CS 294-128: Algorithms and Uncertainty	Lecture 3	Date: August 31, 2016
Instructor: Nikhil Bansal		Scribe: Antares Chen

1 Introduction

We finish our discussion of online paging from last lecture with a $\Omega(\log k)$ lower bound on the competitive ratio of any randomized online algorithm. We then begin discussing the online primal-dual method. First, we review basic LP duality and then formulate linear programs in an online setting. We then discuss how this framework can model various problems. Finally, we discuss the intuition behind the primal-dual method, and then derive an algorithm for online set-cover.

2 Online paging

In lecture 2, we covered the online paging problem which is set up as follows. Suppose we have a system with a fast memory (cache) of size k and are given a universe of pages $U = [n]$ with a request sequence $\sigma_1, \sigma_2, ..., \sigma_m$. If a request σ_i exists within the fast memory, then it is returned with no cost. Otherwise, a page fault occurs and a page from fast memory must be evicted with cost 1. Our objective is to minimize the number of pages evicted.

2.1 Paging lower bound for randomized algorithms

Recall the definition of an oblivious adversary. This adversary constructs the request sequence a priori to actions made by the online algorithm. The adversary incurs the cost of the optimal offline request sequence.

Theorem 1 *Any randomized online paging algorithm is $\Omega(\log k)$-competitive against an oblivious adversary.*

Proof: Let $n = k+1$ and consider an adversary that requests a page uniformly at random at each step. For any randomized algorithm $\mathcal{A}$, the probability of a page fault is given by $\frac{1}{k+1}$. Thus for a sequence of T requests, the expected cost will be the following.

$$\mathbf{E}\big[\text{cost}(\mathcal{A})\big] = \frac{T}{k+1}$$

We now show that the expected offline cost (over all random sequences of length T) is $O(\frac{T}{k \ln k})$. Recall theorem 5 of lecture notes 2: the optimal offline strategy OPT is one where the page needed furthest in the future is evicted. Consider an arbitrary request sequence. We divide the sequence into phases where all $k+1$ pages appear at least once during the phase.

The following offline caching policy incurs exactly cost 1 per phase. Initialize the fast memory with everything except last page of the first phase. Then upon a page fault, evict the last page of the next phase. It is easily verified that each phase incurs exactly cost 1. Thus across T requests,

for T large enough, the expected cost is given by the following (formally this statement requires renewal theory, but it also follows from elementary considerations so we ignore this here).

$$\frac{\mathbf{E}\big[\text{cost}(\textsc{opt})\big]}{T} = \frac{1}{\text{expected length of phase}}$$

Let X be the length of each phase which is also the number of requests until all pages are seen. Determining $E[X]$ amounts to solving the coupon collector problem. We split X into the sum of random variables X_i representing the number of requests required to first see page i. The probability of seeing the i^{th} new page is equal to the following.

$$P\big(\text{seeing the } i^{\text{th}} \text{ new page}\big) = \frac{(k+1)-i+1}{k+1}$$

Note that each $X_i \sim \text{Geom}(p = \frac{(k+1)-i+1}{k+1})$, thus we have the following for the expected value.

$$\mathbf{E}[X_i] = \frac{k+1}{(k+1)-i+1}$$

Finally the overall expected value is given by the following.

$$\mathbf{E}[X] = \sum_{i=1}^{k+1} \frac{k+1}{(k+1)-i+1} = (k+1)\Big(\frac{1}{k+1} + \frac{1}{k} + ... + \frac{1}{2} + 1\Big) \leq (k+1)\ln(k+1)$$

This gives the claimed $\Omega(\log k)$ lower bound. □

It is possible to prove a H_k lower bound for the randomized online paging algorithms [3]. The 1-bit LRU scheme provided in the previous lecture is actually a factor of 2 off from optimality.

3 Basic linear program duality

Before we discuss the online primal-dual method, let us discuss basic linear programming theory. In linear programming, we wish to find values x_i that minimizes an objective function with respect to the linear constraints below.

$$\begin{aligned} P &= \min \sum_i c_i x_i \\ \text{s.t.} \forall j &: \sum_i a_{ji} x_i \geq b_j \\ a_{ji}, b_j, c_i, x_i &\geq 0 \end{aligned}$$

We will consider the setting where a_{ji}, b_j, c_i are all non-negative. Such programs are called *covering* LPs. Let us call above the primal LP.

Consider weak duality for an LP in the form above. Taking a linear combination with multipliers $y_j \geq 0$, we obtain that any feasible solution x must satisfy

$$\sum_i \sum_j y_j a_{ji} x_i \geq \sum_j b_j y_j$$

If we can clevery pick y_j to satisfy $\sum_j y_j a_{ji} \leq c_i$ for all i, we obtain

$$\sum_j b_j y_j \leq \sum_i c_i x_i$$

As this sum lower bounds any feasible primal value, it also lower bounds the optimal objective value P^*. The above conditions on y also give an LP. The *dual problem* is thus defined as the following.

$$\begin{aligned} D = \max & \sum_j b_j y_j \\ \text{s.t.} \forall i : & \sum_j y_j a_{ji} \leq c_i \\ & y_j \geq 0 \end{aligned}$$

This immediately gives us the "weak-duality" theorem.

Theorem 2 *Let x_i and y_j be feasible solutions to the primal and dual problems respectively. We have the following.*

$$\sum_i c_i x_i \geq \sum_j b_j y_j$$

Equivalently, if P^* and D^* are the minimal and maximal objective values for the primal and dual linear programs respectively, we have $P^* \geq D^*$.

When a_{ji}, b_j, c_i are non-negative as in the covering LP, the dual problem is also known as a *packing* LP.

Interestingly, LPs also satisfy "strong duality" (under the reasonable condition that they have finite optimal values). We skip the proof here, and refer to any textbook on optimization.

Theorem 3 *If either the primal or dual has a finite optimal value, then so does the other and $P^* = D^*$.*

Finally, we mention *complimentary slackness*, which gives an important characterization of optimum primal and dual feasible solutions. These are often very useful while designing primal-dual algorithms, and can often guide how the primal/dual variables must be raised.

The following result directly follows from the weak duality computation we performed above.

Theorem 4 (Duality Gap) *Let x be some primal feasible solution and y be some dual feasible solution. Let $s_j := \sum_i a_{ji} x_i - b_j$ denote the slack in the j-th primal constraint and $t_i := c_i - \sum_j y_j a_{ji}$ the slack in the i-th dual constraint. Then the duality gap is given by $P - D = c \cdot x - b \cdot y = s \cdot y + t \cdot x$.*

So $P - D = 0$ if and only if $s_j y_j = 0$ for each j and $t_i x_i = 0$ for each i.

3.1 Offline primal dual framework

The primal-dual approach is a widely used method for finding approximate (offline) solutions to NP-Hard problems. The idea is as follows. Formulate the problem to be solved as a linear program (say of the covering form). Next, suppose one can find a $\{0, 1\}$ solution (or even a fractional solution) in some iterative way based on updating the primal and dual solutions x and y, such that

- The increase in primal objective at current step, $c\Delta(x)$ is at most α times the increase the dual objective $b\Delta(y)$.
- At the end, the primal solution x^f is feasible, and the dual y^f is β-feasible. The latter means that $\sum_j y_j^f a_{ji} \leq \beta c_i$ for each i

By weak duality, this directly implies that the solution x^f is an $\alpha\beta$-approximation. As by the second property $\frac{y^f}{\beta}$ is dual feasible and hence $P^* \geq b \cdot \frac{y^f}{\beta}$. By the first property $c \cdot x^f \leq \alpha b^T \cdot y^f$. Together this gives $c \cdot x^f \leq \alpha\beta P^*$.

4 Online primal-dual framework for linear programming

We now consider the online variant of linear programming. While this problem may seem artificial initially, we will later see several applications.

Covering Version We first describe the version for covering problems. We can assume the objective function is known in advance (or the coefficient c_i of x_i becomes known when the first constraint arrives with $a_{ji} > 0$). However, the LP is initially empty with the covering constraints arriving over time.

The algorithm must maintain a solution x over time, satisfying the following rules for updating x: (1) a constraint must be satisfied by x upon arrival, and (2) no variable x_i may be decreased over time.

Strictly speaking, as x is a function of time, we must use $x_i^{(t)}$ to denote the value of variables i at time t. However, unless there is a cause for confusion, we will usually drop t for convenience of notation.

Packing Version One can similarly consider a dual *packing* version of the problem. Here, the variables y_j arrive over time, and the right hand side values for c_i are known in advance. Upon the arrival of y_j, the entire column corresponding to y_j, i.e. all the entries a_{ij} involving y_j are made available with coefficient b_j of y_j also revealed. The algorithm can only increase the most recently arrived y_j, and eventually must produce a solution that (approximately) satisfies all the packing constraints.

Finally, for both the online covering and packing setting, the goal of each problem is to minimize and maximize its respective objective function.

4.1 Applications

We now see how various problems can be modeled in this framework.

Ski-rental problem. Each day we are given the choice of renting skis for cost 1 or buying it for cost B. We do not know how many days we will ski in advance. What should be our strategy?

Let z_j for $j = [k]$ be the indicator variable for if we rent the skis on day j. Let x be the indicator variable for if we buy the skis. Consider the following LP. The constraints require that we must

either rent each day or have bought them.

$$\begin{aligned} \min B \cdot x + \sum_{j=1}^{k} z_j \\ \text{s.t.} \forall j \in [k] : x + z_j \geq 1 \\ x \geq 0; z_j \geq 0 \end{aligned}$$

There is a deterministic 2-competitive algorithm, and this is the best possible. With randomization, $\frac{e}{e-1}$ can be achieved which is asymptotically optimal. The primal-dual approach allows us to obtained these bounds in a simple manner.

Online set cover In the online set cover problem we are given a collection of sets $\mathcal{C} = \{S_1, ..., S_n\}$ where each S has cost c_S. Elements arrive one by one in some online manner, and it must be covered by some set immediately upon arrival. The goal is to minimize the overall cost compared to the offline optimal set cover for the elements that have arrived.

One has the following LP where the constraints arrive over time. Denote an element of S_i as e and let $x_S = 1$ if $S \in \mathcal{S}$, otherwise $x_S = 0$. The linear program is as follows.

$$\begin{aligned} \min \sum_{S \in \mathcal{S}} c_S x_S \\ \text{s.t.} \forall e : \sum_{S : e \in s} x_S \geq 1 \\ x_S \geq 0 \end{aligned}$$

We will see a fractional $O(\log n)$ competitive algorithm, and an integral $O(\log n \log m)$ competitive algorithm based on an online rounding of this LP. It is also known that no polynomial time algorithm can do better [5].

Paging We can also model the paging problem. Interestingly, the primary ingenuity here is in computing up with a covering LP formulation. We shall see in the next lecture how this gives another $O(\log k)$ competitive algorithm. An advantage of this approach, is that it also gives an $O(\log k)$ competitive algorithm for weighted paging, which was an open problem for a long time, without any additional effort.

Online network design Consider the online steiner tree problem. Given a graph with root r and edge costs, some terminals $t_1, \ldots, t_k$ that arrive over time, and as soon as a terminal arrives one needs to connect it to the root. The goal is to minimize to cost of the tree constructed.

To do this, we will construct an exponential sized LP, which turns out, can still be solved in polynomial time. Let x_e be variables for each edge that indicate if e lies in the tree. When a terminal t_i arrives, we add a constraint for each r-t_i cut in the graph.

$$\begin{aligned} \min \sum_{e} c_e x_e \\ \forall S \subset V \text{ with } t_i \in S, r \notin S : \sum_{e \in S} x_e \geq 1 \\ x_e \geq 0 \end{aligned}$$

Congestion minimization and call admission Recall the routing problem, where edges have capacity $c(e)$, demand $b_i(e)$ could be edge dependent and requests for some s_i, t_i path arrive over time. The goal is to route online, minimizing the maximum edge congestion. One can also consider a closely related problem where each request has profit p_i and the goal is to maximize profit incurred by accepted requests subject to congestion constraints.

Let us consider the latter problem. For congestion minimization the reader may refer to [2]. We consider the following (exponential size) LP. Define a variables $y(i, P)$ for each request i, and possible $s_i - t_i$ path P_i. Let $\mathcal{P}_i$ be the set of all possible s_i-t_i paths.

$$\begin{aligned} \max \sum_i \sum_{P \in \mathcal{P}_i} p_i y(i,P) \\ \text{s.t.} \forall e : \sum_i \sum_{P: e \in P, P \in \mathcal{P}_i} b_i(e) y(i,P) \leq c_e \\ \forall i : \sum_i \sum_{P \in \mathcal{P}_i} y(i,P) \leq 1 \\ \forall i, P : y_{i,P} \geq 0 \end{aligned}$$

4.2 General results

To apply the above framework to an online problem, usually there are the following steps. First, design a good *fractional* algorithm for the online LP problem. Second, convert this to a randomized algorithm. Sometimes this is easy as the fractional solution may already be viewed as a randomized algorithm (e.g. ski rental). But in other cases, one has to do an additional *online rounding* step, which is usually problem specific. We will see examples of this later.

We first note some general results about solving the online LP problem.

Theorem 5 *For a covering problem with $a_{ji} \in \{0, 1\}$, there is an $O(\log n)$ competitive algorithm, where n is the number of variables. If each row has sparsity d, this can be improved to $O(\log d)$.*

Both these bounds are tight in general. See for example [2]. Similar results also holds for general matrix entries a_{ji} [2, 4]. For packing problem one can show the following guarantee.

Theorem 6 *For a packing problem with arbitrary entries a_{ji}, for any $B \geq 1$ there is an $O((\log n + \log(a_{\max}/a_{\min}))/B)$ competitive algorithm, where B is the violation of the packing constraints, $a_{\max}$ is the maximum entry a_{ji} and $a_{\min}$ is the minimum non-zero entry.*

In general the dependence on $\frac{a_{\max}}{a_{\min}}$ cannot be avoided, even if the LP has just one constraint. Consider the LP with objective $y_1 + y_2 + \ldots + y_n$ and a single constraint $y_1 + \frac{1}{2} y_2 + \frac{1}{4} y_3 + \ldots + \frac{1}{2^{n-1}} y_n \leq 1$. Every time a new variables y_i arrives, it must be set to $\Omega(2^i)$ to be constant competitive, and the constraint must be violated by $\Omega(n)$.

4.3 Online primal-dual approach intuition

We consider the covering variant and describe the main idea underlying most applications of this approach.

Upon arrival of the t-th constraint in the primal setting, we get a corresponding dual variable y_t. We raise y_t as long as the constraint is unsatisfied, and raise the primal variables so that the following hold:

1. The increase in the primal cost is at most α times the dual cost, i.e. $\Delta P \leq \alpha \Delta D$.
2. Eventually y is β-infeasible, i.e. $\frac{y}{\beta}$ is dual feasible.

As discussed above, clearly this would imply an $\alpha\beta$-approximate primal solution.

Let us try to see how one might ensure the first property. Suppose the t-th constraint is $\sum_i a_{ti} x_i \geq b_t$. The corresponding dual variable y_t appears as $b_t y_t$ in the dual objective. If we increase y_t by dy, the dual increases by $b_t dy$. If we are raising the primal variables, it must be that $\sum_i a_{ti} x_i < b_t$, and so the dual increases by at most $\sum_i a_{ti} x_i dy$. To relate this to the primal increase $\sum_i c_i(dx_i)$, a natural choice is to set $dx_i = a_{ti}(\frac{x_i}{c_i})dy$. This would ensure that

$$\Delta P = \sum_i c_i(dx_i) \leq \Big(\sum_i a_{ti} x_i\Big) dy \leq b_t dy = \Delta D$$

This gives a multiplicative update-type rule. To ensure that variables rise when they start from 0, one adds a small additive term to get the rule $dx_i = \frac{a_{ti}}{c_i}(x_i + \eta_i)$ e.g. if $\eta_i = \frac{a_{\min}}{n a_{\max}}$, the effect of η is negligible and one still has $\Delta P \leq 2\Delta D$. Usually, the competitive ratio depends on the smallest η as $\ln(1/\eta)$, and hence it is often that one must use the problem structure to make η as large as possible.

Let us consider a few examples.

4.4 Application to set cover

Let us now apply the online primal-dual method to set cover. Recall the LP formulation given $x_s, y_e \geq 0$.

$$\text{primal} : P = \min \sum_{s \in \mathcal{S}} c_s x_s \qquad \text{dual} : D = \max \sum_e y_e$$
$$\text{s.t.} \forall e : \sum_{s : e \in s} x_s \geq 1 \qquad \text{s.t.} \forall s \in \mathcal{S} : \sum_{e \in s} y_e \leq c_s$$

As discussed above, consider the following continuous algorithm.

Algorithm:
Initialize all $x_s = 0$. Whenever a new primal constraint $\sum_{s:e \in s} x_s \geq 1$ arrives, the corresponding dual variable y_e is created. We initialize $y_e = 0$ and update as follows:

While $\sum_{s:e \in s} x_s < 1$ do: For each $\{s : e \in s\}$ increase $\frac{dx_s}{dy_e} \frac{1}{c_s}\big(x_s + \eta\big)$

To determine the competitive ratio for this algorithm, we show the properties mentioned above.

Theorem 7 *Let $\eta = \frac{1}{n}$. The algorithm produces maintains the following properties: (1) It produces a primal feasible solution. (2) For every iteration, we have $\Delta P \leq 2\Delta D$. (3) Upon termination, each packing constraint in the dual program is violated by at most a $O(\log n)$ factor.*

Proof: Let P and D be the primal and dual solutions and $\Delta P, \Delta D$ be the change in the primal and dual solutions across the arrival of a covering constraint.

The first property follows by design, as we keep increasing the primal variables until the current element e is covered.

For the second property, suppose we increase y_e by dy_e. Then $\Delta D = dy_e$. For ΔP we have the following.

$$\Delta P = \sum_{s:e\in s} c_s dx_s \leq \sum_{s:e\in s} c_s \left(\frac{1}{c_s}\Big(x_s + \frac{1}{n}\Big)dy_e\right) = \sum_{s:e\in s}\Big(x_s + \frac{1}{n}\Big)dy_e \leq 2dy_e$$

The last inequality follows as there can be at most n elements in s, and the variables are increased only as long as the constraint is unsatisfiable.

Finally, we need to bound the dual violation. The dual has a constraint for each set S. We wish to bound the total change in y_e for all $e \in S$ that arrive over the course of the algorithm. Let $x_s^{(t)}$ and $y_e^{(t)}$ be the values of x_s and y_e after the execution of the algorithm on the constraint arriving at t. As the change in x_s with respect to y_e is given by $\frac{dx_s}{dy_e} = \frac{1}{c_s}\big(x_s + \eta\big)$, solving for this first order differential equation obtains

$$y_e = c_s \ln\big(x_s + \eta\big) + C$$

As y_e is only raised during step t, we have the following.

$$\begin{aligned} y_e = y_e\Big|_t^{t-1} &= c_s \ln\big(x_s + \eta\big)\Big|_{t-1}^{t} \\ &= c_s \ln\Big(\frac{x_s^{(t)} + \eta}{x_s^{(t-1)} + \eta}\Big) \end{aligned}$$

Suppose the algorithm terminates after T steps. Summing up over both the sides gives

$$\sum_{e:e\in S} y_e = c_s \sum_t \ln\Big(\frac{x_s^{(t)} + \eta}{x_s^{(t-1)} + \eta}\Big) = c_s \ln\Big(\frac{x_s^{(T)} + \eta}{x_s^{(0)} + \eta}\Big)$$

Now $x_s^{(T)} - x_s^{(0)} \leq 1$, as all the right hand sides of the LP are 1 and the algorithm will never increase any x_s above 1. This gives

$$\sum_{e:e\in S} y_e \leq c_s \ln\Big(\frac{1+\eta}{\eta}\Big) \in O\big(\log n\big)$$

It follows that the dual constraints are only ever violated by a factor of $O(\log n)$. □

Note that if the maximum set size was d, we could choose $\eta = \frac{1}{d}$ in the analysis above to get an $O(\log d)$ competitive algorithm.

4.5 Online Rounding for set cover

The above algorithm gives a fractional algorithm. To obtain an integral algorithm, we can do the following at a cost of an additional $\log m$ factor on top. For each set S, pick $\alpha_S \in [0, 1]$ uniformly and independently. Fix some $c > 1$. As the fractional algorithm executes, have the deterministic

algorithm pick S when x_S first exceeds $(c \log m)\alpha_S$. Clearly, the probability that an element e is not covered is at most

$$\prod_{S:e \in S} (1 - x_S) \leq \exp(-x_S) \leq m^{-c}$$

Every element is covered with probability at least $1 - m^{1-c}$. Otherwise, if an element e is uncovered, just pick a set containing e from the distribution x_S. The probability that set S is picked is at most $c \log(m) x_S + m^{1-c} x_S = O(\log m) x_S$.

Alon et al. [1] gave a deterministic algorithm for online rounding based on derandomizing the above using pessimistic estimators.

References

[1] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder and Joseph Naor. The online set cover problem, *Symposium on Theory of Computing*, 2003, 100–105.

[2] Niv Buchbinder and Joseph Naor. The Design of Competitive Online Algorithms via a Primal-Dual Approach, *Foundations and Trends in Theoretical Computer Science*, 3(2-3), 93–263, 2009.

[3] Fiat, Amos, et al. "Competitive paging algorithms." *Journal of Algorithms* 12.4 (1991): 685-699.

[4] Anupam Gupta and Viswanath Nagarajan. Approximating Sparse Covering Integer Programs Online, *Math. Oper. Res.*, 39(4), 998–1011, 2014.

[5] Korman, Simon. On the use of randomization in the online set cover problem. Diss. Weizmann Institute of Science, 2004.