

EE 520: Topics in Communications: Network Coding

Aug. 27 and 31, 2007

Scribe: Aditya Ramamoorthy

Lectures 2 and 3**1 Network Flows**

Our model of a communication network is one of a directed graph with capacity constraints on edges. Given such a network a question of great interest is : *what is the maximum amount of information that can be transmitted over it from the source to the terminal.* In these notes we shall just consider networks with one source and one terminal. It is useful to imagine the information as water that is being injected by the source node at a particular rate (cu. ft. per sec.) into a network of pipes with each edge's capacity constraint defining the maximum rate at which a given pipe can allow the water to flow. The question then becomes what is the maximum rate of water flow per second from the source to the terminal. To answer these questions we first need to precisely formulate the problem.

Definition 1 *Flow Network.* A flow network is a finite directed graph $G = (V, E)$ with the following features

- a) Each edge $e \in E$ has a capacity c_e which is a non-negative number.
- b) There is a single source s .
- c) There is a single terminal t .

We shall refer to nodes other than s and t as internal nodes. We shall also make the following assumptions none of which is restrictive.

- No edge enters s and no edge leaves t .
- At least one edge is incident to each node.
- All capacities are integers.

Note that if the capacities are rational numbers, then we can define a new time unit that is large enough so that all capacities are integers with respect to it. An example of a flow network is shown in Fig. 1. Next we define a flow.

Definition 2 An $s - t$ flow is a function f that maps each edge e to a non-negative real number $f : E \rightarrow \mathbb{R}^+$. A flow f must satisfy the following conditions.

- a) For $e \in E$, $0 \leq f(e) \leq c_e$. (Capacity constraint)
- b) For each node $v \in V - \{s, t\}$

$$\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e) \quad \text{Conservation constraint} \quad (1)$$

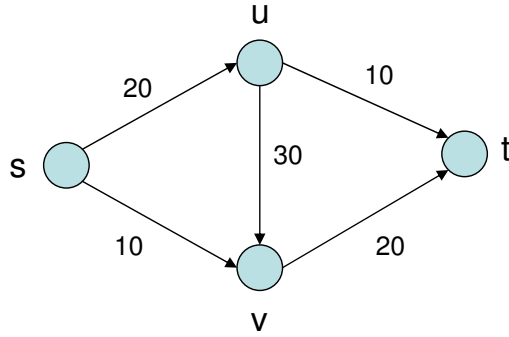


Figure 1: An example of a flow network. The labels on the edges denote the capacities of the respective edges.

Intuitively $f(e)$ represents the amount of flow (or traffic) carried by an edge.

Definition 3 *Value of a flow.* The value of a flow f denoted $v(f)$ and is given by $\sum_{e \text{ out of } s} f(e)$.

Thus, the value of a flow is the total flow leaving the source. Note that the source has no incoming edges. For notational convenience we shall denote (for a given node v)

$$f^{out}(v) = \sum_{e \text{ out of } v} f(e) \tag{2}$$

$$f^{in}(v) = \sum_{e \text{ into } v} f(e) \tag{3}$$

and for a set of vertices $A \subseteq V$

$$f^{out}(A) = \sum_{e \text{ out of } A} f(e) \tag{4}$$

$$f^{in}(A) = \sum_{e \text{ into } A} f(e) \tag{5}$$

2 Maximum-Flow Problem

The maximum flow problem is the following.

Given a flow network $G = (V, E)$ find a flow f such that $v(f)$ is maximum.

Intuitively what is the maximum one can hope to transmit from s to t if one thinks of edges as carrying traffic (or water). We shall find the answer to this problem by outlining an algorithm for its solution. Towards this end we first try a greedy heuristic solution.

In Fig. 1, start with $f(e) = 0, \forall e \in E$. Try to increase the flow by *pushing* some flow from s to t while respecting capacity constraints e.g. push 20 units of flow on the path $s \rightarrow u \rightarrow v \rightarrow t$ so that the value of the flow is 20. The resultant flow assignment is shown in Fig. 2(a). Now is it true that 20 is the maximum flow possible in this network. On inspection we realize that the answer is no, since we can have an assignment as shown in Fig. 2(b).

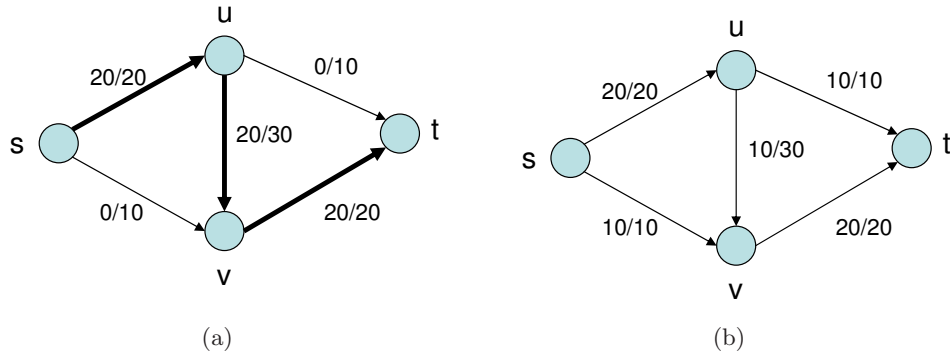


Figure 2: The labels on the edges denote $f(e)/c_e$ for the edge e . (a) The dark edges depict the chosen path that yields a flow of value 20. (b) A flow assignment that has a value of 30 which is maximal.

We realize that the solution in Fig. 2(b) can be arrived at by modifying the solution in Fig. 2(a) by pushing an additional 10 units along s to v , undoing 10 units along u to v and pushing 10 units along u to t . The key idea is to systematically perform the following operations.

- a) Push forward on edges with leftover capacity. This uses us potentially unused capacity that can increase the flow value.
- b) Push backward on edges that are already carrying flow. This allows us to divert the currently flow that can allow more efficient allocations.

For doing these operations systematically we need the concept of a residual graph.

Definition 4 *Residual graph.* Given a flow network G and a flow f we define the residual graph G_f of G with respect to f as follows.

- a) The set of vertices of G_f is the same as the vertex set of G .
- b) For $e = (u, v) \in G$, if $f(e) < c_e$ then introduce the edge (u, v) in G_f with capacity $c_e^f = c_e - f(e)$. Such an edge in G_f is called a forward edge.
- c) For $e = (u, v) \in G$ if $f(e) > 0$ then include edge (v, u) in G_f with $c_e^f = f(e)$. Note that the direction of the edge in G_f is reversed with respect to e . Such an edge in G_f is called a backward edge.

Thus, an edge $e \in G$ can give rise to two edges in G_f if $0 < f(e) < c_e$. Fig. 3 shows the residual graph for our running example under the sub-optimal flow assignment shown in Fig. 2(a). It turns out the residual graph allows us to think systematically about pushing and diverting the existing flow.

2.1 Augmenting paths in a residual graph

Let P be a simple $s - t$ path in G_f i.e. P does not visit any node more than once. We define $\text{bottleneck}(P, f)$ to be the minimum capacity of any edge in P . Now consider the following procedure.

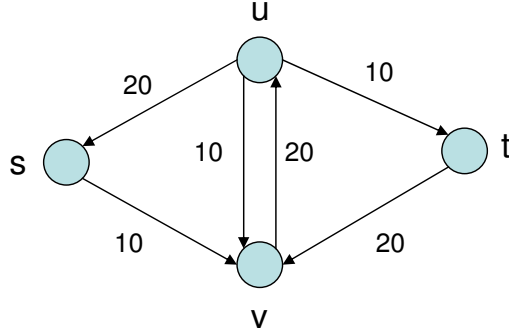


Figure 3: Residual graph for the suboptimal flow assignment shown in Fig. 2(a).

```

Augment( $P, f$ )
Let  $b = \text{bottleneck}(P, f)$ 
For each edge  $e = (u, v) \in P$ :
  If  $e$  is a forward edge then
    Increase  $f(e)$  by  $b$ .
  else  $((u, v)$  is a backward edge  $\implies (v, u)$  exists in  $G$ )
    Decrease  $f((v, u))$  in  $G$  by  $b$ .
  endif
endfor
Return ( $f$ ).

```

One usually refers to the $s - t$ path P in G_f as an augmenting path. Suppose we denote the new flow obtained after running the $\text{Augment}(P, f)$ procedure f' .

Lemma 1 f' is a valid flow for G .

Proof. Recall that a valid flow satisfies the capacity and conservation constraints. We first verify that f' satisfies the capacity constraints. Note that since f' differs from f only on the edges in P we need to check the conditions only on those edges. Let (u, v) be an edge in P and note that $\text{bottleneck}(P, f) \leq c_{(u,v)}^f$. There are two possible cases

- **Case 1.** (u, v) is a forward edge.
In this case $c_{(u,v)}^f = c_e - f(e)$. Therefore $0 \leq f(e) \leq f'(e) = f(e) + \text{bottleneck}(P, f) \leq f(e) + c_e - f(e) = c_e$.
- **Case 2.** (u, v) is a backward edge.
Since (u, v) is a backward edge, we have that it corresponds to edge $e' = (v, u)$ in G . Next we have $c_{(u,v)}^f = f(e')$. Now $c_{e'} \geq f(e') \geq f'(e') = f(e') - \text{bottleneck}(P, f) \geq f(e') - f(e') = 0$.

The flow conservation conditions can also be verified in a similar manner. ■

We shall now see that the augmentation procedure can be used to solve the maximum flow problem. Towards this end we have the following algorithm

Maximum Flow Algorithm

```
Initialize  $f(e) = 0$  for all  $e \in G$ .
While there is an  $s - t$  path in  $G_f$ 
  Let  $P$  be a simple  $s - t$  path in  $G_f$ 
   $f' = \text{Augment}(P, f)$ .
  Update  $f$  to be  $f'$ .
  Update  $G_f$ .
EndWhile
Return  $(f)$ .
```

The algorithm as defined above is reasonably simple. Basically we keep augmenting as long as we can find an $s - t$ path P in the residual graph G_f . However there are some important questions that we need to answer before we can be satisfied about the validity of this algorithm.

1. Does the algorithm ever terminate ?
2. If the algorithm terminates how can we be sure that it indeed returns the maximum flow from s to t .

We now present a sequence of arguments that show that the algorithm definitely terminates.

Lemma 2 *At every intermediate stage of the Maximum Flow algorithm, the flow values $\{f(e)\}, e \in E$ and the residual capacities i.e. the capacities of the edges in G_f are integers.*

Proof. Recall that we started with the assumption that c_e for $e \in G$ is an integer. Now, the statement of the lemma is clearly true before the algorithm enters the while loop (since $f(e) = 0$ for all $e \in G$). Suppose it is true after j iterations. Note that at this stage $\text{bottleneck}(P, f)$ will be an integer $\implies f'$ will have integer values \implies the capacities in the residual graph corresponding to f' will also be integers. Thus the statement of the lemma holds true at the $j + 1$ stage as well. By the principle of mathematical induction we have the required proof. ■

We now show that the value of the flow $v(f)$ strictly increases after an augmentation procedure.

Lemma 3 *Let f be a flow in G , let P be a simple $s - t$ path in G and let f' denote the flow obtained by running $\text{Augment}(P, f)$. Then $v(f') = v(f) + \text{bottleneck}(P, f) > v(f)$.*

Proof. The first edge e in P must be an edge out of s in G_f . Recall that an edge (s, v) in G_f exists either because $(s, v) \in G$ or $(v, s) \in G$. However by assumption there is no edge $(v, s) \in G$ since there are no incoming edges into s in G . Therefore the edge $(s, v) \in G_f$ must be a forward edge. Thus at the end of $\text{Augment}(P, f)$ the flow on edge (s, v) increases by $\text{bottleneck}(P, f)$. Note that we do not change the flow value on any other edge incident on s . Therefore

$$v(f') = v(f) + \text{bottleneck}(P, f) > v(f) \tag{6}$$

This concludes the proof. ■

Finally note that the maximum value of flow is at most $\sum_{e \text{ out of } s} c_e = C^s$ (say). Now, since the augmentation procedure increases the value of the flow by at least one unit each time, it is clear that the algorithm can run for at most C^s iterations.

Our next goal is to show that the Maximum flow algorithm indeed returns a flow whose flow is as large as it can possibly be.

Definition 5 *s - t cut.* An *s - t cut* is a partition of the vertex set V into sets A and B such that $s \in A$, $t \in B$ and $A \cup B = V$ and $A \cap B = \phi$.

Definition 6 *Capacity of an s - t cut.* The capacity of an *s - t cut* denoted by $c(A, B) = \sum_{e \text{ out of } A} c_e$.

Intuitively it should be clear that any cut acts as a bottleneck to any flow since $s \in A$ and $t \in B$ i.e. the capacity of any cut should be an upper bound on the value of any flow. We now make this intuition precise and present a series of arguments that shall culminate in the required proof.

Lemma 4 *Let f be any s - t flow and (A, B) be an s - t flow. Then*

$$v(f) = f^{out}(A) - f^{in}(A). \quad (7)$$

Proof. By definition we have $v(f) = f^{out}(s)$ and by assumption $f^{in}(s) = 0$ which implies that we can write $v(f) = f^{out}(s) - f^{in}(s)$. Next note that all nodes in A except s are internal i.e. flow conservation is maintained at them. This yields for all $v \in A - \{s\}$

$$\begin{aligned} f^{out}(v) - f^{in}(v) &= 0, \text{ so that,} \\ v(f) &= \sum_{v \in A} (f^{out}(v) - f^{in}(v)) \text{ (since only non-zero term corresponds to } v = s.) \end{aligned} \quad (8)$$

Now, if an edge e has both endpoints in A then $f(e)$ appears twice in the above sum, once with a positive sign and once with a negative sign i.e. the contribution of e to the above sum is 0. On the other hand if e only has its head in A then $f(e)$ appears only once with a negative sign and if e only has its tail in A then $f(e)$ appears only once with a positive sign. Of course if e has neither its head nor its tail in A then it does not participate in the sum at all. Therefore, we have

$$\sum_{v \in A} (f^{out}(v) - f^{in}(v)) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) = v(f) \quad (9)$$

which concludes the proof. ■

The next lemma shows that the value of any flow is at most the capacity of any cut.

Lemma 5 *Let f be an s - t flow and (A, B) be an s - t cut. Then*

$$v(f) \leq c(A, B). \quad (10)$$

Proof. We have,

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ into } A} f(e) \quad (11)$$

$$= f^{out}(A) - f^{in}(A) \quad (12)$$

$$\leq f^{out}(A) \text{ (since } f^{in}(A) \geq 0) \quad (13)$$

$$\leq \sum_{e \text{ out of } A} c_e \quad (14)$$

$$= c(A, B) \quad (15)$$

which yields the required result. ■

This is a rather fundamental result. Note that the value of the upper bound does not depend on f i.e. the upper bound holds for any flow. Thus the above lemma states that

$$\text{“The value of every } s - t \text{ flow”} \leq \text{“The capacity of every } s - t \text{ cut”}$$

Now that we have access to a large number of upper bounds on $v(f)$ we proceed to show the maximality of the flow returned by the Maximum Flow algorithm. Let \bar{f} be the flow returned by the algorithm. We shall demonstrate a $s - t$ cut (A^*, B^*) such that

$$v(f) = c(A^*, B^*) \tag{16}$$

which shall imply that $v(f)$ is maximal.

Theorem 6 *If \bar{f} is an $s - t$ flow such that there is no $s - t$ path in the residual graph $G_{\bar{f}}$, then there is a cut (A^*, B^*) such that $v(f) = c(A^*, B^*)$. This means that \bar{f} is a maximum flow and (A^*, B^*) is a minimum cut.*

Proof. We shall define the cut (A^*, B^*) . Let $A^* = \{v \in G \mid \text{there is a } s \rightarrow v \text{ path in } G_{\bar{f}}\}$ and $B^* = V - A^*$. Clearly (A^*, B^*) is an $s - t$ cut since $s \in A^*$ and $t \in B^*$ (because the algorithm cannot find an $s - t$ path in $G_{\bar{f}}$). We have two claims that we shall prove now.

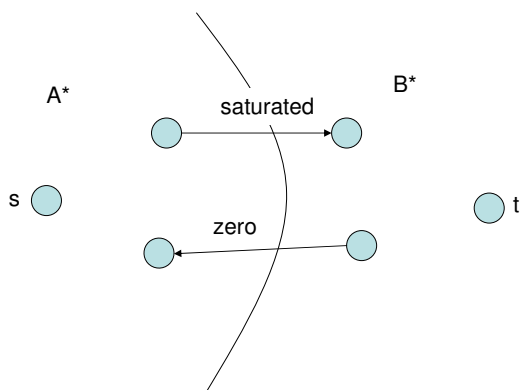


Figure 4: For appropriately defined A^* and B^* the flow on any edge belonging to the cut (A^*, B^*) is either saturated or zero depending on its orientation.

- a) Consider an edge $e = (u, v) \in G$ such that $u \in A^*$ and $v \in B^*$. We claim that $\bar{f}(e) = c_e$. To see this assume otherwise i.e. $\bar{f}(e) < c_e$. In this case (u, v) would be a forward path in $G_{\bar{f}}$. This would imply that the path from s to u in $G_{\bar{f}}$ could be extended to form a path from s to v . This is a contradiction since v was assumed to belong to B^* .
- b) Next, consider an edge $e' = (u', v')$ such that $u' \in B^*$ and $v' \in A^*$. We claim that $\bar{f}(e') = 0$, since if $\bar{f}(e') > 0$ then there exists a backward edge (v', u') in $G_{\bar{f}}$ which implies that we can extend the path from s to v' in $G_{\bar{f}}$ to obtain a path from s to u , which is a contradiction.

Thus as shown in Fig. 4, the chosen A^* and B^* and the flow \bar{f} are such that any edge from A^* to B^* is saturated (i.e. the flow value on the edge equals capacity) while any edge from B^* to A^* carries zero flow.

Now observe that

$$v(\bar{f}) = \bar{f}^{out}(A^*) - \bar{f}^{in}(A^*) \tag{17}$$

$$= \sum_{e \text{ out of } A^*} \bar{f}(e) - \sum_{e \text{ into } A^*} \bar{f}(e) \tag{18}$$

$$= \sum_{e \text{ out of } A^*} c_e - 0 \tag{19}$$

$$= c(A^*, B^*). \tag{20}$$

This finally shows that the flow \bar{f} returned by the Maximum flow algorithm is indeed the maximal flow. ■

Note that in the process we have also found a minimum $s-t$ cut i.e. an $s-t$ cut whose value is minimum. This is precisely the cut (A^*, B^*) . To see this suppose that (A^*, B^*) is not the minimum $s-t$ cut. This means that there exists another cut whose value is lower than $c(A^*, B^*)$. However that is a contradiction since $v(\bar{f})$ (as determined above) equals $c(A^*, B^*)$ but as shown earlier the value of any flow must be smaller than the value of any cut.