Problem Set 7

Due: Wednesday, October 24, 2011.

Collaboration policy: collaboration is *strongly encouraged.* However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so keep your collaboration groups limited to 3 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

NONCOLLABORATIVE Problem 1. Another way to formulate the maximum-flow problem as a linear program is via flow decomposition. Suppose we consider all (exponentially many) s-t paths P in the network G, and let f_P be the amount of flow on path P. Then maximum flow says to find

$$z = \max \sum f_P$$

subject to

$$\begin{aligned} \sum_{P \ni e} f_P &\leq u_e & \qquad \text{for all edges } e, \\ f_P &\geq 0 & \qquad \text{for all paths } P. \end{aligned}$$

(The first constraint says that the total flow on all paths through e must be less than u_e.) Take the dual of this LP and give an English explanation of the objective and constraints.

Problem 2. Consider a directed graph in which edges have costs (possibly negative). Suppose you want to find a *minimum mean cycle* in this graph: one with the minimum ratio of cost to length (number of edges). Going around such a cycle repeatedly (assuming it is negative) provides you with the maximum possible profit per unit length/time, so it is the fastest way to earn money if you are, for example, a delivery service. Finding a minimum mean cycle is also an essential step in certain strongly polynomial min cost flow algorithms. Consider the following linear program, with a variable f_{ij} for every pair of vertices (i, j):

$$\begin{aligned} w &= \min \sum c_{ij} f_{ij} \\ \sum_j f_{ij} - f_{ji} &= 0 \quad (\forall i) \\ \sum_{i,j} f_{ij} &= 1 \\ f_{ij} &\geq 0 \end{aligned}$$

(a) Explain why this captures the minimum mean cycle problem. (**Hint:** $\{f_{ij}\}$ is a circulation, so it can be decomposed into cycles.)

(b) Give the dual of this linear program—it will involve maximizing a variable λ.

(c) Give an explanation (in terms of min-cost-flow reduced costs) for why this dual formulation also captures minimum mean cycles. (**Hint:** how much is added to the cost of a k-edge cycle?)

(d) Let's assume the costs c_{ij} are integers. Suggest a combinatorial algorithm (not based on linear programming) that uses binary search to find the right λ to solve the dual problem. Can you use this to find a minimum mean cycle? **Note:** to know when you can terminate the search, you will need to lower bound the difference between the smallest and next smallest mean cost of a cycle.

Problem 3. Although the dual can tell you a lot about the structure of a problem, knowing an optimal dual solution does not in general help you solve the primal problem. Suppose we had an algorithm that could optimize an LP with an $m \times n$ constraint matrix in $O((m+n)^k)$ time given an optimal solution to the dual LP, where k is some positive constant. The algorithm is not guaranteed to work if given an incorrect solution to the dual.

(a) Argue that any LP optimization problem can be transformed into the form

$$\begin{array}{ll} \text{minimize} & 0 \cdot x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

This LP has optimum 0 if it is feasible and ∞ if no. **Hint:** Recall our argument that optimizing an LP can be reduced to checking feasibility of an LP.

(b) What is the dual of this linear program?

(c) Argue that if the primal is feasible then the dual has an obvious optimum solution.

(d) Deduce that, given the hypothetical algorithm above, you can build an LP algorithm that will solve any LP *without* knowing beforehand a dual solution, in the same asymptotic time bounds as the algorithm above.

Problem 4. The simplex algorithm moves from feasible point to feasible point until reaching an optimum. The interior point algorithm also requires a feasible starting point. But it is not always easy to find a feasible point of an LP (in fact, we showed that being able to do so is equivalent to being able to optimize any LP). So how can we start the simplex algorithm?

(a) Given a standard form LP $\min\{cx \mid Ax = b, x \geq 0\}$, show how to define a different LP with an obvious feasible point, whose optima are precisely the feasible points of the original LP. **Hint:** optimize the amount by which a point diverges from feasibility.

(b) Explain how this solves our problem, allowing us to use a simplex solver (which requires a feasible starting point) to solve arbitrary LPs without being given a feasible starting point.

Problem 5. A *network design problem* specifies a collection of *nodes* V that need to be connected by paths, a nonnegative *cost matrix* C of nonnegative, rational values specifying, for each pair of nodes, the cost per unit to buy or build capacity connecting these two nodes, and a nonnegative *demand matrix* D specifying, for each pair of nodes, the amount of capacity that the designed network must be able to provide between that pair. More precisely, this means that, for any two vertices s and t, there must exist a flow of value at least $D_{s,t}$ between s and t. Assuming that it is permissible to buy arbitrary fractional amounts of capacity between nodes, this problem can be solved in weakly polynomial time by linear programming (if capacity must be purchased in integer quantities the problem is NP-hard). But the number of constraints in an explicit linear program for the problem is quite large.

(a) Explain how a network design problem can be solved using the ellipsoid algorithm. In particular, give a linear program involving exponentially many constraints, and provide an algorithm that, given any proposed network, finds a violated constraint in polynomial time (**Hint:** the demands must be satisfied by flows, which are intimately related to cuts).

(b) Argue that one can therefore find the *minimum cost* network design for the given input (remember to explain how you will minimize the objective, not just find a feasible point).

Problem 6. In a 0-*sum 2-player game*, Alice has a choice of n so-called *pure* strategies and Bob has a choice of m *pure* strategies. If Alice picks strategy i and Bob picks strategy j, then the *payoff* is a_{ij}, meaning a_{ij} dollars are transfered from Alice to Bob. So Bob makes money if a_{ij} is positive, but Alice makes money if a_{ij} is negative. Thus, Alice wants to pick a strategy that minimizes the payoff while Bob wants a strategy that maximizes the payoff. The matrix $A = (a_{ij})$ is called the *payoff matrix*.

It is well known that to play these games well, you need to use a *mixed* strategy—a random choice from among pure strategies. A mixed strategy is just a particular probability distribution over pure strategies: you flip coins and then play the selected pure strategy. If Alice has mixed strategy x, meaning he plays strategy i with probability x_i, and Bob has mixed strategy y, then it is easy to prove that the expected payoff in the resulting game is xAy. Alice wants to minimize this expected payoff while Bob wants to maximize it. Our goal is to understand what strategies each player should play.

We'll start by making the pessimal assumption for Alice that whichever strategy she picks, Bob will play best possible strategy against her. In other words, given Alice's strategy x, Bob will pick a strategy y that achieves $\max_y xAy$. Thus, Alice wants to find a distribution x that minimizes $\max_y xAy$. Similarly, Bob wants a y to maximize $\min_x xAy$. So we are interested in solving the following 2 problems:

$$\min_{\sum x_i = 1} \max_{\sum y_j = 1} xAy$$
$$\max_{\sum y_j = 1} \min_{\sum x_i = 1} xAy$$

Unfortunately, these are nonlinear programs!

(a) Show that if Alice's mixed strategy is known, then Bob has a pure strategy serving as his best response.

(b) Show how to convert each program above into a linear program, and thus find an optimal strategy for both players in polynomial time.

(c) Give a plausible explanation for the meaning of your linear program in terms of the zero-sum matrix game (why does it give the optimum?)

(d) Use strong duality (applied to the LP you built in the previous part) to argue that the above two quantities are *equal*.

The second statement shows that the strategies x and y, besides being optimal, are in *Nash Equilibrium*: even if each player knows the other's strategy, there is no point in changing strategies. This was proven by Von Neumann and was actually one of the ideas that led to the discovery of strong duality.

Problem 7. How long did you spend on this problem set? Please answer this question using the Google form that is sent to you via a separate email. This problem is mandatory, and thus counts towards your final grade. It is due by the Monday 2:30pm after the pset due date. You can find the link to the form on the course website.