CS 6200: Algorithmics II	Fall 2020
Instructor: Avah Banerjee, S&T.	
Lecture: 9	**Dates: 9/14/2020**

1 Maximum Flow

In this section, we give a graph-theoretic definition of flow networks, discuss their properties, and define the maximum-flow problem precisely. We also introduce some helpful notation.

1.1 Flow Networks

A flow network $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a non-negative capacity $c(u, v) \leq 0$. It is also required that if E contains an edge (u, v) then there is no edge (v, u) in the reverse direction (No antiparallel edges). Also, there are no self-loops allowed. If $(u, v) \notin E$, then we define $c(u, v) = 0$. We distinguish two vertices in a flow network: a source s and a sink t. Vertices s and t can also be assumed to be demand and supply vertices respectively. For convenience, we assume that each vertex lies on some path from the source to the sink (Otherwise, there cannot be any flow from s to t). That is, for each vertex $v \in V$, the flow network contains a path $s \rightsquigarrow v \rightsquigarrow t$. The graph is therefore connected since each vertex other than s has at least one entering edge, since $|E| \geq |V| - 1$. Figure 1 shows an example of a flow network.

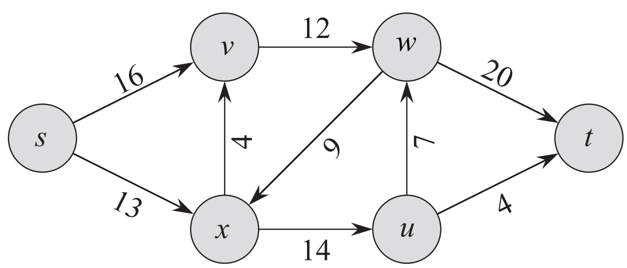


Fig. 1: A flow network $G = (V, E)$. Each edge is labeled with its capacity.

1.2 Flow

We now define flows more formally. Let $G = (V, E)$ be a flow network with a capacity function c. Let s be the source of the network, and let t be the sink. A flow in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

1. **Capacity Constraint:** For all $(u, v) \in E$, we require $0 \leq f(u, v) \leq c(u, v)$. Simply put this property says that the flow from one vertex to another must be nonnegative and must not exceed the given capacity.

2. **Flow Constraint:** For all $u \in V - \{s, t\}$ we require

$$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

This means that the total flow into a vertex other than the source or sink must equal the total flow out of that vertex—informally, "flow in equals flow out."

When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$. We call the nonnegative quantity $f(u, v)$ the flow from vertex u to vertex v. The value f_{val} of a flow f is defined as the total flow out of the source minus the flow into the source, which can be given by the following equation:

$$f_{val} = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \tag{1}$$

Consequently, The value f_{val} of a flow f can also be defined as the total flow in the sink minus the flow out of the sink, which can be given by the following equation:

$$f_{val} = \sum_{v \in V} f(v, t) - \sum_{v \in V} f(t, v) \tag{2}$$

An example of flow in a flow network in given in Figure 2.

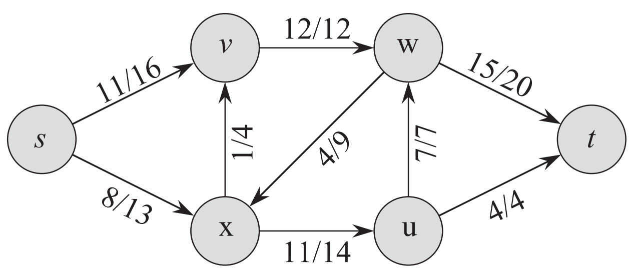


Fig. 2: A flow f in flow network $G = (V, E)$ with value $f_{val} = 19$. Each edge (u, v) is labeled by $f(u, v)/c(u, v)$, where the slash notation separates the flow and capacity.

1.3 Flow Networks with Multiple Sources and Sinks

A flow network may have several sources and sinks, rather than just one of each. We can reduce a flow network with multiple sources and multiple sinks to one with just a single source and sink. Figure 3 shows one such example. We add a supersource s and add a directed edge (s, s_i) with capacity $c(s, s_i) = \infty$ for each $i = \{1, 2, \ldots, m\}$. We also create a new supersink t and add a directed edge (t_i, t) with capacity $c(t_i, t) = \infty$ for each $i = \{1, 2, \ldots, n\}$.

1.4 Residual Networks

Given a flow network G and a flow f, the residual network G_f consists of edges with capacities that represent how we can change the flow on edges of G. An edge of the flow network can admit an

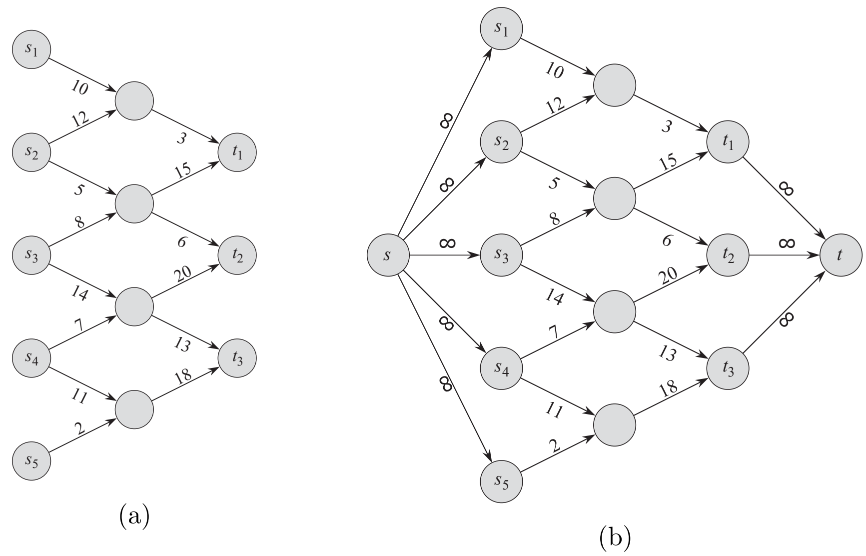


Fig. 3: Converting a multiple-source, multiple-sink flow network into a single source and a single sink. (a) flow network with five sources $S = \{s_1, s_2, s_3, s_4, s_5\}$ and three sinks $T = \{t_1, t_2, t_3\}$. (b) An equivalent single-source, single-sink flow network.

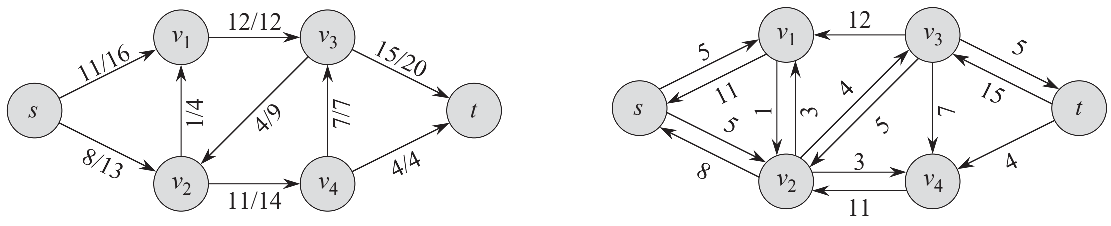
amount of additional flow equal to the edge's capacity minus the flow on that edge. If that value is positive, we place that edge into G_f with a residual capacity of $c_f(u, v) = c(u, v) - f(u, v)$. The only edges of G that are in G_f are those that can admit more flow; those edges (u, v) whose flow equals their capacity (saturated edges) have $c_f(u, v) = 0$, and they are not in G_f. The residual network G_f may also contain edges that are not in G. In order to represent a possible decrease of a positive flow $f(u, v)$ on an edge in G, we place an edge (v, u) into G_f with residual capacity $c_f(v, u) = f(u, v)$, This means that, an edge that can admit flow in the opposite direction to (u, v), at most canceling out the flow on (u, v). Figure 4 shows a flow network and its corresponding residual network. More formally, suppose that we have a flow network $G = (V, E)$ with source s and sink t. Let f be a flow in G, and consider a pair of vertices $u, v \in V$. We then define the **residual capacity** $c_f(u, v)$ by

$$c_f(u, v) = \begin{cases} \text{c(u-v) - f(u,v)}, & \text{if } (u, v) \in E \\ \text{f(u,v)}, & (v, u) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

If f is a flow in G and f' is a flow in the corresponding residual network G_f, we define $f \uparrow f'$, the **augmentation** of flow f by f', to be a function from $V \times V \to \mathbb{R}$, defined by

$$f \uparrow f'(u, v) = \begin{cases} \text{f(u,v)} + \text{f}'(u, v) - f'(v, u), & \text{if } (u, v) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The intuition behind this definition follows the definition of the residual network. We increase the flow on (u, v) by $f'(u, v)$ but decrease it by $f'(v, u)$ because pushing flow on the reverse edge in the residual network signifies decreasing the flow in the original network. Pushing flow on the reverse edge in the residual network is also known as cancellation.



(a) (b)

Fig. 4: (a) The flow network G with flow f and capacity c. (b) The residual network G_f. Edges with residual capacity equal to 0, such as (v_1, v_3), are not shown.

Lemma 1.1 *Let $G = (V, E)$ be a flow network with source s and sink t, and let f be a flow in G. Let G_f be the residual network of G induced by f, and let f' be a flow in G_f. Then the function $f \uparrow f'$ defined in equation (4) is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.*

Proof: CLRS chapter 26 page 18-19 ■

1.5 Augmenting Paths

Given a flow network $G = (V, E)$ and a flow f, an augmenting path p is a simple path from s to t in the residual network G_f . By the definition of the residual network, we may increase the flow on an edge (u, v) of an augmenting path by up to $c_f(u, v)$ without violating the capacity constraint on whichever of (u, v) and (v, u) is in the original flow network G. The shaded path in Figure 5 shows an augmenting path to the residual network G_f shown in Figure 4b. Treating the residual network G_f in the figure as a flow network, we can increase the flow through each edge of this path by up to 1 unit without violating the capacity constraint, since the smallest residual capacity on this path is $c_f(v_1, v_2) = 1$. We call the maximum amount by which we can increase the flow on each edge in an augmenting path p the **residual capacity** of p, given by

$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ } is \text{ } on \text{ } p\}$$

1.6 Cuts of Flow Networks

A cut (S, T) of flow network $G = (V, E)$ is a partition of V into S and $T = V - S$ such that $s \in S$ and $t \in T$. If f is a flow, then the **net flow** $f(S, T)$ across the cut (S, T) is defined to be

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u) \tag{5}$$

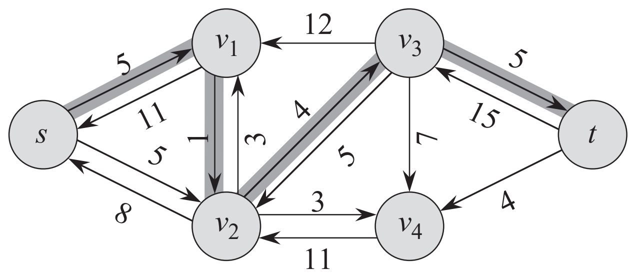


Fig. 5: The residual network G_f with augmenting path p shaded; its residual capacity is $c_f(p) = c_f(v_1, v_2) = 1$

The **capacity** of the cut (S, T) is

$$c(S,T) = \sum_{u \in S} \sum_{v \in T} c(u,v) \tag{6}$$

A minimum cut of a network is a cut whose capacity is minimum over all cuts of the network. For capacity, we count only the capacities of edges going from S to T, ignoring edges in the reverse direction. For flow, we consider the flow going from S to T minus the flow going in the reverse direction from T to S. Figure 6 shows the cut in the flow network of Figure 4a.

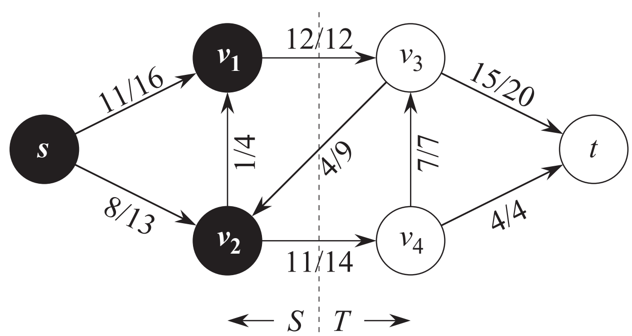


Fig. 6: A cut (S, T) in the flow network of Figure 4a, where $S = \{s, v_1, v_2\}$ and $T = \{v_3, v_4, t\}$. The vertices in S are black, and the vertices in T are white. The net flow across (S, T) is $f(S, T) = f(v_1, v_3) + f(v_2, v_4) - f(v_3, v_2) = 12 + 11 - 4 = 19$, and the capacity is $c(S, T) = c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26$.

1.7 The Ford-Fulkerson Algorithm (Generic Flow Algorithm)

The Ford-Fulkerson algorithm iteratively increases the value of the flow. We start with $f(u, v) = 0$; for all $u, v \in V$, giving an initial flow of value 0. At each iteration, we increase the flow value in G by finding an "augmenting path" in an associated "residual network" G_f. Once we know the edges of an augmenting path in G_f, we can easily identify specific edges in G for which we can change the flow so that we increase the value of the flow. Although each iteration of the Ford-Fulkerson method increases the value of the flow, we shall see that the flow on any particular edge of G may increase or decrease; decreasing the flow on some edges may be necessary in order to enable an algorithm to send more flow from the source to the sink. We repeatedly augment the flow until the residual network has no more augmenting paths.

Algorithm 1 FORD-FULKERSON-ALGORITHM (G, s, t)

```
initialize flow f to 0
while there exists an augmenting path p in the residual network G_f do
    augment flow f along p
    update G_f
end while
return f
```

Running Time: The analysis of Ford-Fulkerson depends heavily on how the augmenting paths are found. If we use either depth-first search or breadth-first search to find the path, Ford-Fulkerson runs in polynomial time. We assume that the capacities are integral capacities. If the capacities are rational numbers, we can apply an appropriate scaling transformation to make them all integral. The while loop in this case runs at most $|f^*|$ times, where f^* is the maximum flow. This is because the flow is increased, at worst, by 1 in each iteration. Finding the augmenting path inside the while loop takes $O(V + E')$, where E' is the set of edges in the residual graph G_f. This can be simplified to $O(E)$. So, the runtime of Ford-Fulkerson is $O(E|f^*|)$.

Remark 1.1 *Since the running time of FORD-FULKERSON depends on how we find the augmenting path, the algorithm might not even terminate if we choose p poorly. The value of the flow will increase with successive augmentations, but it need not even converge to the maximum flow value. The Ford-Fulkerson method might even fail to terminate if edge capacities are irrational numbers.*

Example 1.2 *Consider the six-node network shown in Figure 7a. Six of the nine edges have some large integer capacity say 10, two have capacity 1, and one has capacity* $r = \frac{\sqrt{5}-1}{2}$*, chosen so that* $1 - r = r^2$*. To prove that the Ford-Fulkerson algorithm can get stuck, we can watch the residual capacities of the edges with capacities* $1, 1$ *and* r *as the algorithm progresses. However, for simplicity we will not be updating the capacities on edges with value 10.*

Lets start the algorithm by choosing the augmenting path $s \rightsquigarrow u \rightsquigarrow v \rightsquigarrow t$ *with* $c_f(p) = 1$*. The edges* $(x, v), (v, u)$ *and* (u, y) *will now have residual capacities* r*,* 0*, and* 1 *(Figure 7b). Next, the augmenting path* $s \rightsquigarrow x \rightsquigarrow v \rightsquigarrow u \rightsquigarrow y \rightsquigarrow t$ *with* $c_f(p) = r$*, will give us residual capacities for the edges* $(x, v), (v, u)$ *and* (u, y) *as* 0*,* r^2 *and* r^2 *(Figure 7c). Augmenting the path for the graph in Figure 7c with* $c_f = r$ *we get the Figure 7d. Continuing along this line with* $c_f(p) = r$ *we get Figures 7e and 7f . This will go on if we continue finding augmenting paths with* $c_f = r$*.*

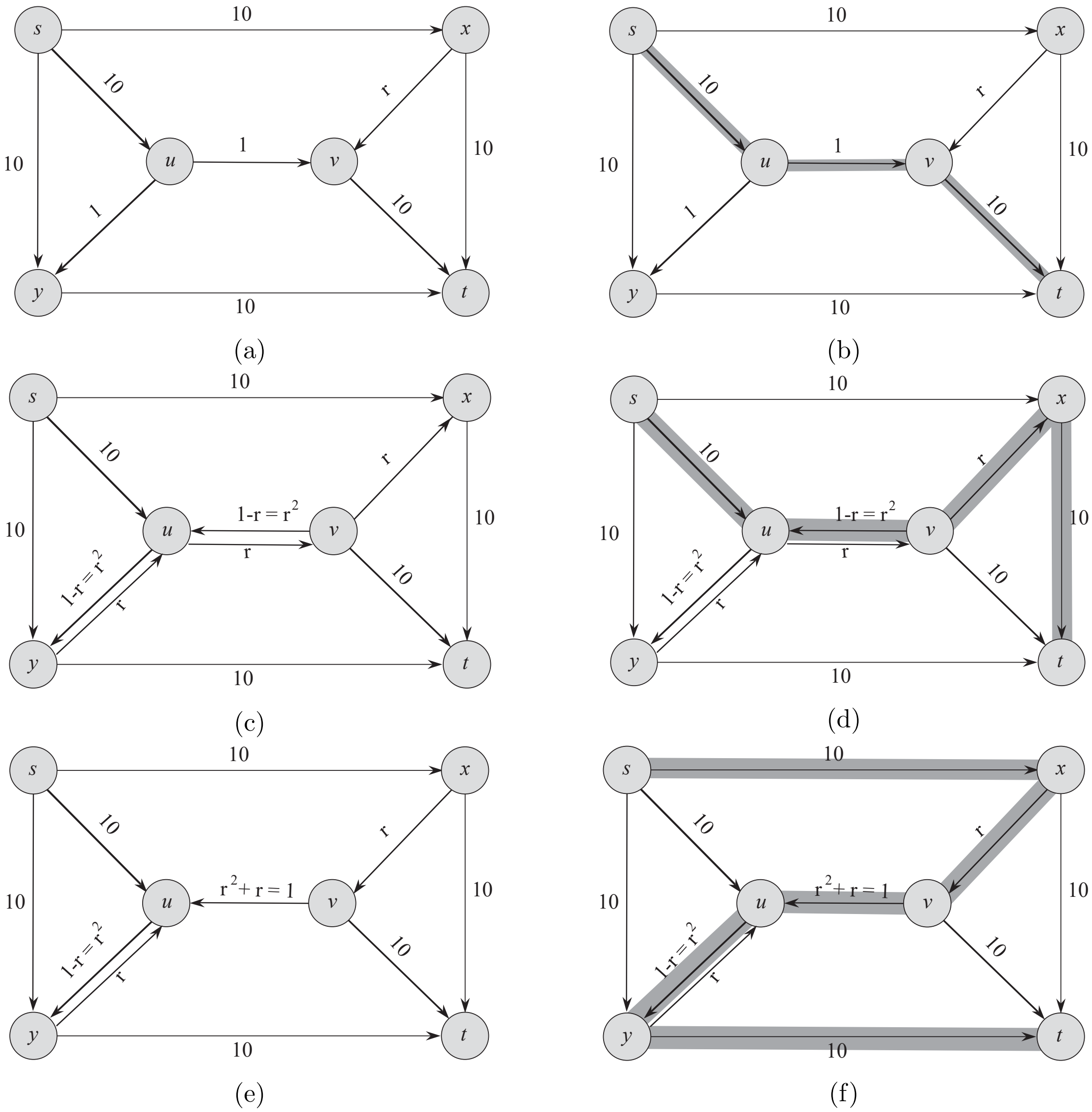


Fig. 7: The Ford-Fulkerson algorithm with one edge capacity as an irrational number

References and Further Reading

[1] Cormen, Thomas H., et al. Introduction to algorithms. MIT press, 2009. [Chapter 6- Section 26.1, 26.2]