Matthew Feng October 17, 2018

6.046 Problem Set 5

Collaborators: *Lauren Oh, Lillian Bu*

Problem 1

(a)

We wish to maximize the revenue $\sum_{i=1}^{n} c_i x_i$ that Melon makes subject to the constraint(s) that he cannot use more materials than he has. Concretely, those two statements translate into the following equations (in standard form):

$$\begin{aligned} \text{max} \quad & \sum_{i=1}^{n} c_i x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} a_{ij} x_i \leq b_j, \quad \forall j \in [1, n] \\ & x_i \geq 0, \qquad\qquad \forall i \in [1, n] \end{aligned}$$

where x_i represents the (not necessarily integral) number of type i vehicles to manufacture.

(b)

The general idea behind getting a $O(\log P^*)$ time algorithm is to binary search on the value of P. Specifically, we want to introduce an additional constraint for our friend such that the revenue P is greater than some amount P'. We will have found $P*$ when we find the value of P' such that for any value $P'' > P'$, adding the constraint $P > P''$ results in an infeasible LP.

Specifically, let us define a value P that represents the minimum revenue that we want Melon to make. Then we can add this constraint to the constraints we have listed in part (a) as

$$\sum_{i=1}^{n} c_i x_i \geq P$$

where P is a constant value that we control. Let us start with $P^{(0)} = 1$, where the superscript represents the iteration index that we are on. If we hand this LP to our MIT oracle, then

we will see whether or not the LP is feasible. If the LP is feasible, double the value of P so that $P^{(k)} = 2^k$ (i.e. $P^{(k)} = 2P^{(k-1)}$). We continue until the problem becomes infeasible, meaning that there is no combination of numbers of vehicles to manufacture that can result in such a return $P^{(K)}$. Then P^* must be between $P^{(K-1)}$ and $P^{(K)}$. We can binary search on this remaining interval (which is guaranteed to be of size $O(P^*)$) until we find the exact P' such that P' is feasible, but any value greater than P'' makes the LP infeasible. That P' is P^*, the maximum amount that Melon can make.

Binary search is viable because the predicate "Melon can make more than P in revenue" is true for all values $\leq P^*$, and false for any value $> P^*$. Furthermore, because we bounded the value of $P*$ with $O(\log P*)$ calls to the oracle, and the final binary search looks in a region that is of size $O(\log P^*)$, this algorithm will make $O(\log P^*)$ uses of the oracle.

Since it will take of $O(n)$ time to set up this additional constraint initially (due to the n different types of vehicles and their corresponding selling prices), the total runtime of our procedure is $O(n + \log n)$.

(c)

We can formulate the desired optimization problem into the following LP (not in standard form):

$$\begin{aligned} \text{min} \quad & \sum_{j=1}^{n} b_j y_j & \\ \text{subject to} \quad & \sum_{j=1}^{n} a_{ij} y_j \geq \frac{c_i}{2}, & \forall i \in [1, n] \\ & y_j \geq 0, & \forall j \in [1, n] \end{aligned}$$

where y_j is the price of resource j. Upon closer inspection, we realize that the dual to this LP is the same as the LP in part (a), with the only change being that the maximization objective is now multiplied by 1/2. By strong duality, we know then that the solution to our minimization LP is equal to the optimal solution for the dual maximization problem, which is $\frac{P^*}{2}$. Thus, the lowest feasible price Melon can pay such that the seller is satisfied is $\frac{P^*}{2}$.