1 Ford-Fulkerson

Consider the following flow network G. Compute a maximum flow in this network, using the Ford-Fulkerson algorithm.

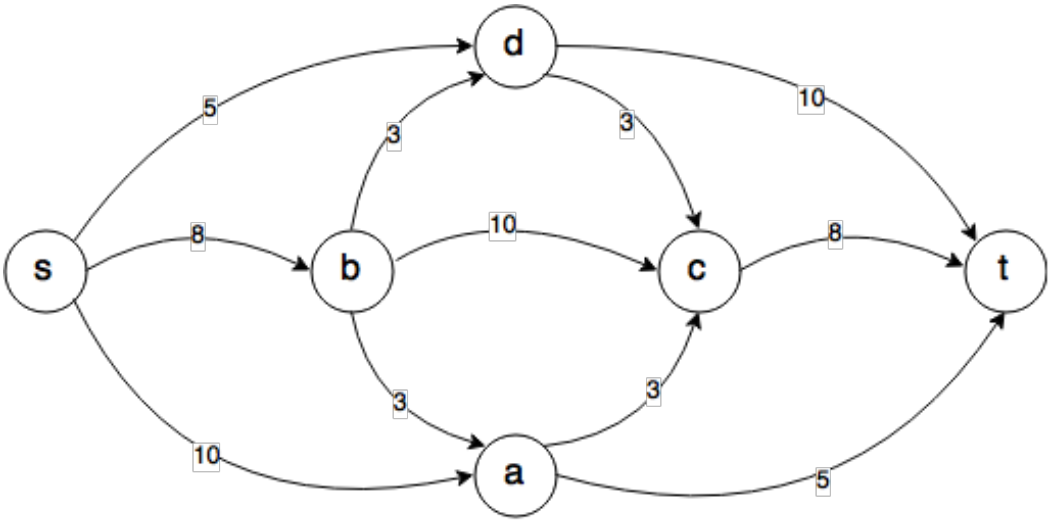

Figure 1: A flow network G

Let's start with constructing the residual flow network G_f. The initial residual flow network looks just the same as the graph diagram above. However, the number on each edge now represents the *unused capacity* - the maximum amount of flow that can be pushed through the edge. It is important to note that the flows in the residual network G_f are not real flows - not in the same sense as the actual flows in G. You can think of G_f as a data storage for storing the intermediate steps.

We look for an $s \to t$ path in this network, and we find the path $s \to b \to c \to t$.

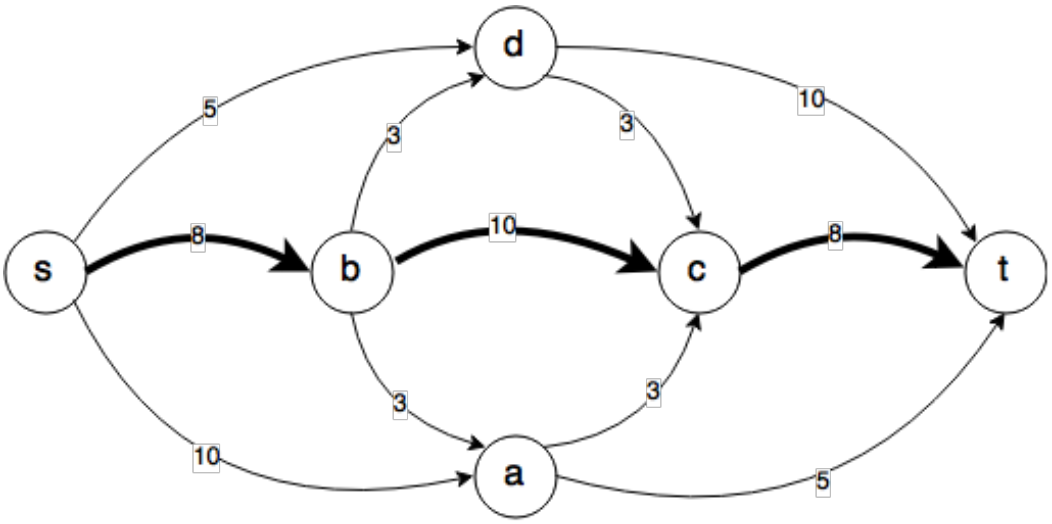

Figure 2: Picking an $s \to t$ path on the residual network G_f

The edges with minimum unused capacity on this path are $s \to b$ and $c \to t$, both have unused capacity 8. So we push amount of flow 8 through this path. Now we update the residual network G_f:

- Update forward edges on this path with the new unused capacities - subtracting each original unused capacity with the amount of flow pushed through this path. If the unused capacity was reduced to 0 for an edge, we simply remove the edge from the network.
- Add a backward edge for every forward edge on this path. Each new backward edge will have unused capacity equals to the amount of flow pushed through this path. If a backward edge already exists for a forward edge, add the amount of flow pushed to the unused capacity of the backward edge.

Basically, we use forward edge to store the "left-over" capacity, and backward edge to store the "surplus" capacity that can be reversed in the future when the backward edge becomes an forward edge in a new $s \to t$ path.

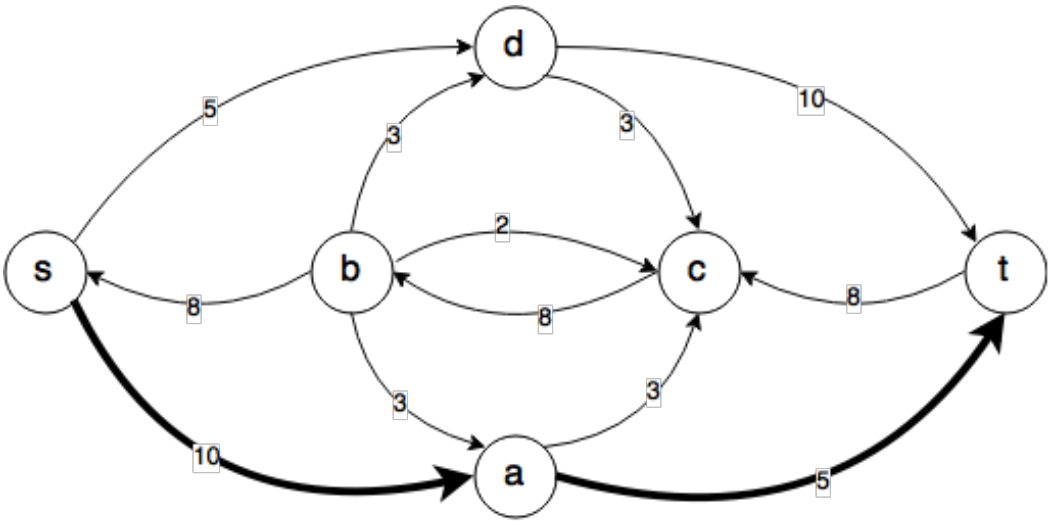

Figure 3: On G_f, update path $s \to b \to c \to t$, then pick a new $s \to t$ path $s \to a \to t$

After updating G_f, we look for a new $s \to t$ path in this network. We use the path $s \to a \to t$. We can push amount of flow 5 through this path. Update the forward edges on this path and add backward edges with new unused capacities.

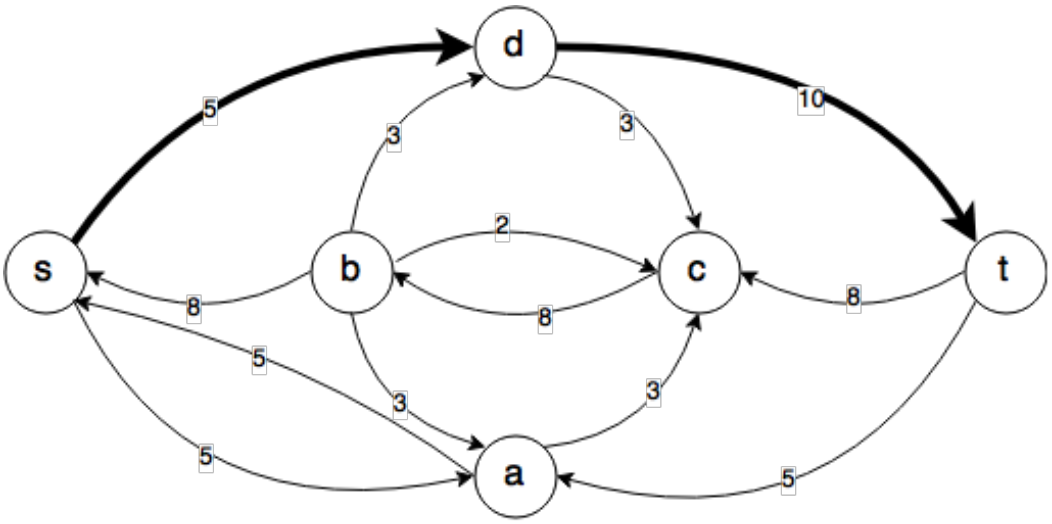

Figure 4: On G_f, update path $s \to a \to t$, then pick a new path $s \to d \to t$

We find another $s \to t$ path: $s \to d \to t$ to push amount of flow 5. Update the forward edges on this path and add backward edges with new unused capacities.

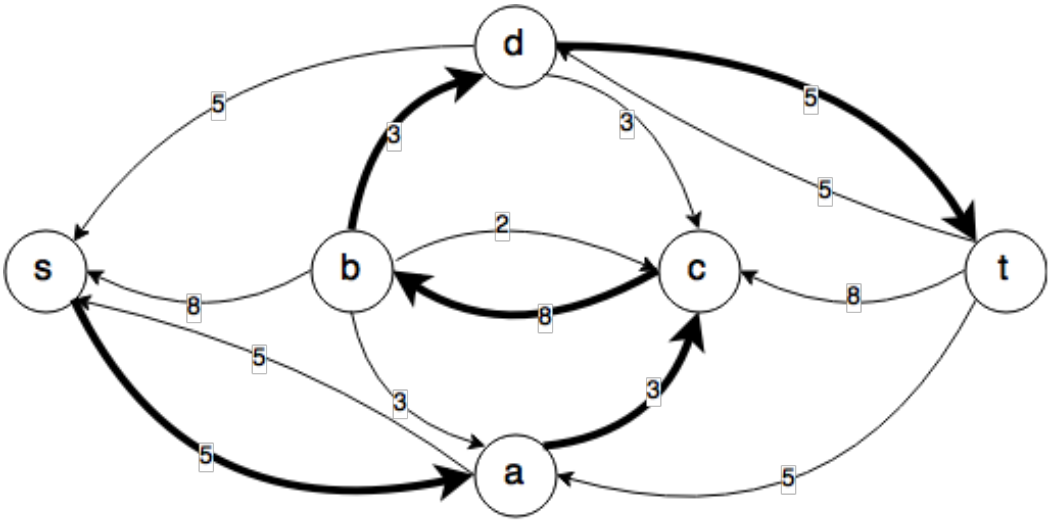

Figure 5: On G_f, update path $s \to d \to t$, then pick a new path $s \to a \to c \to b \to d \to t$

We find yet another path: $s \to a \to c \to b \to d \to t$. The maximum amount of flow we can push through this path is 3, because the minimum unused edge capacity in this path is 3 (edges $a \to c$ and $b \to d$). Note the edge $b \to c$, it was a backward edge in the path $s \to b \to c \to t$ which we picked in the first step, and now it becomes a forward edge in the current path. So by pushing flow through the current path, we are reversing the "surplus" flow in the original edge $b \to c$.

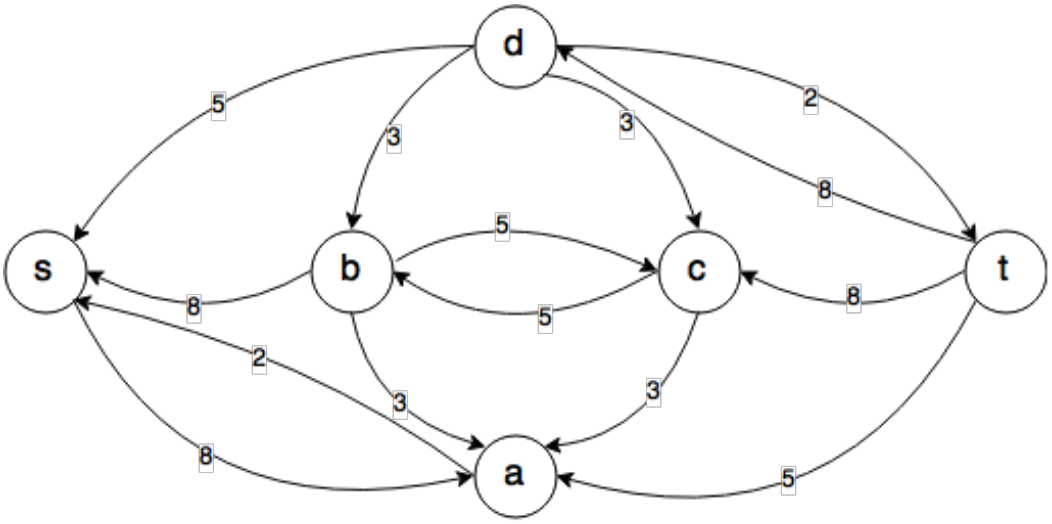

Figure 6: On G_f, update path $s \to a \to c \to b \to d \to t$, no more $s \to t$ path

After updating the last path, we cannot find any more $s \to t$ path in the residual network G_f. The total amount of flow we have pushed is $8 + 5 + 5 + 3 = 21$. According to the Ford-Fulkerson algorithm, this is the maximum flow in this network.

2 Cut Capacity and Flow

Consider the cut $X_0 = (\{s, b, c, d\}, \{a, t\})$. Identify all forward and all backward edges across X_0, then compute the capacity and the flow across X_0.

Let's first label the original network G with the result we obtained from the final residual network G_f. On G_f, each reverse edge (reverse w.r.t. G) stores the flow through the corresponding original edge on G. We label each edge on G with the amount of flow and capacity.

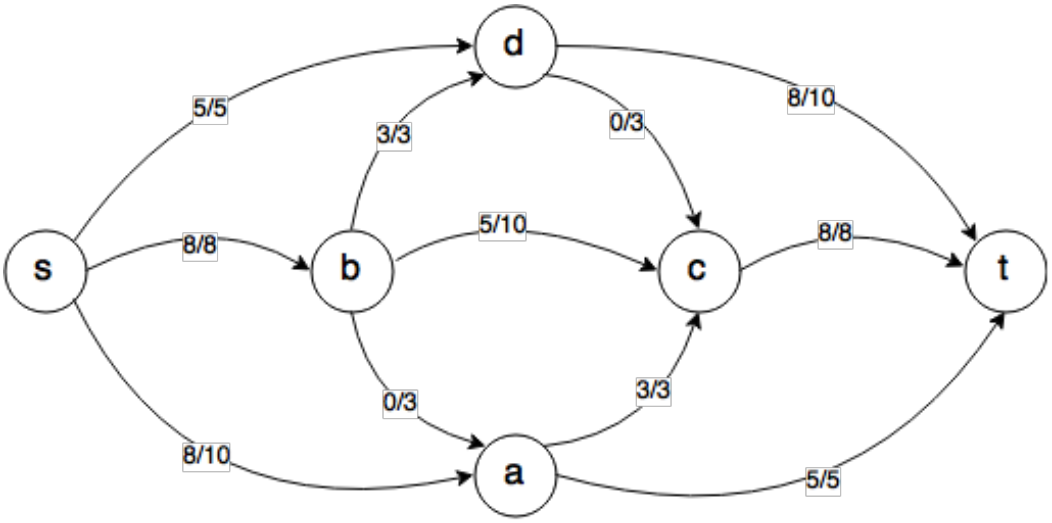

Figure 7: Label edges in G with [flow]/[capacity]

Now let's find the cut $X_0 = (\{s, b, c, d\}, \{a, t\})$.

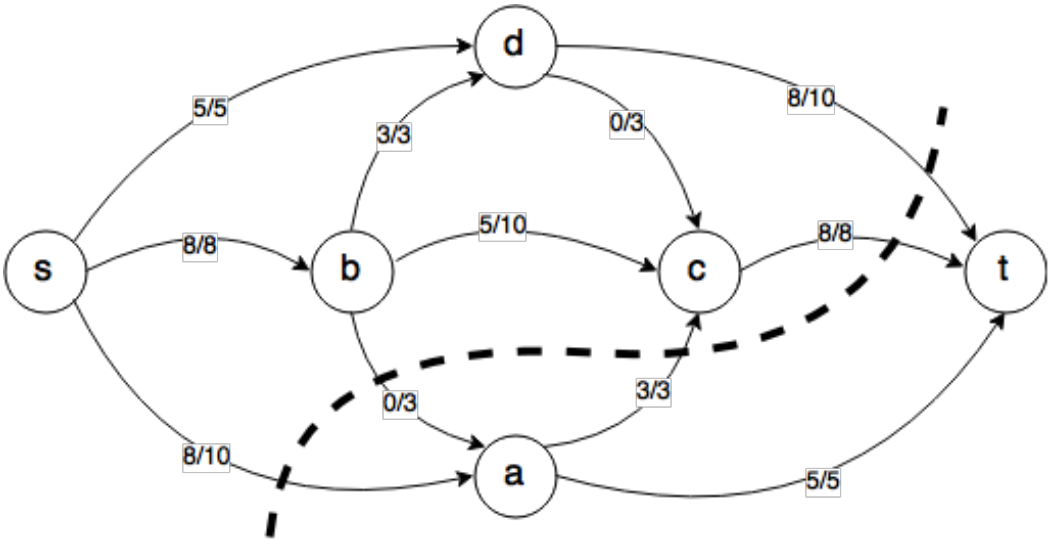

Figure 8: The cut $X_0 = (\{s, b, c, d\}, \{a, t\})$

The cut separate nodes in the network into two groups: $V_s = \{s, b, c, d\}$ and $V_t = \{a, t\}$. The forward edges are defined as the edges going from V_s to V_t, and the backward edges are defined as the edges going from V_t to V_s. Note the terms "forward edges" and "backward edges" are different from the ones we used in the last section. Here they refer to edges on the original network G.

- Forward edges: $\{s \to a,\ b \to a,\ c \to t,\ d \to t\}$
- Backward edges: $\{a \to c\}$

The capacity of the cut X_0 is:

$$\begin{aligned} c(X_0) &= \sum_{e_i \in forward\ edges} c(e_i) \\ &= c(s \to a) + c(b \to a) + c(c \to t) + c(d \to t) \\ &= 10 + 3 + 8 + 10 \\ &= 31 \end{aligned} \tag{1}$$

The flow across the cut X_0 is:

$$\begin{aligned} f(X_0) &= \sum_{e_i \in forward\ edges} f(e_i) - \sum_{e_i \in backward\ edges} f(e_i) \\ &= f(s \to a) + f(b \to a) + f(c \to t) + f(d \to t) - f(a \to c) \\ &= 8 + 0 + 8 + 8 - 3 \\ &= 21 \end{aligned} \tag{2}$$

Remember, when looking for the forward and backward edges of a cut, always work on the original network G. The edges and labels on the residual network G_f are not real edges and not real flows.

3 Minimum Cut

Find a cut in the network above whose capacity is equal to the value of your maximum flow (this provides a guarantee that your flow really is maximum). Use the algorithm outlined in the proof of the Ford-Fulkerson theorem.

Start with $X_1 = (\{s\}, \{a, b, c, d, t\})$ and flow from before. Edge $s \to a$ crosses cut forward with residual capacity 2, so set $X_1 = (\{s, a\}, \{b, c, d, t\})$. Now no more forward edge across the cut with residual capacity - all the forward edges across the cut are used at their full capacities.

As before, the capacity of the cut X_1 is:

$$\begin{aligned} c(X_1) &= \sum_{e_i \in forward\ edges} c(e_i) \\ &= c(s \to b) + c(s \to d) + c(a \to c) + c(a \to t) \\ &= 8 + 5 + 3 + 5 \\ &= 21 \end{aligned} \tag{3}$$

The flow across the cut X_1 is:

$$\begin{aligned} f(X_1) &= \sum_{e_i \in forward\ edges} f(e_i) - \sum_{e_i \in backward\ edges} f(e_i) \\ &= f(s \to b) + f(s \to d) + f(a \to c) + f(a \to t) - f(b \to a) \\ &= 8 + 5 + 3 + 5 - 0 \\ &= 21 \end{aligned} \tag{4}$$

Since $f(X_1) = c(X_1)$, X_1 is a minimum cut of G.

An alternative way to find the minimum cut in G is to look for "blocking edges" in the residual network G_f: starting with $V_s = \{s\}$ and $V_t = \{a, b, c, d, t\}$, do Depth-First Search starting from s and keep track of visited nodes: whenever a node n (other than s) is reached, $V_s = V_S \cup \{n\}$, and stops when no more out-going path that leads to un-visited node can be find for the current node – "blocked" by the in-coming edges. The resulting V_s and V_t form a minimum cut.

4 Multi-source and Multi-sink Network

Explain carefully how to solve the maximum flow problem in a multi-source, multi-sink network – one where there can be more than one source $s_1, \ldots, s_k$ and more than one sink $t_1, \ldots, t_l$. Justify that your solution is correct.

Solution Add "super-source" s with edges $s \to s_1, \ldots, s \to s_k$ each of capacity inf; add "super-sink" t with edges $t_1 \to t, \ldots, t_l \to t$ each of capacity ∞. (Instead of using ∞, can set capacity to sum of outgoing/incoming capacities).

Max flow in resulting network = max flow in original network because:

- Any flow in original network can be extended to a flow in resulting network (for new edges from super-source to source, set flow equal to total flow out of source; for new edges from sink to super-sink, set flow equal to total flow into sink) – hence, max flow in new network $\geq$ max flow in original network;
- Any flow in resulting network induces flow in original network (flow out of every source and into every sink limited only by edges in original network because of "infinite" capacities on new edges) – hence, max flow in original network $\geq$ max flow in new network.