Math 482: Linear Programming[1] Mikhail Lavrov

Lecture 25: Augmenting Paths

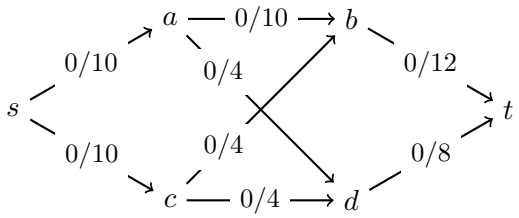
April 3, 2020 *University of Illinois at Urbana-Champaign*

1 Greedily increasing flow

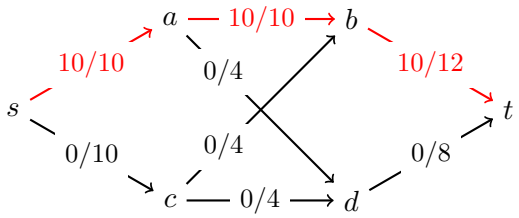
We've found a linear program for max-flow problems. But it turns out that there are better ways to solve this problem than by using a linear program. In the next few lectures, we'll explore some of these approaches.

To build intuition, we'll start by discussing an approach that seems like it ought to work, but doesn't quite finish the job. The idea here is to find directed paths from s to t along which we can increase the flow, and just—keep doing that.

Here's a reasonably typical network that we can use as an example. (As usual, I'll mark an arc with a fraction x/y to represent that x flow is being sent along that arc, and it has a maximum capacity of y.)

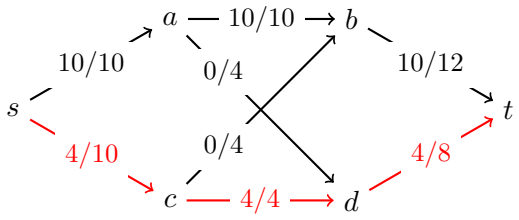


Our first step is to notice the promising path that goes $s \to a \to b \to t$. The arcs along this path all have capacity at least 10, so we can send 10 flow along this path:

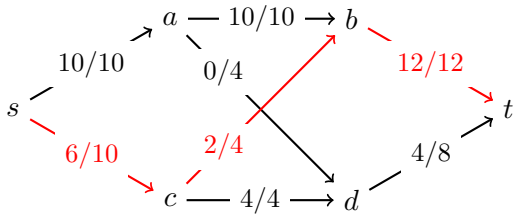


Similarly, there is the path $s \to c \to d \to t$. This one doesn't, let us make as much progress, but the bottleneck capacity along that path is 4, so we can send 4 flow along this path:

[1] This document comes from the Math 482 course webpage: https://faculty.math.illinois.edu/~mlavrov/courses/482-spring-2020.html



And there is room to send 2 more flow along the path $s \to c \to b \to t$ (but no more, because the arc (b, t) reaches its capacity of 12 when we do so):



At this point, we seem to be done. There are no paths that go from s to t along which we can increase the flow.

However, this still not the maximum flow: there are flows with larger value. It's just that we've gotten stuck at a feasible flow where we can't increase all the flows without decreasing all the flows. We need a better strategy.

(Disclaimer: if we had chosen the right paths from s to t in this argument, we could have gotten to the maximum flow. But the point is that we don't know which paths are the right ones, so we need to be smarter than that.)

2 An augmenting path

If you try to describe how we can improve the flow in the most recent step, you might say something like "we want to send less flow from a to b, and more of it to d. This lets us increase the flow from c to b." So, there's a super complicated explanation, and you can imagine that making arguments like this would get pretty tricky bigger networks.

There's a trick to describing such improvements that's simpler to think about, while still being powerful enough that we can always use it to get to the maximum flow. That trick is to find *augmenting paths*, which are slightly more general paths from s to t.

In an augmenting path, we go from s to t, but we're allowed to ignore the directions of the arcs: we can travel both forward and backward. For example, the following is a valid augmenting path:

$$s \xrightarrow{6/10} c \xrightarrow{2/4} b \xleftarrow{10/10} a \xrightarrow{0/4} d \xrightarrow{4/8} t$$

Notice that the arc from a to b is being traversed in the "wrong" direction.

An augmenting path is not allowed to cheat however you like, however. The rule is:

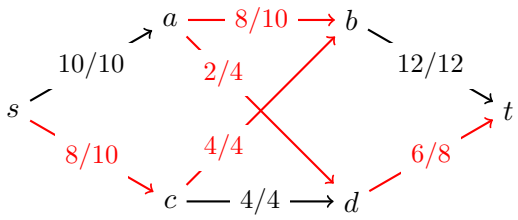
- You can go *forward* along an arc if it's still below its maximum capacity.

- You can go *backward* along an arc if it's at positive capacity.

If we find an augmenting path, we can let δ be the smallest margin by which it's valid: the smallest amount by which forward arcs are below maximum capacity, or backward arcs are above zero capacity. In this example, $\delta = 2$.

Then we can modify our current feasible flow by adding δ flow along every forward arc of the path, and subtracting δ flow along every backward arc. In our example, here is the change we make, and the result:

$$s \xrightarrow{+2} c \xrightarrow{+2} b \xleftarrow{-2} a \xrightarrow{+2} d \xrightarrow{+2} t$$



This is called "augmenting a flow along a path".

It's maybe not as obvious that this operation still gives a feasible flow: we should check that flow conservation is still satisfied at each node other than s and t. There are four cases for nodes that are affected by augmenting. Here they are (as fragments of an imaginary augmenting path in some other network):

$$\cdots \xrightarrow{+2} p \xrightarrow{+2} \cdots \qquad \cdots \xrightarrow{+2} q \xleftarrow{-2} \cdots \qquad \cdots \xleftarrow{-2} r \xrightarrow{+2} \cdots \qquad \cdots \xleftarrow{-2} s \xleftarrow{-2} \cdots$$

A node like p simply has two more units of flow coming through it: two less going in, and two less going out. So flow is still conserved (assuming flow was conserved before). Similarly, a node like s simply has two fewer units of flow coming through it.

Nodes like q and r are slightly different, but still fine. A node like q still has the same amount of flow coming in, but from a different source. Flow coming out is not changed, so flow conservation is not destroyed. Similarly, a node like r still has the same amount of flow coming out, but it goes somewhere else now; flow coming in is not changed at all.

3 The residual graph

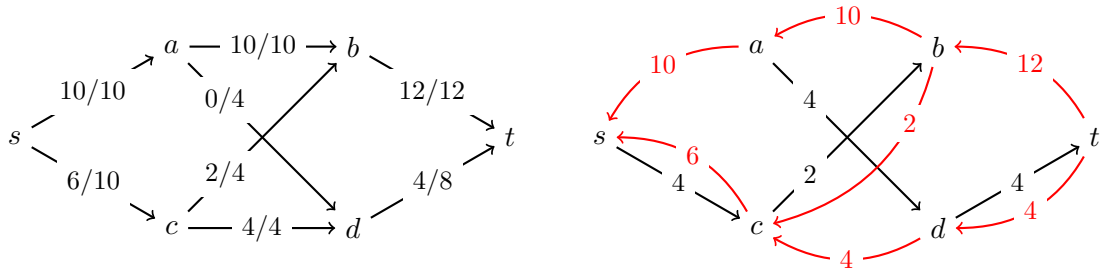
Augmenting paths are convenient to have, but hard to find. To try to do this systematically, we'll construct a *residual graph*.

In a residual graph, the nodes will remain the same, but the arcs we consider are different. We will add every arc that corresponds to a direction that an augmenting path could go. Namely:

- We keep an arc of our network in the residual graph if it's still below maximum capacity. We label it with its *residual capacity*, which is the amount of room still left to increase the flow. (For an arc (i, j) the residual capacity is defined to be $c_{ij} - x_{ij}$.)
- When an arc of our network has positive flow, we add a reverse arc to the residual graph. We also label it with its residual capacity, which is now the amount of room left to decrease the

flow. If the arc of our network was (i, j) with flow x_{ij}, the residual capacity of the reverse arc (j, i) in the residual graph is also x_{ij}.

Here is how this shakes out for the flow we had before the last augmenting step we did. (Forward arcs in the residual graph are drawn in black, backward arcs in red.)



Now, all it takes to find an augmenting path is to explore the residual graph along the arcs it has, starting from s, until we successfully get to t. For example, the augmenting path $s \to c \to b \leftarrow a \to d \to t$ is an actual path $s \to c \to b \to a \to d \to t$ in the residual graph.