RC24109 (W0611-099) November 20, 2006
Computer Science

IBM Research Report

Randomized Algorithms for Weighted Paging

Nikhil Bansal
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Randomized algorithms for weighted paging

Nikhil Bansal

Abstract

We consider randomized online algorithms for the weighted paging problem. Here we are given a fast memory (cache) of size k and n pages of unit size. Requests arrive for pages over time, and if the requested page is already in the cache the system does nothing, otherwise there is a miss and the system must fetch the page into the cache, possibly evicting some other page. In the weighted version, pages can have different fetching costs. We give an $O(\ell \log k)$ competitive algorithm where ℓ is the number of distinct weight classes. Prior to our work $O(\log k)$ competitive algorithms were known only for the unweighted case (i.e. $\ell = 1$) due to Fiat et al. [14] and for the case of two weights, $\ell = 2$ due to Irani [15]. We also give an $O(\ell \log(k/(k-h+1)))$-competitive algorithm for the weighted (h,k)-paging problem where the online algorithm is compared against the offline algorithm with a (smaller) cache of size h. Prior to our work, such a guarantee was known only for the unweighted case due to Young [22]. The key to our results is a novel potential function analysis of the classic randomized marking algorithm for unweighted paging that generalizes easily to the weighted case.

1 Introduction

We consider the following classic online paging problem. We are given a collection of n pages, and a fast memory (cache) that can hold up to k of these pages. At each time step one of the pages in 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. Paging is one of the earliest and most extensively studied problems in online computation.

The unweighted paging problem is very well understood. In their seminal paper, Sleator and Tarjan [20] showed that any deterministic algorithm is at least k-competitive, and also showed that LRU (Least Recently Used) algorithm is k-competitive. They also considered the more general (h, k)-paging problem where the online algorithm with cache size k is compared to the offline algorithm with cache size h. They showed that any deterministic algorithm is at least $k/(k-h+1)$-competitive, and that LRU is actually $k/(k-h+1)$ competitive. Fiat et al. [14] obtained the first non-trivial randomized algorithm, known as the Randomized Marking and showed that it was $2H_k$ competitive against an oblivious adversary, where H_k is the k-th Harmonic number. They also showed that any randomized algorithm is at least H_k competitive. McGeoch and Sleator [17] gave a matching H_k competitive algorithm. Subsequently, Achlioptas, Chrobak and Noga [1] gave another H_k competitive algorithm that is easier to state and analyze. They also showed that the Randomized Marking algorithm is exactly $2H_k - 1$ competitive. Young [22] showed that Randomized Marking is about $2\ln(k/(k-h))$ (ignoring lower order terms) for the (h, k)-paging problem and that any algorithm is at least $\ln(k/(k-h))$ competitive. There has been extensive work on paging along other directions, and we refer the reader to the excellent text by Borodin and El-Yaniv [8] for details.

Deterministic algorithms for weighted paging are well understood. A tight k-competitive algorithm follows from the work of Chrobak et al. [9], and Young [21] gave a tight $k/(k-h+1)$ competitive algorithm for the (h, k)-paging version of the problem. However, randomized algorithms for weighted paging are not well understood. Prior to our work, the only known $o(k)$ guarantee was an $O(\log k)$ competitive algorithm due to Irani [15] for the two weight case, i.e. when each page weight is either 1 or some fixed $M > 1$. Her algorithm applied the randomized marking algorithm to each weight class, while carefully balancing the amount of cache space used for pages of each class.

Paging can viewed as a special case of the much more general and challenging k-server problem. In this problem, there are k servers located on points in an n-point metric space. At each time step a request is placed at one of the points and the algorithm must move one of the servers to this point to serve the request. The goal is to minimize the overall distance traveled by the servers. The unweighted paging problem is exactly the k-server problem on a uniform metric space. The weighted paging problem is identical (up to an additive constant) to k-server on the metric where the distance between any two pages a and b is $(w(a) + w(b))/2$, where $w(a)$ denotes the weight of page a.

The k-server problem has a fascinating history, and substantial progress has been made on deterministic algorithms for the problem. It is known that any deterministic algorithm must be at least k competitive on any metric space with more than k points. Fiat, Rabani and Ravid [13] gave the first algorithm for which the competitive ratio was only a function of k. Their algorithm was $O((k!)^3)$ competitive. After a series of results, a breakthrough was achieved by Papadimitriou and Koutsoupias [16] who gave an almost tight $2k-1$ competitive algorithm. This is still the best known bound (both for deterministic and randomized algorithms) for general metric spaces. The tight guarantee of k is also known for many special cases. In

particular, Chrobak et al. [9] gave a k competitive algorithm for trees. We refer the reader to [8] for more details on the k-server problem.

However, randomized algorithms for the k-server problem remain poorly understood. No lower bound better than $\ln k$ is known for any metric space. Interestingly, Bartal, Bollobas and Mendel [4] also showed that no metric space with more than k points can admit a $o(\log k/\log\log k)$ competitive algorithm. A widely believed conjecture is that $O(\log k)$ competitive algorithms exist for every metric space. Given the results on probabilistic embedding of arbitrary metrics into trees [3, 2, 11], and our understanding of k-server on the uniform metric, a natural line of attack to obtain poly-logarithmic guarantees (in k, n and possibly even the diameter of the space Δ) has been to consider algorithms for HST's Hierarchically Separated Trees. A breakthrough in this direction was achieved by Bartal et al. [5] who gave the first poly-logarithmic competitive algorithm for the related Metrical System Problem[1]. Their results implied a competitive ratio of about $O(\log^5 k)$ for the k-server problem on a space with $k+c$ points, where c is a constant independent of k. This guarantee was improved by Fiat and Mendel [12] to about $O(\log^2 k)$. However, for n much larger than k, no algorithms with a sublinear competitive ratio are known except for very few special cases. Besides paging and weighted paging with two weights, a poly-logarithmic competitive algorithm is known for a special subclass of certain well-separated spaces [19]. Csaba and Lodha [10] gave an $O(n^{2/3})$ competitive algorithm, which is $o(k)$ competitive for $n = o(k^{3/2})$, for n uniformly spaced points on a line[2].

Abstracting and extending the insights from paging is a key step in the resolution of the k-server problem. In this paper we show how the randomized marking algorithm for unweighted paging extends to the weighted version. Our main result is an $O(\ell \log k)$ competitive algorithm for weighted paging with ℓ distinct weight classes. Viewed as a k-server problem, this implies an $O(\log k \log \Delta)$ competitive algorithm for the weighted paging metric with diameter Δ. This result relies on a new potential function based analysis of randomized marking. A novel aspect of the analysis is that it does not rely on "phases" to lower bound the cost of the optimum algorithm. Our result also generalizes to the weighted (h, k)-paging problem, for which we obtain an $O(\ell \log(k/(k-h+1)))$-competitive algorithm.

1.1 Preliminaries

The state under a randomized algorithm is completely specified by the probability distribution on the various configurations (deterministic states). In the context of k-server, this corresponds to specifying the distribution on k-tuples of server positions. Such a distribution induces another distribution $p(x,t)$ on the points in the metric space which specifies the probability that a server is placed at point x at time t. Clearly, this map is not a bijection. For example, the distribution $(1/2, 1/2, 1/2, 1/2)$ on four points A, B, C, D could be induced by the distribution D_1 on two server states where each state (A, B) and (C, D) occurs with probability $1/2$ each, or it can also be induced by the distribution D_2 where each of six possible states (A, B), $(A, C), \ldots, (C, D)$ occur with probability $1/6$ each.

The induced distribution on points turns out to substantially easier to work with than actual distributions. One can simply view a probability mass of k units distributed among the n points, and the "move" of an algorithm simply corresponds to moving this mass among the points. In this view when the algorithm moves ϵ units of mass from point i to j, it incurs a cost of $\epsilon \cdot d(i, j)$. We call this the fractional view. This fractional view has been considered previously [5, 7]. Blum et al. [7] showed that the fractional view and the actual view are within a factor of 2 for the unweighted paging problem. We adopt a similar approach.

[1] We do not define HST's, the MTS problem and other related concepts since this would take us far afield. We refer the reader to [8] for details.

[2] A generalization was considered was considered by Bartal and Mendel [6], who proposed a $\Delta^{1-\epsilon}\mathrm{polylog}k$ competitive algorithm for bounded growth metrics with diameter Δ. Unfortunately, their result seems to have a serious error [18].

However, for weighted caching the relation between fractional view and actual view is much more subtle. Consider for example the distribution $(1/2, 1/2, 1/2, 1/2)$ induced by cache states (A, B) and (C, D) each occurring with probability $1/2$. Pages A and B have weight 1, and pages C and D have a large weight M. Suppose the fractional algorithm moves $1/2$ unit of mass from page A to page B leading to the state to $(0, 1, 1/2, 1/2)$. In the fractional view, this algorithm incurs a cost of $1/2$. However, it is instructive to see that it is impossible to modify the actual distribution without incurring a cost of $\Theta(M)$. In fact, the only actual distribution consistent with $(0, 1, 1/2, 1/2)$ is to have probability $1/2$ on state (B, C) and probability $1/2$ on state (B, D). Thus, starting from the previous configuration, at least $1/2$ unit of mass from either C or D must be moved which incurs cost $\Theta(M)$.

Interestingly, for weighted caching we get around this problem by restricting our actual distributions to a certain subclass of distributions. In particular, in section 5 we show how to maintain an online mapping from induced distributions to actual distributions, such that any fractional move with cost c can be mapped to a move on actual distributions with cost $5c$. Hence, throughout this paper we work with the fractional view. The paper is organized as follows: In section 2 we describe our alternate proof of unweighted caching. This section develops most of the ideas that we need later. In section 3 we describe our results for weighted paging, and in section 4 we consider the weighted (h, k)-paging problem.

2 Unweighted Paging

Recall the Randomized Marking algorithm of Fiat et al. [14]. Initially, all the pages in the cache are unmarked. If the requested page q is already in the cache but unmarked, we mark q. If q is not in the cache, we mark q and bring it into the cache, while evicting an unmarked page chosen uniformly at random. If all pages are marked, we first unmark all the pages and then bring q in.

The fractional view of Randomized Marking is the following: All the n pages in the universe are either classified as marked, unmarked or absent. With each page q, we associate a probability $p(q)$ that indicates the probability that it is present in the cache. The mass of marked pages is always 1, the mass of absent pages is always 0, and the mass of all unmarked pages is equal to p, where p satisfies the condition $m + pu = k$. Here m denotes the number of marked pages, and u denotes the number of unmarked pages. When q is requested, it is marked and the probability mass of q is increased to one. The cost incurred by the algorithm is $1 - p(q)$, where $p(q)$ is the probability mass of q before q was requested ($p(q)$ could either be 0 if q was absent, or p if it was unmarked, or 1 if it was already marked). The mass of q is increased by borrowing the mass uniformly from the unmarked pages. If no mass can be borrowed from the unmarked pages (because either $p = 0$ or $u = 0$), we unmark all the marked pages, and all the previously unmarked pages (that had $p = 0$) are classified as absent.

In this section we will consider the fractional view of the online algorithm and prove a competitive ratio of $2 \ln k + 3$. Without loss of generality, we assume that best offline solution is deterministic, and hence at any time a page is either completely present or else is absent from the offline cache. Our presentation will be somewhat more general than a standalone proof for unweighted caching needs to be (we point this out at appropriate places). This generality will be very useful in the analysis of weighted caching.

Let h and k be size of the offline and online cache respectively, and m, u and p be as defined above. We have the relation $m + pu = k$. Note that m and u are non-negative integers, but we not require k to be an integer. This does not make sense for unweighted caching, but will be crucial for weighted caching. We will not worry about what is meant by $k = 3.4$, for example. We just think of p determined by the relation $p = (k-m)/u$, so if $k = 3.4$, $m = 2$ and $u = 4$, we view it is a cache with 2 marked pages, and 4 unmarked pages each having probability $1.4/4 = 0.35$.

Let $\mathrm{Off}(i)$ and $\mathrm{On}(i)$ denote the offline and online cost incurred at step i, We will define a potential function Φ, that is always bounded by a fixed constant independent of the request sequence and show that each step i,

$$c \cdot \mathrm{Off}(i) \geq \mathrm{On}(i) + \Phi_i - \Phi_{i-1} \tag{1}$$

where c is the desired competitive ratio, and Φ_i is the value of Φ at step i.

Let $\mathrm{M}_{\mathrm{both}}$ and (resp.) $\mathrm{U}_{\mathrm{both}}$ denote the set of marked and (resp.) unmarked pages that lie both in the offline and the online cache. Similarly, let M_{on} and (resp.) U_{on} denote the set of marked and (resp.) unmarked pages that lie in the online cache but not in offline cache. Finally, let $\mathrm{Off}_{\mathrm{only}}$ denote the pages that only lie in the offline cache. See figure 1. Let $u_{\mathrm{on}}, u_{\mathrm{both}}, m_{\mathrm{on}}$ and m_{both} denote the cardinalities of the corresponding sets. Thus $m = m_{\mathrm{both}} + m_{\mathrm{on}}$ and $u = u_{\mathrm{both}} + u_{\mathrm{on}}$. As $|\mathrm{Off}_{\mathrm{only}}| = h - m_{\mathrm{both}} - u_{\mathrm{both}}$, we do not define a new symbol for it. We will use $\overline{p}$ to denote $1 - p$. Let $f : [0, 1] \to [0, 1 + \ln k]$ be the following function:

$$f(p) = \begin{cases} \ln(1/\overline{p}) & \text{if } p \leq 1 - 1/k \\ pk - k + 1 + \ln k & \text{if } p \in [1 - 1/k, 1] \end{cases}$$

The reader can view $f(p)$ as simply $\ln(1/\overline{p})$. The case $p > (1 - 1/k)$ never arises when k is an integer. It is added for technical reasons so that $f(p)$ is well-behaved when p approaches 1, and it will also allow us to simplify the analysis by making it "continuous". The following is easily observed.

Observation 1 *$f(p)$ lies in the range $[0, 1 + \ln k]$ for all $p \in [0, 1]$. The derivative, $\frac{d}{dp} f(p) = 1/\overline{p}$ for $p \in [0, 1 - 1/k]$ and is equal to k for $p \in [1 - 1/k, 1]$. Moreover, $f(p)$ and its derivate $f'(p)$ are both continuous at $p = 1 - 1/k$.*

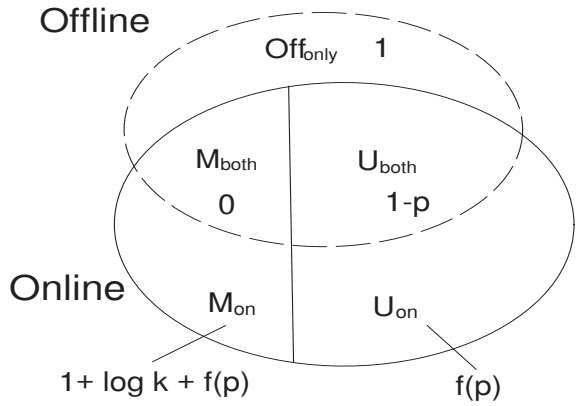

Figure 1: The Potential Function and classification of pages

To define the potential function, we associate a potential with each page depending on the set it lies in. The overall potential Φ is the sum of the potential for each page. See figure 1.

1. Each page q in the offline cache has potential $\overline{p}(q)$. Thus, each page in $\mathrm{M}_{\mathrm{both}}$ has potential 0, each page in $\mathrm{U}_{\mathrm{both}}$ has potential $1 - p$, and each page in $\mathrm{Off}_{\mathrm{only}}$ has potential 1.

2. Each $q \in \mathrm{M}_{\mathrm{on}}$ has potential $1 + \ln k + f(p)$.

3. Each page $q \in \mathrm{U}_{\mathrm{on}}$ has potential $f(p)$.

Thus the overall potential Φ is given by

$$\Phi = (1-p)u_{\mathrm{both}} + (h - m_{\mathrm{both}} - u_{\mathrm{both}}) + (1 + \ln k + f(p))m_{\mathrm{on}} + f(p)u_{\mathrm{on}}$$

Intuitively, if the adversary requests a page q already present in the offline cache, then online will incur a cost of $1-p(q)$, where as offline incurs no cost. We use the potential of $1-p(q)$ on this page to pay for this cost. The potential on pages in U_{on} and M_{on} can be viewed as a penalty term. In a sense, online is wasting valuable cache space on pages in U_{on} and M_{on} (as offline does not have these pages, the adversary has no intention of requesting these any time soon). The closer p is to 1, the higher the penalty $f(p)$ on pages in U_{on}. For marked pages in M_{on} the penalty is even higher since the online algorithm is not going to get rid of these useless pages any time soon. The following are three useful properties of Φ.

Lemma 2 *Whenever the offline algorithm evicts a page, the potential increases by at most* $2\ln k + 2$.

Proof: Let q' be the evicted page. Before eviction, q' either lies in M_{both} or U_{both} or is absent for the online cache. Note that the eviction only affect the potential of q'. In the first case, q' moves from M_{both} to M_{on} and hence the potential of q' increase by at most $1 + \ln k + f(p) \leq 2\ln k + 2$. In the second case, q' moves from U_{both} to U_{on}, and hence the potential change is at most $f(p) - (1-p) \leq f(p) \leq 1 + \ln k$. In the last case, the potential only decreases by 1. Thus the result follows. ■

Lemma 3 *The potential does not increase at the step when all marked pages get unmarked, i.e. when* p *changes from 0 to 1.*

Proof: When all the pages are unmarked, the probability p changes instantaneously from 0 to 1. We first consider the changes to the marked pages. The pages in M_{both} move to U_{both} and the pages in M_{on} move to U_{on}. In the first case, the potential of each page changes from 0 to $1-p=0$, and in the second case it changes from $1 + \ln k + f(0)$ to $f(1) = 1 + \ln k$. Thus the potential remains unchanged for each previously marked page.

For the previously unmarked pages, those originally in U_{both} move to $\mathrm{Off}_{\mathrm{only}}$ and the potential of each page remains 1. And finally, those originally in U_{on} disappear from online's cache. Their potential remains 0 (i.e. changes from $f(0) = 0$ to 0). ■

Lemma 4 *If* $h < k$*, then the following relation holds for every value of* $p \in [0,1]$.

$$(u_{\mathrm{on}} + m_{\mathrm{on}})\frac{d}{dp}f(p) \geq u_{\mathrm{both}} + u_{\mathrm{on}} \tag{2}$$

Remark: The strict inequality above is unrelated to the conditions under which our results for unweighted and weighted caching hold. Our results hold even when $k = h$.

Proof:[of Lemma 4] As $df/dp = \min(k, 1/\overline{p})$ we need to show that $(u_{\mathrm{on}} + m_{\mathrm{on}})\min(k, 1/\overline{p}) \geq (u_{\mathrm{both}} + u_{\mathrm{on}})$. We first show that $(u_{\mathrm{on}} + m_{\mathrm{on}})(1/\overline{p}) \geq (u_{\mathrm{both}} + u_{\mathrm{on}})$.

As the offline cache contains M_{both}, U_{both} and (possibly) more pages, and as each page in U_{both} and U_{on} has mass p in the online cache, it follows that

$$h \geq m_{\mathrm{both}} + u_{\mathrm{both}} \qquad \text{and} \qquad k = m_{\mathrm{both}} + m_{\mathrm{on}} + p(u_{\mathrm{both}} + u_{\mathrm{on}})$$

Subtracting these, we get $0 < k - h \leq m_{\text{on}} - \overline{p} \cdot u_{\text{both}} + p \cdot u_{\text{on}}$, or equivalently

$$(m_{\text{on}} + u_{\text{on}}) > \overline{p}(u_{\text{both}} + u_{\text{on}}). \tag{3}$$

Finally, we show that $(u_{\text{on}} + m_{\text{on}})k \geq (u_{\text{both}} + u_{\text{on}})$, or equivalently that $(km_{\text{on}} + (k-1)u_{\text{on}}) \geq u_{\text{both}}$. Now, $u_{\text{both}} \leq h \leq k-1$, and as u_{on} and u_{on} are integers, the strictly inequality in (3) implies that $(m_{\text{on}} + u_{\text{on}}) \geq 1$. Thus the inequality $((km_{\text{on}} + (k-1)u_{\text{on}}) \geq u_{\text{both}}$ always holds. ■

The above Lemma has the following implication which will be used repeatedly.

Lemma 5 *Given any configuration of offline and online caches, suppose we remove an infinitesimally small ϵ units of probability mass from the online cache, i.e. we reduce $\Delta p = \epsilon/(u_{\text{both}} + u_{\text{on}})$ mass from each unmarked page, and k becomes $k - \epsilon$. Then,*

1. *If $k > h$ and ϵ is small enough such that $k - \epsilon \geq h$, then the cumulative potential of pages in $U_{\text{on}} \cup M_{\text{on}}$ decreases by at least ϵ.*

2. *Irrespective of whether $k > h$ or not, the cumulative potential of pages in U_{both} increases by at most ϵ. Moreover, the potential of other pages does not increase.*

Proof: It is easily verified that only the potential on pages in U_{both} increases as p decreases. The potential on each page in U_{both} increases by Δp, and hence the cumulative increase on these pages is at most $\Delta p \cdot u_{\text{both}} = (\epsilon/(u_{\text{both}} + u_{\text{on}})) \cdot u_{\text{both}} \leq \epsilon$, which implies the second part of the lemma.

The potential for each page in $U_{\text{both}} \cup M_{\text{both}}$ decreases by $(df/dp) \cdot \Delta p$. Thus the cumulative decrease is $(u_{\text{on}} + m_{\text{on}}) \cdot \Delta p \cdot df/dp$, which by Lemma 4 is at least $(u_{\text{both}} + u_{\text{on}}) \cdot \Delta p = \epsilon$, which implies the first part of the lemma. ■

We now give our proof of $O(\log k)$ competitiveness of Randomized Marking.

Theorem 6 *The properties of the potential function mentioned above imply that Randomized Marking is $2 \ln k + 3$ competitive in the fractional view.*

Proof: Let q be the requested page. We divide the analysis into two steps. First the offline serves the request, and then online serves it. We show that inequality (1) is satisfied at each step with $c = 2 \ln k + 3$.

If q is already present in the offline cache, $\text{Off}(i) = 0$ and the potential remains unchanged, thus (1) is trivially satisfied. Now, suppose offline evicts some q' and brings q in the cache. This only affects the potential of q and q'. By Lemma 2 the potential of q' increases by at most $2 \ln k + 2$. As any page in the offline cache has potential at most 1, the potential for q can increase by at most 1. Thus, $\Delta\Phi \leq 2 \ln k + 3$. As offline pays 1 unit to fetch q, it follows that $\text{Off}(i) = 1$ and hence (1) holds for $c = 2 \ln k + 3$.

We now consider online. Clearly, if q is already marked, nothing is affected and both $\text{On}(i) = \Delta\phi = 0$. Thus, we assume that q is not marked. As q already lies in offline cache (since offline served it in the previous step), it must lie in either U_{both} or in Off_{only}. To serve q, we view online as gradually borrowing mass from unmarked pages (unmarking all the marked pages at some point if needed) while increasing the mass $p(q)$ on q and eventually marking it when $p(q)$ reaches 1. For convenience of description, we remove q immediately from U_{both} or Off_{only} when online begins serving it, and add it to M_{both} only when $p(q)$ reaches 1. Thus q does not lie in any of the sets, as $p(q)$ increases to 1.

We now analyze the cost. The online algorithm pays $(1 - p(q))$ to increase the mass of q from $p(q)$ to 1. As the potential of q was $(1 - p(q))$ initially (as q is in the offline cache already) and decreases to 0 eventually, we use this decrease in potential to pay the online cost. To finish the proof we show that as $p(q)$

is increased, the change $\Delta\phi$ due to pages other than q is no more than 0. Consider Lemma 5 applied to the caches $OFF \setminus \{q\}$ and $ON \setminus \{q\}$. Here offline cache size is $h' = h - 1$ as q occupies one unit space, and $k' > k - 1$, as q occupies strictly less than one unit of mass in the online cache as we are still increasing $p(q)$. Thus $k' > h'$ (as $k \geq h$), and hence by Lemma 5, as ϵ units of mass in removed from $ON \setminus \{q\}$ the decrease in potential on pages in $U_{\text{both}} \cup M_{\text{both}}$ compensates for the increase in potential for pages in U_{both}.
■

3 Weighted Caching

Let $1, \ldots, \ell$ be the ℓ weight classes and let $w_1 < w_2 < \ldots < w_\ell$ denote their weight. Our algorithm for weighted caching will apply randomized marking with each weight class, while carefully deciding how much cache space is allocated to each weight class. As previously, we take the fractional view of the online algorithm. Let h_i, k_i, u_i and m_i denote of weight i pages in opt's cache, online's cache, unmarked pages and marked pages. Note that h_i,u_i and m_i are integral but k_i's could be fractional. Let p_i denote the probability of an unmarked page in class i, thus we have that $p_i = (k_i - m_i)/u_i$.

Our algorithm for weighted caching is the following: Suppose the current request is for page q with weight w_i. We will raise the probability mass of q gradually to 1, and then mark it. Again, we view this increase as a continuous process. For a class $j \neq i$, we say that it is unavailable if $k_j = 0$. For $j = i$, class j is unavailable if the mass of class i pages other than q in online cache is 0. Let A denote the set of available classes. To raise the mass of q by amount dx, we remove $dx/w_j N$ amount of mass from class j for each available class j. Here $N = \sum_{i \in A} 1/w_i$ is normalizing factor. If at some point, a class becomes unavailable, we remove this class from A, readjust N and continue increasing the mass of q. As in the unweighted caching, to remove ϵ units of mass from class j, we withdraw probability mass uniformly from the unmarked pages in that class. If class j is available but all the pages are marked, we unmark them and continue to withdraw the mass.

Note that our algorithm is quite simple. It simply withdraws mass from every class that can contribute, and the amount of mass withdrawn is inversely proportional to the weight of the class, and hence each class contributes equally to the online algorithm's cost. As in the unweighted case, we define the sets $U^i_{\text{both}}, U^i_{\text{on}}, M^i_{\text{both}}, M^i_{\text{on}}$ and OFF^i_{only} for each class i. Consider the following potential function for class i.

$$\Phi_i = w_i \left(\ell(1 + \ln k + f(p_i)) m^i_{\text{on}} + \ell \cdot f(p_i) \cdot u^i_{\text{on}} + \overline{p}_i \cdot u^i_{\text{both}} + (h_i - m^i_{\text{both}} - u^i_{\text{both}}) \right)$$

The overall potential is $\Phi = \sum_{i=1}^{\ell} \Phi_l$. Observe that the only difference between Φ_i above and that used in section 2 is that the potential for pages in M_{on} and U_{on} is multiplied by the number of classes.

Our main result in Theorem 7 below. The idea of the analysis is simple. Recall Lemma 5, there, as the mass of $p(q)$ is increased, the decrease in potential of pages in M_{on} and U_{on} pays for the increase in potential of pages in U_{both}. For the weighted case, we are scaling up the contribution of pages in U_{on} and M_{on}. As the overall cache size of both offline and online are equal, there is some class i^* for which $k_{i^*} \geq h_{i^*}$, and the decrease in potential for this class will offset the possible increase in all other classes. We now give the details.

Theorem 7 *The competitive ratio for the above algorithm is* $(2\ell \ln k + 2\ell + 1)$, *where* ℓ *is the number of weight classes and* k *is the cache size.*

Proof: Suppose the requested page q lies in class i. We again divide the analysis in two parts. First offline serves this request and then online serves it.

If q is already in the offline cache, then offline incurs no cost, and nothing changes. Suppose q is not in offline cache and it evicts some page q' of class j to accommodate q, then by the argument in Lemma 2 (applied to the current potential function), the potential of class j goes by at most $\ell \cdot w_j \cdot (2 + 2\ln k)$. After q is brought into the cache, it can have a potential of at most w_i. Thus the overall potential increases by at most $w_j \cdot \ell(2 + 2\ln k) + w_i$. To show the desired competitive ratio, we can charge offline w_i to fetch a page of weight w_i and $\ell \cdot w_i(2 + 2\ln k)$ when it evicts it. Thus up to an additive term of $O(hw_\ell)$ the overall offline cost can be charged to pay for the overall increase in potential.

We now analyze the online cost. First we note that Lemma 3 remains true here, so we need not worry about the change in potential when marked pages get unmarked. Suppose we increase the mass $p(q)$ on q by dx, then online pays $w_i \cdot dx$ to pay for this increase. The potential of page q decreases exactly by exactly $w_i \cdot dx$, and we use this decrease in potential of q to pay for the online cost. It remains to show that the cumulative potential due to the pages other than q does not increase. Let A denote the set of available classes. For $j \in A$ let $dx(j) = dx/(w_j N)$ be the amount mass that is removed from class j. By design, $\sum_{j \in A} dx(j) = dx$ and $dy = w_j \cdot dx(j)$ is identical for all $j \in A$.

For any arbitrary class $j \in A$, let $dp_j = dx(j)/(u^j_{\text{both}} + u^j_{\text{on}})$ denote the change in p_j (for $j = i$, we do not include the requested page q among the unmarked pages). By Lemma 5, irrespective of whether $k_i > h_i$ or not, the potential only increases for pages in U^j_{both}, and the contribution of such pages to class j is $w_j \cdot u^j_{\text{both}} \cdot dp_j \le w_j dx(j) = dy$. Now, let j^* be some class in A such that $k_{j^*} > h_{j^*}$, clearly such a class must exist as $k \ge h$, and h has one unit of mass on q, whereas online has strictly less than one unit of mass on q. Since we remove $dx(j^*)$ units of mass from this class, by Lemma 5, the decrease in potential due to pages in $U^{j^*}_{on}$ and $M^{j^*}_{on}$ is at least $\ell \cdot w_{j^*} \cdot dx(j^*) = \ell \cdot dy$. This decrease in potential can pay for the increase of potential due to U^j_{both} in each class. Since the potential for every page other than those in U_{both} is non-increasing, the result follows. ■

4 Weighted (h, k)-Paging

We first show how to modify the potential function to show that Randomized Marking is $O(\log(k/k - h + 1))$-competitive for the unweighted problem. Let $\alpha = (k - h)/k$ be a fixed constant, and let the function $f(p)$ be defined as

$$f(p) = \begin{cases} \ln((1+\alpha)/(\overline{p}+\alpha)) & \text{if } p \le 1 - 1/k \\ \frac{kp-(k-1)}{(k-h+1)} + \ln\left(\frac{2k-h}{k-h+1}\right) & \text{if } p \in [1-1/k, 1] \end{cases}$$

As in Section 2, each page q in the offline cache has potential $\overline{p}(q)$ and pages in U_{both} have potential $f(p)$. Pages in M_{on} have the potential $1/(k - h + 1) + \ln((2k - h)/(k - h + 1))$. Note that when $k = h$, i.e. $\alpha = 0$, this potential is identical to that in Section 2.

The following analog of Observation 1 is easily verified.

Observation 8 *$f(p)$ lies in the range $[0, 1/(k - h + 1) + \ln((2k - h)/(k - h + 1))]$ for all $p \in [0, 1]$ and hence it is at most $1 + \ln 2 + \ln(k/(k-h+1))$. The derivative, $\frac{d}{dp} f(p) = 1/(\overline{p}+\alpha)$ for $p \in [0, 1-1/k]$ and is equal to $k/(k - h + 1)$ for $p \in [1 - 1/k, 1]$. Moreover, $f(p)$ and its derivate $f'(p)$ are both continuous at $p = 1 - 1/k$.*

To analyze a request, as previously we assume that offline services the request first and then online services it. Repeating the argument in Lemma 2 it follows that the potential increases by at most $2/(k-h+$

$1) + 2\ln((2k-h)/(k-h+1))$ when a page is evicted. This implies that when offline services a request, inequality (1) is satisfied with $c = 1 + 2/(k-h+1) + 2\ln((2k-h)/(k-h+1))$.

To analyze the online cost, we first note that Lemma 3 holds, i.e. the potential does not change when pages get unmarked. This can be verified by repeating the argument in proof of Lemma 3 and observing that $f(0) = 0$ and $f(1) = 1/(k-h+1) + \ln((2k-h)/(k-h+1))$. We now state and prove the analog of Lemma 5 which is the crux of the argument. Note that condition 2 below is analogous to but different than in Lemma 5.

Lemma 9 *Let $\alpha = (k-h)/k$, and consider the (h', k')-paging problem. Suppose that dx units of removed by the online algorithm, then:*

1. *If $h' < k'(1-\alpha)$, the cumulative potential of pages in $U_{\text{on}} \cup M_{\text{on}}$ decreases by at least dx.*
2. *Whether $h' \leq k'(1-\alpha)$ or not, the cumulative potential of pages in U_{both} increases by at most dx.*

Proof: The proof of the second part follows exactly as in Lemma 5. As $df(p)/dp = \min(k'/(k'-h'+1), 1/(\overline{p}+\alpha))$, for the first part, it suffices to show that $(m_{\text{on}} + u_{\text{on}}) \min(k'/(k'-h'+1), 1/(\overline{p}+\alpha)) \geq (u_{\text{both}} + u_{\text{on}})$. As in Lemma 4, we have the inequalities $h' \geq u_{\text{both}} + m_{\text{both}}$ and $k' \leq m_{\text{both}} + m_{\text{on}} + p(u_{\text{both}} + u_{\text{on}})$. Subtracting the first inequality from the second and using that $h' < (1-\alpha)k'$ gives us that

$$m_{\text{on}} + u_{\text{on}} > k - h + \overline{p}(u_{\text{both}} + u_{\text{on}}) \geq (\alpha + \overline{p})(u_{\text{both}} + u_{\text{on}})$$

The last step follows as $u_{\text{both}} + u_{\text{on}} \leq k'$, since all unmarked pages in the current phase must have been marked in the previous phase and hence their number cannot exceed k'. Finally, as $m_{\text{on}} + u_{\text{on}} \geq k' - h' + 1$, it follows that $(m_{\text{on}} + u_{\text{on}})(k'/(k'-h'+1)) \geq k' \geq u_{\text{both}} + u_{\text{on}}$ which implies the desired result. ■

Theorem 10 *The weighted caching algorithm in Section 3 is $O(\ell \log k)$ competitive for (h, k)-paging.*

The idea of the proof is identical to that of Theorem 7. We define an analogous potential function for the weighted case, such that the contribution of M_{on} and U_{on} is scaled up ℓ times. Since, $k - h = \alpha k$, and since offline has strictly more weight on q than online (as long as online serves q), there exists some class j^* such that $k_{j^*} - h_{j^*} > \alpha(k_{j^*})$. By Lemma 9, as the mass on q is increased, the drop in potential due to pages in $M_{\text{on}}^{j^*}$ and $U_{\text{on}}^{j^*}$ offsets the increase of potential in all other classes.

5 Actual Distribution on Caches

Let $w_1, \ldots, w_\ell$ denote the page weights, and consider the (equivalent) cost version where we pay $w_i/2$ both to fetch and to evict a class i page. We will assume that weights are integral powers of two. This affects competitive ratio by at most 2. Consider a distribution P on the pages specified by $p_{ij} \in [0, 1]$ such that $\sum_i \sum_j p_{ij} = k$, where p_{ij} is probability mass on the the j-th page of weight class i. Let P' specified by p'_{ij} be another such distribution. Let $C_f(P, P') = \sum_i (w_i/2)(\sum_j |p_{ij} - p'_{ij}|)$ denote the fractional cost incurred when moving from P to P'. Let $\mathcal{P}$ denote all possible distributions on pages with total mass k.

Given any distribution D on cache states, we let $\Pi(D)$ denote the distribution induced on the pages by D. Finally, given two distributions D and D' on the cache states, let $C(D, D')$ denote the cheapest way of moving from D to D' (and hence the actual cost incurred by a randomized algorithm). Clearly, $C_f(\Pi(D), \Pi(D')$ is a lower bound on $C(D, D')$. Recall the example in section 1.1 which shows that it may not always be possible to reach D' with reasonable cost if we do not choose D carefully. Interestingly, we

get around this problem by carefully restricting the space of distributions D. The main idea is to maintain the distribution on cache states that are almost similar in the sense of how many pages they contain of each class. This similarity can be maintained in an online manner, and prevents the cache states from getting too "far" from each other. We give the details below.

Theorem 11 *Let the weights w_i be such that $w_{i+1}/w_i \geq 2$ for $1 \leq i \leq \ell - 1$. There is a subclass $\mathcal{D}$ of distributions on cache states, along with a map T from $(\mathcal{D} \times \mathcal{P}) \rightarrow \mathcal{D}$ with the following property:*

Given any two distributions on pages P and P', and given any $D \in \mathcal{D}$ satisfying $\Pi(D) = P$, we can obtain another distribution $D' = T(D, P')$ such that $\Pi(D') = P'$ and $C(D, D') \leq 5C_f(P, P')$.

This gives us the desired mapping between the distribution P on pages and distribution D on cache states. Whenever the fractional algorithm moves from state P to P', the randomized algorithm moves from D to $D' = T(D, P')$. Since $\Pi(D') = P'$ and $D' \in \mathcal{D}$, the process can be applied repeatedly.

Proof: Given a distribution P on pages, we characterize all distributions $D \in \mathcal{D}$ such that $\Pi(D) = P$. We denote this set $\mathcal{D}(P)$. Specifying $\mathcal{D}(P)$ for each P describes $\mathcal{D}$ completely. Let $k_i = \sum_j p_{ij}$ denote the mass on class i pages. Consider the interval $I = [0, k]$, and imagine this interval partitioned into $I_1, \ldots, I_l$ where $I_1 = [0, k_1)$, $I_2 = [k_1, k_1 + k_2)$,..., $I_l = [k_1 + \ldots, k_{l-1}, k_1 + \ldots + k_l)$. Each element of $\mathcal{D}(P)$ has the same structure. We associate a cache state $S(\alpha)$ with each real number α in the range $[0, 1)$. Consider an $\alpha \in [0, 1)$. Let $T(\alpha)$ denote the set of real numbers $\{\alpha, 1 + \alpha, 2 + \alpha, \ldots, k - 1 + \alpha\}$. For every $D \in \mathcal{D}(P)$, the cache $S(\alpha)$ has n_i pages of w_i where $n_i = |T(\alpha) \cap I_i|$. By construction, each cache $S(\alpha)$ has either $\lfloor k_i \rfloor$ or $\lceil k_i \rceil$ pages of weight w_i, and the expected number of pages of weight w_i is k_i.

Consider any arbitrary way of filling the caches $S(\alpha)$ with pages such that, no $S(\alpha)$ contains two identical pages, and it is consistent with P (i.e. the probability measure of caches that contain page j of class i is exactly p_{ij}). Such a filling always exists since, for example, we can put first page of class 1 in $S(\alpha)$ corresponding to $\alpha = [0, p_{11})$, the second page of class 1 in $S(\alpha)$ corresponding to $\alpha = [p_{11}, p_{11} + p_{12})$ (where the range of α is considered modulo 1) and so on. Any way of filling $S(\alpha)$ that satisfies the properties above is a valid element $D \in \mathcal{D}(P)$.

We now describe the transformation T. Suppose we are given some $D \in \mathcal{D}(P)$, and the fractional algorithm changes state from P to P'. By decomposing the pages into those for which $p'_{ij} > p_{ij}$ and those for which $p'_{ij} < p_{ij}$ and arbitrarily matching the increases in mass with decreases, we can decompose the move P to P' into the sequence $P = P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \ldots \rightarrow P'$ such that $C_f(P, P') = \sum_{i \geq 0} C_f(P_i, P_{i+1})$ and each move P_i to P_{i+1} is a transposition where some infinitesimally small ϵ units of mass is moved from some page p_a to some page p_b.

Thus it suffices to prove the theorem for moves $P \rightarrow P'$ of this type. Let i be the weight class of page a, and j be that of page b. Suppose $i < j$. Recall the interval I associated with P. When we move from P to P' the right boundary of I_i shifts ϵ units to the left, the intervals $I_{i+1}, \ldots, I_{j-1}$ shift to the left by ϵ units, and finally, the left boundary of I_j shifts left by ϵ and its right boundary stays fixed. The move where $i > j$ is analogous. For this move, the fractional algorithm pays $\epsilon(w_i + w_j)/2$.

We now describe and analyze the move $D \rightarrow D'$. Consider first the weight classes h with weight strictly between w_i and w_j (i.e. $i < h < j$). The intervals corresponding to these classes shift by ϵ each to the left. For each such h at most ϵ fraction of caches $S(\alpha)$ must lose a page of weight w_h (as their quota for class h shrinks from $\lceil k_h \rceil$ to $\lfloor k_h \rfloor$ and similarly, at most ϵ fraction of caches must gain a page. Moreover, the fraction of caches that must lose a weight w_h page is exactly equal to the fraction that must gain such a page. We arbitrarily pair these caches. As any cache that must lose a page is strictly larger than a cache that must gain one, for every matched pair of caches, there is some page in the larger cache that does not lie in the smaller cache and hence can be transferred to it. The movement cost incurred per class is at most $2\epsilon(w_h/2)$,

and hence the total contribution due to all such classes h is $\sum_{i<h<j} \epsilon w_h \leq \epsilon(w_i + w_j)$, as consecutive weights differ by a factor of at least 2.

We now describe the procedure for classes i and j. We first consider the case when $i = j$. Here, none of the sizes $k_1, \ldots, k_\ell$ change, and hence the structure $S(\alpha)$ remains unchanged. We arbitrarily remove page a from an ϵ measure of caches that contain a, and arbitrarily add b to an ϵ measure of caches that do not contain b. We say that first type of cache has a hole, and the second type has an excess. Any cache with a hole has size either $\lfloor k_i \rfloor - 1$ or $\lfloor k_i \rfloor$, and every cache with excess has size either $\lceil k_i \rceil$ or $\lceil k_i \rceil + 1$, and hence is strictly larger. We arbitrarily pair up the caches with a hole to those with excesses, and transfer some page from the larger cache that does not lie in the smaller cache. The cost incurred is at most $2\epsilon w_i/2 + 2\epsilon w_i/2 = 2\epsilon w_i$.

The argument when $i \neq j$ is similar. First, wlog we assume that both $\lceil k_i \rceil = \lceil k_i - \epsilon \rceil$ and that $\lceil k_j \rceil = \lceil k_j + \epsilon \rceil$ (otherwise we can split ϵ into at most 4 parts $\epsilon_1, \ldots, \epsilon_4$, and apply the argument separately). Consider the caches $S(\alpha)$ that are supposed to lose a weight w_i page (because k_i becomes $k_i - \epsilon$). We say that these caches have an excess, and note that they all contain exactly $\lceil k_i \rceil$ class i pages. Next, we arbitrarily choose ϵ measure of caches that contain a, and remove a from them. These caches have a hole, and strictly fewer class i pages than caches with excess. We arbitrarily pair caches with an excess to caches with a hole, and transfer some page from the larger cache that does not lie in the smaller cache. The total cost incurred is $3\epsilon w_i/2$. Applying an identical argument to class j, we incur a cost $3\epsilon w_j/2$.

The distribution D' thus obtained satisfies all the conditions required to lie in the set $\mathcal{D}(P')$. Moreover, the total cost incurred in moving from D to D' is $5\epsilon(w_i + w_j)/2$ which is at most 5 times the fractional cost. ■

6 Concluding Remarks

It would be very interesting to see if the techniques in this paper can be extended and applied to more general spaces. We believe that the key difference from previous techniques is that our analysis of paging does not depend on "phases" to lower bound the cost of the offline algorithm.

Another natural question is whether our linear dependence on the number of weight classes can be improved. This seems hard using our approach since the only information we use in our algorithm is the existence of some class for which the online has no fewer pages in the cache than the offline algorithm, that we use to pay for all cost incurred by all other classes. Thus it seems that improving this dependence will require a deeper understanding of the problem. A possibly easier, yet interesting problem would be to obtain a $\mathrm{polylog}(k)$ competitive ratio, independent of the value of ℓ.

Acknowledgments

I would like to thank Tracy Kimbrel, Konstantin Makarychev and Anupam Gupta for useful discussions about the problem.

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] Y. Bartal. On Approximating Arbitrary Metrics by Tree Metrics. *STOC* 1998, 161–168.

[3] Y. Bartal. Probabilistic Approximations of Metric Spaces and Its Algorithmic Applications. *FOCS* 1996, 184–193.

[4] Y. Bartal, B. Bollobas and M. Mendel. A Ramsey-type Theorem for Metric Spaces and its Applications for Metrical Task Systems and Related Problems. *FOCS* 2001, 396–405.

[5] Y. Bartal, A. Blum, C. Burch and A. Tomkins. A polylog(n)-Competitive Algorithm for Metrical Task Systems. *STOC* 1997, 711–719.

[6] Y. Bartal and M. Mendel. Randomized k-server algorithms for growth-rate bounded graphs. *J. Algorithms*, 55(2), 192–202, 2005.

[7] A. Blum, C. Burch and A. Kalai. Finely-Competitive Paging. *FOCS* 1999, 450–458.

[8] A. Borodin and R. El-Yaniv, Online Computation and Competitive Analysis, 1998, Cambridge University Press.

[9] 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.

[10] B. Csaba and S. Lodha. A randomized on-line algorithm for the k-server problem on a line. *Technical Report 2001-34*, DIMACS, October 2001.

[11] J. Fakcharoenphol, S. Rao and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *STOC* 2003, 448–455.

[12] A. Fiat and M. Mendel. Better Algorithms for Unfair Metrical Task Systems and Applications. *SIAM Journal on Computing*, 32(6), 1403–1422, 2003.

[13] A. Fiat, Y. Rabani and Y. Ravid. Competitive k-Server Algorithms. *Journal of Computer and System Sciences*, 48(3), 410–428, 1994.

[14] 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.

[15] S. Irani. Randomized Weighted Caching with Two Page Weights. *Algorithmica*, 32(4), 624–640, 2002.

[16] E. Koutsoupias and C. H. Papadimitriou. On the k-Server Conjecture *Journal of the ACM*, 42(5), 971–983, 1995.

[17] L. A. McGeoch and D. D. Sleator. A Strongly Competitive Randomized Paging Algorithm. *Algorithmica*, 6(6), 816–825, 1991.

[18] M. Mendel. Personal Communication.

[19] S. S. Seiden. A General Decomposition Theorem for the k-Server Problem. *Information and Computation*, 174(2), 193–202, 2002.

[20] D. D. Sleator and R. E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of the ACM*, 28(2), 202–208, 1985.

[21] N. E. Young. The k-Server Dual and Loose Competitiveness for Paging. *Algorithmica*, 11(6), 525–541, 1994.

[22] N. E. Young. On-Line Caching as Cache Size Varies. *SODA* 1991, pages 241–250.