Ford-Fulkson's Algorithm

At the end of Lecture 18 we gave an example (network) for which there exists a flow f and an s-t cut such that $|f| = c(S,T)$. Therefore, both f and (S,T) are optimal. Is this the case for *any* network? That is, for an arbitrary network, does there *always* exist f and (S,T) such that $|f| = c(S,T)$? The answer is yes! (This elegant, surprising result is referred to the max-flow min-cut theorem.) The proof is constructive: we will design an algorithm (i.e., the Ford-Fulkson algorithm) that actually always finds an s-t flow f and an s-t cut (S,T) that satisfies $|f| = c(S,T)$.

The Ford-Fulkson algorithm aims to find a maximum-flow of the given network. It employs the idead of *iterative improving*. The algorithm starts from a trivial flow f with $f(e) = 0$ for every $e \in E$. It then iteratively improve f. In each iteration, it finds a path p from s to t in the so-called *residual graph w.r.t. the current flow* f, and then improve f by *augmenting* path p. The value of f will be increased after each iteration, and the algorithm will terminate when such path cannot be found. Below see the pseudo-code of the Ford-Fulkson's algorithm.

```
Algorithm Ford-Fulkson (G = (V,E), s, t, c)
    init an s-t flow f with f(e) = 0 for any e ∈ E;
    while (true)
        build the residual graph G_f w.r.t. the current flow f;
        find an s-t path p in G_f;
        if such path cannot be found: return f;
        f ← augment(f, p);
    end;
end;
```

Residual Graph. The residual graph plays a central role in the Ford-Fulkson algorithm and in proving the max-flow min-cut theorem. Let f be an s-t flow of a network $(G = (V,E), s, t, c)$. We denote by $G_f = (V, E_f)$ the residual graph w.r.t. f. We emphasize that a residual graph is always associated with (i.e., w.r.t.) a flow of a network. Each edge $e \in E_f$ in the residual graph is also associated with a capacity, denoted as $c_f(e)$. The construction of the residual graph is given below. See Figure 1.

1. The residual graph has the same set of vertices with the network.

2. For each edge $(u,v) \in E$, there are two corresponding edges in E_f: the *forward-edge* (u,v) with capacity $c_f(u,v) = c(u,v) - f(u,v)$, and the *backward-edge* (v,u) with capacity $c_f(v,u) = f(u,v)$.

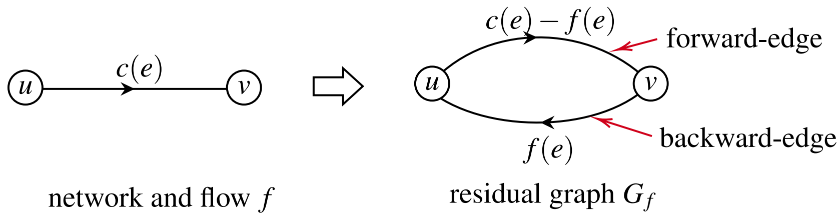
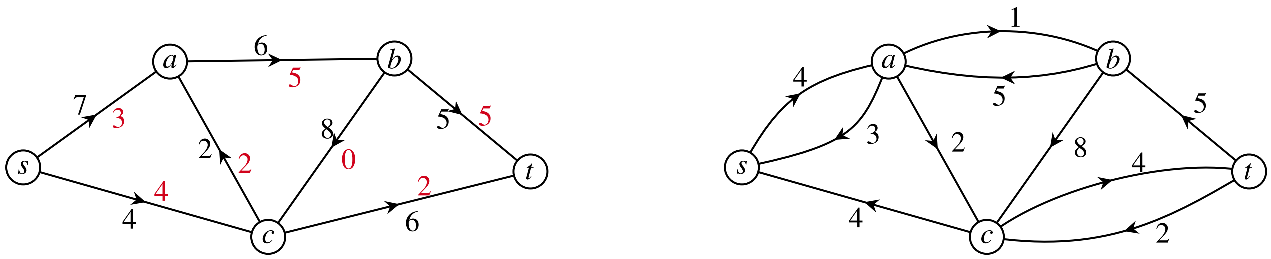


Figure 1: Definition of edges and capacities of the residual graph.

Edges in the residual graph with capacity of 0 will be removed. In other words, we always assume that edges in the residual graph have positive capacities. Therefore, if an edge $e = (u,v) \in E$ in the network carries a flow of 0, i.e., $f(e) = 0$, then the residual graph only includes the corresponding forward-edge (u, v) with capacity $c_f(u, v) = c(e)$; if an edge $e = (u, v) \in E$ is *saturated*, i.e., $f(e) = c(e)$, then the residual graph only includes the corresponding backward-edge (v, u) with capacity $c_f(v, u) = f(e) = c(e)$. See Figure 2.



network and flow f residual graph G_f

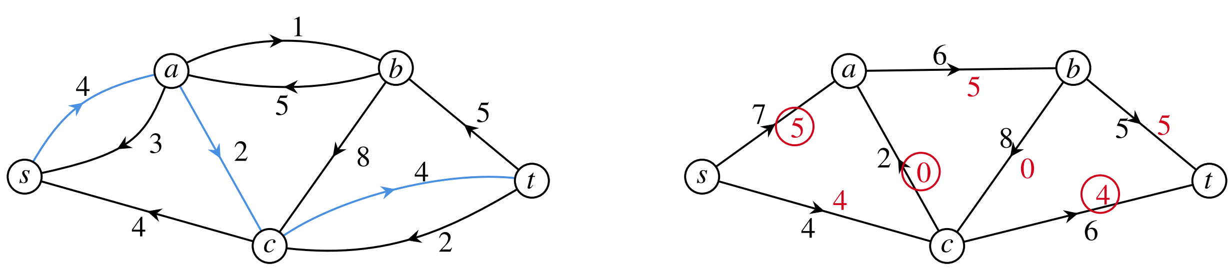
Figure 2: An example of residual graph.

Finding an s-t path in G_f. In each iteration of the Ford-Fulkson algorithm, after building G_f wrt the current flow f, it seeks a path p from s to t, called an s-t path, in residual graph G_f. The searching of such s-t path can be done by either BFS or DFS, starting from s. If such path cannot be found, the algorithm terminates and returns the current flow f. Otherwise, it *augments* p to obtain a flow with increased value.

Augmenting an s-t path p in G_f. To augment an s-t path p in G_f, we first calcualte the bottleneck capacity of p, which is defined as the smallest capacity (in the residual graph G_f) of all edges in p, formally written as $x(p) := \min_{e \in p} c_f(e)$. We then examine each edge $e \in p$, and update the flow of the corresponding edge in the network with the following rule.

1. If $e = (u, v) \in p$ is a forward edge, i.e., (u, v) is in the network, we update $f(u, v) \leftarrow f(u, v) + x(p)$;
2. If $e = (u, v) \in p$ is a backward edge, i.e., (v, u) is in the network, we update $f(v, u) \leftarrow f(v, u) - x(p)$;

The flow of other edges in the network (i.e., none of their corresponding forward edges or backward edges is in p) will not get affected. See an example in Figure 3.



$x(p) = 2$ $|f| = 7 + x(p) = 9$

Figure 3: Illustrating the procedure of augmenting continuing Figure 2. Suppose in the residual graph G_f given in Figure 2 the algorithm identifies s-t path $p = (s, a, c, t)$. Note that in this path (s, a) and (c, t) are forward edges and (a, c) is backward edge. After augmenting p the flow f is given in the right figure, where affected flow are circled.

We emphasize that, after augmenting, f remains a valid flow, i.e., f satisfies both the capacity condition and the conservation condition. The capacity condition remains satisfied owes to how the residual graph is constructed. Consider the two cases in augmenting: in either case, after augmenting, the flow remains non-negative and bounded by the capacity. Specifically,

1. if $e = (u,v) \in p$ is a forward edge, we know that $c_f(e) = c(u,v) - f(u,v)$ based on the construction of the residual graph. Since $x(p) \leq c_f(e)$, after augmenting the flow of edge (u,v) becomes $f(u,v) + x(p) \leq f(u,v) + c_f(e) = f(u,v) + c(u,v) - f(u,v) = c(u,v)$. And it's obvious that $f(u,v) + x(p) \geq 0$.

2. if $e = (u,v) \in p$ is a backward edge, we know that $c_f(e) = f(v,u)$ based on the construction of the residual graph. Since $x(p) \leq c_f(e)$, after augmenting the flow of edge (v,u) becomes $f(v,u) - x(p) \geq f(v,u) - c_f(e) = f(v,u) - f(v,u) = 0$. And it's obvious that $f(v,u) - x(p) \leq f(v,u) \leq c(v,u)$.

The reason why the conservation condition remains satisfied is that we augment an entire s-t path. Hence, for any vertex v in path p except s and t, the augmenting procedure adjusts the flow of two adjacent edges of v in the network, and such adjustments always cancel out. For example, suppose that in path p the two edges

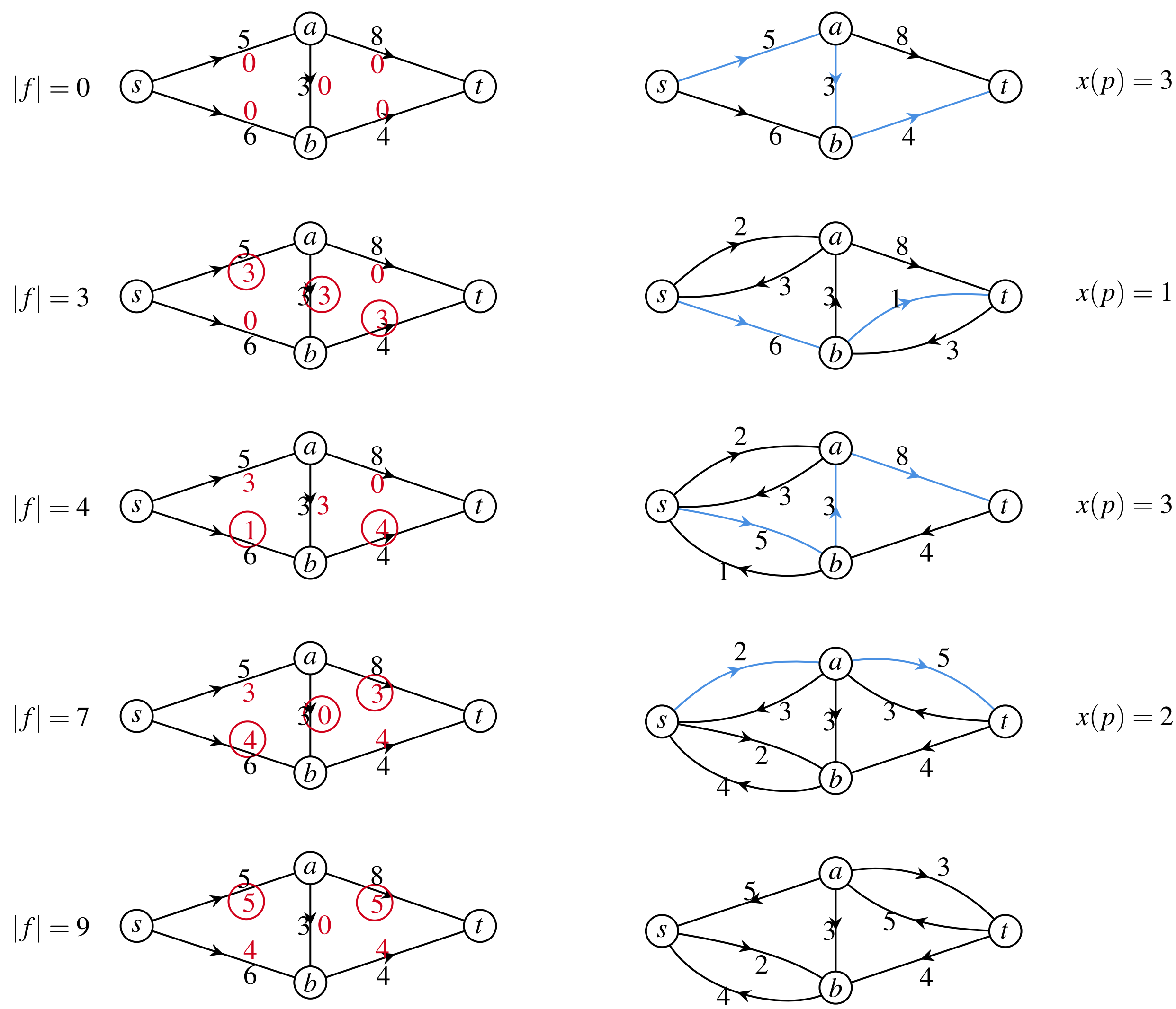


Figure 4: Running the FF algorithm on an instance. Each row shows an iteration. The left column shows the current flow f. The right column shows the residual graph w.r.t. f and the s-t path found (in blue).

are (u,v) and (v,w). If both edges are forward edges, then both $f(u,v)$ and $f(v,w)$ are increased by $x(p)$; so v remains balanced; If (u,v) is a forward edge and (v,w) is a backward edge, then $f(u,v)$ is increased by $x(p)$ and $f(w,v)$ is decreased by $x(p)$; again, v remains balanced; other two cases can be verified similarly.

After augmenting path p, the value of the new flow becomes $|f| + x(p)$. Since we assume that all edges in the residual graph has positive capacity, we have that $x(p) > 0$, which means the value of the new flow always gets improved. See an example of running Ford-Fulkson on a small network, given in Figure 4.