Homework 3

Due Friday March 10, 2017.

1. Warm-up.

(a) What is the dual of the linear program $\min cx, Ax \leq b$? Be careful!

(b) Consider a linear program $\min cx, Ax \geq b, x \geq 0$ and its dual $\max y^T b, y^T A \leq c, y \geq 0$. Which of the following statements are always true?

(i) For any optimal solutions x, y to the primal and dual programs, $y_i \neq 0 \implies a_i x = 0$.

(ii) For any optimal x, y, $x_j = 0 \implies (y^T A)_j = c_j$.

(iii) For any feasible x, y $y^T b \geq cx$.

(iv) For feasible x, y where $y^T(Ax - b) = 0$, x and y are optimal primal and dual solutions.

(Here, a_i is the ith row of A, and the notation $(v)_j$ is the jth element of the vector v.)

2. Another algorithm for solving linear programs.

(a) Recall that the perceptron algorithm would only make $O(1/\gamma^2)$ mistakes when classifying n points which had a separating hyperplane on a ± 1 pointset with margin γ. Use the perceptron algorithm as a blackbox to find a separating hyperplane that will make no mistakes on a set of n such points.

(b) Now, suppose you are given a $n \times m$ matrix A and an n-dimensional vector b. Using part (a), describe an algorithm for solving $Ax \geq b$, if you are given the following information:

(i) Every number a_{ij} in A and b_i in b is in the set $\{-1, 0, 1\}$.

(ii) There is a solution x with the property that $|x| \leq M$, and for every i, $a_i x \geq b + \epsilon$.

Your algorithm should be as fast as running $O(M^2 m/\epsilon^2)$ rounds of the perceptron algorithm (where a round is running the perceptron algorithm on $O(n)$ points.)

3. Maximum weight matching.

In class, we learned how to find a maximum-weight matching on a weighted bipartite graph.

In this question, we'll see the beginning of an algorithm that works for non-bipartite graphs.

For convenience, we'll only look at graphs which have an edge between every pair of vertices, and with an even number of vertices. (We can make any graph satisfy these conditions by possibly adding a dummy vertex, and then adding zero-weight edges.)

(a) Show that the linear program might not have an integer optimum solution when the graph is not bipartite, by providing an example.

(b) Edmonds showed that if one adds the condition where the total value of x_e for edges across every odd cut $(S, \bar{S})$ is at least 1, then there is an integer optimum. Add a set of constraints that correspond to this condition. (The answer is simple. There will be an exponential number of constraints. By odd cut, we mean $|S|$ is odd. Remember that we're assuming the number of vertices is even; in fact, graphs with odd numbers of vertices can't satisfy this condition.)

(c) Take the dual of your linear program. Use the dual to show that the example you used in part (a) has no integer solution of value equal to the fractional solution for the previous linear program (without the odd set constraints.)

(If you answer to (a) has an odd number of vertices, add a dummy vertex first.)

4. (a) Briefly argue that any two convex bodies with an empty intersection can be separated by a hyperplane. (You can appeal to arguments from class, and thus should provide a very brief answer. Your answer could follow the outline: For two convex nonintersecting bodies, we find blah, we construct a hyperplane, if it does not separate the bodies, then blah was, in fact, not blah.)

(b) Prove Farkas version (B) from the slides (also appears in the Goemans notes.)

5. Facility location.

Turn in either Option A or Option B.

Option A In Lecture 10, we saw a few different algorithms for the facility location problem. The last algorithm we learned didn't require running a general linear program solver. It worked in two phases:

- Phase 1: Find a solution to the dual linear program. (This phase begins by initializing every α_j and β_{ij} to 0, and then starts growing the α_js....)
- Phase 2: Round the dual solution to get a solution to the problem.

Show how to implement Phase 1 in $O(nm \log(nm))$ time, where n and m are the numbers of clients and facilities. (The distances d_{ij} are given to your algorithm as a full $n \times m$ matrix.)

Option B (Turn in either Option A or Option B; your choice.)

Consider the facility location algorithm from Lecture 10 that begins by solving the primal and dual linear programs, and then rounds the solution. We'll use that as a starting point in this question.

(a) Notice that the algorithm for rounding the LP relaxation solution had expected facility cost of expected LP facility cost , but had a possibly much larger connection cost (2 times the LP solution plus the LP solution connection cost).

Let F and C be the facility and connection cost of some optimal solution. Given a different solution with connection cost C', show that there is a facility i with facility cost f whose opening reduces the connection cost by Δ where $\frac{\Delta}{f} \geq \frac{C'-C}{F}$.

(b) Argue that you can find a facility location solution with cost at most $F + C$ plus $f_{\max} + F(1 + \ln(\frac{2F+2C}{F}))$, where $f_{\max} = \max_i f_i$. (Think the solution from class (discussed in part (a)) and then reducing the cost by adding facilities and doing some calculus.)

(c) Consider an instance of the facility location problem where the distances between clients and facilities are either 1 or 3. Give a randomized rounding method that produces a solution with expected connection cost of $(1 + \frac{2}{e})C$ and expected facility cost of at most $F + f_{\min} \prod_i (1 - y_i)$ where F and C are the facility and connection costs of an optimal solution to the linear program relaxation used in class; y_i are the facility decision variables from that same program; and $f_{\min}$ is the least cost facility. (Our solution doesn't use the dual.)

(d) Research Question (I believe). Get a facility cost F, connection cost $(1+2/e)C$ solution for a distance function from an arbitrary bipartite graph (with no edge weights) where F and C are the facility, connection costs in an optimal solution.

6. (a) Consider the problem of finding the origin. Oh, look, here it is: $(0, 0)$. Alas, if life were only so easy. Consider doing it by noticing that it is at the common intersection of a set of lines, and repeating the following process. Start with a point x_0 on one of the lines, pick another line, ℓ_1, and project x_0 onto ℓ_1 to get x_1. Projection means that assuming the dot product is d, the new point is at distance d from the origin on the line.

(i) Draw a picture with say eight lines with symmetry, and illustrate this process.

(ii) Does it converge in this case? In general? How long does it take to reduce the distance to the origin by a factor of two given a set of n intersecting lines and worst case selection of line to project onto. Hint: it is about an angle between two lines.

(iii) Is this an algorithm that could be used to solve linear systems in higher dimensions? What is the analogy to projecting onto a hyperplane? What is the normal to a hyperplane? Why do you need to use all the hyperplanes?

(b) Recall that a way to find an electrical flow, f, that routes a current demand vector, χ, is to find an f, satisfying $B^T f = \chi$ and for all cycles, C, we have the constraint $\sum_{e \in C} r(e) f(e) = 0$, where $r(e)$ is the resistance on edge e.

Recall that the algorithm began with a feasible flow (i.e., $B^T f = \chi$) supported on a spanning tree. Then it repeatedly chose an non-tree edge and routed flow around the induced cycle.

Argue this is a version of the above algorithm. Note that it is working in a subspace already (where $B^T f = \chi$.) Specifically argue that we can work only with the cycles induced by non-tree edges.

7. Provide a project description of at most two paragraphs. (If it is really good, one paragraph will surely suffice.)