CSC 373 - Algorithm Design, Analysis, and Complexity **Summer 2016**

Lalla Mouatadid

Network Flows: Introduction & Maximum Flow

We now turn our attention to another powerful algorithmic technique: **Local Search**. In a local search algorithm, we start with an arbitrary solution to our problem, then keep refining this solution by making small repeated local changes that will increase the quality of our solution. Once a solution converges to a local optimum, then no number of small changes will increase its quality and we are done. But this does not always mean that this local optimum is the optimal solution to our problem. In this lecture, we focus on a well structured instance of local search where the local optimum is indeed the global optimum.

Definitions

- A **flow network** is a directed graph $G(V, E)$ with the following properties:

  1. There is a positive weight function on E, called *capacity*, $c : E \to \mathbb{R}^+$.

  2. There are two designated vertices s and t in V, such that $s \neq t$. s is the source of the network and t the terminal.

  3. There are **no** incoming edges towards s and **no** outgoing edges from t.

- We use (G, s, t, c) to denote a flow network, with source s, terminal t, and edge capacities c.

- Given a flow network (G, s, t, c), a **flow** in G is a function $f : E \to \mathbb{R}^+$ that satisfies the following constraints:

  1. **Capacity constraint** : For all edges $e \in E$, $0 \leq f(e) \leq c(e)$. Meaning no edge gets a flow that exceeds its capacity.

  2. **Flow conservation** : For every vertex $v \in V \backslash \{s, t\}$, the flow going into v equals the flow coming out of v:

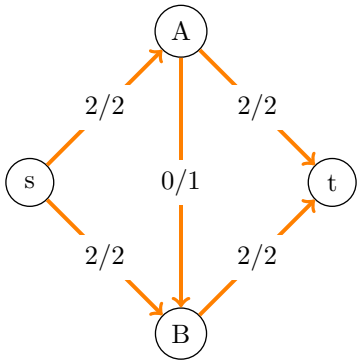
$$\sum_{u:(u,v)\in E} f(u, v) = \sum_{w:(v,w)\in E} f(v, w)$$

- The **value** of the flow, denoted $val(f)$ is the total value of the flow leaving the source s:

$$val(f) = \sum_{u:(s,u)\in E} f(s, u)$$

Example : Consider the network below[1]. $val(f) = 4$, as we can send 2 units of flow down the paths $s - A - t$ and $s - B - t$.

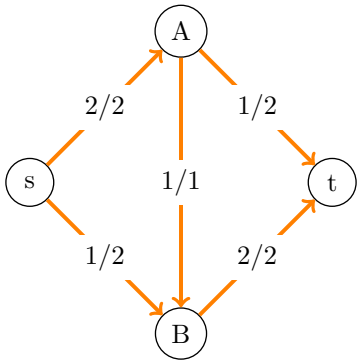
[1] note that the label on an edge f/c denotes the flow and the capacity of that edge



Given a flow network, our goal is to send as much flow as possible from s to t. We'll see how a local search algorithm solves this problem. Formally, we have the following problem:

The Maximum Flow Problem:
Input: A flow network (G, s, t, c)
Output: A flow f on G where $val(f)$ is maximized.

Since we said this is an instance of local search, we should be able to start with any solution and try to improve it over a number of iterations. For instance, we could start by send two units of flow down the path $s - A - t$ in the example above, then improve on this solution by sending another 2 units of flow down $s - B - t$. Since the capacities of the source s are now saturated, we know we can't improve on this solution; and thus $val(f) = 4$ is the local optimum. It turns out for this example, 4 is also an optimal solution for this network
Now suppose our initial starting solution was the path $s - A - B - t$, this path gives us a flow of value 1. We can improve it by sending another unit of flow down $s - A - t$ and another unit of flow down $s - B - t$. The flow network would then look like:



The total value of this flow is $val(f) = 3$, but we know there is a better solution. Which path can we use to increase the flow? If we claim that local search on the instance of max flow is optimal, then we should be able to send one more unit of flow the network.

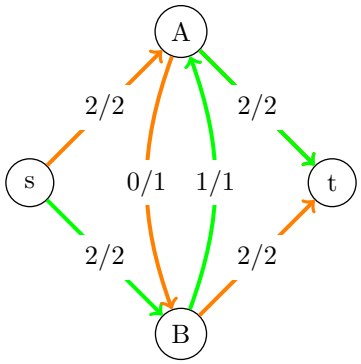
Before we attack this problem, we need a few more definitions. Given a flow network (G, s, t, c), an **augmenting path** $\mathcal{P}$ is an s, t path in G where we can push more flow down the network. That is, $\mathcal{P} = v_1, v_2, ..., v_k$ where $c(v_i, v_{i+1}) - f(v_i, v_{i+1}) > 0$ for all $1 \leq i < k$ and $v_1 = s, v_k = t$.

In the example above, there are no augmenting paths to push that 4^{th} unit of flow, but we get around this

problem by *redirecting* the flow in the network. There are two types of augmenting paths in a network:

Case 1. An s, t path where every edge has some unused capacity: $c(e) - f(e) > 0$ for every edge $e \in \mathcal{P}$. We just push at least one more unit of flow, thus increasing $val(f)$.
Case 2. An s, t path where some edges have unused capacity and some are saturated, but we can redirect the flow on the saturated edges.

To illustrate this 2^{nd} case, consider the example above where we $val(f) = 3$. Since there is no augmenting path that falls in case 1, we will try to redirect some flow in G. In particular, we will augment the flow along the path $s - B - A - t$ and in doing so, redirect 1 unit of flow down the edge (B, A). Notice that when we redirected the flow down (B, A), we decreased the flow on the edge (A, B) since it is the exact flow we redirected.



To formalize this concept, we introduce a data structure that allows us to keep track of augmenting paths in a network.

Definition:

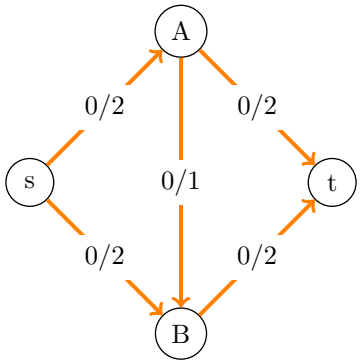
Let (G, s, t, c) be a flow network and f a flow on G. A **residual graph** G_f of G is an edge-weighted directed graph on the same vertex set V as G and the edge set E' where E' is constructed as follows: If there is an edge $(u, v) \in G$, we add two edges (u, v) and (v, u) to E' with capacities:

$$c_f(u, v) = c(u, v) - f(u, v) : \text{the amount of available capacity left on the edge } (u, v)$$
$$c_f(v, u) = f(u, v) : \text{The amount of flow we are allowed to redirect}$$

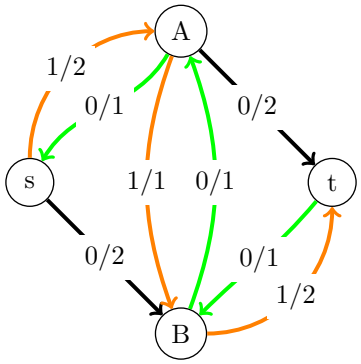
c_f denotes the capacity of an edge in the residual graph G_f. If an edge $e' \in E'$ has capacity $c_f(e') = 0$, we just remove e' from G_f.

Now we can redefine an augmenting path with respect to the residual graph: An augmenting path $\mathcal{P}$ on G_f is an s, t path on G_f consisting of edges with positive capacity: $c_f(e) > 0$ for all $e \in \mathcal{P}$.

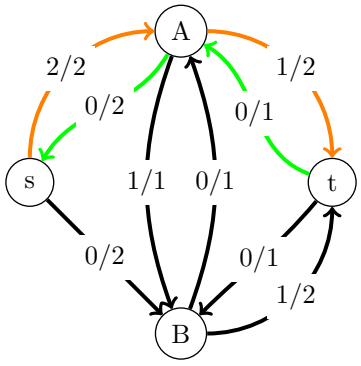
For further clarity, let's consider the example above and produce what G_f looks like at every iteration:



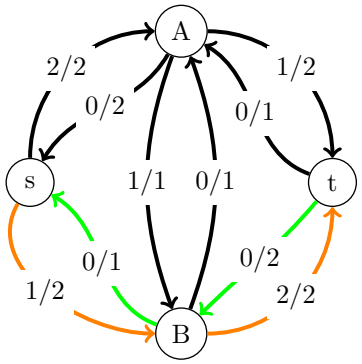
Send 1 unit of flow using $\mathcal{P} = s - A - B - t$:



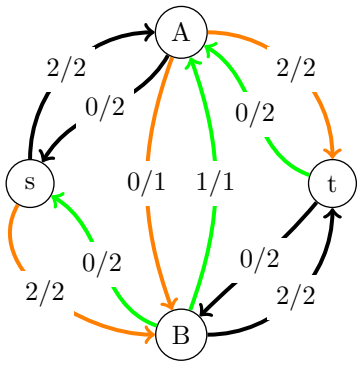
Send 1 unit of flow using $\mathcal{P} = s - A - t$



Send 1 unit of flow using $\mathcal{P} = s - B - t$



Send 1 unit of flow using $\mathcal{P} = s - B - A - t$



There are no outgoing edges from s with leftover capacity, so we can't start an augmenting path, and we are done. It is not always the case however that we saturate the outgoing edges from s before we are done, there could be leftover capacity on an (s, v) but no augmenting path in G_f.

So in general, we keep performing augmentations along paths $\mathcal{P}$ by increasing the flow on edges with available capacity, and decreasing the flow when redirecting. How much flow can we send at every augmentation step? Let $\mathcal{P} = v_1, v_2, ..., v_k$ be an augmenting path with $v_1 = s, v_k = t$. Since $\mathcal{P}$ is an augmenting path, we know that $c_f(v_i, v_{i+1}) > 0$ for all edges (v_i, v_{i+1}) in $\mathcal{P}$ with $1 \leq i < k$. Therefore the most flow we can send down $\mathcal{P}$ is just $\min_{1 \leq i < k} c_f(v_i, v_{i+1})$. Why?

So far, we haven't said how to solve our Maximum Flow problem! Ford and Fulkerson proposed this local search algorithm to solve it: Keep augmenting f using augmenting paths until there are no more augmenting paths! The claim is when the algorithm halts, the flow we end up with is maximized.

Algorithm 1 Ford-Fulkerson

Input: A flow network (G, s, t, c)
Output: A flow f on G where $val(f)$ is maximized

```
Set f(e) = 0 for every edge in G.
while There is an augmenting path P in G_f do
    Augment f using P
end while
Return f
```

We'll prove the correctness of this algorithm in the next lecture. The proof is an immediate corollary of a well know theorem: The Max-Flow/Min-Cut Theorem.