Lectures 18
Network Flow
• Algorithms:
  • Ford-Fulkerson
  • Capacity Scaling
• Applications

Sofya Raskhodnikova

CSE 565

Algorithm Design and Analysis
Network Flow
**Minimum Cut Problem**

**Def.** An s-t cut is a partition \((A, B)\) of \(V\) with \(s \in A\) and \(t \in B\).

**Def.** The capacity of a cut \((A, B)\) is: 

\[
\text{cap}(A, B) = \sum_{e \text{ out of } A} c(e)
\]

**Goal.** Find an s-t cut of minimum capacity.

![Graph Diagram](attachment:graph.png)

Capacity = 9 + 15 + 8 + 30 = 62
**Maximum Flow Problem**

**Def.** An s-t flow is a function that satisfies:
- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V - \{s, t\}$: $\sum_{e \text{ in to } v} f(e) = \sum_{e \text{ out of } v} f(e)$ (conservation)

**Def.** The value of a flow $f$ is: $v(f) = \sum_{e \text{ out of } s} f(e)$.

**Goal.** Find s-t flow of maximum value.

![Graph of a network flow problem](image)

Value = 4
What we proved about flows and cuts

**Flow value lemma.** Let $f$ be any flow, and let $(A, B)$ be any $s$-$t$ cut. Then the net flow sent across the cut is equal to the amount leaving $s$.

$$\sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) = v(f)$$

**Augmenting path theorem.** Flow $f$ is a max flow iff there are no augmenting paths.

**Max-flow min-cut theorem.** [Ford-Fulkerson 1956] The value of the max flow is equal to the value of the min cut.
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: Analysis

Ford-Fulkerson summary:

- While you can,
  - Greedily push flow
  - Update residual graph

Feasibility lemma: Ford-Fulkerson outputs a valid flow.

Optimality: If Ford-Fulkerson terminates then
- the output is a max flow;
- set of vertices reachable from s in residual graph forms a minimum cut.

Still to do:
- Running time (in particular, termination!)
Running Time

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

**Invariant.** Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains an integer throughout the algorithm.

**Theorem.** The algorithm terminates in at most $v(f^*) \leq nC$ iterations.

**Proof.** Each augmentation increases flow value by at least 1. □

Running time of Ford-Fulkerson on a graph with integer capacities?
Augmenting Path Algorithm

Ford-Fulkerson(G, s, t, c) {
    foreach e ∈ E, f(e) ← 0
    $G_f$ ← residual graph

    while (there exists augmenting path P) {
        f ← Augment(f, c, P)
        update $G_f$
    }
    return f
}

Augment(f, c, P) {
    b ← bottleneck-capacity(P)
    foreach e ∈ P {
        if (e ∈ E) f(e) ← f(e) + b
        else f($e^R$) ← f($e^R$) - b
    }
    return f
}
Running Time

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

**Invariant.** Every flow value $f(e)$ and every residual capacity $c_f(e)$ remains an integer throughout the algorithm.

**Theorem.** The algorithm terminates in at most $v(f^*) \leq nC$ iterations.

**Proof.** Each augmentation increases flow value by at least 1. □

**Running time** of Ford-Fulkerson on a graph with integer capacities: $O(mnC)$.

**Space:** $O(m+n)$.

**Important special case.** If $C = 1$, Ford-Fulkerson runs in $O(mn)$ time.
Review Question

• Is this flow a maximum flow?

• **Def:** Integral flow: flows on all edges are integers

• Does this graph have an integral maximum flow?

• Does every graph with integer capacities have an integral maximum flow?
Ford-Fulkerson Summary

• **Assumption:** All capacities are integers between 1 and C.

• **Running time:** The FF algorithm terminates in at most $v(f^*) \leq nC$ iterations.

  Running time = $O(mnC)$. Space: $O(m + n)$.

• **Correctness:**
  – FF outputs a flow with maximum value
  – Set of vertices reachable from s in residual graph forms a minimum cut
  – **Integrality theorem:** FF outputs an integral flow, so every graph with integer capacities has an integral maximum flow.

• **Important special case:** if $C = 1$, Ford-Fulkerson runs in $O(mn)$ time.
Review Question

• Does Ford-Fulkerson always terminate if capacities are rational?
• Does Ford-Fulkerson always terminate if capacities are irrational?

\[ r = \frac{\sqrt{5} - 1}{2} \implies r^2 = 1 - r \]

• **Exercise:** Find a sequence of augmenting paths so that FF does not terminate and does not converge to max flow.
Faster algorithms when capacities are large
Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

A. No. If max capacity is $C$, then algorithm can take $C$ iterations.

Intuition: We’re choosing the wrong paths!
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 $\Delta$. 
Capacity Scaling

Scaling-Max-Flow(G, s, t, c) {
    foreach e ∈ E  f(e) ← 0
    Δ ← smallest power of 2 greater than or equal to C
    G_f ← residual graph

    while (Δ ≥ 1) {
        G_f(Δ) ← Δ-residual graph
        while (there exists augmenting path P in G_f(Δ)) {
            f ← augment(f, c, P) // augment flow by ≥ Δ
        }
        update G_f(Δ)
    }
    Δ ← Δ / 2
}
return f
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.
Proof.
- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths.
Lemma 1. The outer while loop repeats \(1 + \lceil \log_2 C \rceil\) times.
Proof. 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 is at most \(v(f) + m \Delta\). ← proof on next slide

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

Proof. (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 \( A \), source \( s \in A \).
- By definition of \( f \), sink \( t \not\in A \).

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

So, \( v(f^*) \leq \text{cap}(A,B) \leq v(f) + m\Delta \).
General Principle

- Let
  - $G = (V, E)$ be a directed graph with capacities $\{c_e\}_{e \in E}$
  - $f$ be any valid flow in $G$
  - $G_f$ be the residual graph for $f$ in $G$
  - $f^*$ be any maximum flow in $G$

- Then we have
  \[ \nu(f^*) = \nu(f) + (\text{value of max } s-t \text{ flow in } G_f) \]

- In particular, for any cut $A, B$:
  \[ \nu(f^*) \leq \nu(f) + (\text{capacity of } A, B \text{ in } G_f) \]

- Applications:
  - Correctness of Ford-Fulkerson
  - Running time analysis for capacity scaling

10/11/10
S. Raskhodnikova; based on slides by E. Demaine, C. Leiserson, A. Smith, K. Wayne
Best Known Algorithms For Max Flow

- Reminder: The scaling max-flow algorithm runs in $O(m^2 \log C)$ time.
- There are algorithms that run in time
  - $O(mn)$ (Orlin, 2013)
  - $O(m^{10/7} \log^a m)$ for constant $a$ and $C = 1$ (Madry, 2013)
  - $O \left( \min \left( n^{2/3}, m^{1/2} \right) \cdot m \cdot \log n \cdot \log C \right)$
- Active topic of research:
  - Flow algorithms for specific types of graphs
  - Special cases (bipartite matching, etc)
  - Multi-commodity flow
  - ...
Applications when $C=1$

- Maximum bipartite matching
  - Reducing MBM to max-flow
  - Hall’s theorem

- Edge-disjoint paths
  - another reduction
Matching

- Input: undirected graph $G = (V, E)$.
- $M \subseteq E$ is a matching if each node appears in at most 1 edge in $M$.
- **Maximum matching**: find a matching with as many edges as possible.
Bipartite Matching

Bipartite matching.

- **Input**: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most one edge in $M$.
- **Maximum matching**: find a matching with as many edges as possible.

![Bipartite Matching Diagram]

We cannot add edges to this matching.
- It is **maximal** (local max)
- But **not maximum** (global max)
Bipartite Matching

Bipartite matching.

- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in $M$.
- Maximum matching: find a matching with as many edges as possible.

There is no matching in this graph with more than 4 edges
- This matching is both maximal (local max) and maximum (global max)

Do not confuse with stable matching (different inputs and goals)
Reductions

• “Problem A reduces to problem B”
  – Rough meaning: there is a simple algorithm for A that uses an algorithm for B as a subroutine.
  – Denote $A \leq B$

• Usually:
  • Given instance $x$ of problem A we find a instance $x'$ of problem B
  • Solve $x'$
  • Use the solution to build a solution to $x$

• Useful skill: quickly identify problems where existing solutions may be applied.
  • Good programmers do this all the time
Reduction to Max flow.

- Create digraph $G' = (L \cup R \cup \{s, t\}, E')$.
- Direct all edges from $L$ to $R$, and assign capacity 1.
- Add source $s$, and capacity 1 edges from $s$ to each node in $L$.
- Add sink $t$, and capacity 1 edges from each node in $R$ to $t$. 

Reducing Bipartite Matching to Maximum Flow
Bipartite Matching: Proof of Correctness

Theorem. Max cardinality matching in $G = \text{value of max flow in } G'$. 

Proof: We need two statements
  
  - max. matching in $G \leq$ max flow in $G'$
  - max. matching in $G \geq$ max flow in $G'$
Theorem. Max cardinality matching in $G = \text{value of max flow in } G'$.

Pf. $\leq$

- Given max matching $M$ of cardinality $k$.
- Consider flow $f$ that sends 1 unit along each of $k$ paths.
- $f$ is a flow, and has value $k$. □
Bipartite Matching: Proof of Correctness

Theorem. Max cardinality matching in $G$ = value of max flow in $G'$.

Pf. $\geq$

- Let $f$ be a max flow in $G'$ of value $k$.
- **Integrality theorem** $\Rightarrow$ we can find a max flow $f$ that is integral;
  - all capacities are 1 $\Rightarrow$ $f$ takes values only in $\{0,1\}$
- Consider $M$ = set of edges from $L$ to $R$ with $f(e) = 1$.
  - Each node in $L$ and $R$ participates in at most one edge in $M$
    - Because all capacities are 1 and flow must be conserved
  - $|M| = k$: consider cut $(\{s\}, S \cup R \cup t)$