Algorithm Design and Analysis

CSE 565

Lecture 28
Network Flow
• Choosing good augmenting paths
• Capacity scaling algorithm

A. Smith
Pre-break: Ford-Fulkerson

• Find \textit{max s-t flow} \& \textit{min s-t cut} in \(O(mnC)\) time
  – All capacities are integers \(\leq C\)
  – (Today: removing this assumption)

• \textbf{Duality}: Max flow value = min cut capacity

• \textbf{Integrality}: if capacities are integers, then FF algorithm produces an \textit{integral} max flow
Residual Graph

**Original edge:** $e = (u, v) \in E$.
- Flow $f(e)$, capacity $c(e)$.

**Residual edge.**
- "Undo" flow sent.
- $e = (u, v)$ and $e^R = (v, u)$.
- Residual capacity:
  
  $$c_f(e) = \begin{cases} 
  c(e) - f(e) & \text{if } e \in E \\
  f(e) & \text{if } e^R \in E 
  \end{cases}$$

**Residual graph:** $G_f = (V, E_f)$.
- Residual edges with positive residual capacity.
- $E_f = \{e : f(e) < c(e)\} \cup \{e^R : c(e) > 0\}$. 
Ford-Fulkerson:
• While you can,
  • Greedily push flow
  • Update residual graph

**Theorem:** FF algorithm terminates with a maximum flow.

Two big ideas:
1) \( \text{Max flow} \leq \text{min cut} \)
2) \( \text{Value(FF output flow)} = \text{capacity(some cut)} \)

Hence... FF flow is maximal and the corresponding cut is minimal.

Proof of (b):
• \( f = \text{flow output by FF algorithm} \)
• \( A = \text{set of vertices reachable from } s \text{ in residual graph } G_f \)
• **Lemma:** \( \text{value}(f) = \text{capacity}(A) \) (see Lecture 22)
Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

A. No. If max capacity is $C$, then algorithm can take $C$ iterations.
Choosing Good Augmenting Paths

Use care when selecting augmenting paths.
- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:
- Can find augmenting paths efficiently.
- Few iterations.

Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.
Capacity Scaling

Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.

- Don't worry about finding exact highest bottleneck path.
- Maintain scaling parameter $\Delta$.
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least $\Delta$. 

![Graphs](image-url)
Capacity Scaling

$$\text{Scaling-Max-Flow}(G, s, t, c) \{$$

$$\text{foreach } e \in E \text{ } f(e) \leftarrow 0$$

$$\Delta \leftarrow \text{smallest power of 2 greater than or equal to } C$$

$$G_f \leftarrow \text{residual graph}$$

$$\text{while } (\Delta \geq 1) \{$$

$$G_f(\Delta) \leftarrow \Delta-\text{residual graph}$$

$$\text{while } (\text{there exists augmenting path } P \text{ in } G_f(\Delta)) \{$$

$$f \leftarrow \text{augment}(f, c, P) \text{ // augment flow by } \geq \Delta$$

$$\text{update } G_f(\Delta)$$

$$\}$$

$$\Delta \leftarrow \Delta / 2$$

$$\}$$

$$\text{return } f$$

$$\}$$
Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and $C$.

Integrality invariant. All flow and residual capacity values are integral.

Correctness. If the algorithm terminates, then $f$ is a max flow.

Pf.

- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. •
Capacity Scaling: Running Time

Lemma 1. The outer while loop repeats $1 + \lfloor \log_2 C \rfloor$ times.
Pf. Initially $C \leq \Delta < 2C$. $\Delta$ decreases by a factor of 2 each iteration. □

Lemma 2. Let $f$ be the flow at the end of a $\Delta$-scaling phase. Then the value of the maximum flow $f^*$ is at most $v(f) + m \Delta$ (Sanity check: $|v(f^*) - v(f)| \leq m\Delta$, and $\Delta$ shrinks, so $v(f)$ converges towards $v(f^*)$ )

Lemma 3. There are at most $2m$ augmentations per scaling phase.
- Let $f$ be the flow at the end of the previous scaling phase.
- Lemma 2 $\Rightarrow v(f^*) \leq v(f) + m (2\Delta)$.
- Each augmentation in a $\Delta$-phase increases $v(f)$ by at least $\Delta$. □

Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. It can be implemented to run in $O(m^2 \log C)$ time. □

(Why?)
Lemma 2. Let $f$ be the flow at the end of a $\Delta$-scaling phase. Then value of the maximum flow is at most $v(f) + m \Delta$.

**Pf.** (almost identical to proof of max-flow min-cut theorem)

- We show that at the end of a $\Delta$-phase, there exists a cut $(A, B)$ such that $\text{cap}(A, B) \leq v(f) + m \Delta$.
- Choose $A$ to be the set of nodes reachable from $s$ in $G_f(\Delta)$.
- By definition of $A$, $s \in A$.
- By definition of $f$, $t \notin A$.

\[
\begin{align*}
  v(f) & = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
  & \geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ in to } A} \Delta \\
  & = \sum_{e \text{ out of } A} c(e) - \sum_{e \text{ out of } A} \Delta - \sum_{e \text{ in to } A} \Delta \\
  & \geq \text{cap}(A, B) - m\Delta
\end{align*}
\]

So $v(f^*) - v(f) \leq \text{cap}(A, B) - v(f) \leq m\Delta$. 

original network
Best Known Algorithms For Max Flow

Reminder: The scaling max-flow algorithm runs in $O(m^2 \log C)$ time.

Currently there are algorithms that run in time

- $O(mn \log n)$
- $O(n^3)$
- $O(\min(n^{2/3}, m^{1/2}) m \log n \log C)$

Active topic of research:

- Flow algorithms for specific types of graphs
- Special cases (bipartite matching, etc)
- Multi-commodity flow
- ...