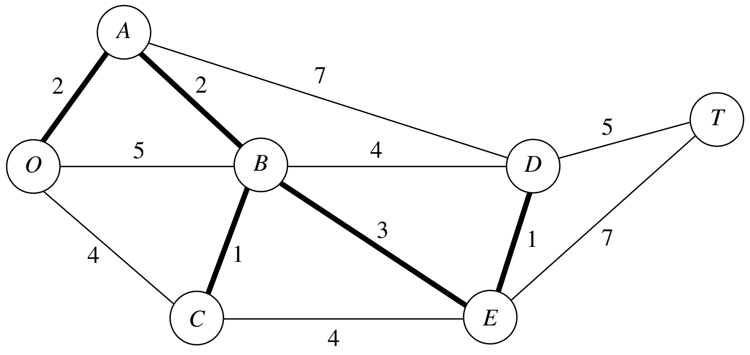
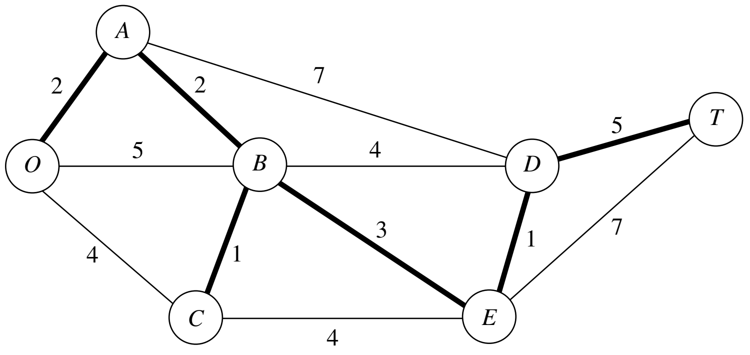
The unconnected node closest to node *O*, *A*, *B*, *C*, or *E* is node *D* (closest to *E*). Connect node *D* to node *E*.



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



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

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

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

MPL 9.5 THE MAXIMUM FLOW PROBLEM

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

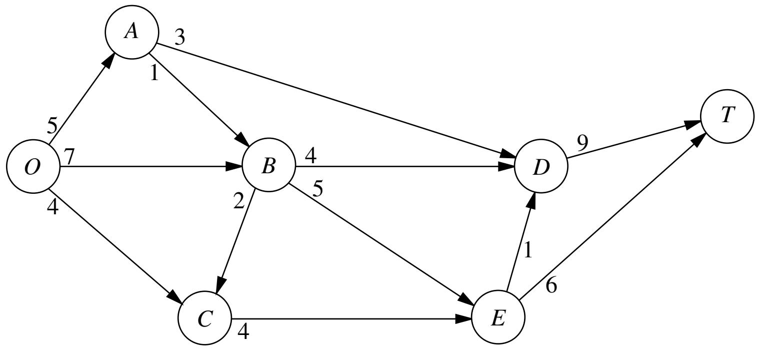


FIGURE 9.6
The Seervada Park maximum flow problem.

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

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

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

Some Applications

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

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

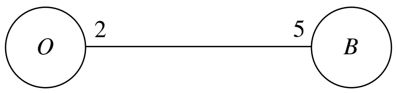
3. Maximize the flow of oil through a system of pipelines.
4. Maximize the flow of water through a system of aqueducts.
5. Maximize the flow of vehicles through a transportation network.

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

An Algorithm

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

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



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

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

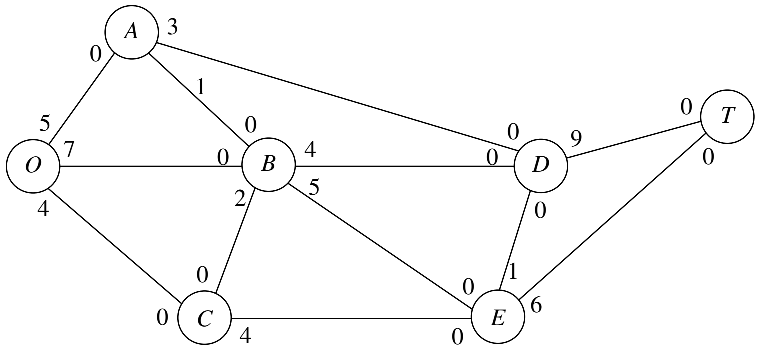


FIGURE 9.7
The initial residual network for the Seervada Park maximum flow problem.

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

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

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

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

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

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

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

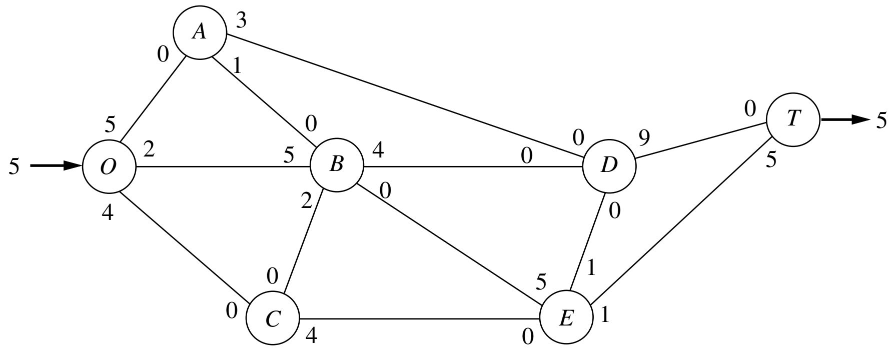
[1]It is assumed that the arc capacities are either integers or rational numbers.

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

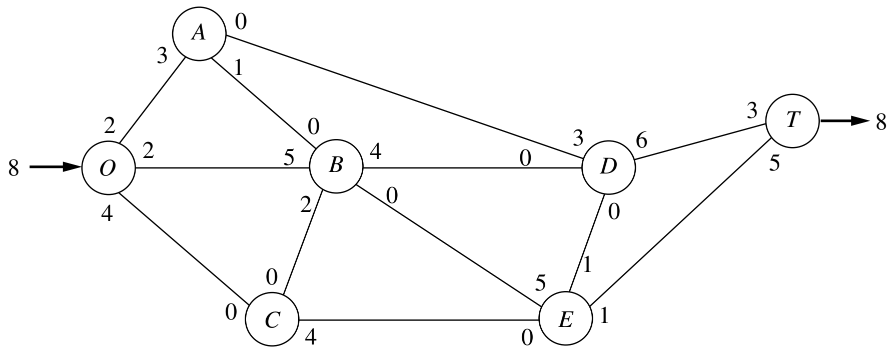
Applying This Algorithm to the Seervada Park Maximum Flow Problem

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

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

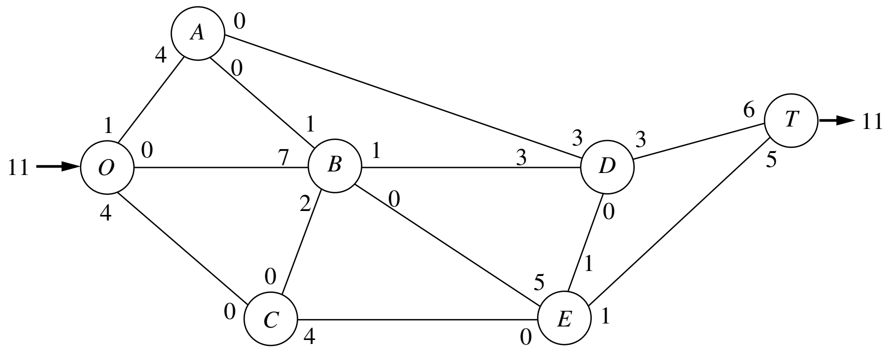


Iteration 2: Assign a flow of 3 to the augmenting path $O \rightarrow A \rightarrow D \rightarrow T$. The resulting residual network is



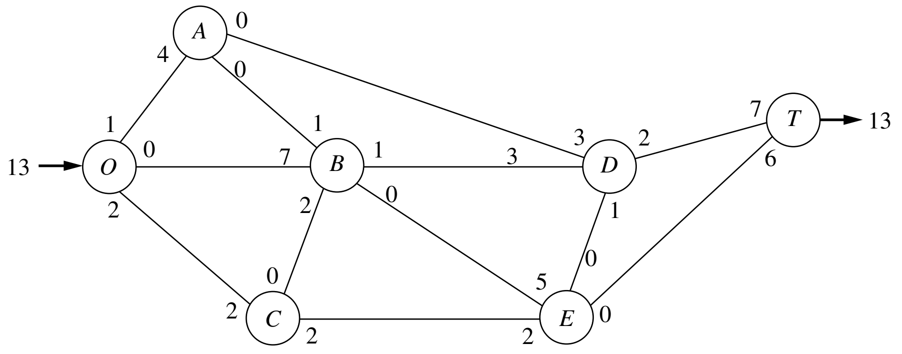
Iteration 3: Assign a flow of 1 to the augmenting path $O \rightarrow A \rightarrow B \rightarrow D \rightarrow T$.

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

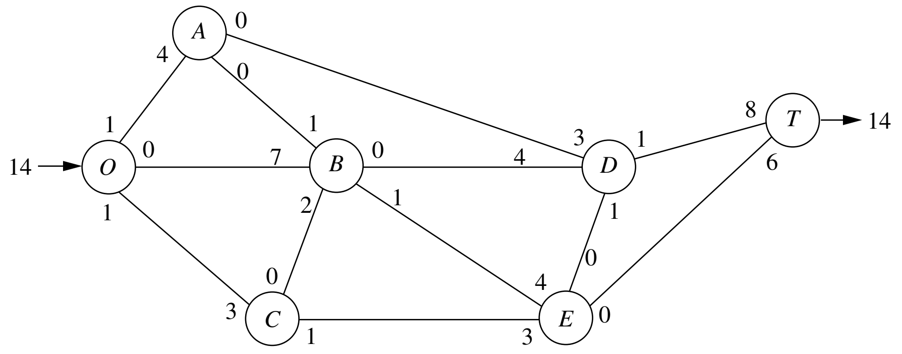


Iteration 5: Assign a flow of 1 to the augmenting path $O \to C \to E \to D \to T$.

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



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



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

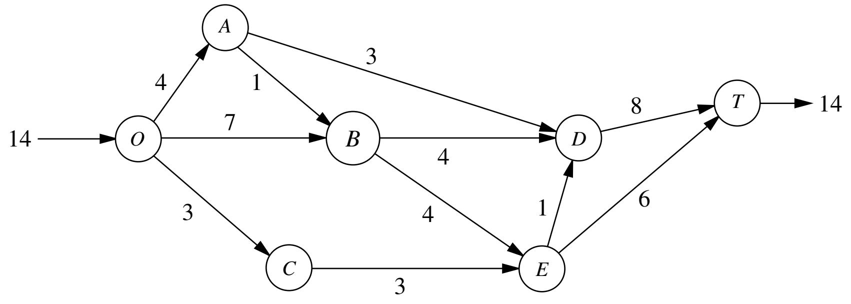


FIGURE 9.8
Optimal solution for the Seervada Park maximum flow problem.

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

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

Finding an Augmenting Path

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

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

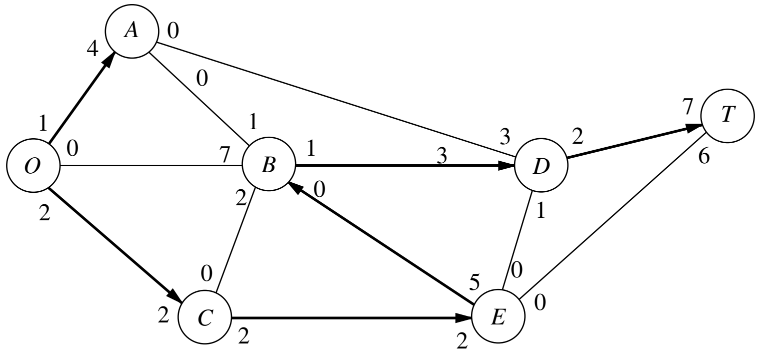


FIGURE 9.9
Procedure for finding an augmenting path for iteration 7 of the Seervada Park maximum flow problem.

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

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

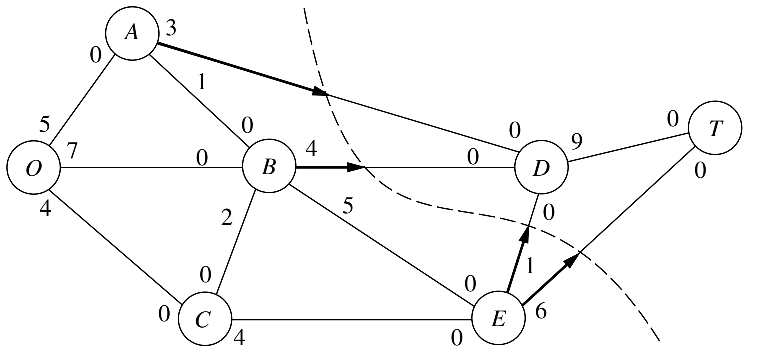


FIGURE 9.10
A minimum cut for the Seervada Park maximum flow problem.