CS 577 Homework 6

Ellie Holzhausen and Andrew Ma

May 1, 2012

Problem 1

For this question, you are given a directed network G = (V,E) with capacities c_e on edges and a sourse-sink pair (s,t). You are also given the max s-t flow in the graph, f* specifying the amount of flow on every edge in the graph. An edge in the network is called a critical edge if reducing its capacity by one unit decreases the s-t max flow in the network.
Develop an algorithm for determining all the critical edges in the given graph. Your algorithm is allowed to use the max flow f* and should run in time O(mn).

1. Construct the residual graph, G'
2. For every saturated edge in the flor $f*$, remove the corresponding forward and back edges from G'
3. Perform BFS off of each node in G' to determine and to record, given a node i, which nodes in G' are connected to i.
4. For every saturated edge, (a, b), in the the $f*$, use the information from step 1 to determine whether there is an alternate path in the G' connecting a to b. If there is no alternate path from a to b, then (a, b) is a critical edge and is recorded.

Claim: If this algorithm determines an edge is critical, it is indeed critical.
Proof: Edges found using this algorithm are those edges which are saturated in the flow $f*$, and when capacity is reduced by one, there is no way to redirect the one unit of flow via some other path to node b. Therefore, an edge found, must be critical as, as reducing the capacity of this edge reduces the the flow along the $s - t$ path that the found edge sits on, which in turn reduces the overall max-flow, $f*$.

Claim: If (a, b) is a critical edge in the graph, the above algorithm finds it.
Proof: If an edge (a, b) is either non-saturated or there exists a non-saturated alternate path from a to b, then it is not a critical edge. If the edge (a, b) is not saturated, then reducing its capacity does not affect the max flow at all, because it had additional unused capacity already. If the edge (a, b) is saturated, but there exists an alternate path from a to b in G' then we can divert the flow lost from reducing the capacity of (a, b) along the other path and no flow is lost from the max flow of the graph, and thus this particular edge should not be a critical edge.Overall, any edge that is saturated which cannot have one unit of flow redirected, must be a critical edge, and our algorithm seraches the residul graph for exactly these edges.

Proof of run time: For step 1 we can construct the residual graph in $O(m+n)$ time. In step 2, we iterate over every edge in $f*$ which is $O(m)$. Step 3, we run BFS off od every node in the graph. Running BFS takes O(m + n) time and each time we are recording information regarding connectivity, so the whole step takes $O(n)O(m+n)$. Step 4 we do another iteration over the edges in $f*$ so we do $O(m)$ iterations. In total this is $O(m+n) + O(m) + O(n)O(m+n) + O(m) = O(mn)$, because this graph is connected and $m \gg n$.

Problem 2

Consider the flollowing problem. You are given a flow network with unit-capacity edges: it consists of a directed graph G = (V,E), a sourse $s \in V$, and a sink $t \in V$; and $c_e = 1$ for every edge $e \in E$. You are also given a parameter k. The goal is to delete k edges so as to reduce the maximum s-t flow in G by as much as possible. In other words, you should find a set of edges F so that the order of $F = k$ and the maximum s-t flow in $G' = (V, E - F)$ is as smal as possible subject to this. Give a polynomial time algorithm to solve this problem.

1. Run Ford-Fulkerson to find a min-cut
2. if number of edges in min cut $> k$ then remove k edges from this min-cut else if number of edges in min cut $< k$ then remove every edge in the min cut, and take the remaining edges randomly out of the leftover graph.

Proof of Correctness:
Claim: Every critical edge is in a min cut and every min cut edge is a critical edge.
Proof: Say that there is some critical edge that belongs to a cut which is not a min cut.Then there is a some unused edge in this cut, and then if we remove our critical edge, we could redirect flow via paths in the residual graph to send the unit of flow along the unused edge in the cut (this is possible because the original graph is connected). Then the max flow would not be changed. Now, assume that you have a min cut. Remove one of the edges contained in the min cut. Because it was a min cut, removing this edge reduced the graph's max flow. Thus, the edge you removed must have been a critical edge.

Note that every edge we remove from the graph that redeuces the max flow is a critical edge, and given the above claim, we know that to remove edges the optimally, we should remove critical edges i.e. edges in a min cut. Then, all we have to do is remove k edges from the min cut. We need run Ford-Fulkerson only once because given an edge which is a min cut, removing an edge will lower the max flow by one, and the leftover cut will again be a min cut since it has a value of one less than the previous max flow value, which is now the current max flow value. Finally, if $k >$ the number of edges in the min cut, we will reach a point where we have disconnected the entire graph, and are thus unable to reduce the flow any further.

Proof of run time:
Running Ford-Fulkerson is $O(mn)$ since the capacities in this graph are 1 the n can be an upperbound on the size of any given cut. Then getting a min cut from Ford-Fulkerson will involve using the algorithm on page 350 of the textbook, which runs in $O(m)$. Finally since we remove k edges there are $O(k)$ operations for the removals. This gives a total run time of $O(n + mn + k) = O(mn + k)$.

Problem 3

Problem 7.27 in the textbook (Pg. 431).

(a)
Claim: There exists a fair driving schedule.
Pf: Construct a bipartite-like graph, G, in the following way (see the attached crude drawing for an example of what I mean):

1. create nodes s, t, p'_i for $i \in [1, k]$ representing people, and nodes S'_j for $j \in [1, d]$ representing the days of carpooling

2. for each node p_i' add an edge (s, p_i') with edge wt. $\lceil \sum_{i:p_j \in S_j} \frac{1}{|S_j|} \rceil$

3. for each person p_i in subset S_j, add an edge (p_i', S_j') of wt. 1

4. for each node S_j', add an edge (S_j', t) of edge wt. 1

Claim: There exists an integral max flow of size d in G
Pf: Note that it is ETS that there is a fractional flow of size d. So for each (s, p_i') edge in the graph send $\sum_{i:p_j \in S_j} \frac{1}{|S_j|}$ units of flow. For each edge (p_i', S_j') send $\frac{1}{|S_j|}$ units of flow along this edge. For each edge (S_j', t), send 1 unit of flow along this edge. First note that the this is a valid flow since the flow sent along each edge is a below that edge's capacity, so the capacitiy condition is fine. Also, by construction and the flow conservation is fine. Next, see that the value of this flow is d, because the value of the flow going into node t is d. Moreover, this fractional flow is maximal because the cut which partitions G into the subgraphs t and everything else is a cut where every edge is saturated, and the value of this cut is d. Since Ford-Fulkerson guarantees a max flow that is also an integral flow, we can conclude that there exists an integral max flow of value d.

Claim: Every fair scheduling corresponds to a max flow in G, and every max flow in G corresponds to a fair scheduling
Pf: Every fair scheduling corresponds to a max flow in G: Given a fair scheduling, for each person, p_i, driving on day S_j, send a unit of flow along the node (p_i', S_j'). Also, for each person p_i send a flow equal to the numberof days they drive along the edge (s, p_i'). Finally send a unit of flow along (S_j', t). In this construction the capacity condition for a flow is satisfied since the number of days a person drives $\leq \lceil \sum_{i:p_j \in S_j} \frac{1}{|S_j|} \rceil$, and the other edges have capacity 1. Finally, this flow also satisfies the flow conservation because for each node S_j' the flow conservation is clear as only one person drives a day, and someone drives each day. Also, the flow into a node p_i' should be the same as the outgoing flow since the both values will equal the number of days person p_i drives. Finally, this induced flow is a max flow as there is a cut made of the edges going into t, and each edge in this cut is saturated.

Every max flow in G corresponds to a fair scheduling: Given an integral max flow in G, for each edge in the flow of the form (p_i', S_j'), assign person p_i to drive on day S_j. Since the max flow satisfies the capacity constraints, this will work as a valid fair schedule.

By earlier claims, a solution to the fair schedule is the same as an integral max flow, and an integral max flow is guaranteed by Ford-Fulkerson. Therefore there exists a fair schedule.

(b)

1. construct graph G as described in part (a)

2. run Ford-Fulkerson to find an integral max flow, $f*$

3. iterate over the edges in $f*$ and for each edge of the form (p_i', S_j'), assign person p_i to drive on day S_j

Proof of correctness: Using the claims in part (a) solutions to the fair schedule is the same as finding a max flow in G

Proof of run time: Step 1 takes $O(k)O(d)$, as there are k people and d days, so in the worst case, each person can drive every day and there would need to be $k \cdot d$ edges added to the graph (Additionally there are k edges to connect the nodes p_i' to s and d edges to connect nodes S_j' to t, however these are smaller factors than $O(k)O(d)$). For step 2, running Ford-Fulkerson will take $O((k+d) \cdot d)$ as there are $k+d$ nodes in G and the min cut has value d. Finally, step 3 is an iteration over the edges in $f*$ and the worst size $f*$ could be

would be the $O(d)$ as d is the number of days of driving and there is exactly one person driving each day, so only one edge leads into any particular day. In total there is a run time of $O(kd)+O((k+d)\cdot d)+O(d) = O(kd)$ as $k \geq d$.