Network Flows

$G = (V, E)$ directed.
Each edge $(u, v)$ has a capacity $c(u, v) \geq 0$.
If $(u, v) \not\in E$, then $c(u, v) = 0$.

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

Source has not incoming edges, and sink no outgoing edges!

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

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 $u, v \in V$, $f(u, v) = -f(v, u)$, $v \in V$
- 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: \( f = |f| = \sum_{(x,y)} f(x,y) \) = total flow out of source.

Maximum-flow problem

Given \( G, s, t, \) and \( c \), find a flow whose value is maximum.

Algorithm 1

Initialize \( f = 0 \)
While true {
  if (there is a path \( P \) from \( s \) to \( t \) such that all edges on that path have a flow strictly less that their capacity)
    then
      increase the flow on that path as much as possible.
    else
      break
  }

Applications
Example where algorithm works

Example where algorithm works

Example where algorithm works

Example where algorithm works

Example where algorithm fail!

Example where algorithm fail!

|f| = 2

|f| = 4

|f| = 5

|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!

Algorithm 2

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

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

Example 2/3
Augmenting path

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?
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 should we add \( \beta \)?

Augmenting a path

\[
\text{augment}(P) \{
\begin{align*}
\beta &= \min \{ c(e) - f(e) \mid e \in P \} \\
\text{for each edge } e &= (u,v) \in P \{
\begin{align*}
\text{if } e \text{ is a forward edge } &:\ f(e) += \beta \\
\text{else } &:// e \text{ is a backward edge } \ f(e) -= \beta
\end{align*}
\}
\}
\}
\]

Ford-Fulkerson algorithm

\[
f = 0 \\
G_f = G \\
\text{while (there is a s-t path in } G_f) \{
\begin{align*}
f \text{.augment}(P) \\
\text{update } G_i \text{ based on new } f
\end{align*}
\}
\]

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 \( C = \sum_{e \in f} c(e) \)
- Finding a 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 \( O(C \cdot |E|) \)