Massachusetts Institute of Technology
6.854/18.415: Advanced Algorithms
David Karger

Handout 14
November 7, 1999

Problem Set 6 Solutions

Problem 1. Assume that we are given graph $G = (V, E)$ with source s and sink t. We compute the max-flow of G. Let M be the max-flow value. We construct a graph G' on which a min-cost max-flow algorithm produces the required minimum cost flow. Graph $G' = (V', E')$ is constructed from G such that $V' = V \cup \{s'\}$ and $E' = E \cup \{s's, st\}$. The source of G' is s' and the sink is still t. The edge $s's$ has capacity M and cost 0, and forces the min-cost max-flow algorithm to send M flow through the remaining graph.

(a) The edge st has capacity $M/10$ and cost 0 and therefore allows at most 10% of the flow to bypass the original graph. Recall that s' is the source of G' and allows at most M units of flow to s. The max-flow value of G' is therefore M, but the flow in its subgraph G can be in the range $[9M/10, M]$. We can thus establish a one-to-one mapping from a feasible 90% flow in G to a flow in G' with same cost. So we compute the min-cost max-flow of G' and look for the required minimum cost flow in its subgraph G.

(b) In this case, the edge st has capacity ∞ and cost K. Again, we have a one-to-one mapping from a feasible flow in G to a flow in G' with same cost. We compute the min-cost flow of G' and extract the required flow from subgraph G.

Problem 2. Given graph G and its min-cost flow F, we can compute the residual graph G_F in $O(m)$ time. A different optimum solution to this problem exists iff there is a zero-cost cycle in G_F. This equivalence follows from the existence of another solution in the presence of a zero-cost cycle and the existence of a zero-cost circulation in the presence of another optimum solution.

Finding a zero-cost cycle involves computing reduced costs for edges in G_F. Since the reduced costs are non-negative, a zero-cost circulation exists iff a cycle of reduced cost 0 edges exists. Such a cycle can be computed by doing a depth first search.

The time complexity of this algorithm is dominated by the $O(mn)$ running time of Bellman-Ford algorithm to find the reduced costs.

Problem 3. Given a graph G and its min-cost flow F, we would like to re-optimize the solution after increasing/decreasing the cost of one edge by 1. We compute the residual graph G_F and the reduced edge costs using Bellman-Ford algorithm. The reduced edge costs are always non-negative and no negative cost cycles exists in G_F. If an edge's reduced cost is strictly positive, the flow in the edge is 0.

Increasing the cost of an edge (say) $e = (v, w)$ in G_F can potentially increase the minimum cost. If the reduced cost of the edge is strictly greater than 0, the prices are valid after the cost of e is incremented. Therefore reduced-cost optimality is also satisfied. If the reduced cost is 0, we need to re-optimize the solution.

We would like to push as little flow as possible on the edge e. Edge e may have some flow $f(e)$ in it if its reduced cost is 0. We perform a max-flow on the subgraph $G_F - e$ constrained by a maximum of $f(e)$ at the source (done by adding an edge $s's$ with capacity $f(e)$). Let F' be the max-flow. We send flow F' from v to w. If $|F'| = f(e)$, we send no flow across e and our solution is optimal. Otherwise, we send flow $f(e) - |F'|$ through edge e. This is optimal since there is no negative cycle induced by edge e. If there does exist one, it contradicts the optimality of F'.

Decreasing the cost of edge e can potentially reduce the minimum cost. If the reduced cost of e is strictly greater than 0, the prices are valid even after the decrement and reduced-cost optimality is satisfied. If the reduced cost of e is 0, we need to re-optimize the solution.

We would like to send as much flow across e as possible. As in the previous section, we compute a max-flow from w to v limiting the flow value by $u(e) - f(e)$. Even if all of $u(e) - f(e)$ units are not shipped, the solution is optimal due to the absence of a negative cost cycle.

From the above discussion, changing the cost by 1 involves one Bellman-Ford reduced costs computation and one max-flow computation. The $O(mn \log n)$ time taken by max-flow domintates the running time.

We can design a cost scaling algorithm using the above re-optimization routine. In a phase of the scaling algorithm, we shift a bit to the costs on the edges. At the beginning of kth phase, we have a min-cost flow on the graph with first k bits of the cost. We double the costs. Notice that doubling the cost does not change the min-cost flow. Then, for each edge e with $(k+1)$st bit set to 1, we increment/decrement the cost based on the sign of cost $c(e)$. Each phase involves upto m unit changes in edge costs. There are $O(\log C)$ phases. So the running time of algorithm is $O(m^2 n \log n \log C)$.

Problem 4. We represent the following problems as min-cost flow computations on a graph G.

(a) Student i is associated with node v_i and faculty member j is associated with node w_j. Suppose the cost of associating a student i with faculty member j is c_{ij}. We add capacity 1 edges with cost $-c_{ij}$ from v_i to w_j. To limit the number of associations per student and faculty member, we add a an edge from source s to each v_i with capacity 1 and cost 0 and an edge from each w_j to sink t with capacity 1. A min-cost max-flow on G gives the maximum money-generating assignment of faculty members to students.

(b) Every day the same student meets the same faculty member and MIT ends up earning d times the money earned in a day.

(c) Each faculty member is associated with two "faces" — one that gives complete attention and the other that gives partial attention. The fully-attentive face of faculty member j is assigned node w_j. We construct the graph as mentioned in the solution to part (a). In addition we assign the partial-attention face of faculty member j to node x_j and connect each v_i to x_j with cost $-c_{ij}/2$. All x_j nodes are connected to t with capacity 1 and cost 0. Now, each student is assigned to exactly one faculty member and each faculty member can give full attention to one student and partial attention to another. A min-cost flow on this graph produces the maximum money-generating assignment.

Problem 5. We start with a ϵ-optimal flow F_0 in the graph G. All negative arcs are saturated and excesses and deficits are created. This "pseudo-flow" is now 0-optimal. We would like to convert the pseudo-flow to a flow that is $\epsilon/2$-optimal. Flow is pushed only along admissible arcs, i.e., arcs that have negative reduced cost. The minimum reduced cost in the residual arc is now maintained as $-\epsilon/2$.

(a) Since there is no admissible path from any excess to any deficit, the set of nodes reachable from an excess using admissible arcs does not include any deficit. If we decrease the price of nodes in this set, the arcs connecting two nodes in this set do not undergo changes in reduced cost. All outgoing arcs from this set had non-negative costs before relabeling. These arcs remain $\epsilon/2$-optimal after relabeling. Incoming arcs have their reduced cost increased and therefore maintain $\epsilon/2$-optimality.

(b) We perform the blocking flow steps on the admissible graph from a super-source connecting all excesses to a super-sink connecting all deficits. After the blocking flow, all remaining uncancelled excesses are disconnected from deficits in the admissible graph.

(c) Consider the ϵ-optimal flow F_0 we start with. After converting this into a pseudo-flow, we have reversed residual arcs with no flow and reduced cost at most ϵ. Let us consider an excess and a deficit that belong to a saturated path. If no other means is economical, the excess can be sent back to the deficit using the residual arc. In the pseudo-flow the path from the excess to deficit has arcs that have reduced cost at most ϵ. So such a path of length l will be made admissible after $3l$ relabels.

After $3\sqrt{m}$ relabel steps, we are left with paths in G from excesses to deficits that are longer than $\sqrt{m}$. For the remaining units of excesses to be shipped at least $\sqrt{m}$ edges need to be saturated per unit flow. So we are left with at most $\sqrt{m}$ units of excesses.

(d) Since G_F is acyclic, we perform a Dijkstra's shortest paths on reduced costs from the super-sink to super-source to determine a price function. This relabelling ensures that there is at least one admissible path from an excess to a deficit. Notice that this is equivalent to the shortest augmenting paths algorithm.

(e) After $3\sqrt{m}$ block/relabel steps we are left with at most $\sqrt{m}$ excesses. These excesses can be sent to deficits by performing $O(\sqrt{m})$ block/Dijkstra-relabels that take $O(m^{3/2}\log n)$ time. Thus a scaling step takes $O(m^{3/2}\log n)$ time.