COMS 4995-3: Advanced Algorithms | March 1, 2017

Lecture 12 – Max Flow Algorithms (continued)

Instructor: *Alex Andoni* | Scribes: *Serena Liu, Lusa Zhan*

1 Introduction

Today's lecture is a continuation on the maximum flow problem. After discussing the maximum bottleneck algorithm introduced in the previous lecture, we continue to discuss to other algorithms for the maximum flow problem: a scaling algorithm and a strongly polynomial-time algorithm.

2 Last Time

2.1 Residual graph G_f

$$u_f(v,w) = u(v,w) - f(v,w), \text{if } (v,w) \in E$$

$$u_f(w,v) = f(v,w) > 0, \text{if } (w,v) \in E$$

2.2 s-t cut

$S, \bar{S}, s \in S, t \in \bar{S}$

$$u_f(S) = \sum_{\substack{(v,w)\in E \\ v \in S \\ w \in \bar{S}}} u_f(v,w)$$

2.3 Augmenting Path

P in G_f (s - t path)

$$u_f(P) = \delta_f(P) = \min_{e \in P} u_f(e) > 0$$

residual capacity / bottleneck

3 Maximum Bottleneck Algorithm

Recall from last lecture that we always aim to:
(1) find an augmenting path with maximum bottleneck capacity in G_f.
(2) The runtime for the algorithm is $O(m \lg U)$ per augmenting path.

Main Question: *How many augmenting paths are there?*

Claim 1. *The number of augmenting paths is bound by $m \ln(2nU)$*

Proof. Fix the current flow f. We start out with the following:

- Let g be the maximum flow in G_f. In particular, this means

$$|f| + |g| = \text{max flow in } G$$

- By the decomposition theorem, we can write

$$g = \text{ sum of } \leq m \text{ paths } P_1, \ldots, P_m$$

- $\exists i$ s.t. $|P_i| \geq \frac{|g|}{m}$
- If P' is the maximum bottleneck path in G_f, then

$$u_f(P') \geq u_f(P_i) \geq |P_i| \geq \frac{|g|}{m}$$

  where $|P_i|$ is the value of P_i in the decomposition.
- Then the value of the remaining flow is at most $|g| - u_f(P') \leq |g|(1 - \frac{1}{m})$
- After k augmenting paths, the value of the remaining maximum flow $\leq (1-\frac{1}{m})^k$[total max flow in G].
- We know total max-flow $\leq nU$
- Since we are working with integers, we are done when

$$(1 - \frac{1}{m})^k nU < 1$$

- Set $k = m \ln(2nU)$, then

$$(1 - \frac{1}{m})^k nU \leq e^{-\frac{1}{m}k} nU \leq \frac{1}{2}$$

- So number of augmenting paths: $k \leq m \ln(2nU)$

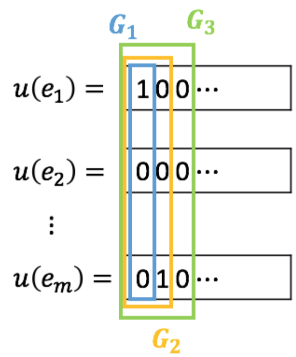
□

Therefore, the total time will be $O(m \lg U \cdot m \ln(2nU)) = O(m^2 \lg^2(nU))$

4 Scaling Algorithm

The idea behind the scaling algorithm is to reduce the algorithm so that the maximum flow is bounded. The goal is to improve the runtime.

There will be $b = \lg U$ scaling stages. Each stage i computes max flow in a graph G^i starting from some flow f_{i-1} (the flow constructed in the previous stage), and where the remaining flow $\leq m$.



If we solve each scaling stage in $O(m^2)$ time (using FF), then the total time is $O(m^2 \lg U)$.

Let G^i denote the graph that we get by replacing the capacities of G with $u^i(e)$, the first i bits of $u(e)$.

- Suppose we have a max flow f^i for G^i.
- G^{i+1} has capacities $u^{i+1}(e) = 2u^i(e) + v^{i+1}(e)$, where $v^{i+1}(e) \in \{0, 1\}$ and represent the last bits added. In other words, $v^{i+1}(e) = (i+1)^{th}$ most significant bit of $u(e)$.
- Note that $2f^i$ is still a valid flow in G^{i+1}, but $2f^i$ might not be the max flow in G^{i+1}
- Question: How much remaining flow is there in G^{i+1}?

  Since f^i is maximal in $G^i \Rightarrow$ exists cut S s.t. all edges $S \to \bar{S}$ are saturated.

$$u^{i+1}(S) \leq 2u^i(S) + m$$

  So the remaining flow in G^{i+1} is upper bounded by

$$u^{i+1}_{2f^i}(S) = u^{i+1}(S) - 2f^i \leq 2u(S)m - 2|f^i| \leq m$$

5 Strongly Polynomial-time Algorithm

We will discuss another algorithm that does not depend on max capacity, a strongly polynomial time algorithm runs in $(n \cdot m)^{O(1)}$ time.

In the real world model:

- input capacities are “words”
- can do reasonable operations on these words in $O(1)$ time.

The idea for the new algorithm (Combinatorial algorithm) is as follows:

- Take the augmenting path that minimizes the $s - t$ distance in the residual graph G_f.
- This is done within the FF algorithm framework.

- If we push/augment a path $s \to t$ and call it P, at least one edge on P will get saturated.

Definition 2. *$d_f(s,v) =$ min distance $s \to v$ in G_f.*

Lemma 3. *Fix $f, G_f, \{d(s,v)\}_v$. Let P be the shortest path $s \to t$.*

Then after augmenting we get flow f' and $G_{f'}$ and $\{d'(s,v)\}_v$

$$\forall v : d'(s,v) \geq d(s,v)$$

Proof. We prove the lemma by contradiction.

Let $A = \{v : d(s,v) > d'(s,v)\} \neq \emptyset$

Let $v \in A$ with minimal $d'(s,v)$.

Consider the shortest $s - v$ path P' after augmenting. Then

$$d'(s,v) = d'(s,w) + 1$$

In Figure 1, let w be the previous node before v. w satisfies $d'(s,w) \geq d(s,w)$ (because of the minimality of $d'(s,v)$).

So this means (w,v) in G_f' appeared when augmenting path P.

The edge $(v,w) \in G_f$ got saturated in P.

But since P was the shortest path before augmentation, we get

$$\begin{aligned} d(s,w) &= d(s,v) + 1 \\ &> d'(s,v) + 1 \\ &= d'(s,w) + 2 \\ &\geq d(s,w) + 2 \end{aligned}$$

which is a contradiction. $\square$

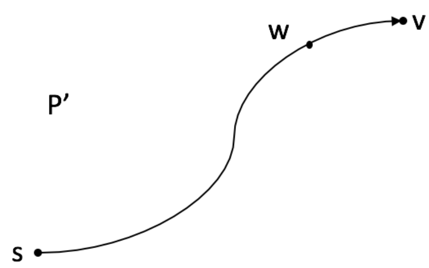


Figure 1: P' in G'_f (after augmenting)

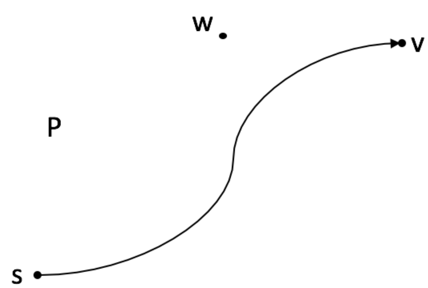


Figure 2: P in G_f (before augmenting)

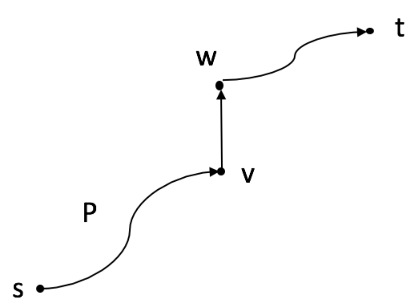


Figure 3: P in G_f