Chapter 9: Maximum Flow and the Minimum Cut

A common question about networks is "what is the maximum flow rate between a given node and some other node in the network"? For example, traffic engineers may want to know the maximum flow rate of vehicles from the downtown car park to the freeway on-ramp because this will influence their decisions on whether to widen the roadways. Another example might be the maximum number of simultaneous telephone calls between two cities via the various land-lines, satellites, and microwave towers operated by a telephone company.

An infinite flow rate is impossible because the individual roads or telephone links have limited capacities to carry flow. On the other hand, there are usually multiple ways to drive between the downtown car park and the freeway on-ramp, or to route calls between two cities. Finding the maximum flow involves looking at *all* of the possible routes of flow between the two end-points in question. When the system is mapped as a network, the arcs represent channels of flow with limited capacities. To find the maximum flow, assign flow to each arc in the network such that the total simultaneous flow between the two end-point nodes is as large as possible.

A further wrinkle is that the flow capacity on an arc might differ according to the direction. For example, a particular road might have two lanes in the A to B direction, but only one lane in the B to A direction. Or it might be simply a one-way street, with no flow in the B to A direction at all. In Figure 9.1, the labels on the arcs indicate the flow capacities in both directions. For example, the label near node A on the arc A-B indicates the flow capacity in the A-to-B direction, while the label near node B on the arc A-B indicates the flow capacity in the B-to-A direction.

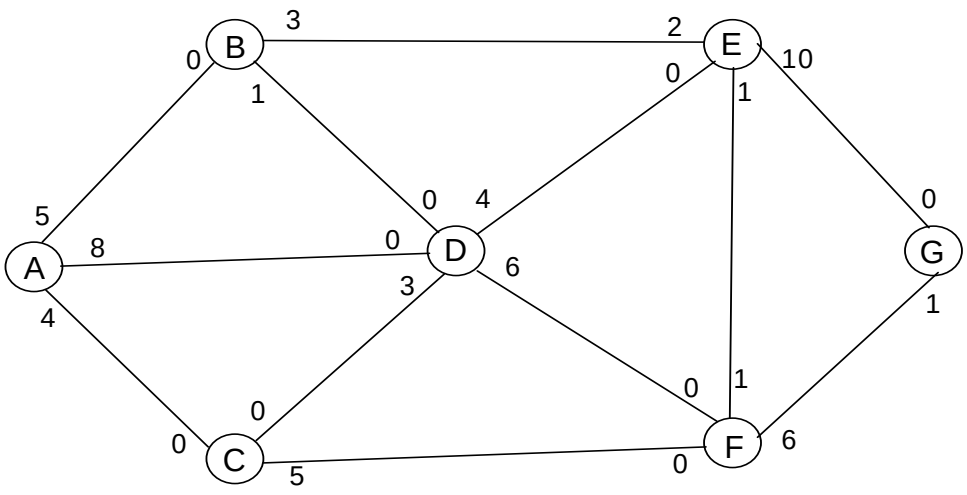

Figure 9.1: Find the maximum flow from A to G.

Let's imagine that the flows in Figure 9.1 are in units of vehicles per minute. Here are some examples of routes on which flow could travel from node A to node G:

- 4 vehicles per minute along the route A-D-E-G. This is the maximum flow on this route because of the bottleneck on arc D-E.
- 3 vehicles per minute along the route A-B-E-G. The bottleneck here is the arc B-E. Note, though, that with the simultaneous flow on route A-D-E-G, the total flow on arc E-G is now 7 vehicles per minute.
- 4 vehicles per minute along the route A-C-F-G. The bottleneck on this route is arc A-C.

This set of flows gives a total flow of 4 + 3 + 4 = 11 vehicles per minute from A to G. But it is not the *maximum* flow attainable from A to G. There remain unused routes for carrying flow between the two end nodes. We need an organized method of tabulating the routes and flows, and this is provided by the Ford and Fulkerson method. As we will see later, the maximum flow problem can be solved by linear programming, but the Ford and Fulkerson method is simple and even faster than linear programming when implemented on a computer. Ford and Fulkerson first published their method in the *Canadian Journal of Mathematics* in 1956 – it is a real classic paper, very often referenced to this day.

The main idea is careful bookkeeping of the flows assigned to different routes from the origin node to the destination node. The steps in the method are:

1. Find *any* path from the origin node to the destination node that has a strictly positive flow capacity remaining. If there are no more such paths, exit.

2. Determine *f*, the maximum flow along this path, which will be equal to the smallest flow capacity on any arc in the path (the bottleneck arc).

3. Subtract *f* from the remaining flow capacity in the forward direction for each arc in the path. Add *f* to the remaining flow capacity in the backwards direction for each arc in the path.

4. Go to Step 1.

On termination, the sum of the flows along the paths found during Step 1 gives the maximum total flow between the origin and destination nodes.

Let's try the Ford and Fulkerson method on the network in Figure 9.1. The results are shown in Figures 9.1 through 9.6. Figures 9.2 through 9.4 show the three flow paths suggested earlier, and Figures 9.5 and 9.6 show two more flow paths that can be added before we are unable to find a path that can support a strictly positive flow *f*. Note the bookkeeping on the flow capacities as the solution progresses, and how it becomes more and more difficult to find a path having positive flow capacity.

The algorithm terminates after the last path is found in Figure 9.6. No more strictly positive flow paths can be found between A and G. This is obvious since all paths must pass through the set of

arcs B-E, D-E, F-E, and F-G, and these arcs have all had their flow capacities in the forward direction reduced to zero.

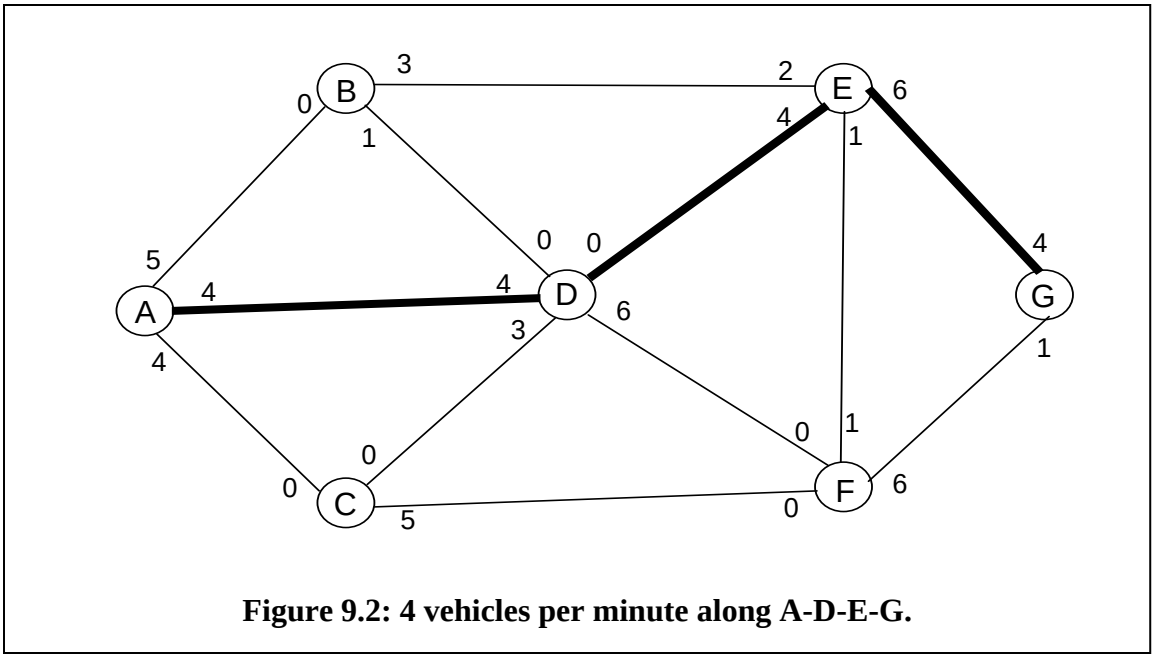

Figure 9.2: 4 vehicles per minute along A-D-E-G.

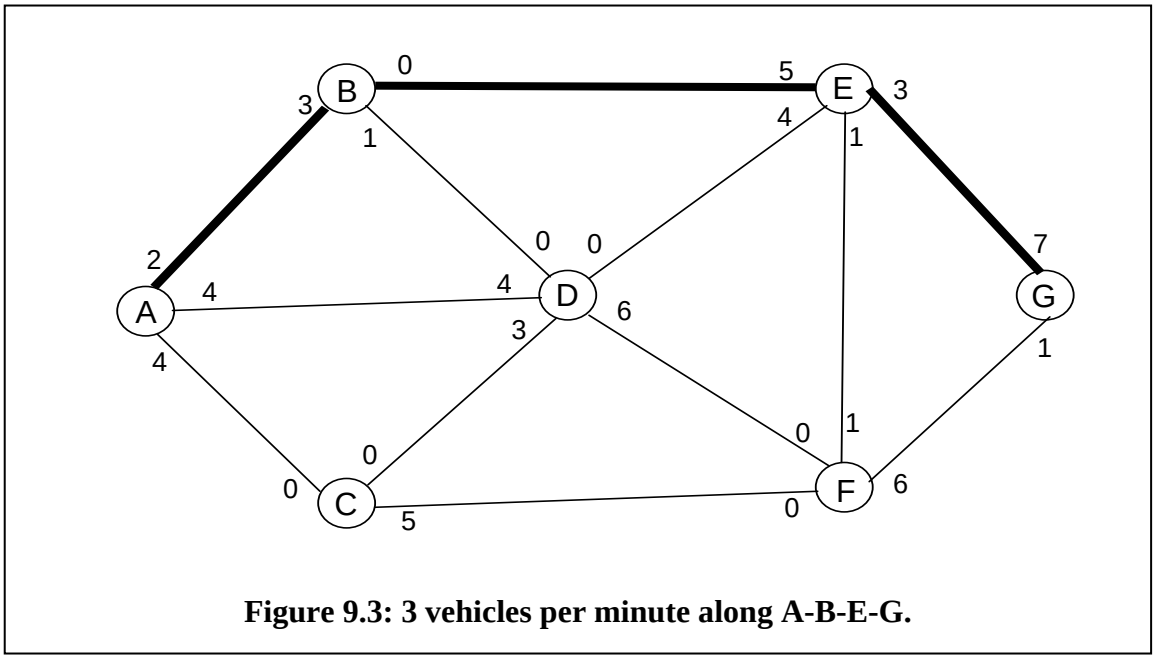

Figure 9.3: 3 vehicles per minute along A-B-E-G.

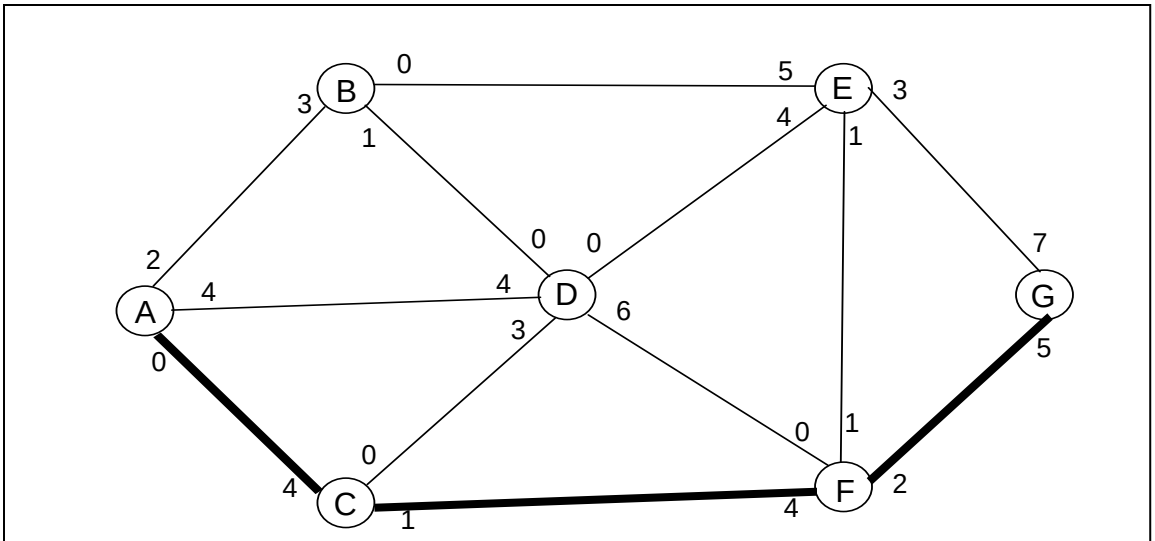

Figure 9.4: 4 vehicles per minute along A-C-F-G.

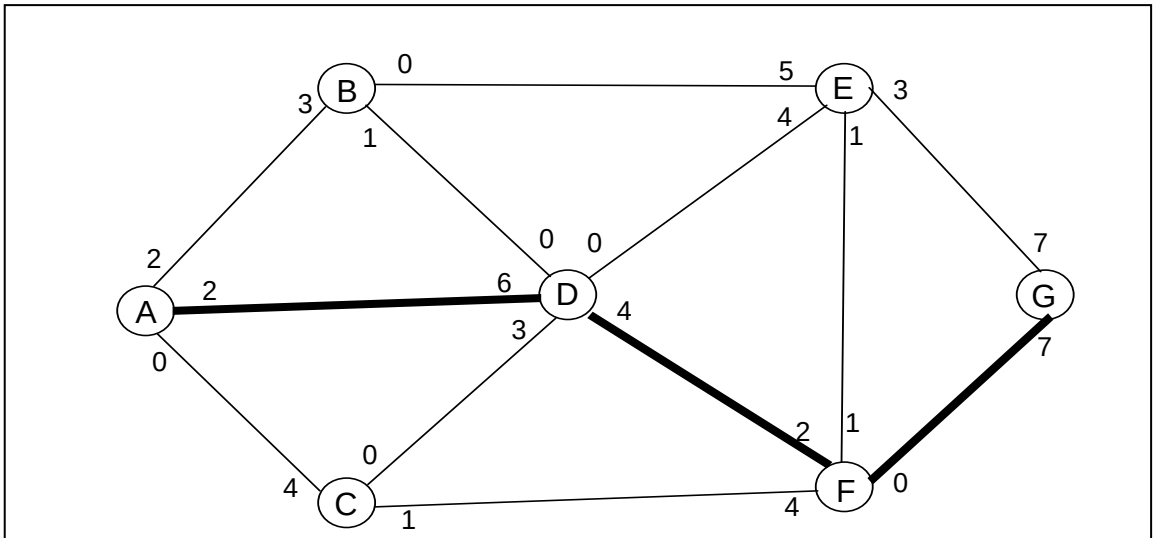

Figure 9.5: 2 vehicles per minute along A-D-F-G.

When the algorithm terminates, the maximum total simultaneous flow of vehicles from A to G is given by summing the flows on the 5 paths we selected: 4 + 3 + 4 + 2 + 1 = 14 vehicles per minute.

But what is the actual *pattern* of flows that gives this optimum? How much flow should go on each arc, and in which direction? This is found by looking at the difference between the initial flow capacity and the final flow capacity: a positive difference indicates a flow in the associated

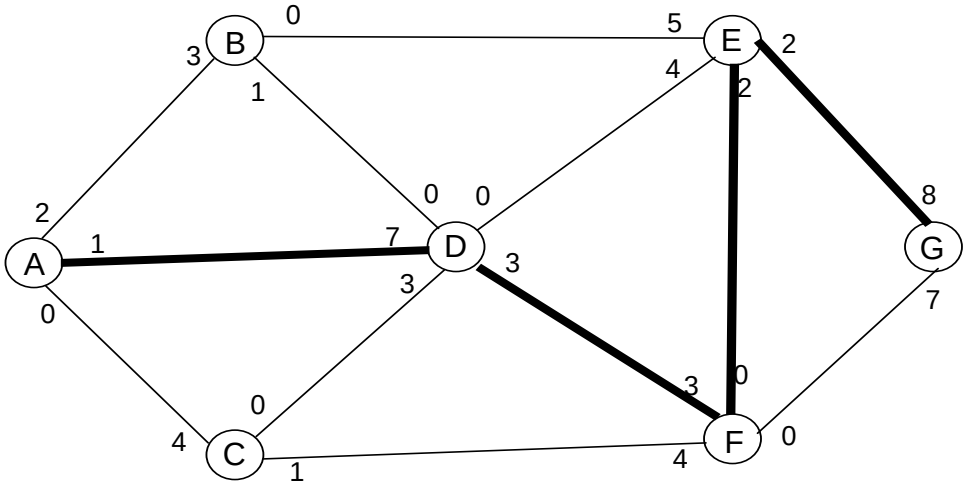

Figure 9.6: 1 vehicle per minute along A-D-F-E-G.

direction (a negative difference is ignored). The pattern of flows – and their directions – which gives the maximum simultaneous flow of vehicles per minute, is shown in Figure 9.7. The arc labels in Figure 9.7 show the amount of flow in each arc. Note that the principle of flow conservation at a node is respected. For example, the flows entering node F total 7 vehicles per minute, as do the flows leaving node F.

You probably noticed that it becomes harder and harder to find a strictly positive flow path as the algorithm progresses and all the easy-to-spot paths are used up. You might think this would be a

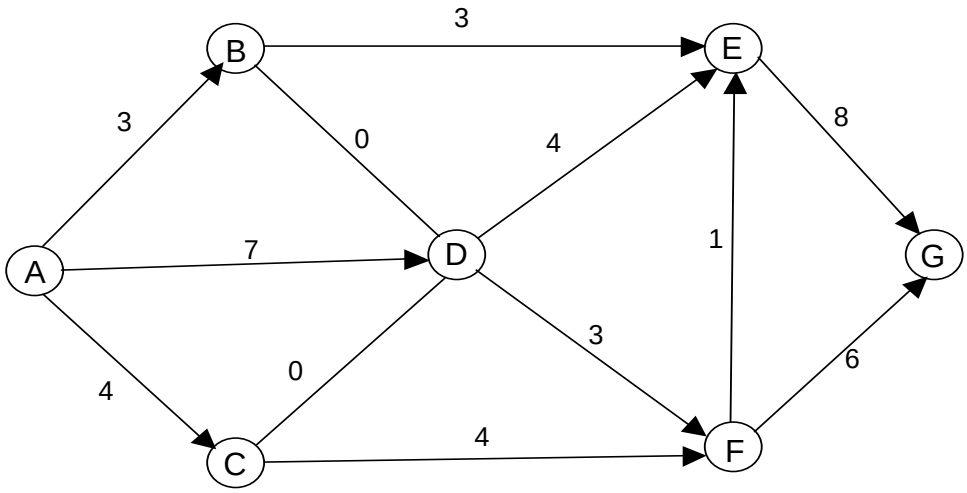

Figure 9.7: Pattern of flows yielding the maximum flow.

problem in a computer implementation of the method, but it turns out that simple depth-first and breadth-first searches are quite efficient for finding positive flow paths.

Students often ask why the Ford and Fulkerson algorithm bothers to update the flow capacities in the backwards directions on the arcs. This is because the backwards capacities that are added are a bookkeeping convention to indicate flow that can be undone if needed. This did not happen in our example, but Figure 9.8 shows a simple example in which the backwards capacities are used in reaching a larger total flow. As you see, after the first path is chosen, the only way for the second path to route more flow from A to B is by *undoing* the flow placed on the vertical arc by the first path. The resulting flow pattern in (d) shows that the vertical arc is not used at all in the final solution.

The maximum flow problem is intimately related to the minimum cut problem. A *cut* is any set of directed arcs containing at least one arc in every path from the origin node to the destination node. In other words, if the arcs in the cut are removed, then flow from the origin to the destination is completely cut off. The *cut value* is the sum of the flow capacities in the origin-to-destination direction over all of the arcs in the cut. The minimum cut problem is to find the cut that has the minimum cut value over all possible cuts in the network. Some possible cuts are illustrated in Figure 9.9; each cut is labeled with the cut value.

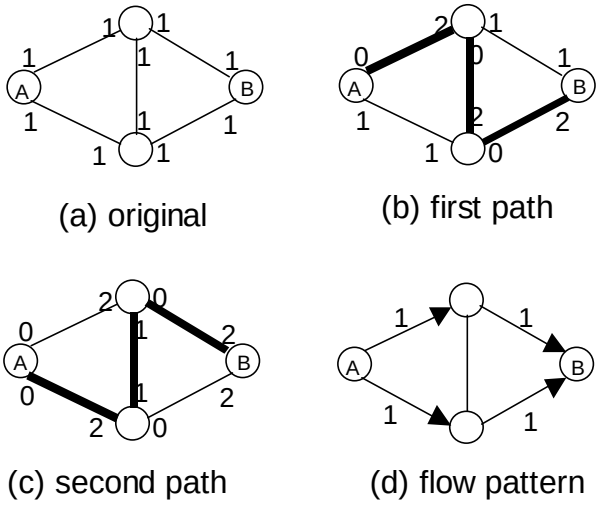

Figure 9.8: Why backwards flow capacities are updated.

One terribly inefficient way to find the minimum cut is to simply try all possible cuts and select the smallest. However the number of possible cuts is extremely large, and it is impossible to enumerate all possible cuts in a larger network, even with extremely fast computers and a lot of computing time.

A better approach is to make use of the *max-flow / min-cut theorem*: for any network having a single origin node and a single destination node, the maximum possible flow from origin to destination equals the minimum cut value for all cuts in the network.

This may seem surprising at first, but makes sense when you consider that the maximum flow through a series of linked pipes equals the maximum flow in the smallest pipe in the series, i.e. the flow is limited by the bottleneck pipe, as illustrated in Figure 9.10. The minimum cut is just

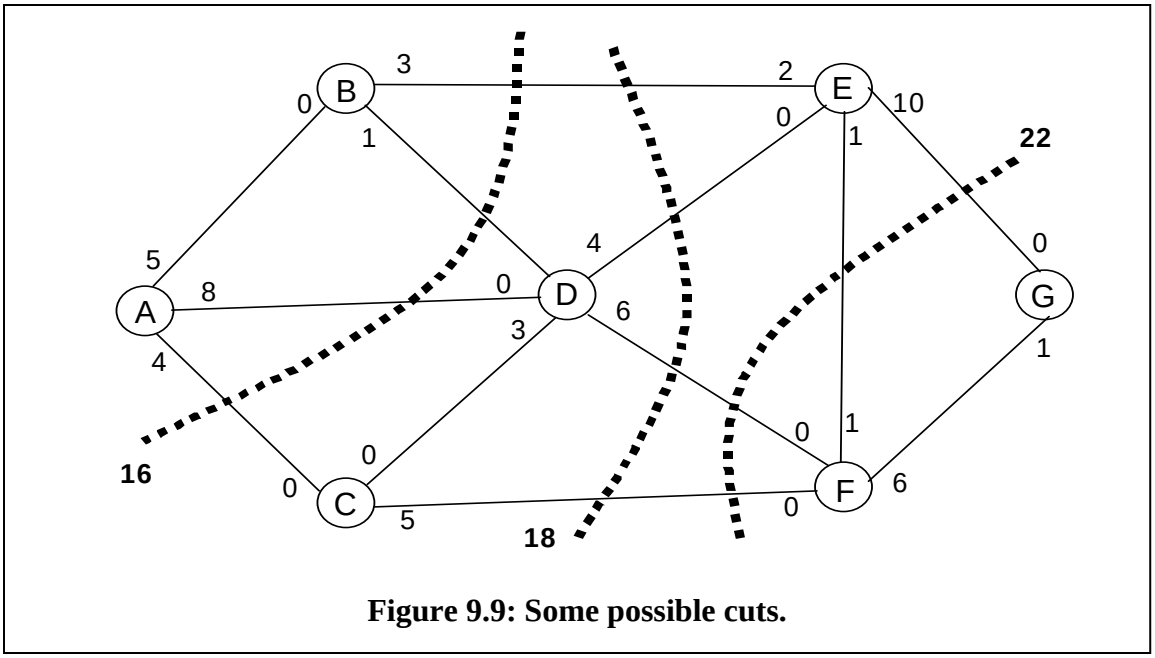

Figure 9.9: Some possible cuts.

a kind of *distributed* bottleneck; i.e. a bottleneck for a whole network as opposed to a simple bottleneck for a series of pipes.

The max-flow / min-cut theorem means that we can determine the minimum cut value using the Ford and Fulkerson maximum flow algorithm. But the real question is *where* the minimum cut is located. If our traffic engineers determine that the maximum flow rate of vehicles from the car park to the freeway on-ramp of 14 vehicles per minute is too small to handle peak rush hour traffic, then we are going to want to expand the roadways. But which ones? There is no point in expanding roadways that are carrying less than their maximum capacity. The only roadways to expand are those that are part of the bottleneck, i.e. the minimum cut.

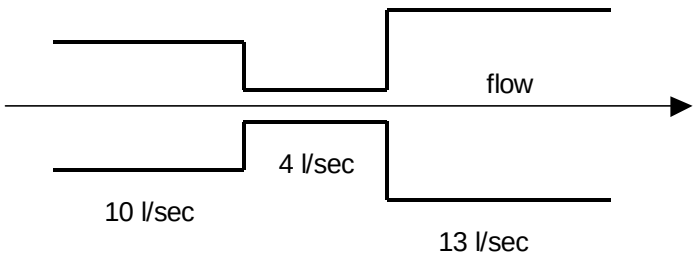

Figure 9.10: The maximum flow through this series of pipes is 4 l/sec.

The minimum cut is actually simple to find after the Ford and Fulkerson algorithm has been completed. Simply mark the arcs that are carrying a flow equal to their maximum flow capacity and look for a cut that consists *only* of marked arcs and no other arcs. It often happens that not all of the marked arcs are used in the cut, and it may also happen that there are multiple cuts that can be made in this way. The results for our vehicle flow example are shown in Figure 9.11. Not surprisingly, the cut is the same set of arcs whose flow capacities were forced to zero by the Ford and Fulkerson algorithm, hence forcing the termination: B-E, D-E, F-E, and F-G. The cut value in the forward (origin-to-destination) direction is 14 for these arcs, the same as the maximum

flow. These four arcs are the bottleneck for the network. These are the roads that the traffic engineers should consider widening to increase the flow from the car park to the freeway on-ramp.

Keep in mind, though, that you may not get one unit of increase in the maximum flow for every unit of added capacity in an arc from the minimum cut. This is because the increased flow through that arc may cause another bottleneck to come into play upstream or downstream from that arc.

There are many applications that make use of the minimum cut, including finding the bottlenecks in traffic applications as we have shown here, or in telecommunications networks, production lines, etc. It's interesting to note that the algorithms for finding the max flow and the min cut do not require that the associated labels be flow capacities. Suppose the flow capacity labels actually represented costs, e.g. costs of building dams over various tributaries of a river system. Then the min cut would actually show the minimum cost of completely damming the water flow in the river network. In a military application during the Vietnam war, the infamous Ho Chi Minh trail, actually a network of trails, was modeled as a network, and estimated "costs" were assigned to cutting the trail at various points. The min cut then showed the set of trail segments that should be attacked in an attempt to cut the flow of enemy men and supplies at the least "cost".

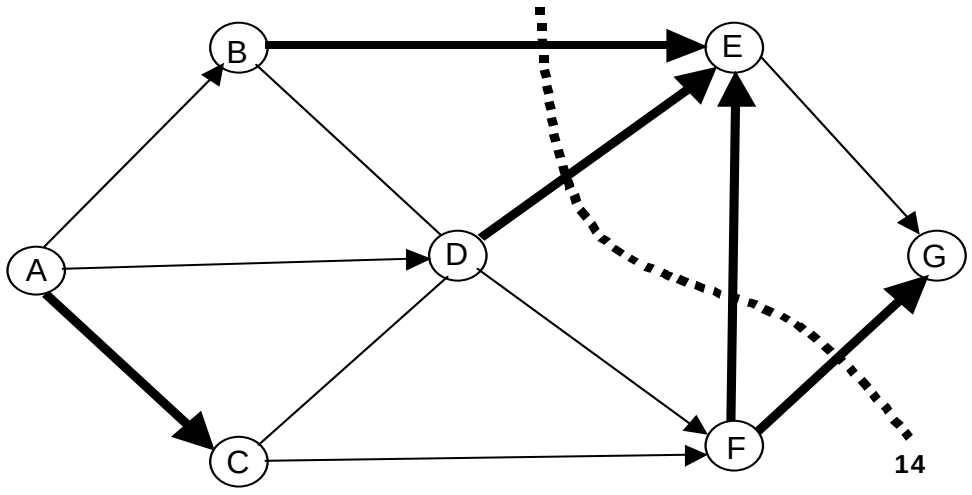

Figure 9.11: Finding the minimum cut.