Lecture 17: Max-Flow & Min-Cut

Rafael Oliveira

University of Waterloo
Cheriton School of Computer Science

rafael.oliveira.teaching@gmail.com

November 9, 2023

Overview

- Ford-Fulkerson Recap
  - Algorithm
  - Running Time
- Max-Flow Min-Cut Theorem & Correctness of Ford-Fulkerson
- Acknowledgements

Residual Graph

- The residual graph is the object we will study to find augmenting paths

Residual Graph

- The residual graph is the object we will study to find augmenting paths
- Given $G(V, E, c)$ and $s \to t$ flow f on G, define the *residual graph* G_f as follows:
  - $V(G_f) = V(G)$
  - For each $(u, v) =: e \in E$ add edges
    - (u, v) to G_f with capacity $c(e) - f(e)$ (forward edges)
    - (v, u) to G_f with capacity $f(e)$ (backward edges)

Augmenting Path

- An *augmenting path* with respect to a flow f is simply an $s \to t$ path[1] in G_f

[1] By path here we mean a simple path, and not a walk.

Augmenting Path

- An *augmenting path* with respect to a flow f is simply an $s \to t$ path[1] in G_f
- Given augmenting path P in G_f, want to push *as much flow as possible* through it:

$$\text{bottleneck}(P, f) := \text{minimum capacity of edge of } P \text{ in } G_f$$

[1] By path here we mean a simple path, and not a walk.

Improving the Flow

- **Input:** flow f and an augmenting path P in G_f
- **Output:** improved flow f'

Improving the Flow

- **Input:** flow f and an augmenting path P in G_f
- **Output:** improved flow f'

augment(f, P) :

- Let $b :=$ bottleneck(P, f) and $f'(e) = f(e)$ for all $e \in E$
- for each $e := (u, v) \in P$:
  - If e forward edge:
    $f'(e) = f'(e) + b$
  - If e backward edge:
    $f'(v, u) = f'(v, u) - b$ (decrease reversed edge)
- **return** f'

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\mathrm{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\mathrm{value}(f') = \mathrm{value}(f) + \mathit{bottleneck}(P, f)$$

and $f'_{\mathrm{in}}(s) = 0$.

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\mathrm{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\mathrm{value}(f') = \mathrm{value}(f) + \mathit{bottleneck}(P, f)$$

and $f'_{\mathrm{in}}(s) = 0$.

- To check that f' is a flow, need to check capacity constraint and flow conservation constraint.

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\text{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\text{value}(f') = \text{value}(f) + bottleneck(P, f)$$

and $f'_{\text{in}}(s) = 0$.

- Let $b :=$ bottleneck(P, f).
- **Capacity constraint:** given $e \in E(G_f)$, we have
  - e forward edge in G_f, then

    $$f'(e) = f(e) + b \leq f(e) + (c(e) - f(e)) = c(e)$$

  - $e := (u, v)$ backward edge in G_f, then

    $$f'(v, u) = f(v, u) - b \leq f(v, u) \leq c(v, u)$$

    and

    $$f'(v, u) = f(v, u) - b \geq f(v, u) - f(v, u) \geq 0$$

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\text{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\text{value}(f') = \text{value}(f) + \textit{bottleneck}(P, f)$$

and $f'_{\text{in}}(s) = 0$.

- Let $b := \text{bottleneck}(P, f)$.
- **Flow Conservation:** let $u \in V$ be a vertex.
  - if $u \notin P$ then flow in and out of u doesn't change.

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\mathrm{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\mathrm{value}(f') = \mathrm{value}(f) + \mathit{bottleneck}(P, f)$$

and $f'_{\mathrm{in}}(s) = 0$.

- Let $b :=$ bottleneck(P, f).
- **Flow Conservation:** let $u \in V$ be a vertex.
  - if $u \in P$, have 4 cases to analyze. Let $e_1 := (w, u)$ and $e_2 := (u, z)$ be the edges in P passing through u in G_f.
    1. e_1, e_2 forward edges: *both* incoming and outgoing flow *increase* by b
    2. e_1, e_2 backward edges: *both* incoming and outgoing flow *decrease* by b
    3. e_1 forward, e_2 backward: *both* incoming and outgoing flow *unchanged*
    4. e_1 backward, e_2 forward: *both* incoming and outgoing flow *unchanged*

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\text{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\text{value}(f') = \text{value}(f) + \textit{bottleneck}(P, f)$$

and $f'_{\text{in}}(s) = 0$.

- Let $b := \text{bottleneck}(P, f)$.
- Value of flow f' and $f'_{\text{in}}(s)$:
  - $f_{\text{in}}(s) = 0 \Rightarrow$ no backward edges incident to s in G_f

$$f'_{\text{in}}(s) = f_{\text{in}}(s) + 0 = f_{\text{in}}(s) = 0$$

Improving Flow

Lemma (Flow Improvement)

Let f be a flow in G with $f_{\text{in}}(s) = 0$ and P an augmenting path with respect to f. If f' is the output from augment(f, P), then f' is a flow with

$$\text{value}(f') = \text{value}(f) + \mathit{bottleneck}(P, f)$$

and $f'_{\text{in}}(s) = 0$.

- Let $b := \text{bottleneck}(P, f)$.
- Value of flow f' and $f'_{\text{in}}(s)$:
  - Value of f': by previous bullet, only forward edges out of s, thus:

$$\text{value}(f') = f'_{\text{out}}(s) = f_{\text{out}}(s) + b = \text{value}(f) + b$$

Ford-Fulkerson Algorithm

Now that we know that augmenting paths can only improve our flow, we can describe Ford-Fulkerson, which simply applies the augmenting operation until we can no longer do it.

- Ford-Fulkerson(G):
  1. Initialize $f(e) = 0$ for all $e \in E$, and initialize G_f accordingly
  2. While there is $s \to t$ path $P \in G_f$:
     - $f \leftarrow$ augment(f, P)
     - update G_f
  3. **return** f

- Use BFS to decide whether there exists $s \to t$ path in G_f, and take P to be the shortest path returned by the BFS, if exists

Running Time Analysis

- Each iteration can be implemented in $O(n + m)$ time (runtime of BFS)

Running Time Analysis

- Each iteration can be implemented in $O(n + m)$ time (runtime of BFS)
- If all capacities are integral, then flow improvement lemma says that the value of our flow increases by at least 1 in each iteration.

Running Time Analysis

- Each iteration can be implemented in $O(n + m)$ time (runtime of BFS)
- If all capacities are integral, then flow improvement lemma says that the value of our flow increases by at least 1 in each iteration.
- If flow has value k, then runtime is

$$O(k \cdot (n + m))$$

Running Time Analysis

- Each iteration can be implemented in $O(n+m)$ time (runtime of BFS)
- If all capacities are integral, then flow improvement lemma says that the value of our flow increases by at least 1 in each iteration.
- If flow has value k, then runtime is

$$O(k \cdot (n+m))$$

For more details & variations on the algorithm we presented (and the proof by Edmonds-Karp), please see references.
Also, if you liked flows and want to learn more, consider taking C&O's Network Flows course.

- Ford-Fulkerson Recap
  - Algorithm
  - Running Time
- Max-Flow Min-Cut Theorem & Correctness of Ford-Fulkerson
- Acknowledgements

Max-Flow Min-Cut Theorem

Theorem (Max-Flow Min-Cut Theorem)

The value of the maximum $s - t$ flow equals the minimum capacity among all cuts.

$$\max_{f \text{ } s-t \text{ flow}} \mathrm{value}(f) = \min_{S \text{ is } s-t \text{ cut}} C_{\mathrm{out}}(S)$$

- **Easy direction:** given any flow f and $s - t$ cut S, we have

$$\mathrm{value}(f) \leq C_{\mathrm{out}}(S).$$

- To prove the above, will prove following claim:

$$f_{\mathrm{out}}(s) - f_{\mathrm{in}}(s) =: \mathrm{value}(f) = f_{\mathrm{out}}(S) - f_{\mathrm{in}}(S)$$

Proof of Claim 1

$$\begin{aligned}
\mathrm{value}(f) &= f_{\mathrm{out}}(s) - f_{\mathrm{in}}(s) \\
&= \sum_{v \in S} (f_{\mathrm{out}}(v) - f_{\mathrm{in}}(v)) && (\textit{flow conservation}) \\
&= \sum_{v \in S} \left(\sum_{z \in N_{out}(v)} f(v, z) - \sum_{w \in N_{in}(v)} f(w, v) \right) && (\text{definition}) \\
&= \sum_{e \in \delta_{out}(S)} f(e) - \sum_{e \in \delta_{in}(S)} f(e) && (\textit{cancellations}) \\
&= f_{\mathrm{out}}(S) - f_{\mathrm{in}}(S)
\end{aligned}$$

Hard direction

Proposition

If f is an $s \to t$ flow such that there is no $s \to t$ path in the residual graph G_f, then there is $s - t$ cut S such that $\mathrm{value}(f) = C_{\mathrm{out}}(S)$.

Hard direction

Proposition

If f is an $s \to t$ flow such that there is no $s \to t$ path in the residual graph G_f, then there is $s - t$ cut S such that $\mathrm{value}(f) = C_{\mathrm{out}}(S)$.

- No $s \to t$ path in G_f, by BFS/DFS, can find the set of visited vertices in G_f starting from s.
  Let S be this set. Then, no $s \to t$ path $\Rightarrow t \notin S$.

Hard direction

Proposition

If f is an $s \to t$ flow such that there is no $s \to t$ path in the residual graph G_f, then there is $s - t$ cut S such that $\mathrm{value}(f) = C_{\mathrm{out}}(S)$.

- No $s \to t$ path in G_f, by BFS/DFS, can find the set of visited vertices in G_f starting from s.
  Let S be this set. Then, no $s \to t$ path $\Rightarrow t \notin S$.
- We will prove that $C_{\mathrm{out}}(S) = \mathrm{value}(f)$. Let's look at G:
  - Let $(u, v) \in \delta_{out}(S)$. S has no outgoing edge in G_f implies $f((u, v)) = c((u, v))$ (otherwise G_f has forward edge)

Hard direction

Proposition

If f is an $s \to t$ flow such that there is no $s \to t$ path in the residual graph G_f, then there is $s - t$ cut S such that $\mathrm{value}(f) = C_{\mathrm{out}}(S)$.

- No $s \to t$ path in G_f, by BFS/DFS, can find the set of visited vertices in G_f starting from s.
  Let S be this set. Then, no $s \to t$ path $\Rightarrow t \notin S$.
- We will prove that $C_{\mathrm{out}}(S) = \mathrm{value}(f)$. Let's look at G:
  - Let $(u, v) \in \delta_{out}(S)$. S has no outgoing edge in G_f implies $f((u, v)) = c((u, v))$ (otherwise G_f has forward edge)
  - Let $(u', v') \in \delta_{in}(S)$. S has no outgoing edge in G_f implies $f((u', v')) = 0$ (otherwise G_f has backward edge)

Hard direction

Proposition

If f is an $s \to t$ flow such that there is no $s \to t$ path in the residual graph G_f, then there is $s - t$ cut S such that $\mathrm{value}(f) = C_{\mathrm{out}}(S)$.

- No $s \to t$ path in G_f, by BFS/DFS, can find the set of visited vertices in G_f starting from s.
  Let S be this set. Then, no $s \to t$ path $\Rightarrow t \notin S$.
- We will prove that $C_{\mathrm{out}}(S) = \mathrm{value}(f)$. Let's look at G:
  - Let $(u, v) \in \delta_{out}(S)$. S has no outgoing edge in G_f implies $f((u, v)) = c((u, v))$ (otherwise G_f has forward edge)
  - Let $(u', v') \in \delta_{in}(S)$. S has no outgoing edge in G_f implies $f((u', v')) = 0$ (otherwise G_f has backward edge)
  - Thus, we have:

$$f_{\mathrm{out}}(S) - f_{\mathrm{in}}(S) = C_{\mathrm{out}}(S) - 0 = C_{\mathrm{out}}(S)$$

Acknowledgement

Based on

- Prof. Lau's Lecture 15

  `https://cs.uwaterloo.ca/~lapchi/cs341/notes/L15.pdf`

- Jeff Erickson's book, Chapter 10

  `https://jeffe.cs.illinois.edu/teaching/algorithms/book/10-maxflow.pdf`

References I

Cormen, Thomas and Leiserson, Charles and Rivest, Ronald and Stein, Clifford. (2009)

Introduction to Algorithms, third edition.

MIT Press

Dasgupta, Sanjay and Papadimitriou, Christos and Vazirani, Umesh (2006)

Algorithms

Kleinberg, Jon and Tardos, Eva (2006)

Algorithm Design.

Addison Wesley