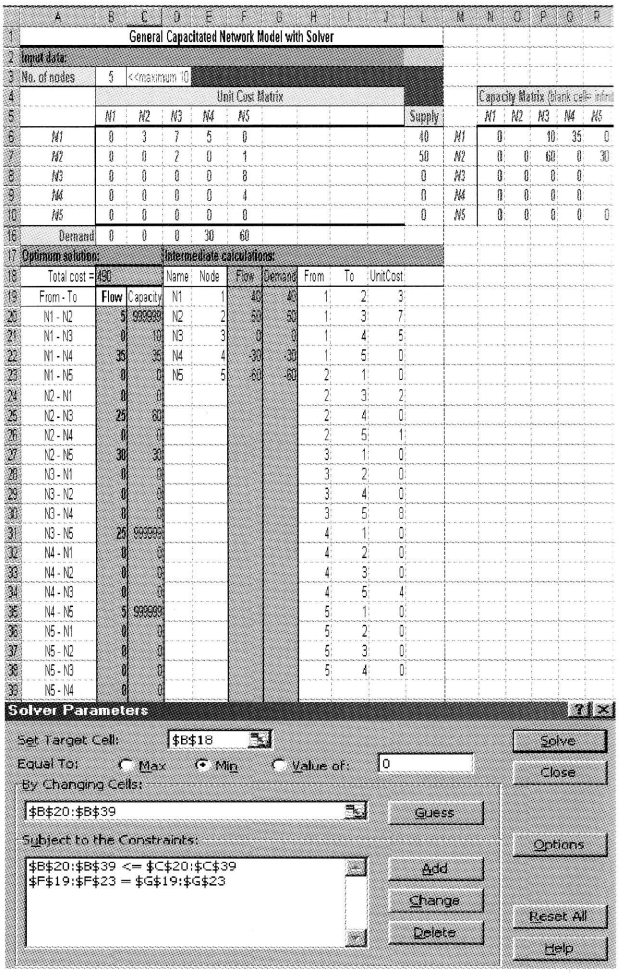


FIGURE 6.49
Excel Solver output for Example 6.5-4

(c) Problem 7, Set 6.5c

(d) Problem 8, Set 6.5c

6.6 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 consuming time and resources. The objective of CPM and PERT is to provide analytic means for scheduling the activities. Figure 6.50 summarizes the steps of the techniques. First, we define the activities of the project, their precedence relationships, and their

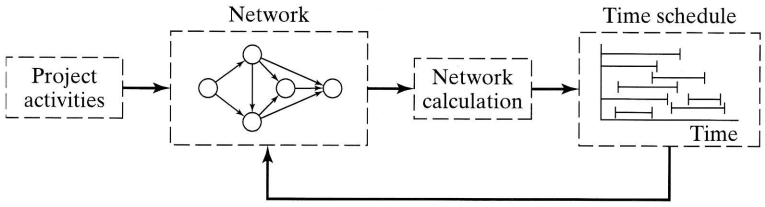


FIGURE 6.50
Phases for project planning with CPM-PERT

time requirements. Next, the project is translated into a network that shows the precedence relationships among the activities. The third step involves specific network computations that form the basis for the development of the time schedule for the project.

During the execution of the project, the schedule may not be realized as planned, causing some of the activities to be expedited or delayed. In this case, it will be necessary to update the schedule 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 6.50.

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

6.6.1 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 of the project.

Two 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 6.51 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 dashed arc, consumes no time or resources. Inserting a dummy activity in one of the four ways shown in Figure 6.51, we maintain the concurrence of A and B, and also 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?*

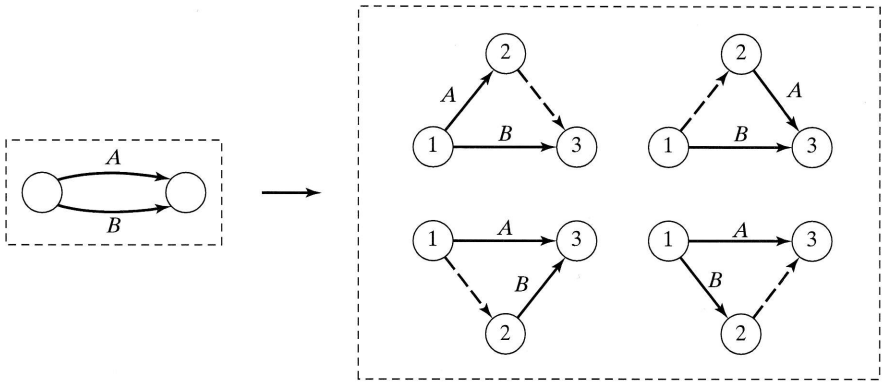


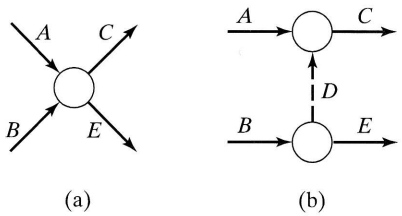
FIGURE 6.51
Use of dummy activity to produce unique representation of concurrent activities A and B

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 after B only has been completed.

Part (a) of Figure 6.52 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.

FIGURE 6.52
Use of dummy activity to ensure correct precedence relationship



Example 6.6-1

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. Develop the associated network for the project.

Activity	Predecessor(s)	Duration (weeks)
A: Manuscript proofreading by editor	—	3
B: Sample pages prepared by typesetter	—	2
C: Book cover design	—	4
D: Preparation of artwork for book figures	—	3
E: Author's approval of edited manuscript and sample pages	A, B	2

F:	Book typesetting	*E*	2
G:	Author checks typeset pages	*F*	2
H:	Author checks artwork	*D*	1
I:	Production of printing plates	*G, H*	2
J:	Book production and binding	*C, I*	4

Figure 6.53 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*. The numbering of the nodes is done in a manner that indicates the direction of progress in the project.

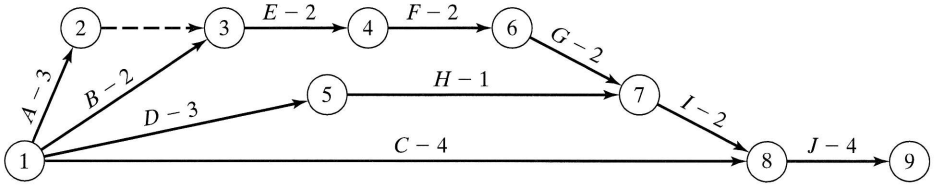


FIGURE 6.53
Project network for Example 6.6-1

PROBLEM SET 6.6A

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 connected 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. After each section of the footings is completed, an additional 5% of the plumbing can be started provided that the preceding 5% portion is completed. 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. Develop 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*	.5
C: Requisition stores	*A*	1
D: Scout job	*A*	.5
E: Secure poles and material	*C, D*	3
F: Distribute poles	*E*	3.5
G: Pole location coordination	*D*	.5
H: Re-stake	*G*	.5
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*	.1
Q: Energize and switch new line	*P*	.5
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

6.6.2 Critical Path (CPM) Computations

The ultimate result in CPM is the construction of the time schedule for the project (see Figure 6.50). 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 may 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

$$\square_j = \text{Earliest occurrence time of event } j$$

$$\Delta_j = \text{Latest occurrence time of event } j$$

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

The definitions of the *earliest* and *latest* occurrence times 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, □). 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.

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 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 *noncritical.*

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

Example 6.6-2

Determine the critical path for the project network in Figure 6.54. 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. $\square_5 = \max\{\square_3 + D_{35}, \square_4 + D_{45}\} = \max\{8 + 2, 13 + 0\} = 13$
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.

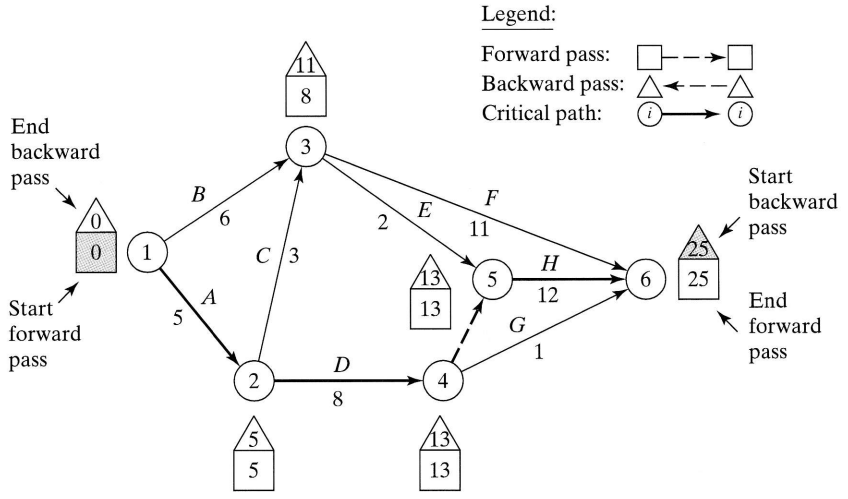
Backward Pass

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. $\Delta_1 = \min\{\Delta_3 - D_{13}, \Delta_2 - D_{12}\} = \min\{11 - 6, 5 - 5\} = 0$

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

The forward and backward pass computations are summarized in Figure 6.54. The rules for determining the critical activities show that the critical path is defined by $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$, which 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 not critical.

FIGURE 6.54
Forward and backward pass calculations for the project of Example 6.6-2



PROBLEM SET 6.6B

1. Determine the critical path for the project network in Figure 6.55.

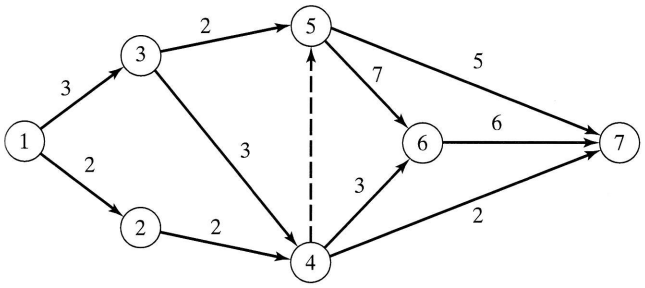
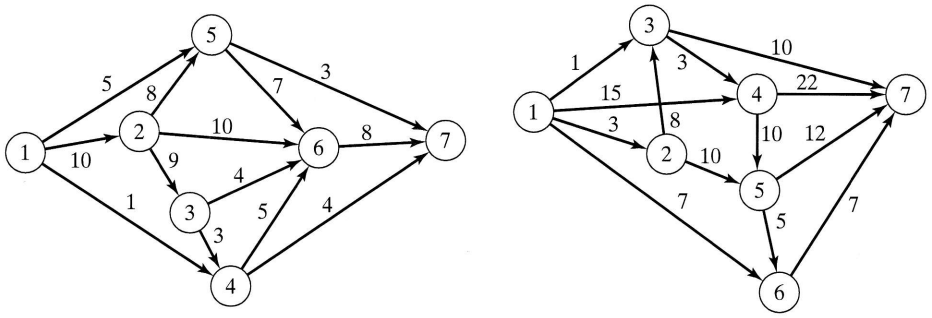


FIGURE 6.55
Project network for Problem 1, Set 6.6b

2. Determine the critical path for the project networks in Figure 6.56.



Project (a) Project (b)

FIGURE 6.56
Project network for Problem 2, Set 6.6b

3. Determine the critical path for the project in Problem 6, Set 6.6a.
4. Determine the critical path for the project in Problem 8, Set 6.6a.
5. Determine the critical path for the project in Problem 9, Set 6.6a.
6. Determine the critical path for the project in Problem 10, Set 6.6a.

6.6.3 Construction of the Time Schedule

This section shows how the information obtained from the calculations in Section 6.6.2 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 $(\square_i, \Delta_j)$ delineates the (maximum) span during which activity (i, j) may be scheduled.

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

Example 6.6-3

Determine the time schedule for the project of Example 6.6-2 (Figure 6.54).

We can get a preliminary time schedule for the different activities of the project by delineating their respective time spans as shown in Figure 6.57. Two observations are in order.

1. The critical activities (shown by solid lines) must be scheduled 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) encompass spans that are larger than their respective durations, thus allowing slack (or "leeway") in scheduling them within their allotted spans.

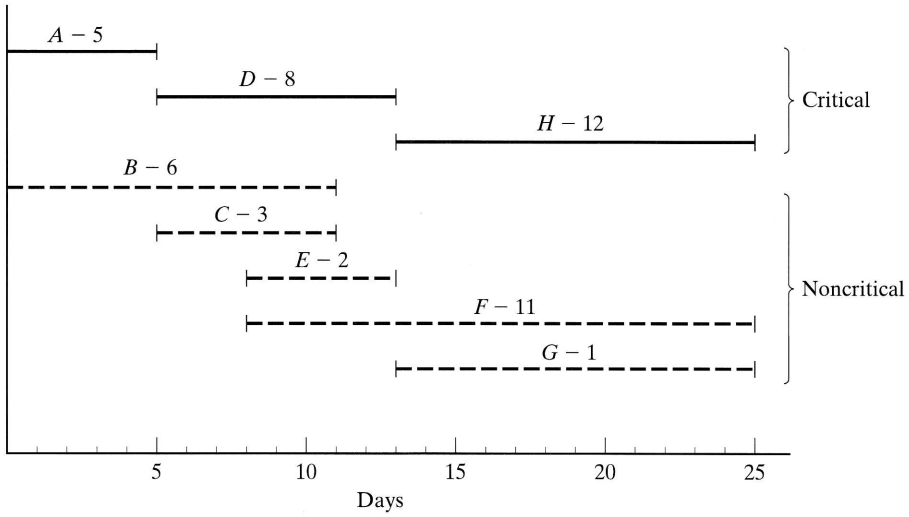


FIGURE 6.57

Preliminary schedule for the project of Example 6.6-2

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 time. For example, in Figure 6.57, 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 6.57. In the project network (Figure 6.54), though C must

be completed before E, the spans of C and E in Figure 6.57 allow us to schedule C between times 6 and 9, and E between times 8 and 10. These spans, however, do not ensure that C will 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 two most common floats are the **total float** and the **free float**.

Figure 6.58 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,

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

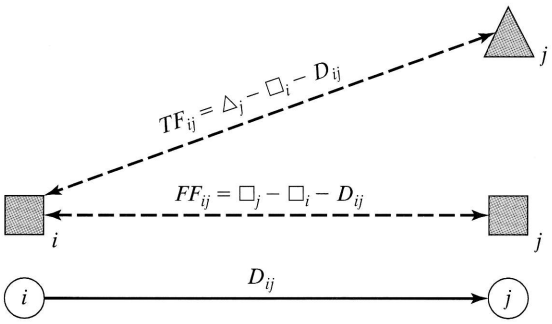


FIGURE 6.58
Computation of total and free floats

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 activity (i, j) 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 accompanied by 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.