CS 580: Algorithm Design and Analysis

Jeremiah Blocki
Purdue University
Spring 2018

Travel:
I will be attending a conference next week.

Tuesday:
Recorded Lecture + return midterms (hopefully)

Midterm Regrade:
Must be completed within 2 weeks (syllabus). Please e-mail us before then.

Midterm Solutions:
Will post on blackboard before Tuesday.

Max Flow Recap

- **Max-Flow Problem, Min-Cut Problem**
  - Definition of a s-t flow f(s, t)
  - Value of a flow f
  - Capacity of a s-t cut \((A, \overline{A})\)

- **Weak Duality Lemma**: For any flow f and s-t cut \((A, \overline{A})\) in the graph, we have \( f(A, \overline{A}) \leq \text{cap}(A, \overline{A}) \).

- **Finding a Max-Flow**
  - **Greedy Algorithm**
  - **Residual Graph**
  - **Ford-Fulkerson Algorithm**

- **Max-Flow Min-Cut Equivalence**

Max-Flow Recap

- **Augmenting Path Theorem**
  - Flow f is a max flow if there are no augmenting paths.

- **Max-Flow Min-Cut Theorem**
  - The value of the max flow is equal to the value of the min cut.

- **Proof of Max-Flow Min-Cut Theorem**
  - Let \( f \) be a flow with no augmenting paths.
  - Let \( A \) be a set of vertices reachable from \( s \) in residual graph.
  - By definition of \( A \), \( s \in A \).
  - By definition of \( f, f(A) \).

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 value by at least 1.

**Corollary**: If \( C = 1 \), Ford-Fulkerson runs in \( O(mn) \) time.

**Integral Flow Theorem**: If all capacities are integers, then there exists a max flow \( f \) for which every flow value \( f(e) \) is an integer.

**Proof**: Since algorithm terminates, theorem follows from invariant.
7.3 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:
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

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.

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 \( \Delta \).
- Let \( \Delta \) be the subgraph of the residual graph consisting of only arcs with capacity at least \( \Delta \).

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.
- By integrality invariant, when \( \Delta = 1 \), \( G_f(\Delta) = G_f(1) \).
- Upon termination of \( \Delta = 1 \) phase, there are no augmenting 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:
- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

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.

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: 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.
- By integrality invariant, when \( \Delta = 1 \), \( G_f(\Delta) = G_f(1) \).
- 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 $v(f) + m\Delta$.

Lemma 3. There are at most $2m$ augmentations per scaling phase.

Let $f$ be the flow at the end of the previous scaling phase.

$L2 \implies v(f^*) \leq v(f) + m(2\Delta)$.

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.

Proof on next slide.
Dinic’s Max Flow Min-Cut Algorithm

Remark: Number of levels increased. This is not a coincidence!

Breadth First Search: Yields minimum s-t cut! We are done!
Dinic's Algorithm: Correctness and Running Time

Correctness Follows Directly from Augmenting Path Theorem:

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

Running Time Analysis: Let $G$ denote residual graph after iteration $k$ ($G_k = G$)

Definition: $\text{depth}(G_k) = \text{length of the shortest directed path from s to t}$.

Key Claim: $\text{depth}(G_k) > \text{depth}(G_{k-1})$ (depth always increases)

Implications: We can use a "time to compute blocking flow in level graph $G_k$".

7.7 Extensions to Max Flow
Circulation with Demands

- Directed graph $G = (V, E)$.
- Edge capacities $c(e), e \in E$.
- Node supply and demands $d(v), v \in V$.

Def. A circulation is a function that satisfies:
- For each $e \in E$: $0 \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \in \delta^+(v)} f(e) = \sum_{e \in \delta^-(v)} f(e) = d(v)$ (conservation)

Circulation problem: given $(V, E, c, d)$, does there exist a circulation?

Max flow formulation.

- Add new source $s$ and sink $t$.
- For each $v$ with $d(v) < 0$, add edge $(s, v)$ with capacity $-d(v)$.
- For each $v$ with $d(v) > 0$, add edge $(v, t)$ with capacity $d(v)$.

Claim: $G$ has circulation iff $G'$ has max flow of value $D$.

Integrality theorem. If all capacities and demands are integers, and there exists a circulation, then there exists one that is integer-valued.

Pf. Follows from max flow formulation and integrality theorem for max flow.

Characterization. Given $(V, E, c, d)$, there does not exist a circulation iff there exists a node partition $(A, B)$ such that:

$$\sum_{v \in B} d(v) > \text{cap}(A, B)$$


Integrality with lower bounds.

- Directed graph $G = (V, E)$.
- Edge capacities $c(e)$ and lower bounds $\lambda(e), e \in E$.
- Node supply and demands $d(v), v \in V$.

Def. A circulation is a function that satisfies:
- For each $e \in E$: $\lambda(e) \leq f(e) \leq c(e)$ (capacity)
- For each $v \in V$: $\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = d(v)$ (conservation)

Circulation problem with lower bounds. Given $(V, E, c, d)$, does there exist a circulation?
**Circulation with Demands and Lower Bounds**

**Idea.** Model lower bounds with demands.
- Send \((\lambda(e))\) units of flow along edge \(e\).
- Update demands of both endpoints.

<table>
<thead>
<tr>
<th>lower bound</th>
<th>upper bound</th>
<th>capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d(v))</td>
<td>(d(v) + 2)</td>
<td>((v))</td>
</tr>
<tr>
<td>(d(w))</td>
<td>(d(w) - 2)</td>
<td>((w))</td>
</tr>
</tbody>
</table>

**Theorem.** There exists a circulation in \(G\) iff there exists a circulation in \(G'\). If all demands, capacities, and lower bounds in \(G\) are integers, then there is a circulation in \(G\) that is integer-valued.

**Pf sketch.** \(f(e)\) is a circulation in \(G\) iff \(f'(e) = f(e) - \lambda(e)\) is a circulation in \(G'\).

---

**7.8 Survey Design**

**Survey design.**
- Design survey asking \(n_1\) consumers about \(n_2\) products.
- Can only survey consumer \(i\) about product \(j\) if they own it.
- Ask consumer \(i\) between \(c_i\) and \(c_i'\) questions.
- Ask between \(p_j\) and \(p_j'\) consumers about product \(j\).

**Goal.** Design a survey that meets these specs, if possible.

**Bipartite perfect matching.** Special case when \(c_i = c_i' = p_j = p_j' = 1\).

**Algorithm.** Formulate as a circulation problem with lower bounds.
- Include an edge \((i, j)\) if consumer \(i\) owns product \(j\).
- Integer circulation \(\Rightarrow\) feasible survey design.