A Simple Analysis for Randomized Online Weighted Paging

Nikhil Bansal * Niv Buchbinder † Joseph (Seffi) Naor‡

Abstract

We describe a new potential function based proof for an $O(\log k)$-competitive algorithm for weighted paging, and more generally an $O(\log k/(k-h+1))$-competitive algorithm for the weighted (h,k)-paging problem. In contrast to previous results that are based on the online primal-dual framework, our analysis is extremely simple, and we believe that the technique might be useful in the analysis of other related problems.

1 Introduction

The classic online paging problem is the following. We are given a collection of n pages, and a fast memory (cache) which can hold up to k of these pages. At each time step one of the pages is requested. If the requested page is already in the cache then no cost in incurred, otherwise the algorithm must bring the page into the cache, possibly evicting some other page, and a cost of one unit is incurred. In the weighted version of the problem, each page has an associated weight that indicates the cost of bringing it into the cache. This models situations where some pages are more expensive to fetch than others because they may be on far away servers, or slower disks, and so on. The goal is to minimize the total cost incurred by the algorithm.

Unweighted paging is one of the earliest and most extensively studied problems in competitive analysis and is well understood. Any deterministic algorithm must be at least k-competitive, and various k-competitive algorithms are known [13]. When randomization is allowed, it is known that any algorithm must be at least H_k-competitive against an oblivious adversary, where H_k is the k-th Harmonic number. The first $O(\log k)$ competitive algorithm was obtained by Fiat et al. [9] and subsequently optimum H_k-competitive algorithms have been developed [11, 1].

The more general (h,k)-paging problem has also been studied. Here the online algorithm with cache size k is compared to the offline algorithm with cache size h, where $h \leq k$. Here a tight $k/(k-h+1)$-competitive deterministic algorithm is known [13], and with randomization, an essentially tight $2\ln(k/(k-h))$-competitive algorithm is also known [15]. There has been extensive work on paging along several other directions, and we refer the reader to the excellent text by Borodin and El-Yaniv [6] for further details.

For weighted paging, a tight k-competitive deterministic algorithm follows from the work of Chrobak et al. [8] on the (more general) k-server problem on trees. Young [14] gave a tight $k/(k-h+1)$-competitive deterministic algorithm for weighted (h,k)-paging. However, until recently, $o(k)$-competitive randomized algorithms were known only in few special cases [12, 5, 10] of weighted paging. The first $O(\log k)$ competitive algorithm for general weighted paging was obtained by [2], based on the primal-dual framework for online algorithms developed by [7]. The algorithm and its analysis were simplified in the journal version [3],

*IBM T. J. Watson Research, Yorktown Hts, NY 10598. nikhil@us.ibm.com

†Microsoft Research, Cambridge, MA. nivb@cs.technion.ac.il

‡Computer Science Department, Technion, Haifa 32000, Israel. naor@cs.technion.ac.il

and subsequently simplified even further in [4], where an alternate primal-dual formulation for the problem was presented. However, all these results are still based on the primal-dual framework.

In this paper, we use a more traditional potential function based analysis to show $O(\log k)$-competitiveness for an algorithm for weighted paging, and more generally $O(\log k/(k-h+1))$-competitiveness for weighted (h,k)-paging. The algorithm we consider here is identical to that in [4], which is closely related to the one in [2] and [3], but the analysis is different. We feel that the new analysis might be more insightful than the previous primal-dual proofs and could be useful in other contexts.

2 Weighted Caching

We first define the problem formally. There is a universe of pages $1,\ldots,n$ and a cache that can hold up to k pages. To fetch page p into the cache the algorithm incurs a cost of w_p. At each time step t, some page $p_t \in [n]$ is requested. If p_t is already in the cache, the algorithm does nothing and no cost in incurred. Otherwise, the algorithm must fetch p_t into the cache, possibly evicting some other page from the cache. The goal is to minimize the total cost.

Instead of describing a randomized algorithm, it is easier to work with the following fractional view of the problem. Let $x_{p,t}$ denote the amount of page p present in the cache at time t. A feasible solution to the problem must satisfy at each time t that $x_{p_t,t} = 1$ and $\sum_p x_{p,t} \leq k$. The fetching cost is measured fractionally. That is, for any p and time t, it costs $w_p \Delta x_{p,t}$ to increase $x_{p,t}$ by $\Delta x_{p,t}$, and nothing to decrease $x_{p,t}$. Using standard techniques, an α-competitive algorithm in the fractional view can be converted to a 5α-competitive randomized (integral) algorithm [2].

For further ease, we define variables $y_{p,t} = 1 - x_{p,t}$, i.e. the amount of page p *missing* from the cache at time t. The fractional view can equivalently be described as $y_{p_t,t} = 0$ and that $\sum_p y_{p,t} \geq n - k$. It costs $w_p \Delta y_{p,t}$ to increase $y_{p,t}$ by $\Delta_{p,t}$ and nothing to decrease (we can charge for evicting pages instead of fetching, since any page that is fetched must eventually be evicted, except possibly for the last k pages).

Algorithm: Let $\eta = (k-h+1)/k$. At time t when request for page p_t arrives, do the following:

1. Set $y_{p_t,t} = 0$.
2. For each page p, $p \neq p_t$, for which $y_{p,t} < 1$, continuously increase $y_{p,t}$ in proportion to $(1/w_p) \cdot (y_{p,t} + \eta)$, until $\sum_{p \in [n]} y_{p,t} = n - k$.

2.1 Analysis of the Algorithm

Let $C(t)$ denote the set of pages in the offline cache at time t (we can assume that the optimal offline algorithm is deterministic, and hence a page is either completely present or completely absent). Define the following potential function at time t:

$$\Phi(t) = -2 \sum_{p \notin C(t)} w_p \cdot \ln\left(\frac{y_{p,t}+\eta}{1+\eta}\right).$$

As $y_{p,t} \in [0,1]$, note that $-\ln\left(\frac{y_{p,t}+\eta}{1+\eta}\right) \in [0, \ln(1+1/\eta)]$. Moreover, the rate of change of the potential with respect to $y_{p,t}$ is:

$$\frac{d\Phi(t)}{dy_{p,t}} = -2\frac{w_p}{y_{p,t}+\eta}. \tag{1}$$

Let $\text{Opt}(t)$ and $\text{On}(t)$ denote the cost of the optimal offline algorithm and the cost of the online algorithm until time t, respectively. We prove that the competitive factor of the algorithm is $2\ln(1+1/\eta) = 2\ln(1+(k/(k-h+1))) \leq 2\ln(2k/(k-h+1))$. For $h=k$ this implies an $O(\ln k)$-competitive algorithm. To this end, it suffices to show that at any time t the following relation holds:

$$\Delta\text{On}(t) + \Delta\Phi(t) \leq 2\ln(1+1/\eta)\Delta\text{Opt}(t). \tag{2}$$

We show (2) in two parts. When a request arrives at time t, we first consider (2) when the optimal algorithm services the request, and then consider (2) when the online algorithm services it.

Optimal Algorithm Moves: If $p_t \in C(t-1)$ then $\Delta\Phi = 0$ and (2) holds. Else, at time t page p_t is fetched and suppose some other page p is evicted. Then, we have $\Delta\text{Opt} = w_p$. The only change to Φ is that p may start contributing to Φ, increasing it by at most $2w_p\ln(1+1/\eta)$, while the contribution of p_t disappears, which can only reduce Φ. Thus, $\Delta\Phi \leq 2w_p\ln(1+1/\eta)$, and hence (2) holds.

Online Algorithm Moves: We do a continuous analysis. Suppose the online algorithm fetches an infinitesimally small dy units of of page p_t, i.e. $y_{p_t,t}$ decreases by dy. Let S denote the set of pages $S = p_t \cup \{p : y_{p_t,t} < 1\}$. The online algorithm increases $y_{p,t}$, for each $p \in S \setminus \{p_t\}$, in proportion to $(y_{p,t}+\eta)/w_p$, such that the sum of these increases is exactly equal to dy. Thus,

$$dy_{p,t} = \frac{1}{N} \cdot \frac{(y_{p,t}+\eta)}{w_p} \cdot dy \tag{3}$$

where N is the normalization factor $\sum_{p\in S\setminus\{p_t\}} \frac{1}{w_p}(y_{p,t}+\eta)$. Thus, the online eviction cost is

$$\begin{aligned}
\sum_{p\in S\setminus\{p_t\}} w_p dy_{p,t} &= \sum_{p\in S\setminus\{p_t\}} \frac{1}{N}(y_{p,t}+\eta)dy \\
&\leq \frac{(|S|-k)}{N} + \frac{(|S|-1)\eta}{N}dy \qquad (4)\\
&\leq \frac{2(|S|-h)}{N}dy. \qquad (5)
\end{aligned}$$

The first term in (4) follows as $y_{p,t} = 1$ for $p \notin S$, and hence

$$\sum_{p\in S} y_{p,t} = \sum_{p\in[n]} y_{p,t} - \sum_{p\notin S} y_{p,t} \leq (n-k) - (n-|S|) = |S| - k.$$

Finally, (5) follows as $(x-1)\eta \leq x-h$ for any $x \geq k+1$, and as $|S| \geq k+1$.

We now upper bound $\Delta\Phi(t)$. As $p_t \in C(t)$, a change in $y_{p_t,t}$ does not affect Φ. Now, as $y_{p,t}$ can only increase for $p \neq p_t$, Φ only decreases. Thus,

$$\begin{aligned}
\Delta\Phi(t) &= \sum_{p\neq p_t} \left(\frac{d\Phi(t)}{dy_{p,t}}\right) dy_{p,t} = -2 \sum_{p\in S\setminus C(t)} \left(\frac{w_p}{y_{p,t}+\eta}\right) \cdot \frac{1}{N} \cdot \left(\frac{y_{p,t}+\eta}{w_p}\right) dy \\
&= -2 \sum_{p\in S\setminus C(t)} \left(\frac{1}{N}\right) dy \leq -2\left(\frac{(|S|-h)}{N}\right) dy. \qquad (6)
\end{aligned}$$

The second equality follows by (1) and (3), and the final inequality follows as $|C(t)| \leq h$, as the optimal algorithm has a cache of size at most h.

By (5) and (6) it follows that $\Delta \mathrm{On}(t) + \Delta\Phi(t) \leq 0$ as desired.

References

[1] D. Achlioptas, M. Chrobak and J. Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234(1-2): 203–218, 2000.

[2] N. Bansal, N. Buchbinder and J. Naor. A Primal-Dual Randomized Algorithm for Weighted Paging. In *Proceedings of Foundations of Computer Science*, 2007, 507–517.

[3] N. Bansal, N. Buchbinder and J. Naor. A Primal-Dual Randomized Algorithm for Weighted Paging. Journal version, submitted.

[4] N. Bansal, N. Buchbinder and J. Naor. Towards the Randomized k-server Conjecture: A primal-dual approach. In *Symposium on Discrete Algorithms*, 2010, to appear.

[5] A. Blum and M. Furst and A. Tomkins. What to do with your free time: algorithms for infrequent requests and randomized weighted caching. *Manuscript*, 1996.

[6] A. Borodin and R. El-Yaniv, Online computation and competitive analysis, 1998, Cambridge University Press.

[7] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. Mathematics of Operations Research, 34, 270–286, 2009.

[8] M. Chrobak, H. J. Karloff, T. H. Payne and S. Vishwanathan. New results on server problems. *SIAM J. Discrete Math*, 4(2), 172–181, 1991.

[9] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. Sleator and N. Young. Competitive paging algorithms. *J. Algorithms*, 12(4), 685–699, 1991.

[10] A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6), 1403–1422, 2003.

[11] L. A. McGeoch and D. D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6(6), 816–825, 1991.

[12] S. Irani. Randomized weighted caching with two page weights. *Algorithmica*, 32(4), 624–640, 2002.

[13] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28(2), 202–208, 1985.

[14] N. E. Young. The k-server dual and loose competitiveness for paging. *Algorithmica*, 11(6), 525–541, 1994.

[15] N. E. Young. On-line caching as cache size varies. In *Proceedings of the 2nd Annual ACM-SIAM Symposium on Discrete Algorithms*, 1991, 241–250.