COMPSCI 330: Design and Analysis of Algorithms 09/30/2014

Lecture #11

Lecturer: Debmalya Panigrahi *Scribe: Wenshun Liu*

1 Overview

This lecture reviews the basic definitions and properties of Network Flow and the Augmenting Path Algorithm for computing the maxflow. The lecture then analyses the correctness of the Augmenting Path Algorithm as part of the introduction to the MaxFlow MinCut Theorem, and discusses potential optimisations to the runtime of the algorithm.

2 Review: Network Flow

2.1 Properties of Network Flows

Network Flows have the following two properties:

$$Capacity\ Constraints:\ f_e \leq u_e,\ \forall e \in E \tag{1}$$

$$Flow\ Balance:\ \Sigma_{e \in E_v^{in}} f_e = \Sigma_{e \in E_v^{out}} f_e,\ \forall v \in V \setminus \{s,t\} \tag{2}$$

2.2 Residual Network G_f

A residual network has the following capacities:

$$u_e^f = u_e - f_e\ \forall e \in E \tag{3}$$

$$u_e^f = f_e^R\ \forall e^R \in E \tag{4}$$

2.3 Augmenting Path Algorithm

pseudocode:

$f_e = 0\ \forall e \in E$
$G_f = G$
while $\exists$ path p from s to t in G_f
  $f'_e = min\ (u_e^f) \forall e \in p$
  augment f by combining f'
  recompute residual network G_f

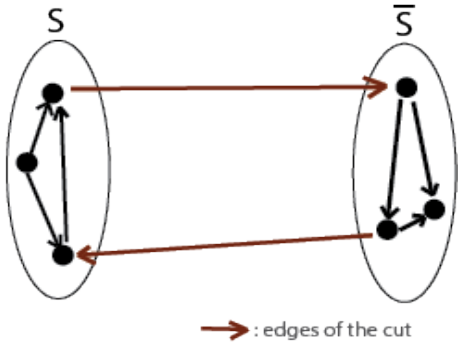
3 Definition

3.1 Cut

A cut is a bipartition of the vertices (S, $\overline{S}$).

3.2 Edges of a Cut

The edges of a cut is the set of edges that have one endpoint in each subset of the partition.



3.3 Value of a Cut

For undirected graphs, the value of a cut is the sum of capacities of the edges of the cut.
For directed graphs, the value of a cut is the sum of capacities of the edges in one direction.

$$value\ of\ the\ cut\ (S,\overline{S}) = \Sigma_{e\in(S,\overline{S})} u_e \tag{5}$$

(here $(S,\overline{S})$ indicates the direction of the edges of the cut, in terms of directed graphs)

3.4 Netflow of Cut(S, $\overline{S}$)

The netflow of a cut is the difference between the sum of edges in one direction and the sum of edges in the other direction.

$$netflow\ of\ the\ cut\ (S,\overline{S}) = \Sigma_{e\in(S,\overline{S})} f_e - \Sigma_{e\in(\overline{S},S)} f_e \tag{6}$$

4 MaxFlow MinCut Theorem

Lemma 1. *For any cut(S, $\overline{S}$), where s belongs to S and t belongs to $\overline{S}$, netflow of cut(S, $\overline{S}$) = value of the flow.*

Proof. $$\begin{aligned} netflow\ of\ the\ cut\ (S,\overline{S}) &= \Sigma_{e\in(S,\overline{S})} f_e - \Sigma_{e\in(\overline{S},S)} f_e \\ &= \Sigma_{e\in(S,\overline{S})} f_e - \Sigma_{e\in(\overline{S},S)} f_e + \Sigma_{e\in(S,S)} (f_e - f_e) \\ &\qquad\qquad (f_e - f_e \text{ is zero for each } e \in (S,S)) \\ &= \Sigma_{e\in(S,V)} f_e - \Sigma_{e\in(V,S)} \end{aligned}$$

(regroup elements so that $e \in (S,V)$ represents all edges start in S and end anywhere, and $e \in (V,S)$ represents all edges that start anywhere and end in S)

$= \Sigma_{u \in S} \Sigma_{e \in E_S^{out}} f_e - \Sigma_{u \in S} \Sigma_{e \in E_S^{in}} f_e$

$= \Sigma_{u \in S} (\Sigma_{e \in E_S^{out}} f_e - \Sigma_{e \in E_S^{in}} f_e)$

$= \Sigma_{e \in E_s^{out}} f_e - \Sigma_{e \in E_s^{in}} f_e$

(for every vertex other than s and t, $\Sigma_{e \in E_S^{out}} f_e - \Sigma_{e \in E_S^{in}} f_e$ is zero because of flow balance. Thus, all entries are cancelled out except when $u = s$)

= value of the flow (by definition) □

Observation 2. *For any cut(S, $\overline{S}$), where s belongs to S and t belongs to $\overline{S}$, netflow of cut(S, $\overline{S}$) $\leq$ value of the cut(S, $\overline{S}$).*

Proof. $net\,flow\,of\,the\,cut\ (S,\overline{S}) = \Sigma_{e \in (S,\overline{S})} f_e - \Sigma_{e \in (\overline{S},S)} f_e$

$\leq \Sigma_{e \in (S,\overline{S})} f_e$

$\leq \Sigma_{e \in (S,\overline{S})} u_e$ (capacity constraint)

= value of the cut(S, $\overline{S}$) (by definition) □

Theorem 3. *For any flow f from s to t, and for any cut(S, $\overline{S}$) s.t. $s \in S$ and $t \in \overline{S}$, value of f $\leq$ value of the cut(S, $\overline{S}$)*

Proof. The theorem can be easily proved, as from Lemma 1 and Observation 2 we know that,

$$value\,of\,the\,flow (= net\,flow\,of\,the\,cut\ (S,\overline{S})) \leq value\,of\,the\,cut$$

□

Corollary 4. *In a flow network, the maximum amount of flow passing from s to t is smaller or equal to the cut with the smallest value. (in short: maxflow $\leq$ mincut)*

Proof. The corollary can be easily proved based on Theorem 3. As maxflow is just some flow, and mincut is one example of the cuts for a flow network, the fact that $value\,of\,the\,flow \leq value\,of\,the\,cut$ indicates that $maxflow \leq mincut$.

□

Theorem 5. *In a flow network, the maximum amount of flow passing from s to t is equal to the cut with the smallest value. (in short: maxflow = mincut) (NOTE: this is the actual MaxFlow MinCut Theorem)*

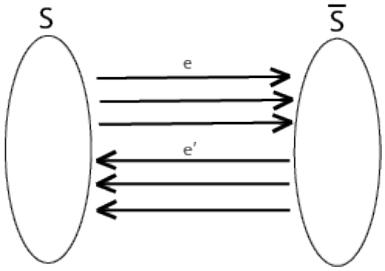
Proof. We prove this theorem using the Augmenting Path Algorithm. At the end of the algorithm, we know that there is no path from s to t in the residual network G_f. Thus, let set S in cut(S, $\overline{S}$) be defined as:

$$S = \{\, v \in V : \exists\ path\ from\ s\ to\ v\ in\ G_f \}$$

Because in the residual network G_f there's no edge going forward, we have,

$f_e = u_e$ (the capacity) and

$f_{e'} = 0$ (otherwise, could add an reverse edge on the residual network)



Thus, we know that at the end of the algorithm, $net\,flow\ on\ cut(S, \overline{S}) = \Sigma_{e \in (S, \overline{S})} u_e = value\ of\ cut(S, \overline{S})$

So far, we know that:

mincut $\geq$ maxflow f (Corollary 4)
$\geq$ Flow f' at the end of the Augmenting Path Algorithm
= netflow on a particular cut(S, $\overline{S}$) at the end of the algorithm
= value of the cut (S, $\overline{S}$) (proven above)
$\geq$ mincut (a trivial observation)

Obviously, mincut = mincut, so all inequalities above have to be equal. Thus, we have mincut = maxflow. □

Remark 1. *The MaxFlow MinCut Theorem also proves the correctness of the Augmenting Path Algorithm, as in proving the theorem we've shown that,*

$$maxflow = Flow\ f'\ at\ the\ end\ of\ the\ Augmenting\ Path\ Algorithm \tag{7}$$

which shows that the Augmenting Path Algorithm correctly computes the maxflow of a flow network.

5 Runtime Analysis of the Augmenting Path Algorithm

For each round of the loop we have:

while $\exists$ path p from s to t in G_f $\Leftarrow$ O(m), as we could check the condition using any traversal algorithm
$f'_e = min\ (u_e^f) \forall e \in p$ $\Leftarrow$ O(n), as there are no more than n vertices
augment f by combining f' $\Leftarrow$ O(m), as it only involves some constant time operations on each edge
recompute residual network G_f $\Leftarrow$ O(m), as it only involves edgewise operations

Thus, each round of the loop takes runtime O(m).

If capacities are *integers*, we know that after each round of the loop, the flow will be increased by at least 1. As a result, the number of loops cannot exceed $O(f)$, with f being the maxflow the algorithm finds. Thus, the total runtime of the Augmenting Path Algorithm is $O(mf)$.

Remark 2. *The Augmenting Path Algorithm is a pseudo-polynomial algorithm.*

Remark 3. *Strongly polynomial algorithms for finding the maxflow also exist. The primary optimisation is to always choose the shortest path from s to t in G_f (while in the Augmenting Path Algorithm, we have a lot of flexibility in terms of which path to choose). We will now introduce another optimisation to the Augmenting Path Algorithm, the result of which would be weakly polynomial.*

6 Scaling Algorithm

6.1 Description

The Scaling Algorithm optimises the Augmenting Path Algorithm by realising that we could actually increase the flow in larger steps. It starts by choosing a scale Δ which is usually initialised at a large value, and try to increase the flow by the scale.

pseudocode:

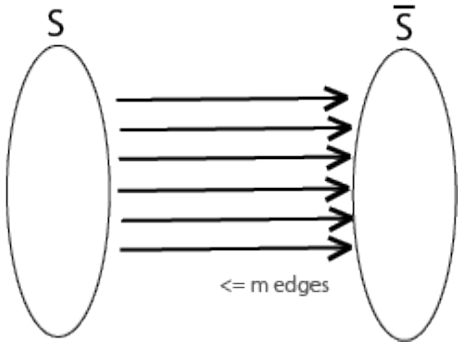
$\Delta = max_{e \in E} u_e$
$f_e = 0 \; \forall e \in E$
$G_f = G$
while $\Delta \geq 1$ (when the flow can still increase, with the assumption that we have integral capacities)
    while $\exists$ path p from s to t in G_f
        $f'_e = min\ (u_e^f) \forall e \in p$
        augment f by combining f'
        recompute residual network G_f
    end while
    $\Delta = \Delta/2$
end while

Remark 4. *We have $G_f(\Delta) = \{e \in E_f : u_e^f \geq \Delta\}$. Thus we have,*
1. if the original value of Δ is 1, there's no difference between the Scaling Algorithm and the Augmenting Path Algorithm, as we are not able to drop any edge (basically, $G_f(\Delta) = G_f$).
2. when the algorithm ends, $G_f(\Delta)$ is just the residual network, as both algorithms have the same termination condition.

6.2 Runtime Analysis

Lemma 6. *When Δ is rescaled to $\Delta/2$, the max remaining flow is at most $m\Delta$, with m being total number of edges.*

Proof. We know that the total number of edges of the cut $\leq m$. We know that each edge has capacity $< \Delta$, or we won't be able to go from Δ to $\Delta/2$. Thus, the total remaining value of the cut must be less than or equal to $m\Delta$.



□

For each round of the inner loop, the runtime is the same as that of the Augmenting Path Algorithm ($O(m)$).
The number of rounds of the outer loop is $O(log\ max_{e \in E} u_e)$, as we are always halving Δ.
The number of rounds of the inner loop (for each fixed Δ) is $O(m)$. This is because Lemma 6 shows that when Δ is rescaled to $\Delta/2$, the max remaining flow is at most $m\Delta$. Thus, at level $\Delta/2$, we can have at most $2m$ iterations.

Thus, the total runtime for the Scaling Algorithm is $O(m^2 log\ max_{e \in E} u_e)$ (weakly polynomial).