CS 170 Homework 8

Due **3/28/2022, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 To Infinity and Beyond

For any vector $\vec{x}$, the infinity norm $||\vec{x}||_\infty$ is the element of $\vec{x}$ with the largest magnitude. Given a matrix A of size $m \times n$ and vector $\vec{b}$ of length m , we want to maximize $||\vec{x}||_\infty$ subject to $A\vec{x} \leq \vec{b}$ and $\vec{x}_i \geq 0$, $\forall i \in [n]$. Provide an algorithm to solve this problem.

Hint: Linear programming

Solution: Note that $\vec{x}$ is of length n. For every index i from 1 to n, solve the following linear program:

$$\text{Maximize } \vec{x}_i$$
$$\text{s.t.} A\vec{x} \leq \vec{b}$$

Return the maximum solution out of the n linear programs.

3 Flow vs LP

You play a middleman in a market of n suppliers and m purchasers. The i-th supplier can supply up to $s[i]$ products, and the j-th purchaser would like to buy up to $b[j]$ products.

However, due to legislation, supplier i can only sell to a purchaser j if they are situated at most 1000 miles apart. Assume that you're given a list L of all the pairs (i, j) such that supplier i is within 1000 miles of purchaser j. Given $n, m, s[1..n], b[1..m], L$ as input, your job is to compute the maximum number of products that can be sold. The run-time of your algorithm must be polynomial in n and m.

For part (a) and (b), assume the product is divisible, that is, it's OK to sell a fraction of a product.

(a) Show how to solve this problem, using a network flow algorithm as a subroutine. Describe the graph and explain why the output from the network flow algorithm gives a valid solution to this problem.

(b) Formulate this as a linear program. Explain why this correctly solves the problem, and the LP can be solved in polynomial time.

(c) Now let's assume you *cannot* sell a fraction of a product. In other words, the number of products sold by each supplier to each purchaser must be an integer. Which formulation would be better, network flow or linear programming? Explain your answer.

Solution:

(a) *Algorithm:* It is similar to the standard matching to flow reduction. We create a bipartite graph with $n+m+2$ nodes. Label two of the nodes as a "source" and a "sink." Label n nodes as suppliers, and m nodes as purchasers. Now, we will create the following edges:

- Create an edge from the source to supplier i with capacity $s[i]$.
- For each pair (i, j) in L, create an edge from supplier i to purchaser j with infinite capacity.
- Create an edge from purchaser j to the sink with capacity $b[j]$. We then plug this graph into our network flow solver, and take the size of the max flow as the number of dollars we can make.

Proof of correctness: We claim that the value of the max flow is precisely the maximum amount transactions we can make. To show this, we can show that a strategy of selling products corresponds exactly to a flow in this graph and vice versa.

For any flow, let $x_{i,j}$ be the amount of flow that goes from the node of supplier i to the node of purchaser j. Then, we claim a product selling strategy will sell exactly $x_{i,j}$ products from supplier i to purchaser j. This is a feasible strategy, since the flow going out of the node of supplier i is already bounded by $s[i]$ by the capacity of the edge from the source to that node, and similarly for the purchasers. Similarly, for the other direction, one can observe that any feasible selling strategy leads to a feasible flow of the same value.

(b) We define a variable $x_{i,j}$, denoting the amount of products we take from supplier i and sell to purchaser j. Then, we have the following linear program

$$\begin{aligned}
\max \sum_{i=1}^{n} \sum_{j=1}^{m} x_{i,j} \\
\sum_{i=1}^{n} x_{i,j'} \leq b[j'], \text{ for all } j' \in [1, m] \\
\sum_{j=1}^{n} x_{i',j} \leq s[i'], \text{ for all } i' \in [1, n] \\
x_{i,j} = 0, \text{ for all } (i,j) \notin L \\
x_{i,j} \geq 0, \text{ for all } (i,j)
\end{aligned}$$

The linear program has nm variables and $O(nm)$ linear inequalities, so it can be solved in time polynomial in n and m.

(c) Network flow is better. Linear programming is not guaranteed to find an integer solution (not even if one exists), so the approach in part (b) might yield a solution that would involve selling fractional products. In contrast, since all the edge capacities in our graph in part (a) are integers, the Ford-Fulkerson algorithm for max flow will find an integer solution. Thus, max flow is the better choice, because there are algorithms for that formulation that will let us find an integer solution.

4 Applications of Max-Flow Min-Cut

Review the statement of max-flow min-cut theorem and prove the following two statements.

(a) Let $G = (L \cup R, E)$ be a unweighted bipartite graph. Then G has a L-perfect matching (a matching with size $|L|$) if and only if, for every set $X \subseteq L$, X is connected to at least $|X|$ vertices in R. You must prove both directions.
Hint: Use the max-flow min-cut theorem.

(b) Let G be an unweighted directed graph and $s, t \in V$ be two distinct vertices. Then the maximum number of edge-disjoint s-t paths equals the minimum number of edges whose removal disconnects t from s (*i.e.*, no directed path from s to t after the removal).
Hint: show how to decompose a flow of value k into k disjoint paths, and how to transform any set of k edge-disjoint paths into a flow of value k.

Solution:

(a) The proposition is known as Hall's theorem. On one direction, assume G has a perfect matching, and consider a subset $X \subseteq L$. Every vertex in X is matched to distinct vertices in R, so in particular the neighborhood of X is of size at least $|X|$, since it contains the vertices matched to vertices in X.

On the other direction, assume that every subset $X \subseteq L$ is connected to at least $|X|$ vertices in R. Add two vertices s and t, and connect s to every vertex in L, and t to every vertex in R. Let each edge have capacity one. We will lower bound the size of any cut separating s and t. Let C be any cut, and let $L = X \cup Y$, where X is on the same side of the cut as s, and Y is on the other side. There is an edge from s to each vertex in Y, contributing at least $|Y|$ to the value of the cut. Now there are at least $|X|$ vertices in R that are connected to vertices in X. Each of these vertices is also connected to t, so regardless of which side of the cut they fall on, each vertex contributes one edge cut (either the edge to t, or the edge to a vertex in X, which is on the same side as s). Thus the cut has value at least $|X| + |Y| = |L|$, and by the max-flow min-cut theorem, this implies that the max-flow has value at least $|L|$, which implies that there must be a perfect matching.

(b) The proposition is known as Menger's theorem. By max-flow min-cut theorem, we only need to show that the max flow value from s to t equals the maximum number of edge-disjoint s-t paths.

If we give each edge capacity 1, then the maxflow from s to t assigns a flow of either 0 or 1 to every edge (using, say, Ford-Fulkerson). Let F be the set of saturated edges; each has flow value of 1. Then extracting the edge-disjoint s-t paths from the flow can be done algorithmically. Follow any directed path in F from s to t (via DFS), remove that path from F, and recurse. Each iteration, we decrease the flow value by exactly 1 and find 1 edge-disjoint s-t path.

Conversely, we can transform any collection of k edge-disjoint paths into a flow by pushing one unit of flow along each path from s to t; the value of the resulting flow is exactly k.

5 Meal Replacement

We are trying to eat cheaply but still meet our minimum dietary needs. We want to consume at least 500 calories of protein per day, 100 calories of carbs per day, and 400 calories of fat per day. We have three options for food we're considering buying: meat, bread, and protein shakes.

- We can consume meat, which costs 5 dollars per pound, and gives 500 calories of protein and 500 calories of fat per pound.
- We can consume bread, which costs 2 dollars per pound, and gives 50 calories of protein, 300 calories of carbs, and 25 calories of fat per pound.
- We can consume protein shakes, which cost 4 dollars per pound, and gives 300 calories of protein, 100 calories of carbs, and 200 calories of fat per pound.

Our goal is to find a combination of these options that meets our daily dietary needs while being as cheap as possible.

(a) Formulate this problem as a linear program.

(b) Take the dual of your LP from part (a).

(c) Suppose now there is a pharmacist trying to assign a price to three pills, with the hopes of getting us to buy these pills instead of food. Each pill provides exactly one of protein, carbs, and fiber.

Interpret the dual LP variables, objective, and constraints as an optimization problem from the pharmacist's perspective.

Solution:

(a) Let m be the pounds of meat we consume, b be the pounds of bread, and s be the pounds of shakes. The objective is to minimize cost, and we have a constraint for each of protein/carbs/fat.

$$\begin{aligned} \min\ & 5m + 2b + 4s \\ 500m + 50b + 300s &\geq 500 \\ 300b + 100s &\geq 100 \\ 500m + 25b + 200s &\geq 400 \\ m, b, s &\geq 0 \end{aligned}$$

(b) Let p, c, f be variables corresponding to the protein, carb, and fat constraints. The dual is:

$$\begin{aligned} \max\ & 500p + 100c + 400f \\ 500p + 500f &\leq 5 \\ 50p + 300c + 25f &\leq 2 \\ 300p + 100c + 200f &\leq 4 \\ p, c, f &\geq 0 \end{aligned}$$

(c) The variables can be interpreted as the price per calorie for the protein, carb, and fat pills.

The objective says that the pharmacist wants to maximize the total revenue he gets from selling enough of these pills to us to meet our dietary needs.

The constraints say that no combination of pills should cost more than a pound of food providing the same dietary needs (otherwise, we would just buy that food instead of these pills).