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

Handout 16
Wednesday, Obtober 26, 2005

Problem Set 7 Solutions

Problem 1.

(a) If P and Q have empty intersection, then the following linear program is infeasible:

$$\begin{aligned} \alpha &= \max 0 \cdot x, \\ \begin{pmatrix} A \\ D \end{pmatrix} x &\leq \begin{pmatrix} b \\ e \end{pmatrix}. \end{aligned}$$

Its dual

$$\begin{aligned} \beta &= \min t \begin{pmatrix} b \\ e \end{pmatrix}, \\ t \begin{pmatrix} A \\ D \end{pmatrix} &= 0, \\ t &\geq 0 \end{aligned}$$

is always feasible ($t = 0$ is a feasible solution), and after substitution $t = (y\ z)$ becomes

$$\begin{aligned} \beta &= \min(yb + ze), \\ yA + zD &= 0, \\ y, z &\geq 0. \end{aligned}$$

Since the dual is feasible, it must be unbounded, otherwise the primal would have to be feasible. Therefore, there exist $y \geq 0$ and $z \geq 0$ such that $yb + ze < 0$, and $yA + zD = 0$.

(b) We consider $c = yA = -zD$ from part (a). We will make extensive use of the fact that $y, z \geq 0$. Let $x \in P$, and $w \in Q$. Then

$$cx = yAx \leq yb < -ze \leq -zDw = yAw = cw.$$

(c) If the two polyhedra have a point in common, then a point that belongs both to P and Q is a quickly verifiable answer.

Otherwise, it is enough to show a feasible solution t to the dual such that

$$t \begin{pmatrix} b \\ e \end{pmatrix} < 0.$$

If such t exists the primal must be infeasible, and on the other hand, if the primal is infeasible, we already know by part (a) that such t exists. This answer can be quickly verified as well.

Problem 2.

The dual is

$$\begin{aligned} y &= \min \sum u_e d_e, \\ \sum_{e \in P} d_e &\geq 1 \qquad (\forall P), \\ d_e &\geq 0 \qquad (\forall e), \end{aligned}$$

where y is the value that we optimize, and d_e's are variables.

For each edge e, d_e describes its length. The first constraint says that the distance from s to t is at least 1, and the second that the length of each edge is nonnegative.

The objective of the goal function is an assignment of lengths to edges of the minimum cost. By strong duality the optimal value of y equals the minimum s-t-cut value in the network, and actually, for a given s-t-cut (S, T) there exists an assignment of the same value. We assign to each edge going from S to T length 1, and to all other length 0. Obviously, the length of each path from s to t is greater than or equal to 1, and y for this assignment equals the value of the s-t-cut.

Problem 3.

(a) Let us take a linear program P in the standard form:

$$\begin{aligned} \alpha &= \min c^T x, \\ Ax &= b, \\ x &\geq 0. \end{aligned}$$

Its dual is

$$\begin{aligned} \beta &= \max b^T y, \\ A^T y &\geq c. \end{aligned}$$

If P is feasible and bounded, then, by strong duality, $\alpha = \beta$, and it is enough to check whether the following linear program is feasible:

$$\begin{aligned} b^T y &= c^T x, \\ Ax &= b, \\ x &\geq 0, \\ \beta &= \max b^T y, \\ A^T y &\geq c. \end{aligned}$$

If P is unbounded or infeasible, the above program does not have a solution. The feasibility of this program is equivalent to minimization of 0, and this program can be expressed in the

form in the problem statement. For each unbounded variable t we substitute $t^+ - t^-$ where t^+ and t^- are new nonnegative variables, and for each inequality of the form $L \geq R$, we add an additional nonnegative variable t, and replace the inequality with $L + t = R$. Finally, we achieve a linear program Q.

(b) The dual R to this program is

$$\begin{aligned} \gamma &= \max b^T y, \\ A^T y &\leq 0. \end{aligned}$$

(c) When the primal is feasible, it has a solution of value 0. The dual is always feasible, since $y = 0$ meets all the constraints. By strong duality the dual has maximum of value 0, and $y = 0$ is an obvious optimum solution.

(d) First, we assume that the algorithm that we are given stops if it is given a nonoptimum solution to the dual when the dual is unbounded. The running time of the algorithm can be bounded by $c(m+n)^k$, where c is a constant. Knowing that, in the worst case we can always stop the algorithm after $c(m+n)^k$ steps, and say that it has looped. Denote this algorithm by $\mathcal{A}$.

In the beginning we check whether P is feasible. This can be done by substitution 0 instead of the goal function of P. The dual to the modified P is almost the same as the dual R, and, if P is feasible, has optimum solution 0. We run $\mathcal{A}$ on modified P with dual solution 0. Algorithm $\mathcal{A}$ returns a correct solution if and only if P is feasible.

Now, we know that P is feasible. If it is bounded, $\mathcal{A}$ ran on Q and 0, the optimum solution to the dual, returns a correct solution to Q (easily transformable to a solution to P); otherwise, P is unbounded.

We have shown how to solve in $O((m+n)^k)$ time a linear program, making use of algorithm $\mathcal{A}$.

Problem 4.

(a) Take an optimal circulation described by the linear program, and decompose it into cycles. Suppose there is a circulation cycle, that does not go over a minimum mean cycle, but over a cycle of length k_1 and the sum of c_{ij}'s is c_1. Let f be the sum of values of circulation over this cycle (f/k_1 per edge). This implies that this cycle contributes $c_1 f/k_1$ to the value of w. In the network there is a cycle of the optimal mean value. It has some length k_2, and the sum of c_{ij}'s c_2. Obviously, $c_2/k_2 < c_1/k_1$. If we took flow f, and let it circulate over the minimum mean cycle, it would contribute $c_2 f/k_2$ to w, and w would be smaller. Therefore, in the optimum solution all circulation goes over minimum mean cycles.

(b) The dual is following:

$$\begin{array}{rcll} & & \max \lambda, & \\ \lambda + p_i - p_j & \leq & c_{ij} & (\forall i, j), \\ & & p_i,\ \lambda \text{ unbounded.} & \end{array}$$

(c) We can transform the constraint to

$$0 \leq (c_{ij} - \lambda) + p_j - p_i.$$

This implies that the dual has a feasible solution as long as all edges (of lengths $c_i j - \lambda$) have positive reduced costs under some potential p. It has been shown in class that this is true as long as there is no cycle of negative length, and obviously, this cannot be done when there is a cycle of negative cost, since lengths of cycles do not change under reduction. For optimal λ at least one cycle has zero length, otherwise we would be able to subtract a very small positive value from length of each edge that would not make any cycle have negative length, and we would have greater λ.

Furthermore, the length of a cycle C equals 0 if and only if λ equals the mean of costs of edges, since

$$\sum_{(i,j)\in C} (c_{ij} - \lambda) = 0$$

is equivalent to

$$\frac{1}{k} \sum_{(i,j)\in C} c_{ij} = \lambda,$$

where k is the length of the cycle.

In total, this means that the optimal λ equals the minimum mean of all cycles.

(d) We want to determine λ (or, as we will see, almost λ) for which all cycles have nonnegative lengths, and there exists a cycle of length 0. For sure, λ as a mean of costs of edges is greater than or equal to the minimum cost of an edge, and less than or equal to the maximum cost of an edge. We will use binary search to find good approximation of λ. Suppose that $\lambda \in [a, b]$. We take $\lambda^* = (a + b)/2$, and check, using Bellman-Ford algorithm, whether there is a cycle of negative cost in a network if we subtract λ^* from the cost of each edge. If there is, $\lambda \in [a, \lambda^*(=: b)]$; otherwise, $\lambda \in [\lambda^*(=: a), b]$.

When can we stop the binary search? It turns out that when

$$b - a < \frac{1}{n^2},$$

where n is the number of vertices. For any two distinct means of cycles in the network the difference between them is at least $1/n^2$. Suppose that one of them is the mean of the cycle

of k_1 vertices, and the other of k_2. If we multiply both means by k_1k_2, they both will be integers, and they need to differ at least by 1, so in the beginning they had to differ at least by $1/(k_1k_2) \geq 1/n^2$. If we find a range of length less than $1/n^2$ in which λ is, then there is no other mean in this range. This means that if we take b, the right end of the range, and subtract it from the cost of each edge, the only cycles of nonpositive cost in the graph will we be cycles of mean λ. This ends the description of the algorithm.