

CS 170 Spring 2008 - Solutions to Homework #10

Problem 7.10 (8 points)

Run the max-flow algorithm as in the paradigm of figure 7.6 in the text-book, iteratively find a flow and update the residual graph. We end up with a maximum flow of 13 units, which corresponds to a min-cut $\{S, C, F\}$ and $\{A, B, D, E, G, T\}$.

Problem 7.12 (8 points)

Multiply the first inequality by y_1 , the second by y_2 (with $y_1, y_2 \geq 0$) and add to obtain

$$(y_1)x_1 + (2y_2 - y_1)x_2 + (-y_2)x_3 \leq y_1 + y_2$$

To obtain the tightest constraint on the given objective function, we would like to obtain y_1 and y_2 , subject to

$$\begin{aligned} \min \quad & y_1 + y_2 \\ & y_1 \geq 1 \\ & 2y_2 - y_1 \geq 0 \\ & -y_2 \geq -2 \\ & y_1, y_2 \geq 0 \end{aligned}$$

This is the *dual* of the given linear program. Solving gives $y_1 = 1, y_2 = 1/2$. Plugging these values into the above inequality gives $x_1 - 1/2x_3 \leq 3/2$ which implies $x_1 - 2x_3 \leq 3$ since $x_3 \geq 0$. Finally, it remains to note that the given solution has value $3/2$ and is hence optimal.

Problem 7.14 (10 points)

Let Joey choose the two possible options with probabilities x_1 and x_2 . Then the gains to Joey (or losses to Tony) corresponding to the three strategies of Tony are $\{2x_1 - x_2, -2x_2, -3x_1 + x_2\}$. Of course, Tony will choose the option that minimizes his loss and hence Joey will try to maximize the minimum of

these three options. Thus, the optimal strategy for Joey can be obtained from the solution of the linear program

$$\begin{aligned}
 \max z, \quad & \text{subject to} \\
 z & \leq 2x_1 - x_2 \\
 z & \leq -x_2 \\
 z & \leq -3x_1 + x_2 \\
 x_1 + x_2 & = 1 \\
 x_1, x_2 & \geq 0
 \end{aligned}$$

Similarly, if Tony chooses his three strategies with probabilities y_1, y_2 and y_3 , then Joey gains $\{2y_1 - 3y_3, -y_1 - 2y_2 + y_3\}$ by his two strategies and Tony tries to minimize the maximum gain. The corresponding linear program is

$$\begin{aligned}
 \min z, \quad & \text{subject to} \\
 z & \geq 2y_1 - 3y_3 \\
 z & \geq -y_1 - 2y_2 + y_3 \\
 y_1 + y_2 + y_3 & = 1 \\
 y_1, y_2, y_3 & \geq 0
 \end{aligned}$$

The solution to the first linear program is $x_1 = x_2 = 1/2$. For the second linear program $y_1 = 0, y_2 = 2/3, y_3 = 1/3$. The value of the game, which is the value of z in both the programs, is -1 i.e. Joey will lose the sale of at least one dozen pizzas to Tony. (Basically, giving free sodas is a good idea!)

Problem 7.24 (20 points total, 5 points each)

- a) D-F-B-E-A-H-C-I is an alternating path.
- b) Let p be an alternating path of length $2m + 1$ with respect to a matching M . Let p_M denote the matching edges in p and let p_{E-M} denote the non-matching edges. Note that $M - p_M$ and p_{E-M} form two matchings of sizes $|M| - m$ and $m + 1$ respectively with no vertices in common. Hence, M cannot be a maximum matching since $(M - p_M) \cup p_{E-M}$ gives a matching of size $|M| + 1$.

For the other direction, let M and M' be two matchings such that $|M| < |M'|$. Since M' is larger, the graph must contain at least one vertex u covered by M' but not by M . Starting from u , we construct a path by alternately following edges from M' and M till we reach a vertex v which has an edge from only one of the matchings. If v is covered by M' but not by M , we have an alternating path. If v is only covered by M , then the path constructed so far has an equal number of edges from M and M' . We can then delete these edges and continue the argument on the rest of the graph. Since we always have more vertices covered by M' than by M , we are guaranteed to find an alternating path.

- c) We start a BFS from an uncovered vertex, with the modification that we only look at non-matching edges at the odd levels of a BFS tree and only at the unique matching edge incident on the vertex at the even levels of the tree. We stop as soon as we reach another uncovered vertex.

If the BFS started from an uncovered vertex does not reach any other uncovered vertex, this means that there is only one uncovered vertex reachable (through alternating paths) from *all* the covered vertices in the tree and they cannot be in any alternating path. Hence, we delete all the vertices in the tree and continue on the rest of the graph. Since we examine each edge at most once, the algorithm takes $O(|V| + |E|)$ time.

- d) We start with $M = (u, v)$ for any edge (u, v) . We then find an alternating path, and if one exists we obtain a larger matching from M and the path p (see part (b)). We continue this till no alternating path can be found, which means M is a maximum matching. Since the size of the matching increases by 1 edge each time we find a matching, and the maximum size is $|V|/2$, the time taken is $O(|V| \cdot (|V| + |E|)) = O(|V| \cdot |E|)$.

Problem 7.29 (16 points total, 8 points each)

(a) We introduce 0 – 1 valued variables y_i that indicate whether investor i is used or not and 0 – 1 valued variables x_i that indicate whether actor i is used or not. The integer linear program follows:

$$\begin{aligned} \max \quad & \sum_{j=1}^m y_j p_j - \sum_{i=1}^n x_i s_i \\ \forall j \quad & \sum_{i \in L_j} \frac{x_i}{|L_j|} \geq y_j \end{aligned}$$

The constraints ensure y_j will be 1 only if all actors required by producer j are included.

(b) Consider the optimal of the linear program above, when variables are relaxed to be in the interval $[0, 1]$. Note that for an optimal solution we must have all the constraints satisfied with equality, as increasing the y_j 's only increases the value of the objective. That is:

$$\forall j \quad \sum_{i \in L_j} \frac{x_i}{|L_j|} = y_j$$

When we replace this in the objective function, we obtain:

$$\max \sum_{j=1}^m \sum_{i \in L_j} \frac{x_i}{|L_j|} p_j - \sum_{i=1}^n x_i s_i$$

This is a linear function in the x_i 's. Hence it is maximized at one of the vertices of the feasible region for x_i , i.e. at an integral solution.

Problem 7.31 (20 points total, 5 points each)

- a) Suppose the sequence of updates alternates path $S - A - B - T$ with path $S - B - A - T$. With each update we send a single unit of flow from s to t . Hence, we will need 1000 updates to achieve the optimal flow.
- b) It suffices to modify the update function used in Dijkstra as follows:

$$\text{procedure update}((u, v) \in E) \\ \underline{\text{dist}}(v) = \max\{\underline{\text{dist}}(v), \min\{\text{dist}(u), c(u, v)\}\}$$

Distances are initialized as in Dijkstra but now $\underline{\text{dist}}(v)$ represent the capacity of the fattest path from s to v . We can argue the correctness of this algorithm by showing that when a vertex v is removed from the priority queue its distance value is the correct capacity of the fattest path to the vertex. We do this by induction on the vertices in order of their removal from the priority queue. Suppose this is not the case and the capacity of the fattest path to v is not that given by the last update to $(\underline{\text{dist}})(v)$ performed by edge (u, v) . Then, the fattest path to v must use another edge $e = (w, z)$ out of the set of current nodes. But the fattest path to z through (w, z) must be thinner than the current value $\underline{\text{dist}}(z)$, which is less than $\underline{\text{dist}}(v)$. Hence, any path through (w, z) must yield a thinner path for v . Hence, our algorithm is correct.

- c) Given a maximum flow, consider the following algorithm which decomposes it into the sum of flow along $|E|$ paths: let e be the edge carrying the least non-zero amount of flow. Remove all flow through e by picking any path from s to t through e . This is always possible, as all other edges in the network are carrying an equal or greater flow. Repeat this procedure on the new flow obtained until all flow has been eliminated. At every iteration we make the flow through one edge to be 0. Hence, this decomposition can yield at most $|E|$ paths.
- d) At every iteration the fattest path strategy picks the algorithm which can carry the most flow from s to t . Because the optimum flow can be decomposed in $|E|$ paths, by c), at least one of these paths must carry at least $\frac{F}{|E|}$ units of flow. Hence, the greedily picked path must achieve at least $\frac{F}{|E|}$ units of flow. But this is true for any flow through the network, as the argument c) always applies. Hence, if we let F_t denote the flow remaining after the t^{th} iteration, then $F_{t+1} < F_t(1 - \frac{1}{|E|})$. Hence:

$$F_t < F \left(1 - \frac{1}{|E|}\right)^t < F(e^{-1/|E|})^t$$

showing that $|E| \log F$ iterations suffice, as required.