CMSC 641, Design and Analysis of Algorithms, Spring 2010

Sandipan Dey,
Homework Assignment - 4

March 8, 2010

Problem 1 Solution

While choosing the augmented path, if we ensure that we always choose an augmenting path containing the edge with minimum flow value, we can argue that the Ford-Fulkerson algorithm must terminate after $|E|$ steps.

Do the following steps iteratively:

1. Start with the current network flow graph (G, f), with maximum flow f.
2. Find $e = \underset{(u,v)\in E(G)}{argmin}\ f(u, v)$. Notice that the maximum flow $f \geq f(e)$, since all the other edges in G must have higher (or equal) flow values.
3. Find an augmenting path p from s to t, containing e. Such a path must exist, by flow property.
4. Augment the path p to obtain new flow graph $(G\prime, f\prime)$, with new maximum flow $f\prime$. Now $c_f(e)$ becomes the residual capacity $c_f(p)$, with e being the critical edge. By Ford-Fulkerson, the residual graph $G\prime_{f\prime}$ will no longer contain the edge e (since already saturated, the corresponding flow value will be set to 0).

Since there will be 0 on one more edge, namely, e, in $f\prime$ than in f in each iteration and there can be at most E edges in the flow graph, we can have at most $|E|$ iterations after which there can be no such augmentation done and the algorithm must terminate. Hence, an edge can always be found after $|E|$ iterative stpes.

Problem 2 Solution

By the MAX-FLOW/MIN-CUT theorem, the maximum flow value f of a network $G(V, E)$ also gives size of the minimum cut (S, T). Also, the edge connectivity os defined by the minimum number of edges needed to be removed to

disconnect G.

Now, by definition of a CUT, if we remove all the edges crossing a CUT, G will be diconnected. Hence, by a single execution of Ford-Fulkerson with any two arbirtary nodes as s and t from G and with capacity $c(u, v) = 1, \forall u, v \in V(G)$, (thus with $O(V)$ nodes and $O(E)$ edges), we can find a CUT size in the graph. Only thing left is to ensure the minimality of the CUT size.

We need to do the following steps:

1. Setup $V - 1 = O(V)$ instances of flow networks from G.
2. Each instance with same source $s = u$ (fixed for all flow networks, chosen arbitrarily in the beginning).
3. Each instance with different sinks $t \in V - \{u\}$ (for different networks)
4. Run maximum-flow algorithm ($V - 1 = O(V)$ instances) on each of the networks to obtain maximum flows (i.e., minimum CUTs), $|f_{uv}|$.
5. Find $\min_{v \in V - \{u\}} |f_{uv}|$, i.e., find the edge-connectivity by taking minimum of all these max-flow sizes.

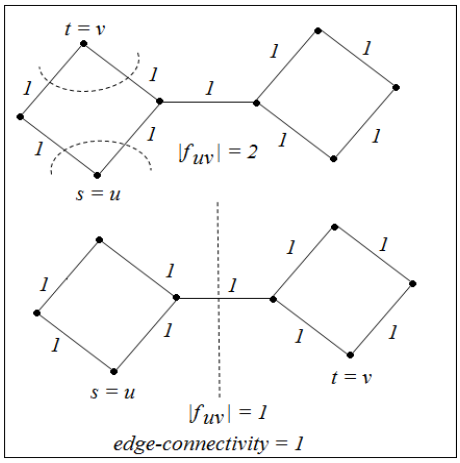

Figure 1: edge-connectivity= $\min_{v \in V - \{u\}} |f_{uv}|$, for any $u \in V(G)$

The correctness proof comes from the fact that the smallest set of edges removing which the graph becomes disconnected must be identical to at least one of the MIN-CUTS obtained by the $O(V)$ instances of the max-flow algorithm, ran on $O(V)$ different flow networks formed.

Since we need at least $\min_{v \in V-\{u\}} |f_{uv}|$ edges to be removed to gurantee that G is disconnected (follows from MAX-FLOW/MIN-CUT), edge-connectivity must be at least this. Also, considering the vertex v with minimum f_{uv}, the MIN-CUT solution obtained in this case must upper bound the edge-connectivity, since removing the edges in the cut-set disconnects G.

Problem 3 Solution

Part (a)

MIN-CUT capacity of $G = C(S,T)$
$= \sum_{u \in S, v \in T} c(u,v)$ (where (S,T) is the MIN-CUT, $S, T \subseteq G,\ S \cap T = \phi$)
$\Rightarrow C(S,T) \leq \sum_{u \in S, v \in T} C$ (since $C = \max_{(u,v) \in E} c(u,v) \geq c(u,v),\ \forall (u,v) \in E$)
$\Rightarrow C(S,T) \leq C|S,T| \leq C|E|$ (since number of edges crossing the cut $\leq |E|$)
$\Rightarrow$ MIN-CUT capacity of G is at most $C|E|$ (Proved).

Part (b)

Capacity of augmenting path = minimum capacity of an edge on the path $\Rightarrow$ to find an augmenting path of capacity at least $K \Rightarrow$ the edge with the least capacity and hence all the edges must be at least K. If such a path exists, we can use BFS/DFS to find such a path in the usual manner, but while looking for the adjacent vertices, we have to perform an additional check whether the capacity of the corresponding edge is at least K.

Part (c)

In order to find the flow in the network, MAX-FLOW-BY-SCALING uses the Ford-Fulkerson method and everytime it chooses an augmenting path with a capacity lower boound, that decreases linearly per iteration with maximum capacity. It repeatedly augments the flow along an augmenting path until there are no augmenting paths of capacity ≥ 1. Since all the capacities are integers, and the capacity of an augmenting path is positive, when the algorithm terminates, there will not be any augmenting path in the residual graph. Thus, by the max-flow min-cut theorem, MAX-FLOW-BY-SCALING returns a maximum flow.

Part (d)

1. At line 4, for the first time (before start of execution of the loop), G_f is same as G, hence $K = 2^{\lfloor lg\ C \rfloor} \Rightarrow 2K = 2^{\lfloor lg\ C \rfloor + 1} \geq 2^{\lceil lg\ C \rceil} \geq 2^{lg\ C} = C$. Also, from part (a), we have, minimum cut capacity of $G \leq C|E| \leq 2K|E| \Rightarrow$ minimum cut capacity of G_f is at most $2K|E|$ at this point.

2. From the next time onwards, do-while loop on lines $5-6$ finds all augmenting paths of length at least K. Hence, when it comes to line 7, there is no augmenting path of capacity K in $G_f \Rightarrow$ maximum flow in G_f must have value $< K|E| \Rightarrow$ min cut in G_f has capacity $< K|E|$ (by MAX-FLOW/MIN-CUT theorem). But at line 7, $K = \frac{K}{2} \Rightarrow$ at line 4 (next iteration) the min CUT capacity of G_f can at most be $2K|E|$.

Part (e)

The maximum flow in G is at most $2K|E|$ more than the current flow in G, since from part (d), (MIN-CUT capacity and hence) the MAX-FLOW of G_f can be at most $2K|E|$ and from the fact that the flow sum in G and G_f is a flow in G. Also, every time the inner do-while on lines $5-6$ finds an augmenting path of capacity at least K, the flow in G increases by at least K. Since the flow cannot increase by more than $2K|E|$, the loop executes at most $(2K|E|)/K = 2|E| = O(E)$ times.

Part (f)

Line 1 runs in $O(E)$ time. Lines $2-3$ and 8 are $O(1)$ operations. Do-while loop $5-6$ is executed $O(E)$ times from part (e) and each time it takes $O(E)$ time to find an augmenting path of capacity at least K, from part (b), hence total time $O(E^2)$ for each K. Line 7 is constant time $O(1)$ again. Again, the while loop on lines $4-7$ runs for $O(lg\ C)$ times, since $K = O(C)$ intially and K halves at every iteration. Hence, the total run time of the algorithm MAX-FLOW-BY-SCALING $= O(E^2 lg\ C)$.