CS 224: Advanced Algorithms Spring 2017

Lecture 11 — February 28, 2017

Prof. Jelani Nelson *Scribe: Meena Jagadeesan*

1 Overview

In this lecture, we look at covering LPs ($A, b, c \geq 0$) under the setting where constraints come online and we must monotonically change x to satisfy the incoming constraints. We give examples of using the primal-dual approach to design and analyze algorithms for ski rental and set cover. For ski rental, we cover a 2-competitive deterministic algorithm and a randomized algorithm that is $\frac{e}{e-1}$-competitive in the limit. For set cover, we investigate two slightly different $\log m$-competitive algorithms – one designed through a primal-only approach and another designed through a primal dual approach.

2 Recap of Approximate Complementary Slackness Result

We recall the approximate complementary slackness theorem from last lecture:

Theorem 1. *Suppose x, y are primal and dual feasible, respectively. Then if $\exists\ \alpha,\ \beta \geq 1$ such that*

$$\forall i, x_i > 0 \implies \frac{c_i}{\alpha} \leq \langle (A^T)_i, y \rangle \leq c_i$$
$$\forall j, y_j > 0 \implies b_j \leq \langle A_j, x \rangle \leq \beta b_j$$

then $c^T x \leq (\alpha\beta) b^T y$.

Recall that the primal is

$$\min c^T x \text{ such that } Ax \geq b, x \geq 0.$$

Recall that the dual is

$$\max b^T y \text{ such that } A^T y \leq c, y \geq 0.$$

3 Ski Rental

We use the primal-dual approach to design a deterministic algorithm and a randomized algorithm for ski rental. Recall the ski rental problem: Suppose you are on a ski trip, and you let your friends dictate how long your group will continue skiing. Each day, you need skis, and there are two options

- rent skis: 1 per day
- buy skis: B, one time payment at any time.

This is equivalent to the following primal:

$$\min(Bx + \sum_{i=1}^{k} z_i)$$

under the following constraints

$$\begin{aligned} \forall i \in [k], x + z_i &\geq 1 \text{ (need to cover every day)} \\ x &\geq 0 \\ \forall i \in [k], z_i &\geq 0. \end{aligned}$$

In the primal set-up, this means that

$$A := \left[\begin{array}{c|ccc} 1 & 1 & & \cdots \\ 1 & & 1 & \cdots \\ \vdots & \vdots & & \ddots \\ 1 & & & 1 \end{array}\right] \left[\begin{array}{c} x \\ z_1 \\ \vdots \\ z_k \end{array}\right] \geq \left[\begin{array}{c} 1 \\ 1 \\ \vdots \\ 1 \end{array}\right] := b,$$

with objective vector

$$c := \left[\begin{array}{c} B \\ 1 \\ \vdots \\ 1 \end{array}\right].$$

Notice that the number of constraints and the number of variables are flipped in the primal and the dual. By definition, the dual is

$$\max \sum_{i=1}^{k} y_i$$

under the following constraints

$$\begin{aligned} \sum_{i=1}^{k} y_i &\leq B \\ \forall i, y_i &\leq 1 \\ y &\geq 0. \end{aligned}$$

3.1 Deterministic Algorithm

Our deterministic primal/dual algorithm works as follows: We initialize all of the variables in the primal and in the dual to 0. When we see a new constraint $x + z_i \geq 1$,

1. If the constraint is already satisfied, then we do nothing.
2. Otherwise, we continuously increment y_i until a dual constraint becomes tight, and then we set the corresponding primal variable to 1.

Notice that case 1) occurs iff $x = 1$. Notice also that y_i is incremented until either $\sum_{i=1}^{k} y_i = B$ or $y_i = 1$. It turns out that if we unravel this algorithm, then it is equivalent to the algorithm covered previously in the course.

We analyze this algorithm by applying Theorem 1. First, we show that $\alpha = 1$. Notice that if $x > 0$, then we must have $\sum_{i=1}^{k} y_i = B$ since this constraint might have been tight at some step, and the values have not been changed since that step. If $z_i > 0$, then we must have had $y_i = 1$. Now, we show that $\beta = 2$. This follows from the fact that $x + z_i \leq 2$ since $x, z_i \leq 1$. Thus, by Theorem 1, the algorithm is 2-competitive.

3.2 Randomized Algorithm

We first design a $\left(1 + \frac{1}{(1+\frac{1}{B})^B - 1}\right)$-competitive algorithm to find a fractional solution for online ski rental, and we will use this to create a randomized algorithm with the same competitive ratio. Notice that we can disregard the case where $B < 1$, since it is always best to rent everyday. Likewise, if $B = 1$, then it is always best to buy on the first day. For the interesting cases where $B > 1$, we have that $\left(1 + \frac{1}{(1+\frac{1}{B})^B - 1}\right) < 2$, so this is better than our deterministic algorithm. Using the approximation that $(1 + \frac{1}{x})^x \approx e$ for large x, we see that the algorithm that approaches $\frac{e}{e-1}$-competitive in the limit as $B \to \infty$.

The algorithm works as follows: We initialize all variables in the primal and the dual to 0. When we see a new constraint $x + z_i \geq 1$,

1. If it is already satisfied, do nothing.
2. Otherwise set

$$\begin{aligned} z_i &\leftarrow 1 - x \\ x &\leftarrow \left(1 + \frac{1}{B}\right) x + \frac{1}{cB} \\ y_i &\leftarrow 1 \end{aligned}$$

   (we choose c later).

We analyze this algorithm without approximate slackness. We want to upper bound $\frac{\text{cost of primal}}{\text{profit of dual}}$. It suffices to bound $\frac{\Delta\text{cost of primal}}{\Delta\text{profit of dual}}$ in each time step. In the case 1), neither quantity changes. In case 2), we have that

$$\begin{aligned} \Delta\text{dual} &= 1 \\ \Delta\text{primal} &= B\Delta x + z_i \\ &= B\left(\left(1 + \frac{1}{B}\right) x + \frac{1}{cB} - x\right) + (1 - x) \\ &= 1 + \frac{1}{c}. \end{aligned}$$

One might wrongly think we can take $c = \infty$ to achieve 1-competitiveness. However, we cannot deduce that $c^T x \leq \left(1 + \frac{1}{c}\right) b^T y$ or $\left(1 + \frac{1}{c}\right) b^T y \leq \left(1 + \frac{1}{c}\right) OPT_D$ unless the x is primal feasible and y is dual feasible.

Notice that our solution to the primal is always feasible since we set z_i to be $1-x$ and then increase x, so we definitely have that $x = z_i \geq 1$. We set c so our solution to the dual is feasible. We require that $\sum y_i \leq B$ – this means the Bth day must end with $x \geq 1$. How long is it before $x \geq 1$ in terms of c? Let $q = 1 + \frac{1}{B}$ and $r = \frac{1}{cB}$. Then we have that the new x is $qx + r$ where x is the old x. Observe that $x = 0$ on day 0, $x = r$ on day 1, $x = (q+1)r$ on day 2, and on day B, we have that

$$\begin{aligned} x &= r \sum_{i=0}^{B-1} q^i \\ &= r \frac{q^B - 1}{q - 1} \\ &= \frac{1}{cB} \left(\frac{\left(1 + \frac{1}{B}\right)^B - 1}{c} \right). \end{aligned}$$

We need $c \leq \left(1 + \frac{1}{B}\right)^B - 1$, so we set $c = \left(1 + \frac{1}{B}\right)^B - 1$.

Now, we show how to convert this fractional solution into an integral primal solution such that

$$\mathbb{E}(\text{cost}) \leq \left(1 + \frac{1}{c}\right) \cdot OPT.$$

Before the first day, we draw a random variable α uniformly in $[0, 1]$. Let $x_i = \Delta x$ on day i. We call the ith interval $[\sum_{j=1}^{i-1} x_j, \sum_{j=1}^{i} x_j)$. We buy on the day corresponding to the interval that α lands in. Let x be the final value of the variable at the end of the sequence. Then we have that

$$\begin{aligned} \mathbb{E}(\text{costs}) &= \mathbb{E}(\text{money spent buying}) + \mathbb{E}(\text{money spent renting}) \\ &= B\mathbb{P}[\text{buy}] + \sum_{i=1}^{k} \mathbb{P}[\text{rent on day } i] \\ &= Bx \sum_{i=1}^{k} (1 - (x_1 + x_2 + \ldots + x_{i-1})) \\ &= Bx + \sum_{i=1}^{k} z_i \end{aligned}$$

which is the objective function of the fractional solution. This means that we have designed a $\left(1 + \frac{1}{(1+\frac{1}{B})^B - 1}\right)$-competitive randomized algorithm as desired.

4 Set Cover

Recall that in set cover, we are given a collection of sets $C = S_1, S_2, \ldots S_m$ that are each a subset of $[n]$. We want to find the smallest sized subcollection $C' \subseteq C$ such that $\cup_{S \in C'} S = [n]$. Finding the optimal solution is NP-hard even in the offline case.

In the online version of set cover, we are given the names of the m sets but not the items that they contain. The items $i \in [n]$ come online. When we see i, we are told which sets contain i. The primal is

$$\min \sum_{S \in C} x_S \text{ such that } \forall i \in [n], \sum_{S \ni i} x_S \geq 1, x \geq 0.$$

The dual is

$$\max \sum_{i=1}^{n} y_i \text{ such that } \forall S \in C, \sum_{i \in S} y_i \leq 1, y \leq 0.$$

The following $\log m$-competitive algorithms for online set cover are due to [1]. In this lecture, we cover algorithms to obtain a fractional solution, but next lecture we will use these algorithms to design an algorithm to find an integral solution.

4.1 Primal Only View

The primal-only algorithm works as follows: We initialize $x = \left(\frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right)$. (The initial primal cost is 1). When we see an item i contained in $S_1, \ldots S_r$, we do the following:

1. If $\sum_{j=1}^{r} x_{S_j} \geq 1$, then we do nothing.
2. Otherwise, for each j in parallel, we evolve according to $\frac{dx_S(t)}{dt} = x_S(t)$ for one time step. (Note that if we start at $x(t) = x_0$ at time 0 then we evolve as $\frac{dx(t)}{dt} = x(t)$, then after T time steps, we have $x(T) = e^T x_0$.)

Now, we claim that the total number of time steps during which an evolve is performed throughout the algorithm is $\leq OPT \log m$, where OPT is the number of sets in the minimum set cover. For every item i, there exists S' in the minimum set cover that contains i. If we evolve at the time step corresponding to item i, then $x_{S'}$ will be multiplied by e. However, we will never evolve any set S at more than $\log m$ time steps – since x_S starts at $1/m$ and multiplies by e every time step. The desired bound follows from the fact that at least one set in the optimal set cover is evolved at each time step during which the algorithm evolves.

Notice that

$$\frac{d(\text{objective})}{dt} = \frac{d(x_{s_1}(t) + x_{s_2}(t) + \ldots + x_{s_r}(t))}{dt} \leq 1$$

since we only evolve when the constraint is not satisfied. This means that we bound the primal objective by $OPT \log m + 1$, so we obtain an algorithm that is $\log m$-competitive.

4.2 Primal Dual View

We now modify the primal-only algorithm as follows. When we evolve x_S, we also evolve y_i as $\frac{dy_i}{dt} = 1$ for one time step (each y_i is initialized to 0). Notice that $\sum_{i \in S} y_i$ counts the number of times that S has been evolved and $\sum_{i=1}^{n} y_i$ counts the total number of evolving time steps.

One might wrongly conclude that this implies that $OPT_{\text{P}} \leq$ objective $\leq$ number of evolving time steps+ $1 \leq OPT_D + 1$ and combining this with the weak duality theorem, conclude that $OPT_P = OPT_D$ or $OPT_P = OPT_D + 1$. However, this analysis incorrectly assumes that y is dual feasible.

We tweak y to actually be a feasible solution by scaling. We claim that $\frac{y}{\log m}$ is dual-feasible. It suffices to show that $\sum_{i \in S} y_i \leq \log m$ – which follows from the fact that no set can be evolved more than $\log m$ times. Let R be the dual profit of y. Then we have that

$$OPT_D \leq OPT_P \leq R \log m + 1 \leq OPT_D \log m + 1$$

. This means that $R \log m$ gives us a $\log m$-competitive algorithm for computing OPT_P.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, Joseph Naor. The Online Set Cover Problem. *SIAM J. Comput*, 39(2):361–370, 2009.