Ford-Fulkerson Algorithm
Overview

- Network flows on directed acyclic graphs
- Ford-Fulkerson Algorithm
  - Residual networks
Network flows on directed acyclic graphs

- **DAG** is a directed graph with no cycles.

- **Flow network:**
  - Flow network $G = (V,E)$ is a directed graph in which each edge $(u,v) \in E$ has a nonnegative capacity $c(u,v) \geq 0$.
  - If $(u,v) \not\in E$, we assume that $c(u,v) = 0$.

- **Distinguished vertices**: source $s$, and sink $t$.
  - Source: produces the material at a steady rate.
  - Sink: consumes the material at a steady rate.

- **Objective**: compute how much material can flow through the network.
Flow: A flow in $G$ is a real-valued function $f : V \times V \rightarrow \mathbb{R}$ that satisfies three properties:

1. Capacity constraint: For all $u,v \in V$, we require $f(u,v) \leq c(u,v)$.
   - The net flow from one vertex to another must not exceed the given capacity.

2. Skew symmetry: For all $u,v \in V$, we require $f(u,v) = -f(v,u)$.
   - The net flow from a vertex $u$ to a vertex $v$ is the negative of the net flow in the other direction.

3. Flow conservation: For all $u \in V - \{s,t\}$, we require
   $$\sum_{v \in V} f(u,v) = 0.$$ 
   The quantity $f(u,v)$, which can be positive or negative, is called the net flow from vertex $u$ to $v$.

   The value of the flow is defined as $|f| = \sum_{v \in V} f(s,v)$ that is the total net flow out of the source.
Residual Capacity

- Given a flow $f$ in network $G = (V, E)$.
- Consider a pair of vertices $u, v \in V$.
- Residual capacity:
  - The amount of additional flow we can push directly from $u$ to $v$.
  - $c_f(u, v) = c(u, v) - f(u, v)$
  - $\geq 0$ (since $f(u, v) \leq c(u, v)$)
- Example:
  - $c(u, v) = 16, f(u, v) = 5 \Rightarrow c_f(u, v) = 11$
Residual Network

- **Residual network:** $G_f = (V, E_f)$,
  
  $E_f = \{(u, v) \in V \times V : c_f(u, v) > 0\}$.

- Each edge of the residual network can admit a positive flow.

- Given flows $f_1$ and $f_2$, the flow sum $f_1 + f_2$ is

  - $(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v)$

  for all $u, v \in V$. 
Flow Networks: Simple Example

original

residual

CS 4407, Algorithms, University College Cork, Gregory M. Provan
Step 1: path \{s,a,b,t\}

Step 2: path \{s,a,t\}

Step 3: Path \{s,b,t\}
Example: p.3

Step 3

Step 4
Example: Max-Flow Min-Cut
Ford-Fulkerson Algorithm

- The Ford_Fulkerson method is iterative,

- Starts with \( f(u,v) \) for \((u,v) \in V\), initial flow of value 0.

- The method is based on the augmenting path
  - a path from \( s \) to \( t \) along which we can push more flow and then augment flow along this path.
Ford-Fulkerson Algorithm

Ford-Fulkerson(G, s, t) ; G = (V, E)
1 for each edge (u,v) in E
2 $f(u,v) = f(v,u) = 0$
3 while $\exists$ path $p$ from $s$ to $t$ in residual network $G_f$
4 do $c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is in } p\}$
5 for each edge $(u,v)$ on $p$
6 $f(u,v) = f(u,v) + c_f(p)$
7 $f(v,u) = -f(u,v)$
Complexity: Ford-Fulkerson Algorithm

- Running time for this algorithms is $O(E \times |f^*|)$ where $f^*$ is the maximum flow found by algorithm.

- First three Lines take time $\theta(E)$.

- The while loop of last five lines is executed at most $|f^*|$ times, since the flow value increases by at least one unit in each iteration.
Example:
CS 4407, Algorithms, University College Cork, Gregory M. Provan
Reference

- Class Notes from Dr. Istvan Jonyer of Oklahoma State University
- Class Notes from UTexas CSE 5311 of 2004, made by Hiren Patel, Ujjval Patel
Backup: simple network
Consider the network $G=(V,E)$ shown in the figure below. Each edge $(u,v) \in E$ in the network is labeled with its capacity $c(u,v)$. Flow: 1
Residual graph

CS 4407, Algorithms, University College Cork, Gregory M. Provan
Max Flow : 5