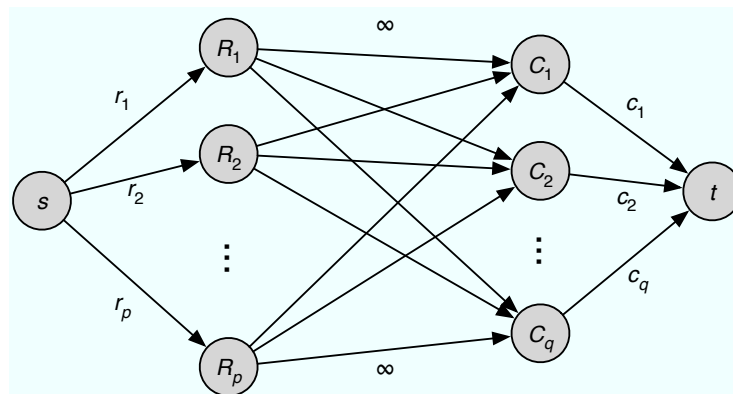


Problem Set 4 Solutions

Friday, October 9, 2009

Problem 1. First, we go through the elements in Y . For all d_{ij} given in Y , we reduce r_i by d_{ij} and c_j by d_{ij} . In other words, we set $r'_i = r_i - \sum_j d_{ij}$ and $c'_j = c_j - \sum_i d_{ij}$, where d_{ij} is given in Y . Next, we construct a graph. We create vertices for each of the rows R_i and columns C_j along with a source s and sink t . We draw edges from the source to each of the rows R_i with capacity equal to the adjusted row sum. That is to say, we draw the edge (s, R_i) , with $u(s, R_i) = r'_i$. Then we draw edges (C_j, t) from each of the column vertices C_j to the sink t with capacities $u(C_j, t) = c'_j$. Finally, we draw edges from all rows R_i to columns C_j with $u(R_i, C_j) = \infty$ provided that d_{ij} wasn't given in Y . If d_{ij} was given, we do not draw the edge (R_i, C_j) .

For example, suppose Y is empty. Then we can draw the graph as follows:



Claim 0.1 Consider a flow f on the graph. This flow corresponds to a solution to the matrix and all the constraints if and only if $f = \sum_i r'_i$ and is feasible.

Proof. (\Rightarrow) Suppose there is a solution $\{d_{ij}\}$ to the matrix. Then we claim that $f(s, R_i) = r'_i$, $f(R_i, C_j) = d_{ij}$, and $f(C_j, t) = c'_j$ provides a feasible flow.

It is fairly obvious that $f = \sum_i r'_i$, as we saturated all the edges leaving s . Next, we just need to show that the flow is feasible. Consider the vertex representing the row R_i . We know that $\sum_j d_{ij} = r'_i$ as we satisfy our matrix constraints. Thus, if we send r'_i flow to R_i , we can send (exactly) the d_{ij} flow necessary along the edge (R_i, C_j) (these edges have infinite capacity). Thus, we are in accord with the capacity and conservation conditions. Similarly for C_j .

(\Leftarrow) This direction is also trivial. Just reverse argument from above. If we're given a flow, we let $d_{ij} = f(R_i, C_j)$, then we argue that that is a matrix solution. If we have a flow with $f = \sum_i r'_i$, then we must be saturating all edges (s, R_i) and (C_j, t) . Therefore, we must have $\sum_j d_{ij} = \sum_j f(R_i, C_j) = r'_i$ for all i . Similarly, we have $\sum_i d_{ij} = c'_j$ for all j , and the solution is valid. ■

Thus, we can just find a solution to the matrix by finding a max flow on the graph. If the max flow f is not equal to $\sum_i r'_i$, then there is no solution. If it is, then we proceed to verify each d_{ij} as being protected or unprotected.

To verify whether d_{ij} is protected, we just need to ask the question, “can any other value work for d_{ij} ?” We can break this into two smaller questions: “Does there exist a valid solution with $d'_{ij} < d_{ij}$?” and “Does there exist a valid solution with $d'_{ij} > d_{ij}$?” If the answer to either of these questions is “yes,” then there is not a unique answer for d_{ij} . Thus, d_{ij} is protected. If the answer to both of these questions is “no,” then there is no other answer for d_{ij} , and we conclude that d_{ij} is unprotected.

Okay, so now we just need to ask these questions about all edges. As we showed in the above claim, a valid $f(R_i, C_j)$ corresponds to a valid solution for d_{ij} . Let us look at each question individually.

“Does there exist a valid solution with $f'(R_i, C_j) < f(R_i, C_j)$?” To answer this question, we find *any* s - t path¹ going through (R_i, C_j) , decrease the flow on this path by 1, and then decrease the capacity $u'(R_i, C_j) = f(R_i, C_j) - 1$.² In other words, we decrease the flow by 1 along a path going through (R_i, C_j) and then decrease the capacity of this edge such that it can only support flow $< f(R_i, C_j)$. Now, we just need to search for an augmenting path. If one exists, then there is a valid solution with $f'(R_i, C_j) < f(R_i, C_j)$.

Answering this question takes $O(m)$ time to find an augmenting path, and we need to ask this question for each of the $O(m)$ edges, for a total of $O(m^2)$ time.

“Does there exist a valid solution with $f'(R_i, C_j) > f(R_i, C_j)$?” Now, instead of forcing the flow to be less than a certain amount on this edge, we want to force it to be greater than a certain amount. Thus, we kinda want to force $f(R_i, C_j) + 1$ flow across (R_i, C_j) . In other words, we assume that $d_{ij} = x + f(R_i, C_j) + 1$ for some value of $x \geq 0$. We reduce the problem to one in which all of r'_i , c'_j , and d_{ij} are decreased by $f(R_i, C_j) + 1$. These problems are isomorphic. To accomplish this on our graph, we first reduce the flow $f'(R_i, C_j) = 0$. We then reduce $u(s, R_i)$ and $u(C_j, t)$ by $f(R_i, C_j) + 1$ and set $f'(s, R_i) = u'(s, R_i) = f(s, R_i) - (f(R_i, C_j) + 1)$ and $f'(C_j, t) = u'(C_j, t) = f(C_j, t) - (f(R_i, C_j) + 1)$. This reduction corresponds to subtracting $f(R_i, C_j) + 1$ off of r'_i , c'_j , and d_{ij} . At this point, we have a flow $f' = f - (f(R_i, C_j) + 1)$. However, this flow is not feasible as we violate conservation at R_i and C_j . We just need to search for a path $C_j \rightsquigarrow R_i$ (in the residual graph) and send 1 flow along this path (augmenting path from C_j to R_i in $O(m)$ time). If we can find such a path, then we satisfy conservation, and we have a feasible max flow. Hence, there

¹Try $s \rightarrow R_i \rightarrow C_j \rightarrow t$.

²Recall $u(R_i, C_j)$ was ∞ before.

exists $f'(R_i, C_j) > f(R_i, C_j)$, and d_{ij} does not have a unique solution. If we cannot find such a path, then we conclude that there is no feasible flow f' with $f' = f - (f(R_i, C_j) + 1)$. Thus, the answer to the question is “no,” there is no solution with $f'(R_i, C_j) > f(R_i, C_j)$.

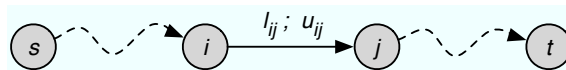
Answering this question takes $O(m)$ time to find an augmenting path, and we need to ask this question for each of the $O(m)$ edges, for a total of $O(m^2)$ time.

We have that asking all questions takes us only $O(m^2)$. Thus, the total running time of our algorithm is $O(\text{max-flow}) + O(m^2)$. Just to ground these numbers, a $p \times q$ matrix results in a graph with $n = \Theta(p + q)$ vertices and $m = \Theta(pq)$ edges.

Problem 2. To solve this problem, construct a new graph G' where each node i with a node capacity is replaced by two nodes i_1 and i_2 . All edges that previously went into the node should now point to i_1 , and all edges that previously exited the node should now exit from i_2 . Add an edge from v_1 to v_2 with the capacity $w(i)$. Finding a max flow on this graph will correctly limit the amount of flow that can pass through each node while still finding a max flow allowed by the arc capacities. In terms of worst-case complexity, this problem is only a constant factor worse since n increases by at most a factor of 2 and m increases by at most n , which we can usually assume is less than m . (If $m < n$, not all of the graph is connected, so we don't even care about the new edges that we can't get to.)

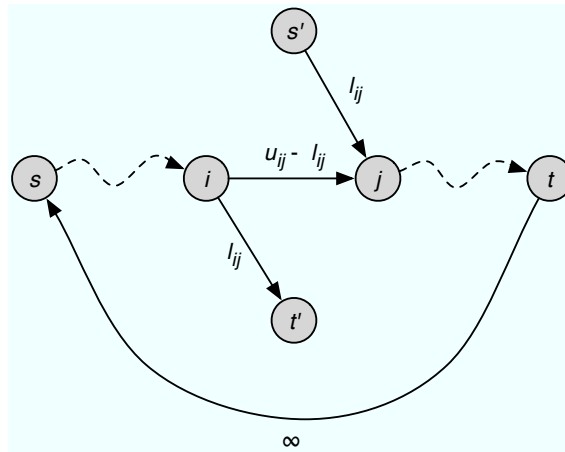
- (a) As suggested by the hint, we first construct some feasible flow on the graph G , and then construct a minimum flow using the feasible flow.

We find a feasible as follows. Note that just finding a max flow from s to t is not sufficient as that does not guarantee that the flow flows along the correct edges (to satisfy the minimums). Instead, we modify the original graph G to get a graph G' as follows. For each edge (i, j) in G , we change the capacity $u'_{ij} = u_{ij} - l_{ij}$. Then we drop the minimum capacities on all edges. Thus, in essence, we are forcing a flow of l_{ij} across each of the edges, but we have a surplus and deficit of l_{ij} at i and j respectively. What ways could we get rid of this deficit? Well, we could find a path from a surplus to a deficit node in G' . Or, we could find a path from a surplus node j to the sink t and the source s to the surplus node i . We want to solve this problem using max flows. So, we add to G' a new source s' and a new sink t' . We add edges (i, t') and (s', j) with capacities $u'_{it'} = u'_{s'j} = l_{ij}$ ³. Finally, we add an edge (t, s) (between the original source and sink) with unbounded capacity $u'_{ts} = \infty$. If we start with the graph:



We would end with the graph

³We may add more than one edge (i, t') for example, or we can add a single edge with the sums of the relevant capacities. More formally, we could say that $u'_{it'} = \sum_j l_{ij}$ and $u'_{s'j} = \sum_i l_{ij}$.



Then all we need to do is find a max flow from s' to t' . We saturate all the edges (s', j) if and only if for every node j with a surplus, there are path(s) to t and/or some other deficit node(s) with enough capacity. Thus, we just check, are all these edges saturated? If no, there is no feasible flow on the graph G' . If yes, then there is a feasible flow, AND we know what it is. All we need to do is drop all the extra edges we added, look at the flow f' we found on G' , and add the original edge minimums. For example, if we find flow of f'_{ij} across an edge (i, j) , then the flow in the original graph is just $f = f'_{ij} + l_{ij}$. Another way of thinking about this is that going from s' to t' in the graph G' is equivalent to going from s to t in the original graph, because we move the flow from (s', j) and (i, t') to the edge (i, j) .

Next, we convert the feasible flow into a minimum flow. Let our feasible flow on any edge from i to j be f_{ij} (in gross flow formulation). We construct a “reverse residual graph” by taking every edge from i to j in the original graph G , and replacing it with two edges in G' : one edge from i to j with capacity $u_{ij} - f_{ij}$, and one edge from j to i with capacity $f_{ij} - l_{ij}$. We might get two edges by this construction going from i to j , so we combine the edges by adding the capacities. So the reverse residual graph has an edge from i to j with capacity $u_{ij} - f_{ij} + f_{ji} - l_{ji}$.

Then we compute a normal maximum flow g_{ij} from t to s in G' , and add the result back into the flow in G to get a flow f' . When we add the flow back, we decompose g into $g_{ij} = g'_{ij} + h_{ij}$, where $0 \leq g'_{ij} \leq u_{ij} - f_{ij}$ and $0 \leq h_{ij} \leq f_{ji} - l_{ji}$. We add the flow back by adding g' in the forward direction and subtracting out h_{ji} in the backward direction. So the new flow from i to j in our original graph G is

$$f'_{ij} = f_{ij} + g'_{ij} - h_{ji}.$$

The inequality relationships we have are

$$\begin{aligned} l_{ij} &\leq f_{ij} \leq u_{ij} \\ 0 &\leq g'_{ij} \leq u_{ij} - f_{ij} \\ 0 &\leq h_{ij} \leq f_{ji} - l_{ji} \end{aligned}$$

Adding f_{ij} to both sides of $g'_{ij} \leq u_{ij}$ and using the fact that $h_{ji} \geq 0$ tells us that $f'_{ij} \leq u_{ij}$. Adding f_{ij} to the inequality $-h_{ji} \geq l_{ij} - f_{ij}$ and using the fact that $g_{ij} \geq 0$ gives us $f'_{ij} \geq l_{ij}$. Therefore, when we add our flow back, we still get a feasible flow.

This flow f' we compute must be a minimum flow. Otherwise, we could reduce the flow further by finding some path from t to s in G . This corresponds to finding some augmenting path in G' from t to s , suggesting that the flow we computed in wasn't a maximum flow for G' . This is a contradiction, so f' must be a minimum flow.

- (b) **Claim 0.2** *Let the lower bound on the cut capacity of an s - t cut S be defined as $L(S) = \sum_{(i,j) \in S \times T} l_{ij} - \sum_{(i,j) \in T \times S} u_{ij}$. Then the minimum value of all feasible flows from node s to t equals the $\max_S L(S)$.*

Proof. We show that for any flow, $f \geq \max_S L(S)$ and that \min flow $f \not> \max_S L(S)$, which implies that minimum value of all feasible flows from s to t equals $\max_S L(S)$.

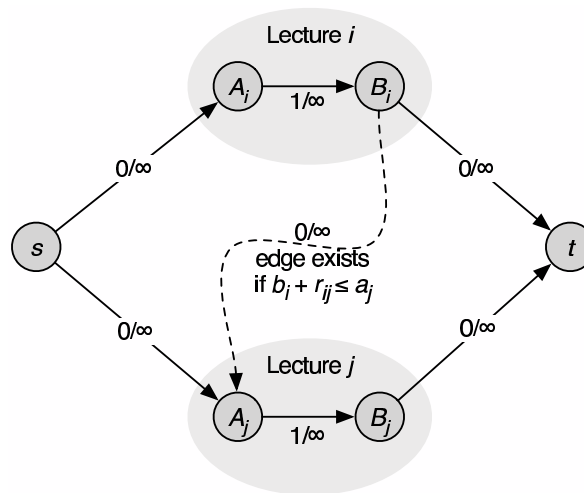
Suppose we have a flow f . Then we want to show that $f \geq \max L(S)$. Instead, we show that $f \geq L(S)$ for any S . Consider any cut S . Since the flow is feasible, we satisfy all the minimum capacities. So, the flow from S to T is at least $\sum_{(i,j) \in S \times T} l_{ij}$. Since the flow is feasible, the most that can be flowing from T to S is bounded by the maximum capacities $\sum_{(i,j) \in T \times S} u_{ij}$. Thus, the net flow across the cut is $\geq \sum_{(i,j) \in S \times T} l_{ij} - \sum_{(i,j) \in T \times S} u_{ij}$. Therefore $f \geq L(S)$.

Suppose that we have a min flow f . Assume for the sake of contradiction that $f > \max L(S)$. Then for all cuts $f > L(S)$. $f > \sum_{(i,j) \in S \times T} l_{ij} - \sum_{(i,j) \in T \times S} u_{ij}$ implies that either we are sending more flow than the minimum from S to T , or we are not sending the maximum amount back from T to S . Consider the first case. Then our modified residual graph G'_f from part (a) has extra capacity available from T to S on the residual (reverse) edges. In the second case, we also have capacity available from T to S . Since there is capacity available across ALL CUTS going from T to S , there is an augmenting path in G'_f from t to s ,⁴ and we don't have a min flow. ■

- (c) For lecture i , we create two vertices in our graph: A_i to represent the start of the lecture and B_i to represent the end. We draw an edge (A_i, B_i) for every lecture

⁴The max-flow min-cut theorem tells us that $|f| \neq u(S)$ for any S implies that there is an augmenting path in the residual graph.

i , with a minimum flow $l(A_i, B_i) = 1$ and a max flow $f(A_i, B_i) = \infty$. We create a source s and a sink t . We create edges (s, A_i) from the source to the start of every class, with unconstrained capacities $l(s, A_i) = 0$ and $f(s, A_i) = \infty$. Similarly, we add edges (B_i, t) from the end of every lecture to the sink, with unconstrained capacities $l(B_i, t) = 0$ and $f(B_i, t) = \infty$. Finally, we introduce edges from the end of lecture i to the start of lecture j if it is possible to commute from lecture i to lecture j . That is, (B_i, A_j) exists if $b_i + r_{ij} \leq a_j$. Again, these edges have unconstrained capacity. The graph looks like:



Now we just solve for the min flow. We claim that the min flow is the minimum number of students necessary to attend all the lectures. Basically, each unit of flow represents a student. We argue correctness by showing that our graph has the same constraints as the students. In particular, a student can only attend two lectures i and j if the commute time allows it. Similarly, we only have an edge in the graph allowing flow between lectures i and j if the commute times allow it. A student can choose any lecture to be the first one he goes to (modeled by edges from s to all lecture starts), and he can choose any lecture to be his last (modeled by edges from lecture ends to t). Finally, the only other constraint we have is that at least one student must be in attendance at each lecture. By creating two vertices for each lecture and having an edge with min capacity of 1 between lecture start and end, we guarantee at least one unit of flow across the lecture—that is, we guarantee that at least one student sits through the entire lecture.

Problem 3. (a) Given a graph G and its min-cost circulation 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 costs using the Bellman-Ford

algorithm. The reduced edge costs are always non-negative and no negative cost cycles exist in G_f . If an edge's reduced cost is strictly positive, the flow on the edge is 0.

Increasing the cost of an edge $e = (u, v)$ 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 $f(e)$ on it if its reduced cost is 0. We perform a max-flow on the subgraph G_f without the edge 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 this max-flow. If $|f'|$ is $f(e)$, 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 an 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 u 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.

Our algorithm takes one Bellman-Ford computation, and one maximum flow computation. The $O(mn \log n)$ time taken by max-flow dominates the running time.

- (b) We can design a cost-scaling 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 the k th phase, we have a min-cost flow on the graph with the 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 up to m unit changes in edge costs. There are $O(\log C)$ phases. So the running time of the algorithm is $O(m^2 n \log n \log C)$.

Problem 4. Define the following sets:

- P = positions on the results page
- C = clicked old results
- U = unclicked old results
- N = new results

and let $\pi: (C \cup U) \rightarrow P$ map old results to their positions in the old list. Then define the network $G = (V, E)$ with edge capacities and costs as follows:

$$\begin{aligned}
 V &= \{s, t, x, y, z\} \cup C \cup U \cup N \cup P \\
 E &= \{(s, x), (s, y), (s, z), (z, y)\} \cup \{x\} \times C \cup \{y\} \times N \cup \\
 &\quad \{z\} \times U \cup (C \cup N \cup U) \times P \cup P \times \{t\} \\
 u(i, j) &= \begin{cases} |C| & (i, j) = (s, x) \\ k & (i, j) = (s, y) \\ n - |C| - k & (i, j) \in \{(s, z), (z, y)\} \\ 1 & \text{otherwise} \end{cases} \\
 c(i, j) &= \begin{cases} v_i - p_{\pi(i),j} & (i, j) \in (C \cup U) \times P \\ v_i & (i, j) \in N \times P \\ 0 & \text{otherwise.} \end{cases}
 \end{aligned}$$

It is easy to check that an integer min-cost max-flow of this graph corresponds to a solution to our problem. Such a solution can be computed in polynomial time using one of the algorithms from class.