CS 170 HW 9

Due **2021-04-06, at 10:00 pm**

1 Study Group

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

2 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.

3 Feasible Routing

In this problem, we explore a question called *feasible routing.* Given a directed graph G with edge capacities, there are a collection of supply nodes and a collection of demand nodes. The supply nodes want to ship out flow, while the demand nodes want to receive flow. The question is whether there exists a flow that satisfies all supply and demand.

Formally, we are given a capacitated directed graph, and each node is associated with a *demand value*, d_v. We say that v is a supply node if it has a negative demand value (namely, flow out > flow in), and a demand node, it has a positive demand value (namely, flow in > flow out). A node can be neither demand or supply node, in which case $d_v = 0$. Let $c(u, v)$ be the capacity of the directed edge (u, v). Define a *feasible routing* as a flow that satisfies

- Capacity constraint: for each $(u, v) \in E$, $0 \leq f(u, v) \leq c(u, v)$.
- Supply and demand constraint: for each vertex v, $f^{\text{in}}(v) - f^{\text{out}}(v) = d_v$.

Here, $f^{\text{in}}(v), f^{\text{out}}(v)$ are the sum of incoming flow and outgoing flow at node v.

Note that this is a feasibility problem, and the answer is simply yes or no, whereas the max flow is an optimization problem, where the answer is a number (max flow value).

(a) Let S denote the supply nodes and T the demand nodes. Define the total demand as $\sum_{u \in T} d_u$ and total supply as $\sum_{u \in S} -d_u$. Is there a feasible routing if total demand does not equal total supply? Explain your answer.

(b) Provide a polynomial-time algorithm to determine whether there is a feasible routing, given the graph, edge capacities and node demand values. Analyze its run-time and prove correctness.
Hint: reduce to max flow.

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.

5 Restoring the Balance!

We are given a network $G = (V, E)$ whose edges have integer capacities c_e, and a maximum flow f from node s to node t. Explicitly, f is given to us in the representation of integer flows along every edge e, (f_e).
However, we find out that one of the capacity values of G was wrong: for edge (u, v), we used c_{uv} whereas it really should have been $c_{uv} - 1$. This is unfortunate because the flow f uses that particular edge at full capacity: $f_{uv} = c_{uv}$. We could run Ford Fulkerson from scratch, but there's a faster way.

Describe an algorithm to fix the max-flow for this network in $O(|V| + |E|)$ time. Also give a proof of correctness and runtime justification.

6 Zero-Sum Games

Alice and Bob are playing a zero-sum game whose payoff matrix is shown below. The ij^{th} entry of the matrix shows the payoff that Alice receives if she plays strategy i and Bob plays strategy j. Alice is the row player and is trying to maximize her payoff, and Bob is the column player trying to minimize Alice's payoff.

Alice \Bob	1	2
1	4	1
2	2	5

Now we will write a linear program to find a strategy that maximizes Alice's payoff. Let the variables of the linear program be x_1, x_2 and p, where x_i is the probability that Alice plays row i and p denotes Alice's payoff.

(a) Write the linear program for choosing Alice's strategy to maximize her payoff.

(b) Write a linear program from Bob's perspective trying to minimizing Alice's payoff. Let the variables of the linear program be y_1, y_2 and p, where y_i is the probability that Bob plays strategy i and p denotes Alice's payoff.

(c) As covered in lecture, Bob's linear program and Alice's are dual to each other. How can you see that this is the case for the LPs you have written here? Either take the dual mechanically or make a concise argument. (Hint: You may want to transform the linear programs into a form where it is easy to take the dual, if they are not already in such a form).

(d) What is the optimal solution and what is the value of the game?