Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout **??**
Wednesday, October 17, 2012

Problem Set 7

Due: Wednesday, October 24, 2012.

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 you should keep your collaboration groups limited to 3 or 4 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

Problem 1. The dual of this program is:

$$\min \sum_{e \in E} u_e x_e \text{ subject to } \begin{cases} \sum_{e \in P} x_e & \geq & 1 & \forall\ s-t \text{ paths } P \\ x_e & \geq & 0 \end{cases}$$

If we force the variables $x_e \in \{0, 1\}$, then we see we have an integer program that computes the minimum cut in the graph. The dual program is just a relaxation of this IP with $x_e \in [0, 1]$, since $x_e > 1$ does not occur in an optimal solution. Note that we know that the LP above can achieve the minimum cut value. By strong duality, though, we know that the minimum cut value is the optimum value of this LP, since the primal LP computes the value of the maximum flow in the graph.

Problem 2. **(a)** The conservation constraints on f_{ij} ensure that the flow defined by f_{ij} is a circulation. The net flow in the circulation is 1. Notice that there are no capacity constraints on the solution.

Consider the cycle decomposition of a solution f_{ij}. Each cycle C with flow f_C contributes $f_C \sum_{(ij) \in C} c_{ij}$ to the objective. Moreover, the cycle contributes $|C| f_c$ to the $\sum f_{ij} = 1$ constraint. The ratio of contribution to objective to the contribution of net flow is minimum for the mean-cost cycle. Thus the cycle decomposition of the solution can have only minimum mean-cost cycles.

(b) The primal is:

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

The dual of the above LP is:

$$\begin{aligned} \max & \quad \lambda \\ y_i - y_j + \lambda &\leq c_{ij} (\forall i, j) \end{aligned}$$

(c) The dual assigns potentials to different nodes and computes reduced costs. All the reduced costs are more than $-\lambda$. The algorithm maximized λ, i.e., minimizes the smallest reduced cost. Recall that the sum of reduced cost over a cycle is same as the sum of original costs. So $-\lambda$ should be at least the mean cost in the minimum mean cost cycle. We will now prove that there exists a potential assignment that works for

$$-\lambda = \text{ mean cost of the minimum mean cost cycle}$$

Consider a graph G^* with costs reduced by λ. This graph does not have a negative cost cycle, since such a cycle will be a smaller mean cost cycle. So there exists a potential assignment for this graph. The same potential assignment for the original graph will have at least $-\lambda$ reduced cost.

(d) Given a λ, we can subtract it from edge costs to get graph G^*. We can check whether we can assign potentials to nodes in G^* such that all reduced costs are positive, using the Bellman-Ford algorithm. Such an assignment is possible for all λ smaller than the optimal λ and is not possible for other values. We can do a binary search to compute the value of λ. If we work with integer costs, we can stop the binary search once the range of investigation is reduced to $1/n^2$. This is because the difference of two unequal fractions m_1/n_1 and m_2/n_2 is at least $1/n_1 n_2 \leq 1/n^2$.

Problem 3. **(a)** We know any LP P can be written in standard form, i.e. minimize $c^T x$ subject to $Ax = b, x \geq 0$. Now, consider the dual LP: maximize $y^T b$ subject to $y^T A \leq c$. If the primal has an optimal solution, the dual has an optimal solution of the same value. Therefore, consider the following LP Q: minimize 0 subject to $Ax = b, x \geq 0, y^T A \leq c, c^T x = y^T b$. Consider P: it is either

feasible with an optimal solution, feasible but unbounded, or infeasible. Clearly, Q will remain infeasible P was infeasible. If there is an optimal solution to Q, we can recover it and produce an optimal solution to P. Finally, we can consider the original LP P without minimization (call it R): minimize 0 subject to $Ax = b, x \geq 0$. If R is feasible and Q was infeasible, then P was feasible but unbounded. This allows us to distinguish all three cases.

(b) The dual of this linear program is maximize $y^T b$ subject to $y^T A \leq 0$.

(c) If the primal is feasible, it has as an optimum value 0. Therefore, the optimum value for the dual is 0. So therefore $y = 0$ is an optimum solution for the dual.

(d) Give the algorithm the LP Q described in part (a). Constructing Q takes less than $O((m+n)^k)$ time. Now, give as the dual solution to Q the zero vector. By part (a), this algorithm will solve the LP, i.e. it can distinguish the three cases that an LP can fall into.

Problem 4. To find a feasible point, we can define the following LP using the original variables x_i and a new slack variable s_i for each of the original variables:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} s_i \\ Ax + Is &= b \\ x &\geq 0 \\ s &\geq 0 \end{aligned}$$

We can assume that all entries in b are nonnegative because if there was some entry in b that was negative, we can negate the appropriate row of A and b to get a positive entry for b and still have equivalent constraints to the original problem. There is an easy feasible solution where we assign $x = 0$ and $s = b$. The optima of this LP are precisely the feasible points of the original LP since the minimum possible objective function is 0 since all $s_i \geq 0$, and when that is true, we have values for x such that $Ax = b$ and $x \geq 0$, the constraints of the original LP. We can then use the simplex method to find an optimal solution to this LP, which can then be converted to a basic feasible solution in the original LP.

Problem 5. **(a)** We construct the following linear program: for each edge (i, j), we have a variable e_{ij} which contains the amount of that node that we buy.

Now, between any two nodes s and t, there must be an s-t flow of value at least D_{st}. In other words, the minimum cut between s and t should be at least D_{st}. So, for every s-t cut, (of which there are exponentially many) we make a constraint saying that the sum of e_{ij} over all edges (i, j) in that cut should be at least D_{st}.

Now, we will create a seperation oracle for this linear program. To find a violated constraint, we simply find the minimum s-t cut for each pair of vertices (s, t). We can do this in polynomial time since we can solve maximum flow in polynomial time. If we find a cut that is too small, then we output that as our violated constraint.

(b) To find the minimum cost network design, we use binary search on the cost. Given some cost c, we want to check if there is a feasible solution to the linear program of cost at most c. To do this, we simply add that $\sum C_{ij}e_{ij} \leq c$ as a constraint, and modify our separation oracle to also check that this constraint is satisfied. Now, we can run the ellipsoid algorithm to see if the linear program is feasible.

Now, we can binary search on the answer. The number of iterations of the binary search is logarithmic in the precision we need; this is polynomial in the input size.

Problem 6. **(a)** We can describe a mixed strategy for Bob as just being an m-tuple $(x_1, x_2, \ldots, x_m)$, and similarly we can describe the known mixed strategy for Alice as $(y_1, y_2, \ldots, y_n)$. So now, we will show that for any mixed strategy there is a pure strategy which performs better, which then implies that the optimal strategy is pure. The expected payoff for a given mixed strategy of Bob is:

$$\begin{aligned}
& \sum_{i=1}^{m} x_i \Big(\sum_{j=1}^{n} y_j a_{ij} \Big) \\
\leq \; & \sum_{i=1}^{m} x_i \Big(\sum_{j=1}^{n} y_j a_{i'j} \Big) \text{ where } i' \text{ is such to make } \sum_{j=1}^{n} y_j a_{i'j} \text{ a maximum} \\
\leq \; & \Big(\sum_{i=1}^{m} x_i \Big) \Big(\sum_{j=1}^{n} y_j a_{i'j} \Big) \\
= \; & \sum_{j=1}^{n} y_j a_{i'j}
\end{aligned}$$

So the pure strategy of always picking i' is better than the mixed strategy we started with.

(b) Bob's linear program:

$$\begin{aligned} & & \max & & x_{m+1} \\ x_{m+1} - \sum_{i=1}^{m} x_i a_{ij} & \leq & 0 \ \forall j \\ \sum_{i=1}^{m} x_i & = & 1 \\ x_i & \geq & 0, \quad i = 1, 2, \ldots, m \end{aligned}$$

Alice's linear program:

$$\begin{aligned} & & \min & & y_{n+1} \\ y_{n+1} - \sum_{j=1}^{n} y_j a_{ij} & \geq & 0 \ \forall i \\ \sum_{j=1}^{n} y_j & = & 1 \\ y_j & \geq & 0, \quad j = 1, 2, \ldots, n \end{aligned}$$

(c) In Bob's linear program, x_i represents the probability that he uses strategy i, for $i = 1, ..., m$. x_{m+1} represents the expected payoff, so that is what he is trying to maximize. By (a), we know that for any given strategy there exists a pure strategy as the best defense against it, so the first constraint says that the best Bob can do is limited by the pure strategies that he could go against. So, the objective function and first constraint are there to capture the notion of optimality we are searching for in this problem. The next two constraints are just there to ensure that the x_i's we pick are actually probabilities.

For an explanation of how Alice's linear program works, replace x with y, m with n, "Bob" with "Alice", "he" with "she", and "maximize" with "minimize" in the above paragraph.

(d) If we look at the constraint matrix for Alice's LP, it is of size $(m+1) \times (n+1)$. The upper left $m \times n$ block is the matrix $-A$. Every other entry is 1, with the exception of the bottom-right entry which is 0 (for when we take the sum of the probabilities to be 1). If we transpose this, we get an $(n+1) \times (m+1)$ size matrix with $-A$ in the bottom left corner and 1's everywhere else except a 0 in the top-right corner. This is just the matrix for Bob's LP! So, Bob's LP is just the dual of Alice's. So, by strong duality their optima are equal.