Chapter 2
Maximum Flow in a Network

2.1 Flows in Networks

The model we use for a network consists of a set of nodes joined by directed edges as in Figure 2.1. Each edge e of the network carries some amount of flow denoted f_e. The flow commodity could be transfer rate through a communications network, or merchandise moving from producer to consumer or simply water flowing in a pipe. Network models can be used in a vast array of applications.

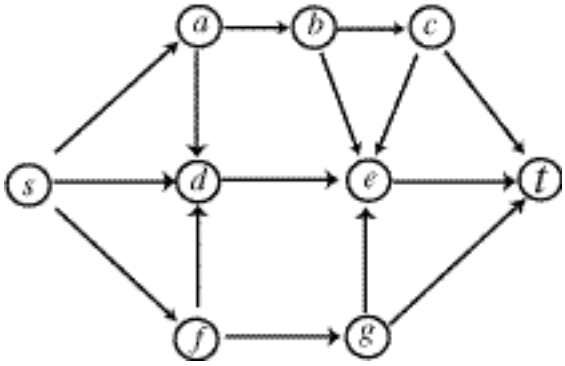


Figure 2.1 An Example of a Network

In the network there are two nodes with special roles: the *source* s and the *sink* t. The network itself is a model of ways to arrange flow moving from node s to node t. (This does not stop us from having edges directed into the source or out of the sink, though they may not be very helpful.)

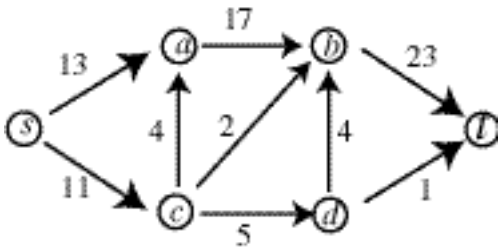


Figure 2.2 A Flow in a Network

There is one key condition that we place on any flow $F = \{f_e\}$ in a network. It is natural to assume that, except for the source and the sink, the nodes themselves neither create nor consume flow; they just pass it on. Thus we require "flow conservation" which means that, at each node $v \neq s, t$ the total flow into v equals the total flow out of v. Any flow F that satisfies this condition is called a *feasible flow*.

When discussing networks we will use the following notation. The network itself will be $N = (V, E)$ with node set V and edge set E. A *flow* in N will be denoted $F = \{f_e\}$ with f_e being the flow in edge e. Each edge e has a *tail* $t(e)$ and a *head* $h(e)$. The flow conservation requirement can now be written

$$\sum_{t(e)=v} f_e = \sum_{h(e)=v} f_e$$

for all nodes $v \in V$ with $v \neq s, t$. In any feasible flow the net flow out of the source is equal to the net flow into the sink (Exercise 2.2). The *value* of the flow is defined to be this common value

$$val(F) = \sum_{t(e)=s} f_e - \sum_{h(e)=s} f_e.$$

Note that in the examples of this and the next section, there are no edges leading into the source or out of the sink so the second sum in the formula for $val(F)$ is empty in these cases.

In any real network there will be an upper bound on the flow in an edge. With this in mind, we specify for each edge e a *maximum capacity* c_e. In addition, we will assume for the moment that the flows are not negative. There are situations where a negative flow makes sense so we will eventually have to deal with this possibility, but for now $f_e \geq 0$. Therefore, for each edge e

$$0 \leq f_e \leq c_e.$$

When drawing pictures of networks with flows and capacities, the label on edge e will have the form (f_e, c_e) with the flow preceding the capacity. The edge $a \to b$ in Figure 2.3 has a flow of 6 with a capacity of 10. The value of the flow is 22. (Note that other authors use different conventions). Finally, we will assume that the edge capacities are rational numbers. This is not a serious constraint since in almost all practical problems any irrational number that arises can be replaced with a rational number. This assumption will simplify some of the analysis.

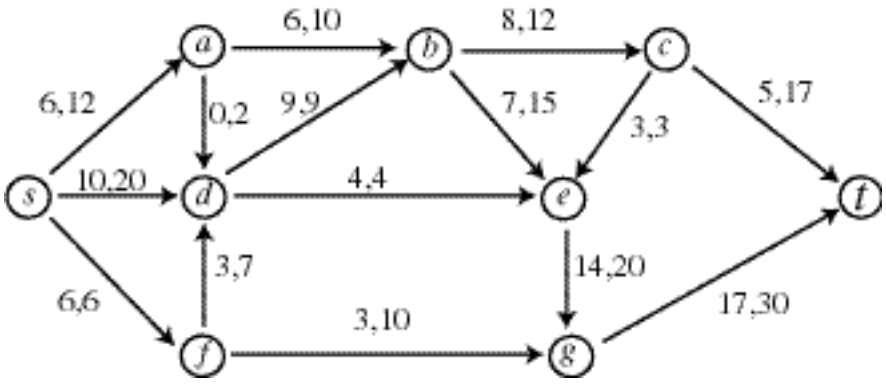


Figure 2.3 A Flow in a Network with edge Capacities

EXERCISE

2.1 In the network in Figure 2.4, the edge labels are part of a flow F with the flow in some of the edges left blank. Specify the flows for these edges so that the flow is feasible and then determine the flow value $val(F)$.

2.2 Show that in any (feasible) flow in a network, $\sum_{t(e)=s} f_e - \sum_{h(e)=s} f_e = \sum_{h(e)=t} f_e - \sum_{t(e)=t} f_e$.

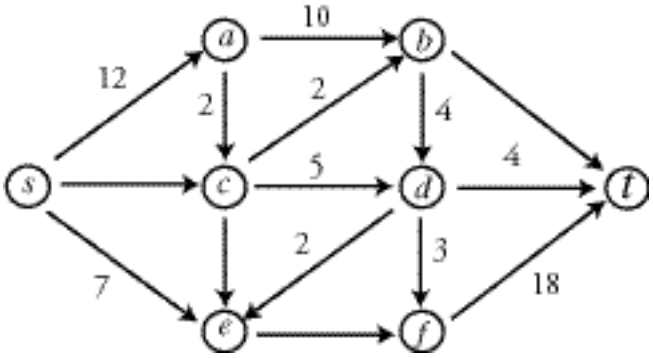


Figure 2.4 Network for Exercise 2.1

We are now in a position to formulate the central problem of this chapter. Given a network with edge capacities, the **Maximum Flow Problem** asks for an algorithm to determine a feasible flow of maximum value subject to the edge capacity constraints.

2.2 Augmenting Paths

The flow in Figure 2.3 has value 22. Is there a feasible flow in this network with a larger value? In Figure 2.5, a particular directed path from s to t has been highlighted: $s \to a \to b \to e \to g \to t$. Along this path each of the edges has flow strictly less than capacity, $f_e < c_e$. Clearly there is room for more flow from s to t along this path. moreover, if we increse the flow in each edge by the same increment δ then we will still have a feasible flow since, at each internal node (i.e. not the source or sink), δ additional units flow in and δ additional units flow out.

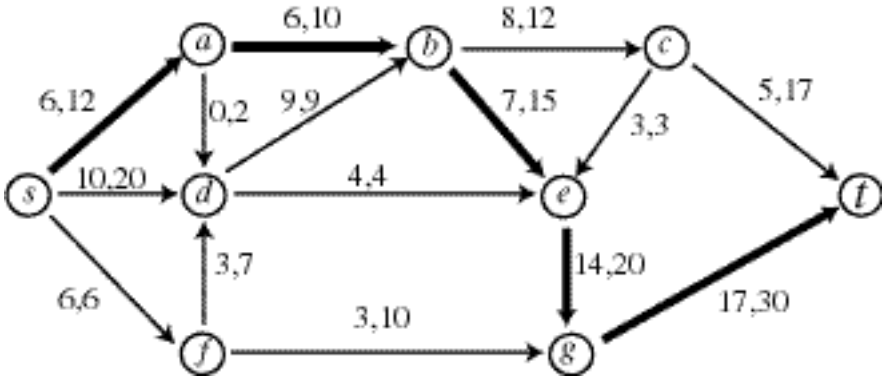


Figure 2.5 An Augmenting Path in a Network

There is a limit to the size of increment we can make. Along an edge e of the path, the flow before was f_e and after is $f_e + \delta$. We must still have $f_e + \delta \leq c_e$ so we need $\delta \leq c_e - f_e$. Thus the largest increment we can make is

$$\delta = min\{c_e - f_e | e \text{ is an edge of the path}\}$$

In Figure 2.5, the values of $c_e - f_e$ along the distinguished path are 6,4,8,6,13. Thus we can increase all flows along this path by the minumum which is 4 units. Notice that once we have augmented the flow along this path by 4 units the flow in edge $a \rightarrow b$ is at capacity; such an edge is called *saturated*. Flow cannot be increased in a saturated edge.

So far we have seen that, if there is a directed path from s to t with $f_e < c_e$ on every edge of the path (so no saturated edges), then the flow is not a maximum flow – it can be increased.

Consider now the simple network in Figure 2.6. The flow shown has value 1. Only two edges are not satured: $s \rightarrow b$ and $a \rightarrow t$. These edges do not form a directed path from s to t so the procedure just developed won't work. In fact,the given flow is not optimal as is clear from Figure 2.7 which exhibits a flow of value 2 in the same network. In order to move from the flow of value 1 to the flow of value 2, it is necessary to decrease the flow in edge $a \rightarrow b$. In other words, the flow in Figure 2.7 has made an unwise choice in sending flow from a to b. If this unit of flow is redirected from a to t, an additional unit of flow can move across the bottom of the network. To make general use of this redirection idea, we will allow some or all of the edges in our distinguished path to point backwards, that is, in the general direction of sink to source.

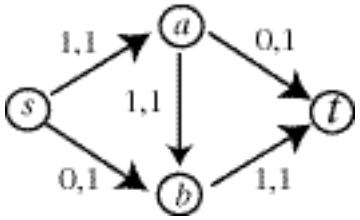
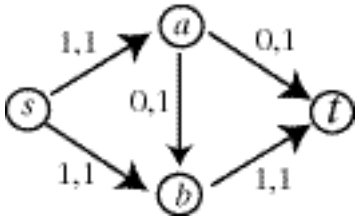


Figure 2.6 A Simple Network



An Optimal Flow in the same Network

An *augmenting path* in a network is a path from source s to sink t consisting of some *forward* edges directed source to sink and some *reverse* edges directed from sink to source along the path. Along the augmenting path we insist that on each forward edge $f_e < c_e$ and on each reverse edge $f_e > 0$.

The highlighted path in Figure 2.8 is an augmenting path: it goes from source s to sink t and all forward edges are non-saturated while the reverse edge $f \rightarrow d$ is non-empty. If δ is chosen so that $\delta \leq c_e - f_e$ for each forward edge and $\delta \leq f_e$ for each reverse edge then we can increase the flow value by δ. The procedure is to add δ units of flow to the forward edges and subtract δ units of flow from the reverse edges. In the example, the largest increment possible along this augmenting path is 3 units.

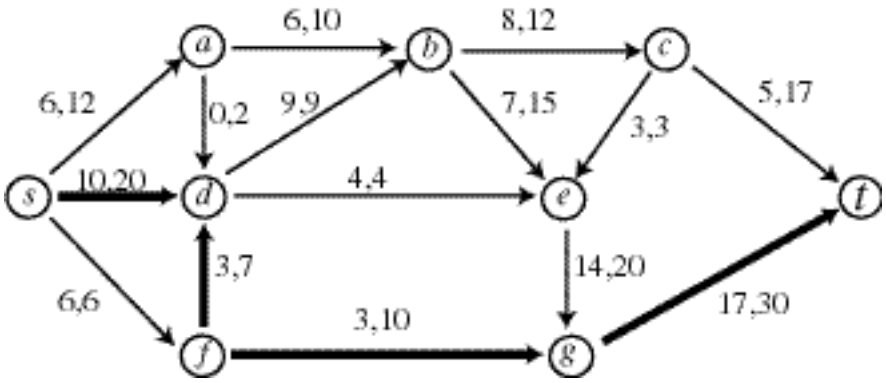


Figure 2.8 An augmenting path with a reverse edge

Staying with the same example, perform both of the augmentations described above. the result is displayed in Figure 2.9.

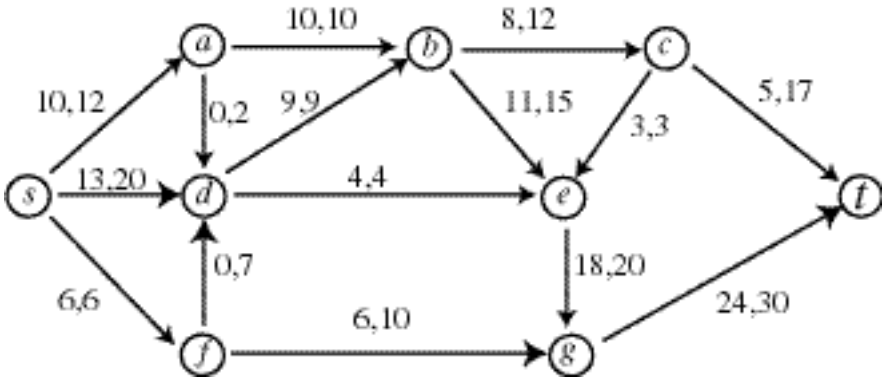


Figure 2.9 The network after both augmentations are performed

There don't appear to be any further augmenting paths in this network. Does this tell us that the flow is optimal? In any optimization situation, one should be on guard for local maxima. Our technique of constructing augmenting paths cannot be used to increase this flow, but does that mean that even if we made more drastic changes in the flow pattern, we would not find a flow of bigger value. This is the case; the next section develops the tool needed to prove this is true.

2.3 Cuts in a Network

In the search for augmenting paths in our current flow, we could start at the source s and try to push flow as far as possible toward the sink. If flow has reached a node x then from here flow can be moved forward along non-saturated edges e with $t(e) = x$ and backwards along non-empty edges with $h(e) = x$. There are several ways to organize such a process into an algorithm but for now just let S be the set of nodes such that, with the current flow in place, flow can be pushed to that node. Thus $x \in S$ if and only if there is a (partial) augmenting path from s to x. Let T be the set of nodes not already placed in S. In Figure 2.10, the set $S = \{s, a, d\}$ is surrounded by a dotted ellipse.

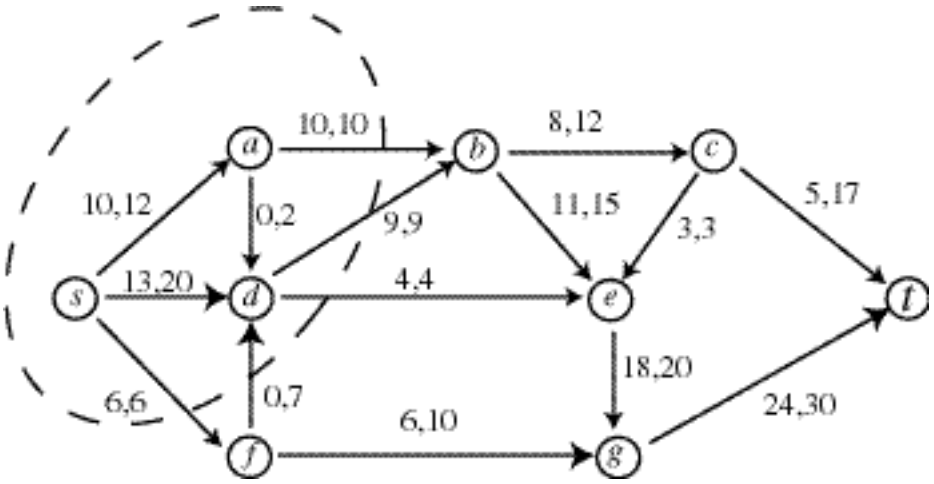


Figure 2.10 The network after both augmentations are performed

The sets S and T have several important features. These are disjoint sets with their union equal to the set of all nodes, so they partition the set of nodes. The source is in S and unless there is an complete augmenting path from s to t, the sink t will be an element of T. Finally, from our construction of these sets it is clear that the edges coming out of S leading into T must be saturated while the edges leading out of T and into S must be empty. The first two properties are those we use to make our next definition.

A *cut* (S, T) in a network $N = (V, E)$ is a partition of the node set $V = S \cup T$ into disjoint sets S and T such that the source $s \in S$ and the sink $t \in T$. The *capacity of the cut* is defined as

$$cap(S, T) = \sum_{t(e) \in S, h(e) \in T} c_e.$$

Thus the capacity of a cut is the sum of all the capacities of the edges leading out of S and into T. This is quite a general definition. In particular, we are not assuming that the cut was determined by an attempt to define an augmenting path. A cut separates the network into two parts and the capacity of the cut sets an upper bound on moving flow from one part to the other. This is the content of the following theorem.

THEOREM 2.1 *Suppose that N is a network with feasible flow $F = \{f_e\}$ and cut (S, T) then*

(i) $val(F) = \sum_{t(e) \in S, h(e) \in T} f_e - \sum_{t(e) \in T, h(e) \in S} f_e$,

(ii) $val(F) \leq cap(S,T)$.

Proof. Using the flow conservation property, the difference

$$\sum_{t(e)=v} f_e - \sum_{h(e)=v} f_e$$

equals the flow value if $v = s$ the source and is zero for all other nodes in S. Thus we can write

$$val(F) = \sum_{v \in S} \left(\sum_{t(e)=v} f_e - \sum_{h(e)=v} f_e \right) = \sum_{t(e) \in S} f_e - \sum_{h(e) \in S} f_e.$$

But an edge e with both ends in S contributes $+f_e$ through the first sum and $-f_e$ through the second sum; these cancel. All that remains is the terms $+f_e$ for edges with tail in S and head in T and terms $-f_e$ for edges with tail in T and head in S. Thus

$$val(F) = \sum_{t(e) \in S, h(e) \in T} f_e - \sum_{t(e) \in T, h(e) \in S} f_e$$

as required for Part (i). To prove Part (ii), simply observe that the first sum is, by definition $cap(S,T)$ and we are subtracting something nonegative. □

This theorem says that the value of any flow in a newtork is at most the capacity of an arbitrary cut.

2.4 The Augmenting Path Algorithm for Max-Flow

We are now in a position to describe the augmenting path algorithm for the Max-Flow problem. We will first describe the algorithm informally then discuss the correctness proof, efficiency and implementation issues.

As input we have a network $N = (V, E)$ with edge capacities $c_e \geq 0$. Our aim is to produce a feasible flow in N of maximum value.

Maximum Flow Algorithm

(1) Initialize the flow F with $f_e = 0$ for all edges $e \in E$.

(2) Construct the set S of all nodes x such that there is a (partial) augmenting path from the source s to x relative to the current flow F.

(3) If the sink $t \in S$ then an augmenting path from s to t has been found. Perform the maximum allowed augmentation along this path. Return to (2).

(4) If the sink $t \notin S$ then stop; the flow is maximal.

The first thing to establish is that if the algorithm ever stops then the flow is indeed maximal. What we learn from Theorem 2.1 is that for any flow F and cut (S,T) we have $val(F) \leq cap(S,T)$. In particular the maximum value of a flow is at most the minimum value of any cut. Indeed, if we ever find a flow and cut with equality, $val(F) = cap(S,T)$ then this must be a maximum flow and a minimal cut.

Suppose the algorithm stops. It must have reached Step (4) so the set S of nodes to which flow can be moved from the source does not include the sink t. Setting $T = V \setminus S$, we have specified a cut (S, T). Since we cannot extend any of the augmenting paths out of S into T, every edge with its tail in S and head in T is saturated and every edge with its head in S and tail in T is empty. Using Theorem 2.1 Part (i), we deduce that for this particular cut, $val(F) = cap(S, T)$. Therefore F is a maximum flow and (S, T) is a minimum cut.

We have shown that if the algorithm ever terminates then it has found a maximum flow (and a minimum cut). Can the algorithm run forever? Each augmentation in Step (3) actually increases the flow by some non-zero amount, so the flow values are a monotonically increasing sequence. By Theorem 2.1, this sequence is bounded above by the capacity of any cut. If the capacities are integers then all flows constructed by the algorithm have integer flows in each edge hence have an integer value. Since there is no infinte, increasing bounded sequence of integers the algorithm must stop at some finite point. Recall that we have assumed that the capacities are rational numbers. A similar argument can be used (there will be an upper bound on the denominators); again the algorithm is sure to stop (see Exercise 2.5). Examples do exist of networks with irrational capacities and particular sequences of augmenting paths that result in an infinite sequence of flows ([Lawler]); the algorithm does not stop in these cases. This is not a very realistic example. In any case, if in Step (3) we are careful about which augmenting path we choose then the algorithm will always terminate. Therefore for any network some sequence of augmentations will lead the Augmenting Path Algorithm to a maximal flow. We summarize these results in the following theorem.

THEOREM 2.2 *Suppose that $N = (V, E)$ is a network with rational edge capacities $c_e \geq 0$. Then*

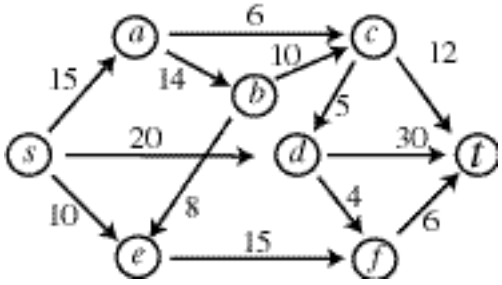
(i) *the augmenting path algorithm determines a maximal flow and a minimum capacity cut in a finite number of augmentations,*

(ii) *if the capacities are integers then there is a maximum flow with integer flows in every edge.*

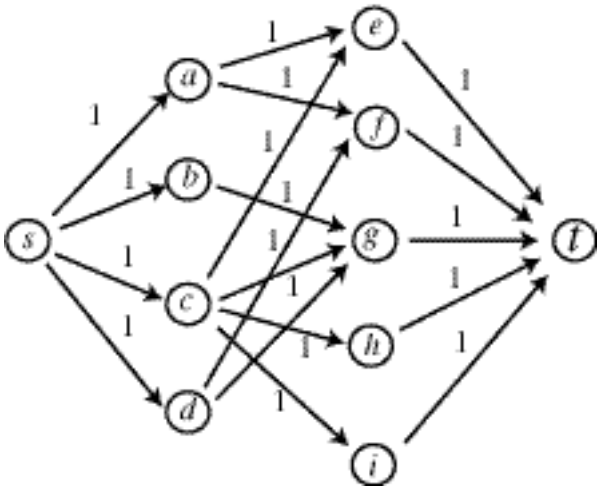
EXERCISES

2.3 In the three networks below the number beside each edge is the capacity of that edge. In each case, determine a maximal flow. Justify your claim that it is maximal.

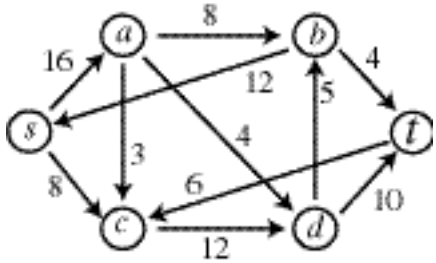
(a)



(b)



(c)



2.4 Give an example of a network which has at least two distinct maximal flows (of equal value, of course!)

2.5 Give an example of a network with at least two distinct minimal cuts.

2.6 Show that the Augmenting Path Algorithm only makes finitely many augmentations in the case that all capcities are rational $c_e \in \mathbf{Q}$.

2.7 Suppose that we have a network where in addition to the edge flow constraints there is a maximum flow c_v that can pass through any internal node v. Thus $\sum_{t(e)=v} f_e \leq c_v$. Describe how to modify the network so that this new problem is simply the standard Max Flow problem in the modified network. [Hint: split each internal node in two.]

2.5 Implementation and Performance

Breadth first search

$O(nm^2)$.

2.6 Applications

Bipartite matching