Advanced Algorithm Fall 2014

Lecture 9 — September 30, 2014

Prof. Jelani Nelson *Scribe: Yi-Hsiu Chen*

1 Overview

In the last lecture, we started discussing online algorithms. In this lecture, we will continue on this topic by proving the marking algorithm is a $2H_k$-competitive algorithm [FKL$^+$91]. Then we will introduce a generic framework – the online primal/dual method of [BN05] – and show how the 2-competitive algorithm for the ski rental problem can be understood under this framework.

2 Marking Algorithm

2.1 Algorithm

Let's recall the marking algorithm. It's an implementation of 1-bit LRU from last lecture, where we evict a uniformly random page. Namely

1. Initially, all pages are unmarked.
2. When bringing page p into cache, evict a page chosen uniformly at random from the set of unmarked pages (if all pages are marked, then first unmark all pages).
3. Mark the requested page.

2.2 Analysis

Denote the marking algorithm as MARK and the optimal one as OPT. Our goal is to show that the marking algorithm is a $2H_k$-competitive algorithm. Namely, for a request sequence s

$$\mathbb{E}[cost_{\text{MARK}}(s)] \leq 2H_k \cdot cost_{\text{OPT}}(s) \text{ where } H_k = \sum_{i=1}^{k} \frac{1}{i} = \ln k + O(1)$$

We will divide the request sequence into phases. A new phase starts when all pages are unmarked. Let's define the following terms:

- A page p is **clean** if it wasn't requested in the last phase and not yet requested in this phase.
- A page p is **stale** if it was requested in the last phase but not yet in this phase.

We also define the variables

- L_i: Number of clean pages requested in phase i.
- S_i: Number of stale pages requested in phase i.
- D_i: Number of pages in OPT's cache at the beginning of phase i which aren't in MARK's cache.
- D'_i: Number of pages in OPT's cache at the end of phase i which aren't in MARK's cache (note $D'_i = D_{i+1}$).

Then we will prove the $2H_k$-competitive by showing in phase i,

1. The cost of OPT is at least $\frac{1}{2}L_i + \frac{1}{2}(D'_i - D_i)$
2. The expected cost of MARK is no more than $H_k L_i$

Once we proved those, we can conclude that

$$cost_{\text{OPT}} = \sum_i \frac{1}{2}L_i + \frac{1}{2}(D'_i - D_i) \geq \sum_i \frac{1}{2}L_i \geq \frac{1}{2H_k}\sum_i H_k L_i \geq \frac{1}{2H_k} cost_{\text{MARK}}$$

proof of 1. For the L_i clean pages, MARK has none of them, so OPT can have at most D_i at the beginning. Thus the cost is not less than $L_i - D_i$. On the other hand, there are D'_i pages in MARK but not in OPT at the end of phase, which means there are at least D'_i evictions. In sum up, the cost of OPT in phase i is $\geq \max(L_i - D_i, D'_i) \geq \frac{1}{2}(L_i - D_i + D'_i)$. □

proof of 2. Only clean or stale requests can cause eviction for MARK. Each of the L_i clean requests causes an eviction. Let's consider a stale request. Suppose we are currently serving a stale request, and we have already had s stale requests and c clean requests prior to it in the current phase. At the moment, the cache consists of c clean, and s pages from the last phase that were stale-requested this phase. Thus of the $k - s$ pages left of the last phase, a random set of c were evicted this phase by the c clean requests. Therefore the expected cost of the stale request is

$$\mathbb{P}[\text{this pages has been evict}] = \frac{c}{k-s} \leq \frac{L_i}{k-s}$$

Then the total expected cost during phase i of serving stale requests is

$$\mathbb{E}[cost_{\text{MARK}} \text{ in phase } i] \leq L_i \cdot \sum_{s=0}^{k-L_i-1} \frac{1}{k-s} = L_i \sum_{j=L_i+1}^{k} \frac{1}{j} = L_i(H_k - H_{L_i}) \leq L_i(H_k - 1)$$

Note $H_{L_i} \geq 1$ since $L_i \geq 1$ (there is at least one clean request per phase, namely the first request that starts the phase).

Then adding the L_i clean requests, we find that the total expected cost is $L_i H_k$. □

2.3 Lower Bound

Theorem 1. *Any randomized algorithm has competitive ratio* $\Omega(\lg k)$.

Proof. Let the number of pages in universe be $n = k + 1$. Consider a uniformly random request sequence. Then for every request, the probability of page fault for an algorithm is $\frac{1}{k+1}$. So on a length m sequence, the algorithm expects to fault $m/(k+1)$ times. In fact for m sufficiently large, it will fault $\Omega(m/(k+1))$ times with high probability on a random sequence of length m.

On the other hand, OPT gets to see the entire random sequence from the beginning. When OPT must evict a page it will evict the page p that appears farthest in the future. Thus before that, it will see all other pages before seeing p again. Therefore by a coupon collector argument, the expected number of steps to see all pages is $k \ln k$ (in fact it is $\Theta(k \ln k)$ with high probability).

In summary, on 2/3rds of the random sequences our algorithm makes $\Omega(m/k)$ page faults. Meanwhile on 2/3rds of random sequences, OPT makes at most $O(m/(k \lg k))$ page faults. Therefore by a union bound there exists a sequence in the intersection, for which our competitive ratio is $\Omega(\lg k)$. □

In fact it is possible to prove an H_k lower bound for the competitive ratio (see [FKL$^+$91]). Thus the marking algorithm is suboptimal by a factor of 2. Later McGeoch et. al developed the "partitioning algorithm" [MS91] which is H_k-competitive. After that, a simpler algorithm with simpler analysis achieving H_K competitive ratio was proposed in [ACN00].

2.4 Generalization

If we compare with the optimal algorithm with different size of caches, it's called **resource augmentation**. Suppose OPT has cache of size h, but you have cache k. Young [You91] showed that you can get $2\ln(k/(k-h+1))$-competitive and less than $\ln(k/(k-h+1))$ is impossible. It seems that it is an open problem to close this factor-2 gap in the resource augmentation case.

The more general version is k-server problem. Your algorithm controls k servers, which lie in a metric space X. The request sequence is a series of points in X. For a request point p, you must move a server to lie on p and the cost is the distance. A $2k-1$ competitive deterministic algorithm is known [KP95] and the lower bound for any deterministic algorithm is k-competitiveness. It's still open whether there is an randomized algorithm with competitive ratio polylog(k) (the conjecture is that $O(\lg k)$ is achievable). A recent result by Bansel et al. achieves the ratio $O((\log)^2(\lg n)^3)\log\log n)$ [BBMN11].

Another way to generalize the paging problem is weighted paging: the costs of bringing different pages into cache are different. This makes sense when some pages are more expensive to fetch than others (e.g. some pages are in local memory, whereas others might live in the cache of another core that you then have to bus over). Bansal, Buchbinder, and Naor gave an online algorithm achieving competitive ratio $O(\lg k)$ for this problem [BBN12].

3 Online Primal / Dual Framework

The Primal / Dual is a powerful technique in algorithm design. Buchbinder and Naor extended this technique to developing online algorithms with good competitive ratios [BN05] (see also the book [BN09]). In this section, we will revisit and define some terminology in linear programming, then apply the online primal / dual framework to the ski rental problem.

3.1 Linear Programming Recap

Linear Programming (LP) In linear programming, we want to find vector x to minimize $c^T x$ and subject to $Ax \geq b$ as well as $x \geq 0$. Here c, b, x are vectors and A is a matrix. Inequalities are coordinate-wise.

Covering We say an LP as above is a *covering LP* if all entries of A, b and c are ≥ 0.

Duality Notice that as long as $y \geq 0$, for a feasible solution x we have $y^T Ax \geq y^T b$. Then if $y^T A \leq c^T$, this implies that $y^T b$ is a lower bound on the optimal solution of our LP.

We could imagine choosing y to get the best lower bound possible. This gives the *dual* linear program (as opposed to the original *primal* one). This dual program is to maximize $b^T y$ subject to $A^T y \leq c$ and $y \geq 0$.

Packing The above formulation of the dual linear program is called a *packing LP* if all entries of A, b, c are ≥ 0.

Notice that by construction, we have the following "weak duality" theorem.

Theorem 2 (Weak duality theorem)**.** *If* OPT_P *is the minimum value possible in primal LP* ($c^T x$) *and* OPT_D *is the maximum value possible in dual LP* ($b^T y$)*, then*

$$\text{OPT}_P \geq \text{OPT}_D$$

In fact the following strong duality theorem also holds, but we will neither prove it nor use it today.

Theorem 3 (Strong duality theorem)**.** *If either primal or dual LP has a finite optimal solution, then*

$$\text{OPT}_P = \text{OPT}_D$$

Now, we will look at the primal/dual framework of [BN05] for developing online algorithms for solving covering LPs. The model is that we know the objective function, but we only see constraints one at a time. Initially $x = 0$, and as we see a new constraint we are allowed to increase some subset of the x_i to satisfy the new constraint. Then, at the end of seeing all constraints, we want to be competitive with the optimal solution in hindsight. Note that the entries of x must be monotonically non-decreasing over time in this model.

3.2 Ski Rental Problem

Suppose renting costs 1 dollar while buying cost B dollars. Let $x \in \{0,1\}$ indicate whether we buy and $z_i \in \{0,1\}$ indicate whether we rent on the i-th day.

Translating the problem into an LP, we are trying to minimize $Bx + \sum_{i=1}^{k} z_i$ subject to $\forall i \in [k],\ x + z_i \geq 1$ and $x \geq 0, z \geq 0$.

While we are considering the online problem, we will see rows of A online and have to satisfy constraints online. Our variables throughout the request sequence start from 0 and are monotonically non-decreasing.

Dual: maximize $\sum_{i=1}^{k} y_i$ subject to $\sum_{i=1}^{k} y_i \leq B$ and $0 \leq y \leq 1$ (here on the right hand side we mean the all 1's vector).

Lemma 4 (approximate complementary slackness)**.** *Let x (resp. y) be solutions to the primal (resp. dual) LP. Suppose there exist $\alpha, \beta > 0$ such that*

$$\begin{cases} \forall i,\ \text{if } x_i > 0,\ \text{then } c_i/\alpha \leq (A^T)_i \cdot y \leq c_i \\ \forall i,\ \text{if } y_i > 0,\ \text{then } b_i \leq A_i \cdot x \leq \beta b_i \end{cases}$$

where A_i is the ith row of A. Then

$$c^T x \leq \alpha\beta \cdot b^T y$$

The lemma is useful because

$$b^T y \leq \mathrm{OPT}_D \leq \mathrm{OPT}_P \leq c^T x \leq \alpha\beta b^T y$$

Therefore $c^T x \leq \alpha\beta\, \mathrm{OPT}_P$. In other words, finding feasible solutions x, y to the primal and dual that satisfy good approximate complementary slackness conditions implies that x is a nearly optimal solution.

Having the approximate complementary slackness, here we state the primal-dual 2-competitive algorithm for ski rental:

Algorithm When constraint $x + z_i \geq 1$ arrives, do nothing if already satisfied. Otherwise, increase y_i until some dual constraint becomes tight. Then set the primal variable corresponding to that dual variable to 1.

More precisely: If $\sum_i y_i \leq B$ becomes tight, then set $x = 1$. If $y_i \leq 1$ becomes tight, then set z_i to 1. At the end, we have pair of feasible solutions for the primal and dual, and it is not hard to check that they satisfy approximate complementary slackness with $\alpha = 1, \beta = 2$. Therefore this algorithm is 2-competitive.

In fact if one closely examines the decisions that this algorithm makes, it is exactly the 2-competitive algorithm from last lecture!

References

[ACN00] Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1):203–218, 2000.

[BBMN11] Nikhil Bansal, Niv Buchbinder, Aleksander Madry, and Joseph Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proceedings of the 52nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 267–276, 2011.

[BBN12] Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19, 2012.

[BN05] Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing problems. In *ESA*, pages 689–701. Springer, 2005.

[BN09] 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.

[FKL+91] Amos Fiat, Richard M Karp, Michael Luby, Lyle A McGeoch, Daniel D Sleator, and Neal E Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

[KP95] Elias Koutsoupias and Christos H Papadimitriou. On the k-server conjecture. *Journal of the ACM (JACM)*, 42(5):971–983, 1995.

[MS91] Lyle A McGeoch and Daniel D Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(1-6):816–825, 1991.

[You91] Neal E Young. On-line caching as cache size varies. In *SODA*, volume 91, pages 241–250, 1991.