Recitation 11 - Max Flow, Maximal Matching

Sebastian Laudenschlager

sebastian.laudenschlager@colorado.edu

April 13, 2018

Flow Networks

- Model a directed graph $G = (V, E)$ as a flow network.
- Each edge (u, v) has some capacity $c(u, v) \geq 0$.
- If $(u, v) \notin E$, then $c(u, v) = 0$.
- There is a source vertex s and a sink vertex t.
- We assume that $s \to v \to t$ for all $v \in V$, i.e. each vertex lies on some path from source to sink.

Flow Properties

- **Capacity constraint:** For all $u, v \in V$, $0 \leq f(u,v) \leq c(u,v)$.
- **Conservation of Flow:** For all $u \in V - \{s, t\}$,

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

- If $(u,v) \notin E$, then $f(u,v) = 0$.
- $f(u,v)$ is called the flow from vertex u to vertex v.

Residual capacity

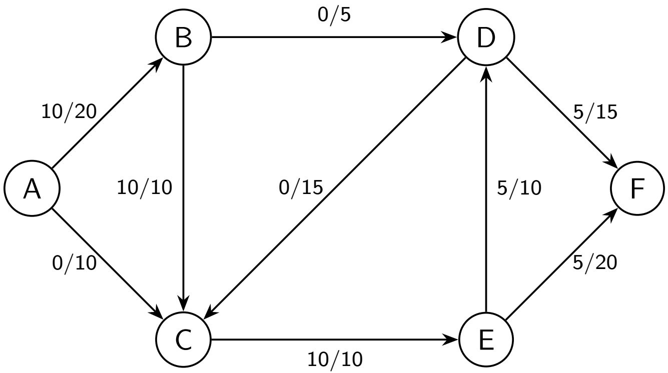
- Given a feasible flow f, we define the residual capacity c_f as $c_f(u, v) = c(u, v) - f(u, v)$.
- Here we assume there are no simultaneous edges (u, v) and (v, u).
  - If there were, we could combine them into a single flow edge.

Residual graph

- Define a residual graph $G_f = (V, E_f)$, where E_f is the set of edges that are not saturated, i.e. the set of edges whose residual capacity is positive.
- Thus, in the residual graph G_f, a back edge (v, u) stores the flow along (u, v), allowing us to store the unused capacity of (u, v) as a forward edge.

Residual Graph example

- Consider the following flow network:



- What would its residual graph look like?

Augmenting paths in G_f

- Suppose there exists a path from s to t in the residual graph G_f, call it $p = \{v_0, v_1, \dots, v_k\}$, where $s = v_0$ and $t = v_k$.
- Call p an augmenting path.
- Define F as the maximum amount of residual flow on path p, i.e. $F = min_i\{c_f(v_i, v_{i+1})\}$.
- So if we can find an augmenting path p, we can improve our current flow f by adding F additional units of flow through G along p.
- Updated flow f':

$$f'(u,v) = \begin{cases} f(u,v) + F & if (u,v) \in p \\ f(u,v) - F & if (v,u) \in p \\ f(u,v) & otherwise \end{cases}$$

Augmenting paths, cont.

- Case 1: There is unused capacity on an existing edge (u, v), so we increase the flow by F units.
- Case 2: Move some flow off edge (u, v), because we moved it somewhere else.
- Case 3: Leave flow unchanged, since (u, v) is not on the path p.

Ford-Fulkerson

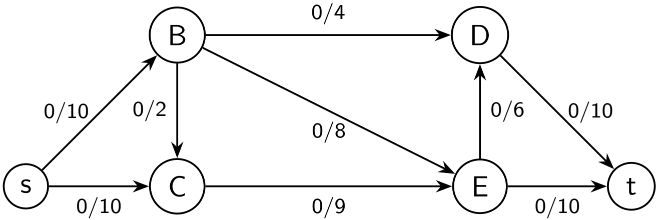
- Start with no flow, i.e. $f = 0$.
- Try to find some augmenting path in G_f. If it can be found, add its flow to f.
- Repeat until no augmenting path can be found.

Pseudocode

```
def FordFulkerson(G, s, t):
    for each edge (u, v) ∈ G.E:
        (u, v).f = 0
    while ∃ augm. path p from s → t in G_f:
        c_f(p) = min{c_f(u, v) : (u, v) ∈ p}
        for each edge (u, v) ∈ p:
            if (u, v) ∈ E:
                (u, v).f = (u, v).f + c_f(p)
            else:
                (u, v).f = (v, u).f − c_f(p)
```

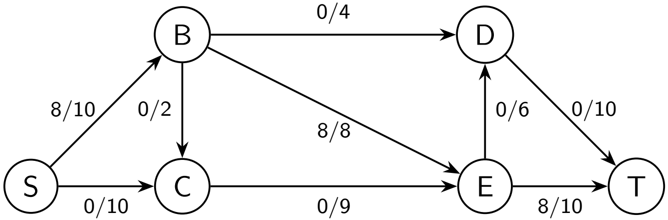
Example

- Consider the following network (initialized to have zero flow):



Example, cont.

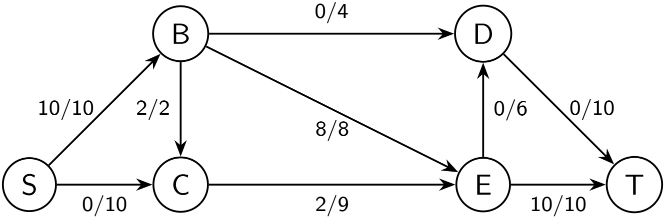
- The first augmenting path we might pick is $S \rightarrow B \rightarrow E \rightarrow T$:



- That gives us a current max flow of 8.

Example, cont.

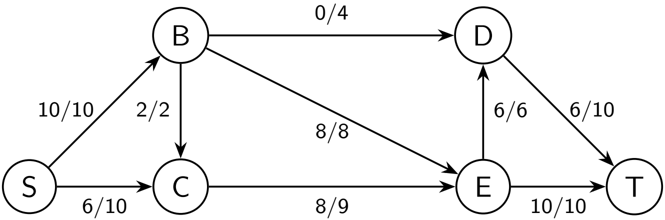
- Next, we push two more units of flow through $S \to B$, and distribute it using the path $S \to B \to C \to E \to T$:



- That gives us a current max flow of $8 + 2 = 10$.

Example, cont.

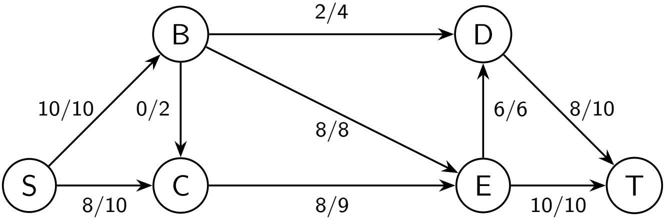
- Next, we push six units of flow through $S \to C$, utilizing the path $S \to C \to E \to D \to T$:



- That gives us a current max flow of $10 + 6 = 16$.

Example, cont.

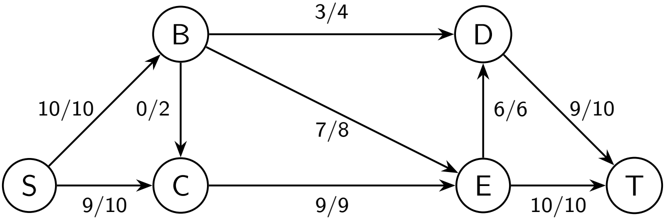
- Now, we push two more units of flow through $S \to C$.
- To do this, we have to move two units of flow away from $B \to C$, and move it to $B \to D$.
- Then we can utilize the path $S \to C \to B \to D \to T$:



- That gives us a current max flow of $16 + 2 = 18$.

Example, cont.

- Finally, we push one more unit of flow through $S \to C$.
- To accomodate this, we have to move one unit of flow away from $B \to E$, and move it to $B \to D$.
- Then we can utilize the path $S \to C \to E \to B \to D \to T$:



- That gives us a final max flow of $18 + 1 = 19$.

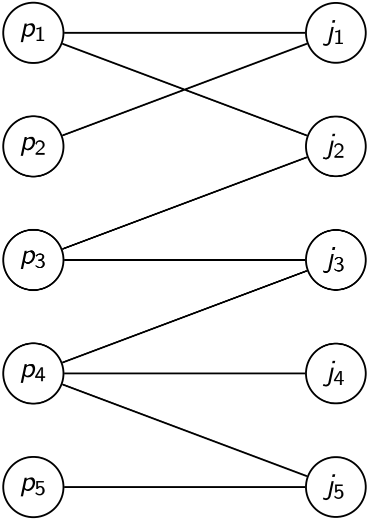
Running time

- If all edge capacities are integers, we increase the flow f by at least 1 at each iteration.
- In this case, the algorithm terminates after at most $|f^*|$ iterations, where $|f^*|$ is the max flow.
- Each iteration of the `while` loop takes $\mathcal{O}(E)$, leading to a total of $\mathcal{O}(E|f^*|)$.
- What if edge weights were rational numbers?
  - Can convert any set of rational numbers into integers by multiplying by a sufficiently large integer (which one?)
  - Thus, Ford-Fulkerson also works for rational edge weights.
- What if the edge weights were irrational?
  - Ford-Fulkerson would not terminate, since we could always find an augmenting path, due to infinite precision.
  - This could actually also happen simply due to a round-off error.

Bipartite Matching

- Suppose we have a set of P people and J jobs.
- Each person can only do one job.
- A bipartite matching is an assignment of people to jobs.
- Goal: complete as many jobs as possible, i.e. find a maximum matching.
- When every person/job is matched, we call the matching a perfect matching.
- Can model this as a bipartite graph:

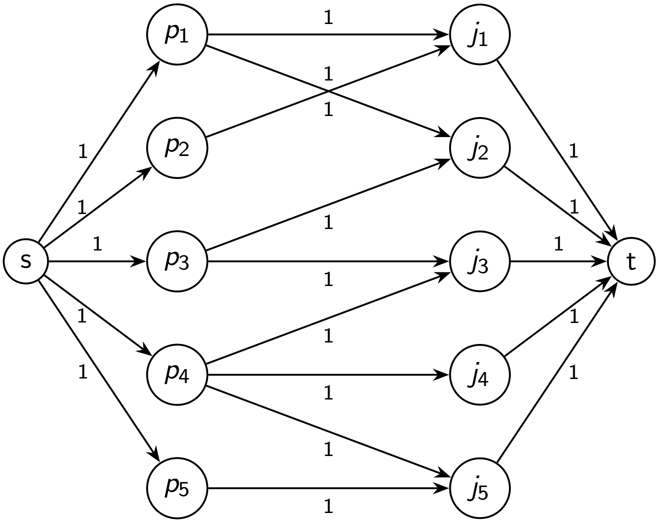
Example



How to solve?

- Turn the bipartite graph into a directed graph.
- Add a source and destination vertex, which connect to all P vertices and J vertices, respectively.
- Assign a capacity of 1 to each edge in this new network.
- Solve the Max Flow problem on this network.

Example



Details

- Since all capacities are 1, we will either use an edge or not.
- After running, say Ford-Fulkerson, on this network, all edges (excluding those from / to source / destination) which have flow going through them will be used in the matching.
- If there exists a matching of k edges, then there exists a flow f with $|f| = k$.
- If there exists a flow f with $|f| = k$, then there exists a matching of k edges.