CS170–Fall 2012 — Solutions to Homework 10

Ben Augarten, section 106, `cs170-bo`

November 25, 2012

1. If we are Joey, then we have to pick a strategy (x_1, x_2) to maximize the minimum of $\{2x_1 - x_2, 0x_1 - 2x_2, -3x_1 + x_2\}$. Let $z = \min\{2x_1 - x_2, 0x_1 - 2x_2, -3x_1 + x_2\}$
   max z
   $z \leq 2x_1 - x_2$
   $z \leq -2x_2$
   $z \leq -3x_1 + x_2$
   $x_1 + x_2 = 1$
   $x_1, x_2 \geq 0$
   The optimal strategy that Joey can take is $(.5, .5)$, resulting in a value of 1.

   Likewise, if Tony has to pick a strategy (y_1, y_2, y_3), then he wants to minimize the maximum of $\{2y_1 - 3y_3, -y_1 - 2y_2 + y_3\}$, so let $z = \max\{2y_1 - 3y_3, -y_1 - 2y_2 + y_3\}$
   $\min z$
   $z \geq 2y_1 - 3y_3$
   $z \geq -y_1 - 2y_2 + y_3$
   $y_1 + y_2 + y_3 = 1$
   $y_1, y_2, y_3 \geq 0$
   Tony's optimal strategy therefore is $(0, \frac{2}{3}, \frac{1}{3})$, yielding value -1.
   So the value of the game is -1.
   Joey's optimal strategy: $(.5, .5)$
   Tony's optimal strategy: $(0, \frac{2}{3}, \frac{1}{3})$

2. So, lets connect a super source to the left side of the bipartite graph and a supersink to the right side (i.e. find the max flow through the bipartite graph). We will use this to find a minimum cut S in the network. Now we have partitioned our vertices again, those that are in the cut and on the left side, $L_S = L \cap S$, those that aren't $L_N = L - S$, those that are in the right side and in the cut, $R_S = R \cap S$, and those on the right side that aren't $R_N = R - S$. We can say that the minimum cover is the set union of $L_N \cup R_S \cup R_{L_1}$, where R_{L_1} are all the vertices in R_N that have neighbors in R_S. This set certainly covers all of the edges that have endpoints in either L_N of R_S, and the only remaining edges are those that have endpoints in L_S and R_N, the ones that go across the cut. I.e. an edge must go across the cut – be in R_{L_1} – or have an endpoint in either R_S or L_N.
   But is this the smallest vertex cover? Well, the size of the cut must include $|L_N|$ because there are paths from the source to these nodes and all of the vertices in R_S because they have paths to the sink. Additionally, the cut must include the edges that are coming out of the cut vertices on the left side, so all $(u, v) \in L_S \text{x} R_N$,
   so min cut$= |L_N| + |R_S| + |(u, v) \in L_S \text{x} R_N|$, so
   min cut$\leq |L_N| + |R_S| + |R_{L_1}| = |C|$
   And the minimum cut equals the maximum flow that we can pass through the graph. The maximum flow must be at least the size of the vertex cover because the maximum flow through a bipartite graph cannot use a vertex on the left or right more than once to pass flow through. Therefore, any vertex cover must have at least one of these vertices in the cover in order for it to be valid, otherwise the edge corresponding to the flow wouldn't be in the cover. Therefore, for every unit of flow there must be at least one more vertex in the cover, so the max flow, or min cut, must be greater than or equal to $|C|$, and we just showed that it also has to be less than or equal to $|C|$ so our solution is optimal.

3. (a) We will make inclusion variables for both actors and investors. These variables are 0 when the actor/investor is excluded and 1 when they are included. The one for actors is a_i and the one for investors is b_j. We want to maximize the investment of all of the actors we include minus the cost of all of the actors we include. This is easy to express:
$\max \sum b_i p_i - \sum a_i s_i$
Our constraints are more subtle. First, before the problem runs we have to compute the set of investors that each actor belongs to. That is, the set of investors that requires actor i in order to invest, we call that C_i. Our first constraint is that the total benefit of investment for any C_i must be at least that of the cost to hire the actor.
$\forall 1 \leq i \leq n \sum_{j \in C_i} (b_j p_j) \geq a_i s_i$
Our next constraint is for the inclusion constraint that for each investor, the money we get from them must be at least that of the actors we hire MINUS the benefit of all of the other investors we get because of the inclusion of each actor, counted once. Note that C_{L_j} is the union of all C_i such that $i \in L_j$. The subtraction makes the constraint valid for intersections of actors without discounting that same investor twice.
$\forall 1 \leq j \leq m,\ b_j p_j \geq \sum_{i \in L_j} (a_i s_i) - \sum_{k \in C_{L_j}/\{j\}} b_k p_k$
Now we need to show that if we include an investor, then we have to include all of the actors:
$\forall 1 \leq j \leq m,\ b_j \leq a_i \forall i \in L_j$
And our last constraint is just that every a_i,b_j must be 0 or 1,
$\forall 1 \leq i \leq n, a_i \in \{0, 1\}$
$\forall 1 \leq j \leq m, b_j \in \{0, 1\}$

   (b) Now lets relax the constraints, such that a_i and b_j don't strictly have to be integer. I will show that there must be an integral solution that IS optimal. First, lets assume the opposite, that there exists a non-integral solution that is optimal. Lets assume that there is a non-integral a_i. If $s_i < \sum_{j \in C_i} p_j$ then we want to increase a_i and increase the corresponding b_j, which must be less than a_i to satisfy the constraints because then we earn more money from investors versus the amount we spend on the actor, thus increasing our objective function. If instead $s_i > \sum_{j \in C_i} p_j$ then we want to decrease a_i and decrease the corresponding b_j because we are spending more money than we receive from the investors and we increase our objective by decreasing both a_i and its b_j's. If they are equal, we can increase or decrease a_i and its b_j's arbitrarily without changing the objective function, so we can achieve an integer solution in this case by increasing or decreasing the terms and in the other cases only an integral solution is possible.

Likewise, if the non-integral solution is for a b_j, and there are no a_i's that are non integral, then we can increase b_j to 1 without fear of violating any constraints, only increasing our objective function, so obviously this case cannot exist – if all a_i's are integral, then so too must the b_j's. Therefore, while a fractional solution can be optimal (third case above), in all cases, an integer solution will be optimal.

4. (a) If we choose $S \to A \to B \to T$ and then $S \to B \to A \to T$ and then the first again, so on and so forth, where we route exactly 1 unit of flow per iteration, we can continually undo and redo the middle path, taking over 1000 iterations.

   (b) All we have to do to change Dijkstra's to work for fattest path is change sums of distances to mins, the lengths of edges to capacities, and the minimization of distance to maximization of fatness.
   Make the priority Q based on fatness, set all the fat values to 0 except the start which is ∞. While the Q isn't empty, pop the fattest vertex off, then for all edges that leave that vertex if that path is of greater fatness to the target node (for edge (u,v) if fat(v) $<$ min(fat(u),capacity(u,v))), then update that vertex's fatness and update its value in the Q.
   Proof of correctness is essentially the same as Dijkstra's – at every iteration of the while loop, every vertex that is not in the Q are such that their fatness is correct. This doesn't change after an iteration in the while loop because using the now fattest vertex we update all of the remaining vertexes to be constrained by that vertex's value, and add the fattest vertex, which cannot possibly have a fatter path coming to it because fatness is a minimization.

   (c) We have a maximum flow in G from a source to a sink. We can trace each path back from the sink. Each time we trace the path back, we can remove the constraining edge, the one that limits the entire path. We can only do this $|E|$ times so there can be a maximum of $|E|$ paths.

   (d) If we have flow F and number of paths $|E|$, then there must be a path with flow $\frac{F}{|E|}$. So at iteration n the fattest path must be at least $\frac{F_n}{|E|}$, so we can remove it. Then the next fattest path, that for iteration $n+1$ cannot be greater than $F_n - \frac{F_n}{|E|}$ because we removed $\frac{F_n}{|E|}$ flow from the graph. So we get the inequality $F_{n+1} \leq F_n(1 - \frac{1}{|E|})$, which implies that $F_t \leq F(1 - \frac{1}{k})^t$. This is the same as greedy set cover in the book, so we can use the same logic and bound, $1-x \leq e^{-x}$ for all x, so $F_t \leq Fe^{-t/|E|}$, and at $t = |E| \ln(F)$ we are done, so it will terminate in at most $O(|E| \ln(F))$ iterations.