CS170 Fall 2013 Solutions to Homework 10

Zackery Field, section Di, 103, `cs170-fe`

November 15, 2013

1. (10 pts.) Pizza Predicament

The pizza business in Little Town is split between two rivals, Tony and Joey. They are each investigating strategies to steal business away from the other. Joey is considering either lowering prices or cutting bigger slices. Tony is looking into starting up a line of gourmet pizzas, or offering outdoor seating, or giving free sodas at lunchtime. The effects of these various strategies are summarized in the following payoff matrix (entries are dozens of pizzas, Joey's gain and Tony's loss)

		Tony		
		Gourmet	Seating	Free Soda
Joey	Lower Price	+2	0	-3
	Bigger slices	-1	-2	+1

For instance, if Joey reduces prices and Tony goes with the gourmet option, then Tony will lose 2 dozen pizzas worth of business to Joey. What is the value of this game, and what are the optimal strategies for Tony and Joey?

We can first determine an LP representation of this problem. Then allow Joey to announce the best strategy based on the best choice Tony can make. By duality of construction, this will reveal the value, V, of the game. In order to show duality, we can ask Tony to reveal his optimal solution first, and show that the value of the game will be the same.

Let $X = (x_1, x_2)$ be the probabilities that Joey chooses lower prices, and larger slices respectively; $Y = (y_1, y_2, y_3)$ be the probabilities that Tony chooses gourmet, seating, and free soda respectively.

Continued on Page 6

2. (15 pts.) Hollywood Hiring

A film producer is seeking actors and investors for his new movie. There are n available actors; actor i charges s_i dollars. For funding, there are m available investors. Investor j will provide p_j dollars, but only on the condition that certain actors $L_j \subseteq \{1, 2, \ldots, n\}$ are included in the cast (all of these actors L_j must be chosen in order to receive funding from investor j).

The producer's profit is the sum of the payments from investors minus the payments to actors. The goal is to maximize this profit.

(a) Express the problem as an integer linear program in which which the variables take on the values $\{0, 1\}$. Let a_i represent whether or not a particular actor is chosen $\{0, 1\}$, and v_i represent whether or not a particular investor is chosen $\{0, 1\}$.

$$\begin{aligned}
\text{max: } & \sum_{j}^{m} v_j p_j - \sum_{i}^{n} a_i s_i \\
v_j &\leq a_i, \forall a_i \in L_j\{a_1, \ldots, a_i, \ldots, a_n\} \\
s_i, p_j &\geq 0 \\
a_i &\in \{0, 1\} \\
v_j &\in \{0, 1\}
\end{aligned}$$

(b) Now relax this to a linear program, and show that there must in fact be an integral solution optimal solution. (as in the case, for example, with maximum flow and bipartite matching).

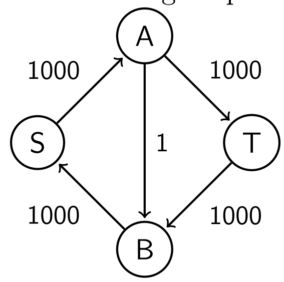
$$\begin{aligned}
\text{max: } & \sum_{j}^{m} v_j p_j - \sum_{i}^{n} a_i s_i \\
v_j &\leq a_i, \forall a_i \in L_j\{a_1, \ldots, a_i, \ldots, a_n\} \\
s_i, p_j &\geq 0
\end{aligned}$$

I will apply the same LP as in (a) and simply relax the conditions on a_i, and v_j. It can be shown by cases that there must be an integral optimal solution. The strategy is to assume a non-integral solution has been found and to then show how making that solution integral will not decrease the objective function.

Continued on Page 7

3. (20 pts.) Fast Flow

Consider the following simple network with edge capacities shown



(a) Show that, if Ford-Fulkerson algorithm is run on this graph, a careless choice of updates might cause it to take 1000 iterations. Imagine if the capacities were a million instead of 1000!

For all edges in a path from $s-t$ each iteration of the Ford-Fulkerson algorithm updates the max flow based on the edge that has the minimum value of $capacity(edge) - current\_flow(edge)$. In the case of this graph, if we first update the path S,A,B,T, and then update the path S,B,A,T, then the max flow will only increase by one each time. If we continue to alternate between these two paths then the maximum flow will only be found after 1000 iterations. That is because the max flow will increase by one as the path $A - B$ is traveled forward, and then increased by one on the next iteration as the path $B - A$ is traveled instead.

We will now find a strategy for choosing paths under which the algorithm is guaranteed to terminate in a reasonable number of iterations. Consider an arbitrary directed network $(G = (V, E), s, t, c_e)$ in which we want to find the maximum flow. Assume for simplicity that all edge capacities are at least 1, and define the capacity of an $s-t$ path to be the smallest capacity of its constituent edges. The fattest path from s to t is the path with the most capacity.

(b) Show that the fattest $s-t$ path in a graph can be computed by a variant of Dijkstra's algorithm.

Since s is the starting vertex, Dijkstra's algorithm usually assigns the distances from all edges away from s to infinity and the distance to s to be zero. This is so that the shortest path can be found from s to all other vertices. A useful variation would be to preprocess all vertices, not s, marking their distances 0. The distance definition in this case is actually a representation of the fat-ness of the path to that node.

Continued on Page 8

4. (15 pts.) Evacuation Emergency

- There are $|V|$ rooms in the building.
- The start room s. You wish to know the maximum number of people who can leave this room safely in T seconds.
- Rooms are connected to each other by one-way hallways. You have $|E|$ hallways total, with the hallway (u, v, c_{uv}, t_{uv}) connecting room u to room v having a maximum capacity of c_{uv} people and transit time t_{uv} seconds. At every second, this hallway can only send out c_{uv} from room u, and it takes those people t_{uv} seconds to make it to room v. Notice, this means that a hallway may contain up to $c_{uv} \times t_{uv}$ people in it at any moment in time. You may also assume the rooms can hold an arbitrarily large number of people who can wait in the room without using a hallway.

You may assume that the capacities, transit times, and T are all integers. You want to know the maximum value of N such that if N people begin at s at time 0, they can all reach t in at most time T using the hallways. Given the rooms V , hallways E , and start and end rooms, show how maximum flow in a modified graph can be used to solve this problem Hint: Consider creating T copies of each room.

Construct a graph G with $T * |V|$ vertices that represent each of the rooms. The reason for making T copies of each of the rooms and hallways is so that each representation of the $|V|$ rooms is that room at some time $t, 0 \leq t \leq T$. There are also T copies of both s and t. For some room v, the state of the room at time t is represented by the vertex v_t in the graph.

The transit time for some maximum group size c_{uv} from room u to room v, t_{uv} is positive. This means that there is a constraint on the capacities of all edges that connect rooms at the same time label, or with a past time label. For example, two rooms u_a and v_b can only have a nonnegative person transfer capacity if $b - a \leq t_{uv}$. This constraint ensures that people will only move forward in time, as defined by transit times t_{uv}.

The remaining constraint is that there can only be $c_{uv} \times t_{uv}$ in a given hallway between u, and v. The careful constraint on person transfer capacities based on transit time also contrains the number of people that will be in a hallway at any one time to $c_{uv} \times t_{uv}$. To illustrate this fact, imagine that there are two rooms u and v that have a transit time $t_{uv} = a$ and capacity $c_{uv} = b$, we know that the maximum capacity of the hallway is $a \times b$. At time $t = 0$ allow b people to enter the hallway from u_0 only to emerge at v_a. Do this again at time $t = 1$, $u_1 \rightarrow v_{a+1}$. Continue this process through u_{a-1}. At time state u_a, when we release b more people to travel along hallway $u - v$, the first group of b people has arrived at v_a, leaving only $a \times b$ people in the hallway. This shows that there are at most $a \times b$ people in the hallway at any time with this capacity contraint.

Continued on Page 9

5. (20 pts.) Directed Decomposition

Consider a directed acylcic graph. We define a path to be starting at any node u and traversing 0 or more edges of the DAG and ending at an arbitrary node v (if we travel through 0 edges, we have $u = v$). A path P consists of the set of nodes that we touch between u and v (inclusive). We wish to find the minimum number of paths such that every vertex is in exactly one of these paths. Show how maximum matching in an appropriate bipartite graph can be used to solve the problem of determining the minimum number of paths needed to cover a DAG.

In order to apply bipartite matching to the DAG we must first construct a bipartite graph that represents the DAG appropriately. Let U and V be the two sets of vertices of the bipartite graph that are initially empty, but will contain all of the vertices of the given DAG. Start a BFS at the root of the DAG, and let the root vertex be called v_0. Add v_0 to U. Let a vertex i connected to v_0 by an edge be called v_{1_i}. Add each of these v_1 vertices to V. Continue the BFS search and at each iteration (level in the DAG) add all even level vertices v_{even} to U and all odd level vertices v_{odd} to V. This construction of U and V does not gurantee a bipartite graph, however. In order to ensure a bipartite graph, the BFS search must only account for vertices connected by tree edges. All cross and forward edges should also be removed in the bipartite representation. This will ensure that no vertex in V (or U) is connected by an edge to another vertex in V (or U).

After the bipartite graph is constructed we can create arbitrary sink and source nodes, s and t. Create edges from s to all vertices in U, and create edges from all vertices in V to t. To ensure that there are no sinks, reverse (track the reversals) the direction of all edges that connect V to U. We can apply a maximum flow to this $s - t$ graph, and if we are careful about the application of the flow, it will return the maximum matching of the graph. It will be shown that this maximum matching defines the minimum number of paths necessary to cover the graph.

Start by assigning a capacity of 2 to all of the edges in the $s - t$ graph. All paths take on the form $s - u - v - t$ for some $u \in U$, and $v \in V$. If one of these paths is selected to be included in the maximum flow, then the maximum flow increases by exactly 2. The choice of 2 comes from the fact that each vertex will have at most two edges in the path cover. Since two units of flow can pass through a vertex in U and vertex in V, the maximum flow represents the number of matches between vertices in U and V. Note, only one unit of flow can pass through any given edge. This maximum matching represents the minimum number of paths necessary to cover the DAG.

Extra space for Problem 1

Continued from Page 1

Joey will try to maximize his defensive strategy by choosing a strategy that will be optimal against the optimal startegy that Tony will choose.

$$\text{Joey choose:}(x_1, x_2) \text{ that maximizes } min\text{:}\{2x_1 - x_2, -2x_2, -3x_1 + x_2\}$$

$$z = min\{2x_1 - x_2, -2x_2, -3x_1 + x_2\}$$

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

Alternatively, Tony will try to maximize his defensive strategy by choosing a strategy that will be optimal against the optimal startegy that Joey will choose.

$$\text{Tony choose:}(y_1, y_2, y_3) \text{ that minimizes } max\text{:}\{2y_1 - 3y_3, -y_1 - 2y_2 + y_3\}$$

$$w = max\{2y_1 - 3y_3, -y_1 - 2y_2 + y_3\}$$

$$\begin{aligned} &\min w \\ -2y_1 + 3y_3 + w &\geq 0 \\ y_1 + 2y_2 - y_3 + w &\geq 0 \\ y_1 + y_2 + y_3 &= 1 \\ y_1, y_2, y_3 &\geq 0 \end{aligned}$$

The optimal solution to the upper LP is $(1/3, 2/3)$ with $z = -4/3$. The optimal solution to the dual LP on the bottom is $w = -4/3$, proving that the solution $-4/3$ is optimal.

Extra space for Problem 2

Continued from Page 2

Take the case where a solution has been found where some v_j is non-integral, and all a_i's are integral (at least all a_i in L_j). By the first constraint v_j can be increased as long as it is below each of the individual a_i in its L_j set. So v_j can be increased to the integral value of $v_j = 1$, since all a_i are known to be 1. By the defined objective function, this increase in v_j without an increase in any a_i will result in an increased solution value.

Now take the case where both some a_i is non-integral in the given solution. In this case, there is some portion of actors salary being paid, specifically $a_i * s_i$. And yet, this portion of an actor does not satisfy the help to satisfy the L_j list constraint for any invenstor. This means that for the portion of the actor's salary being paid, there is not return on investment from any invenstor, v_j, whose list, L_j, contains a_i. Therefore, a more optimal solution can be found by either removing portion of actor that is being paid $a_i \to 0$, or by fully hiring the actor in order to satisfy some investor's list, $a_i \to 1$.

If there is some provided solution that includes some non-integral a_i and some non-integral v_j, the arguments listed above can still be applied. This covers all possible non-integral cases and shows that if there is any variable that has a non-integral value, then a more optimal solution can be found by changing these variables to integral values.

Extra space for Problem 3

Continued from Page 3

(b) cont. Instead of updating each node with minimum distance to that node, as with the usual Dijkstra implementation, update each vertex with the maximum minimum capacity to that vertex. In practice, all nodes in the graph will start at zero and then increase based on the maximum capacity to them. This will continue until all paths to t have been updated. The vertex t will contain the information about the maximum capacity up to t. Note, the maximum capacity path to t must contain edges whose capacities are at least the maximum capacity labeled at t. To reconstruct the path $s - t$ with maximum capacity, simply traceback the path from t to s based on the largest capacity at each step (breaking ties arbitrarily). This maximum capacity path from $s - t$ is the fattest path, by definition.

(c) Show that the maximum flow in G is the sum of the individual flows along at most $|E|$ paths from s to t.

By the max-flow min-cut theorem the maximum flow through G is defined as the minimum capacity cut of G. There are at most $|E|$ paths total from s to t in G. We can prove this by constructing a graph that has minimal edges for each path. In order to add a unique edge to the directed $s - t$ flow, we will need to add a new vertex. This new vertex will require two edges to connect it to the previous vertex in the flow and then to the next vertex in the flow. This added vertex will only contribute one extra path for the two edges that it added. Since this is the minimal edge addition technique, this shows that the number of possible paths whose flows have to be summed to find the maximum flow is at most $|E|$.

(d) Now show that if we always increase flow along the fattest path in the residual graph, then the Ford-Fulkerson algorithm will terminate in at most $O(|E| \log F)$ iterations, where F is the size of the maximum flow.

The claim in the book regarding set cover states that $k \ln n$ sets will be used in the greedy scheme where k sets would have been optimal and there are n elements. In this case, there are $|E|$ paths that are necessary to determine the max-flow (shown in (c)), or $|E|$ elements that are necessary to be covered. With each iteration of the algorithm the fattest path is chosen. And the optimal coverage (k) is analogous to the max-flow$= F$

Using the bound described in the book for the repeated application of the fattest path greedy choice we get

$$e_{t+1} = e_t - \frac{e_t}{F} \Rightarrow e_t \leq |E| e^{-t/F}$$

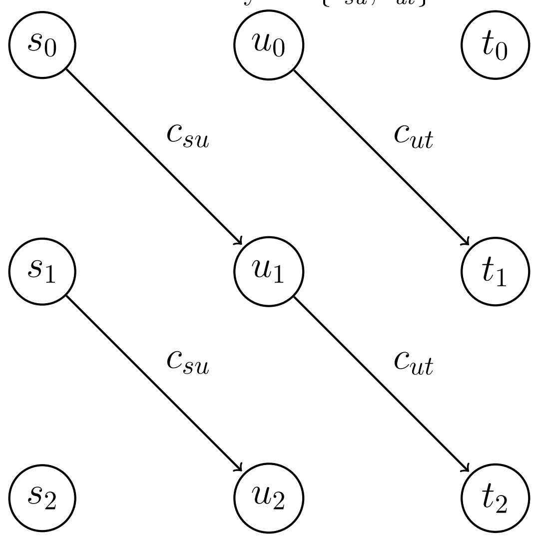
where e_t is the number of paths left to be covered at time t.

This shows that at time $t = |E| \log F$ there are at most one paths left to be covered.

Extra space for Problem 4

Continued from Page 4

Here is an example graph where there is a start room s, one intermediate room u, an end room t, and an exit time of $T = 3$. In this graph there are only unit transit times. By inspection, this example only has one possible path to reach the exit in time and the number of people that can exit is defined by $\min\{c_{su}, c_{ut}\}$



To find the maximum flow of people through the rooms at time T, you can simply run the max-flow algorithm on the constructed graph with $T * |V|$ vertices, and capacities defined by time and space contraints.

Proof: The capacities on the graph constrain the number of people that are allowed to pass and when, as defined by space and time constraints. The defined capacities do not allow people to pass through a hallway unless they have sufficient time and space to do so. Since the capacities take into account the time and space constraints of movement, the maximum flow of people through the school is simply defined by the max-flow through the graph.