COMP251: Network flows (1)

Jérôme Waldispühl
School of Computer Science
McGill University

Based on slides from M. Langer (McGill) & (Cormen et al., 2009)
Flow Network

$G = (V, E)$ directed.

Each edge $(u, v)$ has a **capacity** $c(u, v) \geq 0$.

If $(u,v) \notin E$, then $c(u,v) = 0$.

**Source** vertex $s$, **sink** vertex $t$, assume $s \leadsto v \leadsto t$ for all $v \in V$. 

![Flow Network Diagram](image)
Definitions

**Positive flow:** A function $p : V \times V \rightarrow \mathbb{R}$ satisfying.

**Capacity constraint:** For all $u, v \in V$, $0 \leq p(u, v) \leq c(u, v)$

**Flow conservation:** For all $u \in V - \{s, t\}$,

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

Flow into $u$ \quad Flow out of $u$

Flow in: $0 + 2 + 1 = 3$
Flow out: $2 + 1 = 3$
Example
Cancellation with positive flows

- Without loss of generality, can say positive flow goes either from \( u \) to \( v \) or from \( v \) to \( u \), but not both.
- In the above example, we can “cancel” 1 unit of flow in each direction between \( x \) and \( z \).
- Capacity constraint is still satisfied.
- Flow conservation is still satisfied.
Net flow

A function $f : V \times V \rightarrow \mathbb{R}$ satisfying:

- **Capacity constraint:** For all $u, v \in V$, $f(u, v) \leq c(u, v)$
- **Skew symmetry:** For all $u, v \in V$, $f(u, v) = -f(v, u)$
- **Flow conservation:** For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(u, v) = 0$

\[
\sum_{v \in V; f(v,u) > 0} f(v,u) = \sum_{v \in V; f(u,v) > 0} f(u,v)
\]

- Total positive flow entering $u$
- Total positive flow leaving $u$
Positive vs. Net flows

Define net flow in terms of positive flow:

\[ f(u, v) = p(u, v) - p(v, u). \]

The differences between positive flow \( p \) and net flow \( f \):

- \( p(u, v) \geq 0 \),
- \( f \) satisfies skew symmetry.
Values of flows

Definition: $f = |f| = \sum_{v \in V} f(s,v) = \text{total flow out of source}$.

Value of flow $f = |f| = 3$. 
Flow properties

- Flow in == Flow out
- Source $s$ has outgoing flow
- Sink $t$ has ingoing flow
- Flow out of source $s$ == Flow in the sink $t$
Maximum-flow problem

Given $G$, $s$, $t$, and $c$, find a flow whose value is maximum.
Applications

(http://ais.web.cern.ch/ais/)

(http://driverlayer.com)
Naïve algorithm

Initialize \( f = 0 \)
While true {
    if (\( \exists \) path \( P \) from \( s \) to \( t \) such that all edges have a flow less than capacity)
        then
            increase flow on \( P \) up to max capacity
    else
        break
}
Naïve algorithm

Initialize $f = 0$

While true {
    if (∃ a path $P$ from $s$ to $t$ s.t. all edges $e \in P \: f(e) < c(e)$) then {
        $\beta = \min\{ c(e) - f(e) \mid e \in P\}$
        for all $e \in P$ { $f(e) += \beta$ }
    } else { break }
}
Example where algorithm works
Example where algorithm works

| |f| = 2
Example where algorithm works

$|f| = 4$
Example where algorithm works

\[ |f| = 5 \]
Example where algorithm fail!
Example where algorithm fail!

$|f| = 3$ And terminates...
Challenges

How to choose paths such that:

• We do not get stuck
• We guarantee to find the maximum flow
• The algorithm is efficient!
A better algorithm

Motivation: If we could subtract flow, then we could find it.

Algo 1 terminates here...

Negative value on edge that does not satisfy the definition
Residual graphs

Given a flow network $G=(V,E)$ with edge capacities $c$ and a given flow $f$, define the residual graph $G_f$ as:

• $G_f$ has the same vertices as $G$

• The edges $E_f$ have capacities $c_f$ (called residual capacities) that allow us to change the flow $f$, either by:

  1. Adding flow to an edge $e \in E$
  2. Subtracting flow from an edge $e \in E$
Residual graphs

for each edge \( e = (u, v) \in E \)

if \( f(e) < c(e) \)
then {
    put a **forward edge** \((u,v)\) in \( E_f \)
    with residual capacity \( c_f(e) = c(e) - f(e) \)
}

if \( f(e) > 0 \)
then {
    put a **backward edge** \((v,u)\) in \( E_f \)
    with residual capacity \( c_f(e) = f(e) \)
}

Example 1/3

Flow network

Flow

Residual graph

forward
backward
Example 2/3

Flow network

Flow

Residual graph

forward

backward

3-2=1
Example 3/3
An augmenting path is a path from the source $s$ to the sink $t$ in the residual graph $G_f$ that allows us to increase the flow.

Q: By how much can we increase the flow using this path?
Example

Flow in $G$

Residual graph $G_f$
Example

Residual graph $G_f$

Flow in $G_f$
Example

$|f| = 3$

$G$

$|f| = 5$

$\beta = 2$
Methodology

• Compute the residual graph $G_f$

• Find a path $P$

• Augment the flow $f$ along the path $P$
  
  1. Let $\beta$ be the bottleneck (smallest residual capacity $c_f(e)$ of edges on $P$)
  
  2. Add $\beta$ to the flow $f(e)$ on each edge of $P$.

Q: How do we add $\beta$ into $G$?
Augmenting a path

```java
f.augment(P) {
    \( \beta = \min \{ c(e) - f(e) \mid e \in P \} \)
    for each edge \( e = (u,v) \in P \) {
        if e is a forward edge {
            f(e) += \beta
        } else { // e is a backward edge
            f(e) -= \beta
        }
    }
}
```
Ford-Fulkerson algorithm

\[ f \leftarrow 0 \]

\[ G_f \leftarrow G \]

\textbf{while} (there is a s-t path in } G_f \textbf{) \{ }

\hspace{1cm} f.\text{augment}(P) \\
\hspace{1cm} \text{update } G_f \text{ based on new } f \\
\}

Correctness (termination)

Claim: The Ford-Fulkerson algorithm terminates.

Proof:
- The capacities and flows are strictly positive integers.
- The sum of capacities leaving s is finite.
- Bottleneck values $\beta$ are strictly positive integers.
- The flow increase by $\beta$ after each iteration of the loop.
- The flow is an increasing sequence of integers that is bounded.
Complexity (Running time)

- Let \( C = \sum_{e \in E} c(e) \)

- Finding an augmenting path from \( s \) to \( t \) takes \( O(|E|) \) (e.g. BFS or DFS).

- The flow increases by at least 1 at each iteration of the main while loop.

- The algorithm runs in \( O(C \cdot |E|) \)