

## 8 Maximum flows in networks

Many Operational Research problems can be cast in the form of finding a maximum flow in a network. These ‘structured’ LP’s can be solved very quickly. We shall describe the classical augmenting path method.

**Travellers’ example** A group of travellers wants to fly from San Francisco to New York City at short notice. The seat availabilities on possible flights are as shown.

For example we may send 4 along the ‘middle’ route, 1 along the north route and 3 along the south route.

So 8 people can make it, but can we do better?

By a *network* we mean a directed graph  $G = (V, E)$  with  $n$  nodes in which each arc  $ij$  has a capacity  $u_{ij} \geq 0$ , and in which we have a specified *source* node  $s$  and *sink* node  $t$ . The remaining  $n - 2$  nodes are called *intermediate*. It is convenient to assume that no arc enters the source  $s$  and no arc leaves the sink  $t$ .

A *flow*  $\mathbf{x}$  through the network is an assignment of real numbers  $x_{ij}$  to the arcs  $ij$  satisfying the *conservation equations*

$$\sum_{i \in V} x_{ij} = \sum_{k \in V} x_{jk} \quad (1)$$

for each intermediate node  $j$ . In the first sum it is to be understood that we sum over all nodes  $i$  such that  $ij$  is an arc; and similarly for other sums here.

Call  $\sum_{j \in V} x_{sj}$  the *volume*  $\text{vol}(\mathbf{x})$  of the flow  $\mathbf{x}$ . A flow  $\mathbf{x}$  is *feasible* if  $0 \leq x_a \leq u_a$  for each arc  $a$ . A feasible flow of maximal volume is called a *maximum flow* through the network.

A *cut* is specified by any set  $C$  of nodes with  $s \in C$  and  $t \notin C$ . (This is a convenient notation: really it is the arcs between  $C$  and  $V \setminus C$  which we care about.) Suppose that  $C$  is a cut and  $\mathbf{x}$  is a flow. Summing conservation equations yields

$$\text{vol}(\mathbf{x}) = \sum_{j \in C} \sum_{k \notin C} x_{jk} - \sum_{i \notin C} \sum_{j \in C} x_{ij}. \quad (2)$$

This says that the volume  $\text{vol}(\mathbf{x})$  equals the net export out of  $C$ . For example, taking  $C = V \setminus \{t\}$ , we see that  $\text{vol}(\mathbf{x}) = \sum_{j \in V} x_{jt}$ .

The *capacity*  $\text{cap}(C)$  is defined to be  $\sum_{j \in C} \sum_{k \notin C} u_{jk}$ . A *minimum cut* is a cut of minimal capacity. From (2) we must have

$$\text{vol}(\mathbf{x}) \leq \text{cap}(C), \quad (3)$$

for any feasible flow  $\mathbf{x}$  and cut  $C$ .

### **Example again**

We can send one extra passenger along the south route as far as A, who could then continue to NYC if we route one less passenger  $D \rightarrow A$ ; and we could send this passenger along  $D \rightarrow C \rightarrow \text{NYC}$ . The modifications are represented by

leading to the flow indicated of volume 9.

The shaded vertices give a cut of capacity 9 so we have found a maximum flow, and a minimum cut.

Consider a feasible flow  $\mathbf{x}$  in the network. Let us allow a *path* now to ignore the direction of arcs. An  $\mathbf{x}$ -alterable path is a path such that  $x_a < u_a$  for each forward arc  $a$  and  $x_a > 0$  for each reverse arc  $a$ . An  $\mathbf{x}$ -alterable path from  $s$  to  $t$  is an  $\mathbf{x}$ -augmenting path.

Given an  $\mathbf{x}$ -augmenting path let  $\delta$  be the minimum value of  $c_a - x_a$  over all forward arcs  $a$  and of  $x_a$  over all reverse arcs  $a$ . Thus  $\delta > 0$ . (In the example  $\delta = 1$ .) Increase  $x_a$  by  $\delta$  for each forward arc  $a$  and decrease  $x_a$  by  $\delta$  for each reverse arc. It is easy to see that we have created a new feasible flow  $\hat{\mathbf{x}}$  with volume  $\text{vol}(\hat{\mathbf{x}}) = \text{vol}(\mathbf{x}) + \delta$ . This is the key observation. We are now ready for the famous theorem of Ford and Fulkerson (1956) - though see also the discussion in Chvátal *Linear Programming*.

**Theorem 8.1**      (*The max-flow min-cut theorem*)

*For any network, the maximum value of a flow equals the minimum capacity of a cut.*

**Proof** We saw in (3) that  $\text{vol}(\mathbf{x}) \leq \text{cap}(C)$  for each feasible flow  $\mathbf{x}$  and each cut  $C$ . Thus it suffices to show that there is a feasible flow  $\mathbf{x}$  and a cut  $C$  with  $\text{vol}(\mathbf{x}) = \text{cap}(C)$ .

Consider a maximum flow  $\hat{\mathbf{x}}$ . (The continuous function  $\text{vol}(\mathbf{x})$  must achieve a maximum value on the closed and bounded set of feasible flows.) Let  $C$  be the set of nodes  $i$  such that there is an  $\hat{\mathbf{x}}$ -alterable path from  $s$  to  $i$ . (Include  $s$  in  $C$ .) By our discussion above, there cannot be an  $\hat{\mathbf{x}}$ -augmenting path, and so  $t \notin C$ . Thus  $C$  gives a cut.

For each arc  $jk$  with  $j \in C$  and  $k \notin C$  we must have  $\hat{x}_{jk} = u_{jk}$  (for if  $j \in C$  and  $\hat{x}_{jk} < u_{jk}$  then also  $k \in C$ ). Similarly for each arc  $ij$  with  $i \notin C$  and  $j \in C$  we have  $\hat{x}_{ij} = 0$ . Hence by equation (2)

$$\text{vol}(\hat{\mathbf{x}}) = \sum_{j \in C} \sum_{k \notin C} \hat{x}_{jk} - \sum_{i \notin C} \sum_{j \in C} \hat{x}_{ij} = \sum_{j \in C} \sum_{k \notin C} u_{jk} = \text{cap}(C)$$

which completes the proof. □

In order to design an algorithm to find a maximum flow we want an efficient way to look for augmenting paths. Here is one such method.

## Maximum flow algorithm

### *Start*

Start with a feasible flow in the network, perhaps the zero flow. No nodes are labelled or scanned.

### *Search*

give the source  $s$  the label 0.

**while** there is no breakthrough and there is a labelled unscanned node

    find a labelled unscanned node  $i$  and scan it as follows.

    for each arc  $ij$  such that  $x_{ij} < u_{ij}$  and node  $j$  is not labelled,

        give node  $j$  the label  $i$ .

    for each arc  $ki$  such that  $x_{ki} > 0$  and node  $k$  is not labelled,

        give node  $k$  the label  $i$ .

    if the sink is labelled then declare ‘breakthrough’.

### *Augment*

**if** there has just been a breakthrough **then**

    use the labels to find the corresponding  $\mathbf{x}$ -augmenting path,

    augment the flow by the corresponding  $\delta > 0$ ,

    remove all labels and go to the step *Search*.

### *Finish*

**return** the current flow  $\mathbf{x}$  [which is a maximum flow]

and the set  $C$  of labelled nodes [which gives a minimum cut].

## Example again

From the proof of the max-flow min-cut theorem, the method yields a sequence of feasible flows of strictly increasing volume, and if it stops then indeed the current flow is a maximum flow and the set  $C$  is a minimum cut. Also, if all capacities are integral then each  $\delta$  is integral. So we obtain

**Theorem 8.2**     *Suppose that each capacity  $u_a$  is integral. Then the above method yields a maximum flow  $\mathbf{x}^*$  and a minimum cut  $C$ , in at most  $\text{vol}(\mathbf{x}^*)$  iterations; and further the flow  $\mathbf{x}^*$  is integral.*

**Corollary 8.3**     *(Integrality theorem)     If each capacity is integral then there is a maximum flow which is integral.*

We may implement the method so that each iteration takes  $O(n^2)$  time: in fact we can ensure that the time taken is  $O(m)$  where  $m = |E|$  (see for example Chvátal). This is good, but the bound  $\text{vol}(\mathbf{x}^*)$  on the number of iterations is not good practically or theoretically (we want a bound which at most a polynomial in the number of bits in the input).

### Example

Here each arc capacity is  $M$  (big) except that one arc has capacity 1 as shown. With a bad choice of next node to scan we may repeatedly have  $\delta = 1$  and take  $M$  iterations (although the input has only  $O(\log M)$  bits).

Also what happens if the capacities are not all integers? If all are rationals with least common denominator  $d$  then we must obtain a maximum flow in at most  $d \text{vol}(\mathbf{x}^*)$  iterations (just clear fractions to see this). But, it is possible to construct examples with some irrational capacities such that bad choices of augmenting paths lead to flows with values not converging to the maximum value (see Chvátal).

In a major paper in 1972 Edmonds and Karp showed that if we organise the scanning using ‘first-labelled first-scanned’ then we require at most  $mn/2$  iterations, yielding a good bound of  $O(m^2n)$  steps overall. There are now better methods based on “blocking flows” that run in  $O(n^3)$  steps.

**Infinite capacities**     It is convenient sometimes to allow some capacities  $u_{ij}$  to be infinite. This presents no difficulties. Observe first that if each cut

has infinite capacity then there is an  $s - t$  directed path in the subnetwork containing only the infinite capacity arcs, and so there are (integral) flows of arbitrarily large volume. (To see that there is an  $s - t$  directed path as claimed, consider the set  $C$  of nodes reachable from  $s$  just using infinite capacity arcs: if  $t \notin C$  we would have a cut with finite capacity.) Of more interest is the case when some cut has finite capacity.

**Theorem 8.4**     *Assume that some cut has finite capacity.*

(a) *There is a maximum flow and its volume equals the minimum capacity of a cut.*

(b) *If all finite capacities are integral then there is an integral maximum flow.*

To deduce this from our earlier results just replace each infinite capacity by a suitably large integer  $M$ , say  $M$  greater than the sum of the finite capacities. Similarly, our maximum flow algorithm is easily adapted to allow for infinite capacities. We end this section with a model involving infinite capacities.

### **A strip mining model**

Given a reliable survey of ore deposits, we are to design the open pit to be created. The survey typically divides the volume under consideration into blocks with sides of about 30 feet (depending for example on the distance between bore holes) and there may be several thousands of these.

With each block  $i$  we associate the marginal profit  $w_i$  resulting when  $i$  is added to the pit: to obtain  $w_i$  we subtract the cost of excavating  $i$  (not the accumulated cost of excavating  $i$  along with blocks above it) from the profit from the ore in it. Thus the total profit from a pit  $P$  equals  $\sum_{i \in P} w_i$ .

Not every set  $P$  of blocks is a feasible pit. If we want block  $i$  in a pit  $P$  then we must also include some other blocks  $j$  in  $P$ , usually forming a cone above  $i$ : let us write  $i \rightarrow j$  for each of these. Now a set  $P$  of blocks constitutes a feasible pit if and only if

$$i \in P, i \rightarrow j \Rightarrow j \in P.$$

We are led to the following problem.

**Problem** Let  $G = (V, E)$  be a directed graph, with a value  $w_i$  for each node  $i \in V$ . Call a set  $C$  of nodes *closed* if  $i \in C, ij \in E \Rightarrow j \in C$ . The objective is to find a closed set  $C$  maximising  $\sum_{i \in C} w_i$ .

We may convert this into a network flow problem as follows. Let  $A = \{i \in V : w_i \geq 0\}$ ,  $B = V \setminus A$ . Extend  $G$  by adding a source  $s$  and sink  $t$ , with arcs  $si$  for each  $i \in A$  and  $jt$  for each  $j \in B$ .

Define non-negative upper bounds on the new arcs by setting  $u_{si} = w_i$  for each  $i \in A$  and  $u_{jt} = -w_j$  for each  $j \in B$ . Let the upper bounds on the original arcs of  $G$  be infinite. We now have a network  $N$ .

Observe that a set  $C$  of nodes in  $G$  is closed if and only if the cut  $C \cup \{s\}$  in  $N$  has finite capacity; and then

$$\begin{aligned} \text{cap}(C \cup \{s\}) &= \sum_{i \in A \setminus C} u_{si} + \sum_{j \in B \cap C} u_{jt} \\ &= \sum_{i \in A \setminus C} w_i - \sum_{j \in B \cap C} w_j \\ &= \sum_{i \in A} w_i - \sum_{j \in C} w_j. \end{aligned}$$

Since  $\sum_{i \in A} w_i$  is a constant, minimising  $\text{cap}(C \cup \{s\})$  is equivalent to maximising  $\sum_{i \in C} w_i$ . So a minimum cut in  $N$  is an optimal closed set in the original problem. Thus we can solve our strip mining problem by using the maximum flow algorithm to obtain a minimum cut.