Lecture 4: Network Flow

1 Network Flow

Definition: A network $N$ is a set containing:

- a directed graph $G(V,E)$;
- a vertex $s \in V$ which has only outgoing edges, we call $s$ the source node;
- a vertex $t \in V$ which has only incoming edges, we call $t$ the sink node;
- a capacity function $c : E \mapsto \mathbb{R}^+$,

where $\mathbb{R}^+$ is the set of non-negative real numbers.

Definition: A flow $f$ on a network $N$ is a function $f : E \mapsto \mathbb{R}^+$. Flow $f$ is a feasible flow if it satisfies the following two conditions:

1. Edge capacity limit:
   \[ \forall e \in E, \ 0 \leq f(e) \leq c(e) \]

2. Conservation of flow:
   \[ \forall v \in V \setminus \{s,t\}, \ \sum_{e \text{ leaving } v} f(e) = \sum_{e \text{ entering } v} f(e) \]

A network flow can be used as a model of packet routing in computer networks, finding a route from point $a$ to point $b$ in traffic/congestion grids, a supply chain problem, flow of water through pipes, or electricity flow in a circuit.

Definition: value of flow $f$ is defined as

\[ v(f) \equiv \sum_{e \text{ leaving } s} f(e) \]

Since $s$ and $t$ are the only nodes that do not conserve flow, the value of $f$ can be equivalently stated as the amount of flow entering $t$.

Proposition 1 For any feasible flow,

\[ v(f) = \sum_{e \text{ leaving } s} f(e) = \sum_{e \text{ entering } t} f(e) \]
**Proof:** This follows directly from conservation of flow.

\[
v(f) = \sum_{e \text{ leaving } s} f(e) = \sum_{e \text{ leaving } s} f(e) - \sum_{v \in V \cup \{s,t\}} \left[ \sum_{e \text{ entering } v} f(e) - \sum_{e \text{ leaving } v} f(e) \right] = \sum_{e \text{ entering } t} f(e),
\]

where the last line is due to the fact that each edge \( e \) appears twice in (1) - once as a leaving edge (with positive sign) and once as an entering edge (with negative sign) - except those edges entering \( t \) which appear exactly once and with positive sign.

**Definition:** An \( s-t \) cut, \( \text{cut}(A, B) \), is a partition of \( V \) into subsets \( A \) and \( B \) such that \( s \in A \) and \( t \in B \). We define the cut value, \( c(A, B) \), to be the sum of capacities of all the edges going from set \( A \) to set \( B \).

\[
c(A, B) = \sum_{e \text{ leaving } A, \text{ entering } B} c(e)
\]

**Remark 1:** Using a proof similar to Proposition 1, it is easy to show that for any \( \text{cut}(A, B) \)

\[
v(f) = \sum_{e \text{ leaving } A, \text{ entering } B} f(e) - \sum_{e \text{ leaving } B, \text{ entering } A} f(e).
\]

**Remark 2:** Since \( f(e) \geq 0, e \in E \) and \( f(e) \leq c(e) \)

\[
v(f) = \sum_{e \text{ leaving } A, \text{ entering } B} f(e) - \sum_{e \text{ leaving } B, \text{ entering } A} f(e) \leq \sum_{e \text{ leaving } A, \text{ entering } B} f(e) \leq \sum_{e \text{ leaving } A, \text{ entering } B} c(e) = c(A, B)
\]

Therefore any \( s-t \) cut value is an upper bound on \( v(f) \). Given a network \( N \), the max-flow problem is to find a feasible flow with the maximum possible value. Remark 2 implies that the value of the max-flow is upper bounded by the value of any cut \( (A, B) \). We will in fact show that equality is attained for the minimum cut value i.e.

\[
\max_{f \text{ feasible}} v(f) = \min_{\text{cut}(A,B)} c(A, B).
\]

### 1.1 Ford Fulkerson Algorithm

In the rest of the lecture, we examine a “greedy” algorithm for finding the max-flow, we show its sub-optimality by an example, and then modify it to Ford-Fulkerson algorithm which is an efficient method for finding the max-flow. The greedy algorithm is presented in Algorithm 1.

**Algorithm 1**  First tentative algorithm (greedy)

Initialize \( f(e) = 0 \) for all \( e \in E \).
repeat
Find path $P$ between $s$ and $t$ such that $\min_{e \in P} (c(e) - f(e)) > 0$, we call such a path unsaturated.
Let $df = \min_{e \in P} (c(e) - f(e))$: $f(e) = f(e) + df$, $e \in P$.
until No more unsaturated $s-t$ paths
end

The greedy algorithm does not find the max-flow in general graphs. A simple counterexample can be seen in the Figure 1.

![Figure 1: Two potential outcomes of the greedy algorithm. a) The optimal flow is achieved. b) no more flow can be pushed greedily through the network.](image)

In Figure 1(b), the greedy algorithm made a bad choice for the first unit of flow to push through. There are no remaining unsaturated $s-t$ paths in the network, but clearly we did not find the maximum flow. We modify the algorithm such that we can revise the paths later in the run of the algorithm. This is the rough idea of Ford-Fulkerson algorithm.

In order to describe Ford-Fulkerson algorithm, we first define a residual network of network $N$ with respect to flow $f$.

**Definition:** A residual network $R(N, f)$ is a network with vertex set $V$ and with edge set $E_r$ constructed as follows:

For every $e \in E$:

- if $f(e) < c(e)$ place an edge with capacity $c'(e) = c(e) - f(e)$ in the same direction as $e$.
- if $f(e) > 0$ place an edge with capacity $c'(e) = f(e)$ in the opposite direction of $e$.

The advantage of the residual network $R(N, f)$ is that any path $P$ from $s$ to $t$ in $R(N, f)$ gives a path along which we can increase the flow. Building the residual network and augmenting along an $s-t$ path forms the core of Ford-Fulkerson method described in Algorithm 2.

**Algorithm 2** Ford-Fulkerson, 1956

Start with $f(e) = 0$, \(\forall e \in E\).
while there is a path $P$ from $s$ to $t$ in $R(N, f)$ do
send a flow of value $df = \min_{e \in P} c'(e)$ in $R$ along $P$.
augment $df$ in $(N, f)$ using the above flow.
rebuild the residual network $R(N, f)$.
end while
Output $f^*$. 

Here, augmenting flow $df$ in $(N, f)$, means that for each $e \in P$, $f(e) = f(e) + df$ and $e \in E \setminus P$, $f(e) = f(e)$.

**Lemma 1** If all edge capacities of $N$ are integral i.e., $c(e) \in \mathbb{N} \cup \{0\}$, $e \in E$, Ford-Fulkerson terminates.

**Proof:**
Since the capacities are integral, it is not hard to see that the capacity of every edge in $R(N, f)$ is always integral. At each step, $df$ is at least one; thus the value of flow $f$ increases by at least one. Since $v(f) \leq \sum_{e \text{ leaving } s} c(e)$ and the capacities are finite, $v(f)$ cannot keep increasing, hence the algorithm stops after a finite step.

**Theorem 1** If Ford-Fulkerson terminates, it outputs a maximum flow.

**Proof:**
Suppose the algorithm terminates at step $t$, this means that there is no path from $s$ to $t$ in $R(N, f^*)$; $s$ and $t$ are disconnected. Let $S$ be the set of nodes reachable from $s$, i.e., $v \in S$ iff there exists a path from $s$ to $v$; let $T = V \setminus S$. We claim that $v(f^*) = c(S, T)$; before proving this claim, note that earlier we showed that for every feasible flow $f$ and every cut $(A, B)$, $v(f) \leq c(A, B)$. Thus $v(f^*) = c(S, T)$ implies that $f^*$ is the max-flow and cut $(S, T)$ is the minimum cut.

In order to prove $v(f^*) = c(S, T)$, we consider the residual graph; first, we show that there is no path from $S$ to $T$. By contradiction, assume there exists $e$ from $v_1 \in S$ to $v_2 \in T$; this means that $v_2$ is reachable by $s$ which contradicts with $v_2$ being in $T$. Next, we look back at the original network $N$ suppose $e$ is from $S$ to $T$, the only case that $e$ does not exist in $R(N, f)$ is that $f(e) = c(e)$; for $e'$ from $T$ to $S$, if $f(e') > 0$ then there will be an edge in the opposite direction of $e'$ in $R(N, f)$ i.e., an edge from $S$ to $T$; since there does not exist such an edge, we conclude that $f(e') = 0$. In the last lecture we showed that

$$v(f) = \sum_{e \text{ leaving } S, \text{ entering } T} f(e) - \sum_{e \text{ leaving } T, \text{ entering } S} f(e).$$

The claim follows by substituting $f(e) = c(e)$, for $e$ from $S$ to $T$ and $f(e') = 0$ for $e'$ from $T$ to $S$.

Two important corollaries follow from the proof of Ford-Fulkerson:

**Corollary 1 (Max-Flow/Min-Cut)** The minimum cut value in a network is the same as the maximum flow value.

**Corollary 2 (Integral Flow)** If all edge capacities in a network are non-negative integers, then there exists an integral maximum flow.