MATH 409 LECTURE 14
MAX FLOW MIN CUT THEOREM

REKHA THOMAS

In this lecture we will establish the max flow min cut theorem and see an algorithm for solving the max flow problem in a digraph. The material is taken from partly from [1].

Definition 1.
- Given a digraph G with capacities $(u_e,\, e \in E(G))$, and a flow f, define the upper and lower **residual capacities** of e as follows: $u_f^+(e) := u_e - f(e)$ and $u_f^-(e) := f(e)$.
- The **residual graph** $G_f := (V(G), \{e \in E(G) \,:\, \text{either } u_f^+(e) > 0 \text{ or } u_f^-(e) > 0\})$.
- Given a network (G, u, s, t) and an (s,t)-flow f, an f-augmenting path P is an (s,t)-path in the underlying graph of G_f such that when we view the edges from $E(G)$ in P, all forward edges have $u_f^+(e) > 0$ and all backward edges have $u_f^-(e) > 0$.
- Given a flow f and a path (or circuit) P in G_f, to **augment** f along an f-augmenting path P by γ means to do the following for each $e \in E(P)$: Increase $f(e)$ by γ on all the forward edges of the path and decrease $f(e)$ by γ on all the backward edges of P.

Ford and Fulkerson's algorithm for Max Flows
Input: A network (G, u, s, t).
Output: A max (s,t)-flow in the network.

(1) Set $f(e) = 0$ for all $e \in E(G)$.
(2) Find an f-augmenting path P. If none exists then stop.
(3) Compute $\gamma :=$ minimum of

$$\{u_f^+(e) \,:\, e\,\text{forward edge in P}\} \cup \{u_f^-(e) \,:\, e\,\text{backward edge in P}\}.$$

Augment f along P by γ and go to (2).

We did an example in class to both illustrate the algorithm and to show that in general this algorithm may be exponential in the size of the input.

Date: April 28, 2010.

Note that the way the algorithm works, we will be guaranteed to have an integral optimal flow if all the capacities u_e, $e \in E(G)$ are integers. Thus, if we had imposed integrality restrictions on the variables in the max flow problem, Ford and Fulkerson's algorithm would find us an optimal integral flow as long as the capacities are integral.

Theorem 2. *An (s,t)-flow f is maximum if and only if there does not exist an f-augmenting path in G_f.*

Proof. ($\Rightarrow$): We prove this by proving the contrapositive. Suppose there exists an f-augmenting path. Then Step (3) of the algorithm computes a flow of greater value which shows that f is not a max flow. (Note that if we can augment, then the augmentation is always by a $\gamma > 0$ which shows that the flow will strictly increase if an augmentation is possible.)

($\Leftarrow$): We prove this directly. Suppose there is no f-augmenting path. Then this means that the sink t is not reachable from s in the residual graph G_f. Let R be the set of vertices that can be reached from s in G_f. Consider the sets $\delta_G^+(R)$ which consists of the edges $e \in E(G)$ with tail in R and head in $V \backslash R$ and $\delta_G^-(R)$ which consists of the edges $e \in E(G)$ with head in R and tail in $V \backslash R$. By the definition of G_f, we must have that $f(e) = u_e$ for all $e \in \delta_G^+(R)$ and $f(e) = 0$ for all $e \in \delta_G^-(R)$. By Lemma 5 in the last lecture,

$$value(f) = \sum_{e \in \delta_G^+(R)} f(e) - \sum_{e \in \delta_G^-(R)} f(e) \leq \sum_{e \in \delta_G^+(R)} u_e.$$

In our situation we have

$$value(f) = \sum_{e \in \delta_G^+(R)} f(e) - \sum_{e \in \delta_G^-(R)} f(e) = \sum_{e \in \delta_G^+(R)} u_e$$

where $\sum_{e \in \delta_G^+(R)} u_e$ is the capcity of the cut induced by R. Thus the value of our flow f equals the capacity of a cut which implies that f is a max flow and that $\delta_G^+(R)$ is a min cut. $\square$

Using Lemma 5 from Lecture 13, we have therefore proved the max flow min cut theorem:

Theorem 3. *The max value of a $s-t$ flow in the network (G, u, s, t) equals the min capacity of a $s-t$ cut in the network.*

Remark 4. Note that the Ford-Fulkerson algorithm not only finds a max flow but also the min cut in the network.

Exercise 5. Find the max flow in the following network where each directed and its capacity are given as a pair:

$\{sp, 1\}, \{pa, 1\}, \{ap, 1\}, \{at, 1\}, \{sq, 4\}, \{qb, 1\}, \{bq, 1\}, \{qa, 3\}, \{pb, 3\}, \{bt, 4\}$

References

[1] B. Korte and J. Vygen. *Combinatorial Optimization*. Springer, Berlin, 2000.