A Primal-Dual Randomized Algorithm for Weighted Paging

Nikhil Bansal
IBM T. J. Watson Research,
Yorktown Heights, NY 10598.
nikhil@us.ibm.com

Niv Buchbinder
Computer Science Department,
Technion, Haifa 32000, Israel.
nivb@cs.technion.ac.il

Joseph (Seffi) Naor*
Computer Science Department,
Technion, Haifa 32000, Israel.
naor@cs.technion.ac.il

Abstract

In the weighted paging problem there is a weight (cost) for fetching each page into the cache. We design a randomized $O(\log k)$-competitive online algorithm for the weighted paging problem, where k is the cache size. This is the first randomized $o(k)$-competitive algorithm and its competitiveness matches the known lower bound on the problem. More generally, we design an $O(\log(k/(k-h+1)))$-competitive online algorithm for the version of the problem where the online algorithm has cache size k and the offline algorithm has cache size $h \leq k$. Weighted paging is a special case (weighted star metric) of the well known k-server problem for which it is a major open question whether randomization can be useful in obtaining sublinear competitive algorithms. Therefore, abstracting and extending the insights from paging is a key step in the resolution of the k-server problem.

Our solution for the weighted paging problem is based on a two-step approach. In the first step we obtain an $O(\log k)$-competitive fractional algorithm which is based on a novel online primal-dual approach. In the second step we obtain a randomized algorithm by rounding online the fractional solution to an actual distribution on integral cache solutions. We conclude with a randomized $O(\log N)$-competitive algorithm for the well studied Metrical Task System problem (MTS) on a metric defined by a weighted star on N leaves, improving upon a previous $O(\log^2 N)$-competitive algorithm of Blum et al. [9].

*This work was done while visiting Microsoft Research, Redmond, WA. Research supported in part by ISF Grant 1366/07 and US-Israel BSF Grant 2002276.

1 Introduction

We consider the following classic online paging problem. 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. Paging is one of the earliest and most extensively studied problems in online computation and competitive analysis.

The unweighted paging problem is very well understood. In their seminal paper, Sleator and Tarjan [29] showed that any deterministic algorithm is at least k-competitive, and also showed that LRU (Least Recently Used) is exactly 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 exactly $k/(k-h+1)$-competitive. When randomization is allowed, Fiat et al. [21] designed the Randomized Marking algorithm which is $2H_k$-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. Subsequently, McGeoch and Sleator [25] gave a matching H_k-competitive algorithm, and Achlioptas, Chrobak and Noga [1] gave another H_k-competitive algorithm that is easier to state and analyze. For (h, k)-paging, Young [31] gave a $2\ln(k/(k-h))$-competitive algorithm (ignoring lower order terms) and showed that any algorithm is at least $\ln(k/(k-h))$-competitive. 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 [10] for further details.

For weighted paging, a tight k-competitive deterministic algorithm follows from the more general work of Chrobak et al. [17] for k-server problem on trees (see below). Subsequently, Young [30] gave a tight $k/(k-h+1)$-competitive deterministic algorithm for the more general (h,k)-paging problem. Despite substantial interest no $o(k)$ randomized algorithms are known for weighted paging. The problem has been described as a "challenge" in [30], and stated as a key open problem in [10] (Problems 11.1 and 11.2) and various other papers [27, 2]. Substantial progress was made by Irani [23] who gave an $O(\log k)$-competitive algorithm for the two weight case, i.e. when each page weight is either 1 or some fixed $M > 1$. In another direction, Blum, Furst and Tomkins [9] gave an $O(\log^2 k)$-competitive algorithm for the case of $n = k+1$ pages. Later, Fiat and Mendel [19] gave an improved $O(\log k)$ competitive algorithm for the case of $n = k + c$ pages, where c is a constant. For large n however, no $o(k)$-competitive algorithm was known (prior to our work) even for the case of three distinct weights.

Paging can be 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 the k-server problem on the metric space in which the distance between any two pages a and b is $(w(a)+w(b))/2$, where $w(\cdot)$ denotes the page weights.

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 [20] 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 Koutsoupias and Papadimitriou [24] 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. [17] gave a k-competitive algorithm for trees. We refer the reader to [10] for more details on the k-server problem.

Nevertheless, randomized algorithms for the k-server problem remain poorly understood. No lower bound better than $\ln k$ is known for any metric space. Moreover, from the work of Bartal, Bollobas and Mendel [4] and Bartal, Linial, Mendel and Naor [6], it follows 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. In a breakthrough result, Bartal, Blum, Burch and Tomkins [5] gave a poly-log(N) competitive algorithm for the metrical task system problem (see definitions in the following paragraph) that implies a poly-log(k)-competitive algorithm for the k-server on a space with $k + c$ points, where c is a constant independent of k. This guarantee was improved by Fiat and Mendel [19] to $O(\log^2 k \log\log k)$. However, for n much larger than k, no algorithms with 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 [28]. Csaba and Lodha [18] 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[1].

A closely related problem is the metrical task system problem (MTS). In the MTS problem we are given a finite metric space $\mathcal{M} = (V, d)$, where $|V| = N$. We view the points of $\mathcal{M}$ as states in which the algorithm may be situated in. The distance between the points of the metric measures the cost of transition between the possible states. We use the k-server notation and say that a server that serves the requests is moving between the states. Each task (request) r in a metrical task system is associated with a vector $(r(1), r(2), \dots, r(N))$, where $r(i)$ denotes the cost of serving r in state $i \in V$. In order to serve request r in state i the server has to be in state i. Upon arrival of a new request, the state of the system can first be changed to a new state (paying the transition cost), and only then the request is served (paying for serving the request in the new state). The objective is to minimize the total cost which is the sum of the transition cost and the service cost.

The MTS model was formulated by Borodin, Linial and Saks [11] who gave tight upper and lower bound of $2N-1$ for any deterministic online algorithm for the problem. They also designed a $2H_N$-competitive randomized algorithm for the uniform metric, and showed a lower bound of H_N for this metric. In fact, our proposed algorithm for the weighed star can be seen in retrospect as a direct generalization of their approach. For the MTS problem on a weighted star, Blum et al. [9] gave a random-

[1] A generalization of this result was considered by Bartal and Mendel [7], who proposed a $\Delta^{1-\epsilon}$polylog$k$ competitive algorithm for bounded growth metrics with diameter Δ. Unfortunately, their result seems to have a serious error [M. Mendel, personal communication].

ized $O(\log^2 N)$-competitive algorithm . For general metrics Bartal et al. [5] designed a randomized $O(\log^5 N)$-competitive algorithm which is based on an algorithm for HST's. Fiat and Mendel [19] improved this bound and designed an $O(\log^2 N \log\log N)$-competitive algorithm for general metrics.

1.1 Results and Techniques

Abstracting and extending the insights from paging is a key step in the resolution of the k-server problem. We obtain the following results:

Theorem 1.1 (Weighted Paging). *There is an $O(\log k)$-competitive randomized algorithm for the online weighted paging problem. More generally, there is an $O(\log(k/(k-h+1)))$-competitive algorithm for the case where where the online algorithm has cache size k and the offline algorithm has cache size $h \leq k$.*

Theorem 1.2 (Metrical Task Systems). *There is an $O(\log N)$-competitive randomized algorithm for the online Metrical Task System (MTS) problem on a metric defined by a weighted star on N leaves.*

A novel aspect of our weighted paging algorithm is that it does not rely on "phases" to lower bound the cost of the optimum algorithm. Our proposed algorithm is based on a two-step approach. In the first step we obtain an $O(\log k)$-competitive *fractional* algorithm for weighted paging. In a fractional solution the algorithm is allowed to maintain fractions of pages in the cache, as long as the sum of the page fractions does not exceed the cache size, k. The cost of fetching an ϵ fraction of a page is then ϵ times the weight associated with the page. In order to design an online algorithm that generates a fractional competitive solution we formulate the problem using a *covering* linear program with box constraints [2]. The same formulation was previously used by Bar-Noy et al. [3] for approximating the offline version of the problem. Cohen and Kaplan [15] used the formulation to give an alternative proof of the competitiveness of Young's dual-greedy algorithm [30] (which is $k/(k-h+1)$-competitive). The linear programming formulation allows us to utilize the primal-dual framework of [12] developed for online packing and covering problems. However, the primal-dual framework of [12] can only lead to an $O(\log n)$-competitive algorithm for the paging problem. Thus, obtaining an $O(\log k)$-competitive bound requires new ideas. The main new idea is combining the approach of [12] with the well known primal-dual scheme. We are confident that this more general approach will find further applications to other online problems.

[2]Box constraints are upper bounds on the values of the variables in the linear program.

Finally, we need to obtain a randomized algorithm. To this end, we need to transform the fractional solution to an actual distribution on integral cache solutions. We show that such a transformation is possible while paying only an additional (small) constant multiplicative factor. Combining the two steps together yields the desired randomized $O(\log k)$-competitive algorithm.

The $O(\log N)$-competitive algorithm for the MTS problem on a weighted star is designed along the same lines, but requires an additional initial idea. As a first step we define a new MTS model and show that as a result on a weighted star metric the cost of an optimal solution can only change by a constant factor. We also show that any solution to the new MTS model with cost C is a solution to the MTS model with cost at most $2C$. Finally, we formulate the new MTS model as a covering LP, show how to generate a fractional solution to it in an online fashion, and then show that the fractional solution can be rounded using a simple randomized rounding method with no further cost.

The primal-dual method is one of the fundamental design methodologies in the areas of approximation algorithms and combinatorial optimization. Recently, Buchbinder and Naor [12] have further extended the primal-dual method and have shown its wide applicability to the design and analysis of online algorithms. In further work, [13, 14] have shown more applications of the primal-dual method to online algorithms. The primal-dual method is useful for online problems such as ski-rental, ad-auctions, routing and load balancing, network optimization problems, and more. We use the primal-dual method here for both making online decisions as well as for the competitive analysis. We note that the connection between competitive analysis and linear programming duality was made explicit for the first time by Young [30] in his work on weighted paging, where an optimal deterministic k-competitive algorithm is obtained via a primal-dual approach. The linear programming formulation used by Young [30] was originally suggested by Chrobak et al. [17], however, this formulation is not a covering-packing linear program.

2 Preliminaries

We study the classic weighted paging problem where there is a cache of size k and the total number of pages is $n > k$. Each page i is associated with a positive weight $w(i) \geq 1$, denoting the cost of fetching the page into the cache. A request sequence is a sequence of pages, denoted by $p_1, p_2, \ldots$, where page p_j is requested at time j. The jth request is served by placing page p_j in the cache at time j, for each $j \geq 1$. The objective is to minimize the total cost of fetching pages into the cache. We relax the model and charge for evicting pages instead of loading them. This relaxation can only change the cost of any algorithm by an additive factor (fetching the last k pages is "free").

Linear Programming Relaxation. We formulate a simple linear programming relaxation of the paging problem. Let $x(i,j)$ be an indicator to the event that page i is evicted from the cache between the jth time it is requested and the $(j+1)$st time it is requested. If $x(i,j)=1$, we can assume without loss of generality that page i is evicted in the first time slot following the jth time it is requested. (As we later discuss, this assumption is not necessarily true in the online case.) In a fractional solution, we allow $x(i,j)$ to get any value between 0 and 1. For each page i, denote by $t(i,j)$ the time it is requested for the jth time, and denote by $r(i,t)$ the number of times page i is requested until time t (including t). For any time t, let $B(t)=\{i \mid r(i,t)\geq 1\}$ denote the set of pages requested until time t (including t).

A feasible solution to the paging problem can maintain at any time t at most k pages in the cache. In a fractional solution the sum of the page fractions in the cache is at most k. Also, page p_t must be in the cache at time t. Thus, any feasible paging solution must evacuate at least $|B(t)|-k$ pages out of the set $B(t)\backslash\{p_t\}$, yielding the following linear programming formulation:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{r(i,t)} w(i)x(i,j) \qquad \text{(LP-Paging)}$$

$$\text{For any time } t\text{:} \quad \sum_{i\in B(t)\backslash\{p_t\}} x(i,r(i,t)) \geq \quad |B(t)|-k \quad (1)$$

$$\text{For any } i,j\text{:} \qquad 0\leq x(i,j)\leq 1$$

In the dual program, there is a variable $y(t)$ for each time t and a variable $z(i,j)$ for each page i and the jth time it is requested. The dual program is the following:

$$\max \quad \sum_t \left(|B(t)|-k\right) y(t) \;-\; \sum_{i=1}^{n}\sum_{j=1}^{r(i,t)} z(i,j)$$

For each page i and the jth time it is requested:

$$\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i,j) \leq w(i) \qquad (2)$$

$$\text{For any } i,j\text{:} \qquad z(i,j)\geq 0$$

$$\text{For all } t\text{:} \qquad y(t)\geq 0$$

Randomized Algorithms. A randomized algorithm is completely specified by a probability distribution on the various configurations (deterministic states) in each state of the algorithm. For the k-server problem this corresponds to specifying the distribution on k-tuples of server positions. Such a distribution induces another (simpler) distribution $p(x,t)$ on the points in the metric space, specifying 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 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.

For the k-server problem, the distribution on the points in the metric space can be viewed as a probability mass of k units distributed among the n points, and the "move" of an algorithm simply corresponds to redistributing 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* (in contrast to working with the probability distribution on states, that we call the *actual* view). This fractional view has been considered previously [5, 8]. Blum et al. [8] showed that for the unweighted paging problem it is possible to transform online a fractional view to an actual view such that the expected cost incurred is at most twice the cost of the fractional view. We note that a fractional view can easily be obtained from a solution to linear program (LP-Paging), since the variables in the linear program indicate what fraction of a page is already evacuated from the cache. More formally, at time t, $p(p_t,t)=1$, and for each $i\neq p_t$, $p(i,t)=1-x(i,r(i,t))$.

Our goal is to generate a randomized algorithm from a fractional view. To this end, we adopt a similar approach as [8]. However, for weighted paging the relation between fractional view and actual view is much more subtle. Consider for example the distribution $(1/2,1/2,1/2,1/2)$ on pages A,B,C and D induced by the actual view where 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 of $(0,1,1/2,1/2)$. 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 (to be consistent with the fractional 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, from the previous cache state (C,D), either C or D must be moved to make room for B, which incurs cost $\Theta(M)$.

Interestingly, for weighted paging we get around this problem by restricting our actual distributions to a certain subclass of distributions (for example in the scenario described above, we do not allow the distribution where states (A,B) and (C,D) have probability half each to correspond to the distribution $(1/2, 1/2, 1/2, 1, 2)$). In particular, in Section 4 we show how to maintain an online mapping from induced distributions to actual distributions, such that any fractional move with cost c is mapped to a move on actual distributions with cost at most $5c$. Hence, throughout this paper we work with the fractional view.

The Metrical Task System Problem. We consider the metrical task system (MTS) problem on a metric $\mathcal{M}$ defined by a weighted star. The leaves of the star are denoted by $\{1, 2, \ldots, N\}$, and the distance from the center of the star to leaf i is $d(i)$. There is a single server and the leaf in which the server is located defines the *state* of the system. The cost of moving the server from state i to state j is $d(i) + d(j)$. We can assume that the server is initially in state 1. Each task (request) r in a metrical task system is associated with a vector $(r(1), r(2), \ldots, r(N))$, where $r(i)$ denotes the cost of serving r in state (leaf) i. In order to serve request r in state i the server has to be in leaf i. Upon arrival of a new request, the state of the system can first be changed to a new state (paying the transition cost), and only then the request is served (paying for serving the request in the new state).

3 A Fractional Primal-Dual Algorithm for the Weighted Paging Problem

Our online paging algorithm produces fractional primal and dual solutions to LP-Paging. In the online case, the constraints of LP-Paging (corresponding to the requests to pages) are revealed one-by-one. Upon arrival of a constraint, the algorithm finds a feasible assignment to the (primal) variables that satisfies the constraint. Consider variable $x(i, j)$. In the offline case, we can assume without loss of generality that the value of $x(i, j)$ is determined at time $t(i, j)+1$. However, this is not necessarily true in the online case; thus, we stipulate that the values assigned to $x(i, j)$ in the time interval $[t(i, j) + 1, t(i, j + 1) - 1]$ by the online algorithm form a monotonically non-decreasing sequence.

We start with a high level description of the algorithm. Upon arrival of a new constraint at time t, if it is already satisfied, then the algorithm does nothing. Otherwise, the algorithm needs to satisfy the current constraint by increasing some of the primal variables in the constraint. Satisfying the constraint guarantees that there is a enough space in the cache to fetch the new page. To this end, the algorithm starts increasing (continuously) the new dual variable $y(t)$. This, in turn, tightens some of the dual constraints corresponding to primal variables $x(i, j)$ whose current value is 0. Whenever such an event happens, the value of $x(i, j)$ is increased from its initial setting of 0 to $1/k$. Thus, during the time preceding the increase of $x(i, j)$ from 0 to $1/k$, page i cannot be evicted from the cache. This part is somewhat similar to what happens in the Randomized Marking algorithm [21]. Meanwhile, variables $x(i, j)$ which are already set to $1/k$ are increased (continuously) according to an exponential function of the new dual variable $y(t)$. Note that this exponential function is equal to $1/k$ when the constraint is tight. Thus, the algorithm is well defined. When variable $x(i, j)$ reaches 1, the algorithm starts increasing the dual variable $z(i, j)$ at the same rate as $y(t)$. As a result, from this time on, the value of $x(i, j)$ remains 1. The algorithm is presented in a continuous fashion, but it can easily be implemented in a discrete fashion. The algorithm is the following:

Fractional Paging Algorithm: At time t, when page p_t is requested:

- Set the new variable: $x(p_t, r(p_t, t)) \leftarrow 0$. (It can only be increased in times $t' > t$.)
- If the primal constraint corresponding to time t is satisfied, then do nothing.
- Otherwise: increase primal and dual variables, until the primal constraint corresponding to time t is satisfied, as follows:
  1. Increase variable $y(t)$ continuously; for each variable $x(i, j)$ that appears in the (yet unsatisfied) primal constraint that corresponds to time t:
  2. If $x(i, j) = 1$, then increase $z(i, j)$ at the same rate as $y(t)$.
  3. If $x(i, j) = 0$ and
  $$\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i, j) = w(i),$$
  then set $x(i, j) \leftarrow 1/k$.
  4. If $1/k \le x(i, j) < 1$, increase $x(i, j)$ according to the following function:

$$\frac{1}{k} \cdot \exp\left(\frac{1}{w(i)}\left[\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i, j) - w(i)\right]\right)$$

Note that the exponential function for $x(i, j)$ contains variables $y(t)$ that correspond to future times. However, these variables are all initialized to 0, so they do not contribute to the value of the function.

The analysis of the primal cost is partitioned into two parts. The first one corresponds to the contribution of the increase of the variables $x(i, j)$ from 0 to $1/k$, and the second part corresponds to the increase of the variables $x(i, j)$ from $1/k$ till (at most) 1, according to the exponential function.

Each part is upper bounded separately by the dual solution, yielding the desired result. We now prove the following theorem.

Theorem 3.1. *The algorithm is $O(\log k)$-competitive.*

Proof. First, we note that the primal solution generated by the algorithm is feasible. This follows since, in each iteration, the variables $x(i,j)$ are increased until the new primal constraint is satisfied. Also, each variable $x(i,j)$ is never increased to be greater than 1.

Next, we show that the dual solution that we generate is almost feasible. Whenever $x(i,j)$ increases in some round and reaches 1, the algorithm starts increasing $z(i,j)$ at the same rate as $y(t)$. Therefore, the value of $x(i,j)$ is not going to change anymore, as the exponent of the exponential function will not change any more. Thus, for the dual constraint corresponding to page i and the jth time it is requested, we get that:

$$x(i,j) = \frac{1}{k}\exp\left(\frac{1}{w(i)}\left[\left[\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i,j) - w(i)\right]\right]\right) \leq 1.$$

Simplifying, we get that:

$$\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i,j) \leq w(i)(1+\ln k). \tag{3}$$

Thus, the dual solution can be made feasible by scaling it down by a factor of $(1+\ln k)$. We now prove that the primal cost is at most twice the dual profit, which means that the primal solution produced by the algorithm is $2(1+\ln k)$-competitive.

We partition the primal cost into two parts. Let C_1 be the contribution to the primal cost from Step (3) of the algorithm, due to the increase of variables $x(i,j)$ from 0 to $1/k$. Let C_2 be the contribution to the primal cost from Step (4) of the algorithm, due to the incremental increases of the variable $x(i,j)$ according to the exponential function from $1/k$ up to at most 1.

Bounding C_1: For page i and the jth time it is requested, it follows from the algorithm that if $x(i,j) > 0$, then:

$$\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i,j) \geq w(i) \tag{4}$$

(*primal complementary slackness*)

Next, in the dual solution, if $y(t) > 0$, then:

$$\sum_{i\in B(t)\setminus\{p_t\}} x(i, r(i,t)) \leq |B(t)| - k \tag{5}$$

(*dual complementary slackness*)

The inequality follows since there are $|B(t)| - 1$ variables in the constraint corresponding to t. Thus, even if all the variables are increased from 0 to $1/k$, they add up to $\frac{|B(t)|-1}{k} \leq |B(t)| - k$. The latter inequality holds since $|B(t)| \geq k+1$. Also, it follows from the algorithm that if $z(i,j) > 0$, then:

$$x(i,j) \geq 1 \qquad (\textit{dual complementary slackness}) \tag{6}$$

For completeness, we state what the primal and dual complementary slackness conditions imply.

$$\begin{aligned}
&\sum_{i=1}^{n}\sum_{j=1}^{r(i,t)} w(i)x(i,j) \\
&\leq \sum_{i=1}^{n}\sum_{j=1}^{r(i,t)}\left(\left(\sum_{t=t(i,j)+1}^{t(i,j+1)-1} y(t)\right) - z(i,j)\right)x(i,j) \qquad (7)\\
&= \sum_{t}\left(\sum_{i\in B(t)\setminus\{p_t\}} x(i,r(i,t))\right)y(t) \\
&\quad - \sum_{i=1}^{n}\sum_{j=1}^{r(i,t)} x(i,j)z(i,j) \qquad (8)\\
&\leq \sum_{t}\left(|B(t)| - k\right)y(t) - \sum_{i=1}^{n}\sum_{j=1}^{r(i,t)} z(i,j). \qquad (9)
\end{aligned}$$

Inequality (7) follows from Inequality (4), Equality (8) follows by changing the order of summation, and Inequality (9) follows from Inequalities (5) and (6). Thus, C_1 is at most the profit of a *feasible* dual solution multiplied by $(1+\ln k)$.

Bounding C_2: We bound the derivative of the increase of variables $x(i,j)$ in Step (4) by the derivative of the dual profit accrued in the same round. In each round only variables $x(i,j)$ that belong to the new primal constraint (and correspond to the new dual variable $y(t)$) are being increased. However, variables $x(i,j)$ that belong to the new primal constraint but have already reached the value of 1 are not increased anymore and so do not contribute to the primal cost. In the dual program the new variable $y(t)$ is raised with rate 1, and also all the variables $z(i,j)$ that correspond to $x(i,j)$ in the new primal constraint that are already with value 1. It is beneficial for the purpose of analysis to think of the process as increasing a time variable τ, and then raising the variable $y(t)$ and the appropriate variables $z(i,j)$ with rate 1 with respect to the virtual variable τ. Using this notation we get that:

$$\begin{aligned}
\frac{dC_2}{d\tau} &= \sum_{i \in B(t) \setminus \{p_t\}, 1/k \le x(i,j) < 1} w(i) \cdot \frac{dx(i, r(i,t))}{dy(t)} \cdot \frac{dy(t)}{d\tau} \\
&= \sum_{i \in B(t) \setminus \{p_t\}, 1/k \le x(i,j) < 1} x(i, r(i,t)) \qquad (10) \\
&\le (|B(t)| - k) - \sum_{i \in B(t) \setminus \{p_t\}, x(i,j) = 1} 1 \qquad (11) \\
&= (|B(t)| - k) \frac{dy(t)}{d\tau} - \sum_{i \in B(t) \setminus \{p_t\}, x(i,j) = 1} \frac{dz(i,j)}{d\tau}
\end{aligned}$$

Where Equality (10) follows since $\frac{dy(t)}{d\tau} = 1$ and also $\frac{dx(i,j)}{dy(t)} = \frac{1}{w(i)} \cdot x(i,j)$ for each $1/k \le x(i,j) < 1$. Inequality (11) holds since the new primal constraint is unsatisfied yet and thus:

$$\sum_{i \in B(t) \setminus \{p_t\}, 1/k \le x(i,j) < 1} x(i, r(i,t)) + \sum_{i \in B(t) \setminus \{p_t\}, x(i,j) = 1} x(i, r(i,t)) < |B(t)| - k$$

We also remark that by the properties of the algorithm, any variable $x(i,j)$ which is strictly less than $1/k$ is actually equal to 0. Finally, the last term exactly equals the derivative of the dual profit with respect to τ. Therefore, the change in the dual profit is greater than or equal to the change in C_2. Thus, C_2 is at most the profit of a *feasible* dual solution multiplied by $(1 + \ln k)$.

Completing the analysis. It follows that $C_1 + C_2$ is at most twice the profit of a *feasible* dual solution multiplied by $(1 + \ln k)$. Note that the profit of any dual feasible solution is always a lower bound on the optimal solution. Therefore, we conclude by weak duality that the algorithm is $2(1 + \ln k)$-competitive. □

The weighted (h,k)-paging. In the (h,k) version of the weighted paging problem the online algorithm has a cache of size k, and its performance is compared with an optimal offline algorithm that has a cache of size $h \le k$. The design of a fractional $O(\log(k/(k-h+1)))$-competitive algorithm for this version requires several minor changes to our algorithm. In particular, the variables $x(i,j)$ are initially set to $\frac{k-h+1}{k}$ instead of $\frac{1}{k}$ when their dual constraint is tight. We defer the algorithm and analysis to the full version of the paper.

4 Rounding the Fractional Solution Online

We first round up the page weights to their nearest power of 2 (increasing the competitive ratio by at most a factor of 2). Let $w_1 < w_2 < \ldots < w_\ell$ denote the rounded weights. A page belongs to class i if its rounded weight is w_i. We refer to an individual page as the j-th page of class i. For convenience of analysis, throughout this section we consider the (equivalent) cost version of the problem where we pay $w_i/2$ for both fetching and evicting a class i page.

Recall that in the fractional view of the problem, the algorithm maintains a distribution on the pages with total mass k. Any such distribution P is completely specified by $p_{ij} \in [0,1]$ such that $\sum_i \sum_j p_{ij} = k$, where p_{ij} is the mass on the j-th page of weight class i. Given two distributions P and P' on pages, let $C_f(P, P')$ denote the cheapest way to move from P to P', where it costs $w_i/2$ to move one unit of mass either into or out of a class i page. For those familiar, C_f is just the transshipment cost of flow from P to P' (we refer the reader to [16] for details about transshipment cost between distributions). Let $\delta_{ij} = p_{ij} - p'_{ij}$. Clearly, $C_f(P, P')$ is at least $\sum_i (w_i/2)(\sum_j |\delta_{ij}|)$ since at least $|\delta_{ij}|$ units of mass either needs to enter or leave page j of class i. Moreover, any greedy algorithm that arbitrarily moves mass out of pages with excess ($\delta_{ij} > 0$) to those with a deficiency ($\delta_{ij} < 0$) has cost $\sum_i (w_i/2)(\sum_j |\delta_{ij}|)$ implying that $C_f(P, P') = \sum_i (w_i/2)(\sum_j |\delta_{ij}|)$.

A randomized algorithm on the other hand needs to work with a distribution on valid cache states. Given two distributions D and D' on the cache states, let $C(D, D')$ denote the cheapest way of moving from D to D' (by definition, this is the cost incurred by the randomized algorithm). Let $\Pi(D)$ denote the distribution induced on the pages by D. We say that P and D are *consistent* if $P = \Pi(D)$. Clearly, $C_f(\Pi(D), \Pi(D')$ is a lower bound on $C(D, D')$.

For the unweighted paging problem, Blum, Burch and Kalai [8] showed that given any P, P' and D such that $\Pi(D) = P$, there exists some D' such that $\Pi(D') = P'$ and $C(D, D') \le 2C_f(P, P')$. Their procedure is the following. Suppose without loss of generality that P' is obtained from P by removing ϵ units of mass from page a and putting it on page b. Remove page a arbitrarily from ϵ measure of caches that contain a, and add page b to ϵ measure of caches that do not contain b. Now, some caches may have $k+1$ pages (an excess) while some may have $k-1$ pages (a hole). Arbitrarily match the caches with an excess to those with a hole (clearly, the measure of caches with excess is equal to those with a hole). Consider any matched pair; the cache with an excess must contain a page that does not lie in its matched cache, so we simply transfer this page. It can easily be verified that $C(D, D') \le 2\epsilon$, while the fractional cost $C_f(P, P') = \epsilon$.

However the situation for weighted paging is more involved. Recall the example in Section 2. It shows that there exist P, P' and D consistent with P, such that $C(D, D') \gg C_f(P, P')$ for every D' satisfying $P' = \Pi(D')$. Thus, we cannot work with any arbitrary D that is consistent with P, as in the unweighted case. Interestingly, we get around this problem by carefully restricting the space of distributions D that we are allowed to work with. Formally, we show the following.

Theorem 4.1. *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 $(\mathcal{D} \times \mathcal{P}) \to \mathcal{D}$ 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 $\Pi(D') = P'$, $D' \in \mathcal{D}$ and $C(D, D') \leq 5C_f(P, P')$.*

The theorem gives us the desired mapping between a distribution P on pages and a 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. Let P be a distribution on pages with total mass k. Let $\mathcal{D}(P)$ denote the set of distributions $D \in \mathcal{D}$ that are consistent with P. Specifying $\mathcal{D}(P)$ for each P suffices to describe $\mathcal{D}$ completely. Each distribution $D \in \mathcal{D}$ is specified by associating a cache state $C(\alpha)$ with each real number α in the interval $[0, 1)$.

Let $k_i = \sum_j p_{ij}$ denote the mass on class i pages as determined by P. Consider the interval $I = [0, k]$, and imagine this interval partitioned into $I_1, \ldots, I_l$ where $I_1 = [0, k_1)$, $I_2 = [k_1, k_1 + k_2)$,..., $I_l = [k_1 + \ldots, k_{l-1}, k_1 + \ldots + k_l)$. 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 $C(\alpha)$ has n_i pages of w_i where $n_i = |T(\alpha) \cap I_i|$. By construction, each cache $C(\alpha)$ 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 $C(\alpha)$, for $0 \leq \alpha < 1$, with pages such that: (i) no $C(\alpha)$ contains two identical pages and (ii) 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 the first page of class 1 in $C(\alpha)$ corresponding to $\alpha = [0, p_{11})$, the second page of class 1 in $C(\alpha)$ 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 $C(\alpha)$'s 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 separating the pages 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 $P = P_0 \to P_1 \to P_2 \to \ldots \to P'$ 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 an *exchange* 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 such exchanges $P \to P'$. Let i be the weight class of page a, and j be that of page b. For this move, the fractional algorithm pays $\epsilon(w_i + w_j)/2$.

We now describe and analyze the move $D \to D'$. We divide the cost into two parts. One due to cache size changes, and the second due to the change in the composition of the cache. We first consider the simpler case when $i = j$. Here, the quantities $k_1, \ldots, k_\ell$ and the intervals $I_1, \ldots, I_\ell$ associated with P remain unchanged, and hence the structure of $C(\alpha)$'s remains unchanged. We essentially apply the argument of Blum et al. [8] to class i pages. The only difference is that we need to verify that their argument works even when caches contain either $\lceil k_i \rceil$ or $\lfloor k_i \rfloor$ class i pages (in [8] all caches have the same size). 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 a cache has a hole if it has one fewer page than it is supposed to, and it has an excess if it has one extra page than it is supposed to. 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 $2\epsilon w_i/2 + 2\epsilon w_i/2 = 2\epsilon w_i$.

We now consider the case when $i < j$ (the case when $i > j$ is analogous). Consider the intervals $I_1, \ldots, I_\ell$. 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.

We break the analysis into two parts. We first consider the classes h for $i < h < j$. For each such h at most ϵ fraction of caches $C(\alpha)$ 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 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.

Finally, we consider the case when $h = i$ (the argument for $h = j$ is analogous). Without loss of generality we assume that $\lceil k_i \rceil = \lceil k_i - \epsilon \rceil$ (otherwise we can split ϵ into at most 2 parts ϵ_1, ϵ_2, and apply the argument separately). Consider the caches $C(\alpha)$ 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 (a cache with a hole has either $\lfloor k_i \rfloor$ or $\lfloor k_i \rfloor - 1$ pages). 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 cost incurred is at most $3\epsilon w_i/2$. By an identical argument for class j, the cost incurred is at most $3\epsilon w_j/2$.

The distribution D' 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. □

5 The Metrical Task System Problem on a Weighted Star Metric

We consider in this section the metrical task system (MTS) problem on a metric $\mathcal{M}$ defined by a weighted star. The leaves of the star are denoted by $\{1, 2, \ldots, N\}$. We present an $O(\log N)$-competitive online algorithm. We are going to charge the algorithm by $2d(i)$ whenever the server moves from state i to another state, say j. Thus, we are not going to charge the algorithm for the cost of moving into state j. This assumption can only add an additive term to the total cost which is independent of the request sequence (we do not charge for the last state change). From now on we abuse notation and let $d(i)$ denote the cost of moving from state i to a different state.

We are going to work with the (equivalent) continuous time MTS model. In this model the algorithm is allowed to change states at any time t which is a real number and not only at integral times. The service cost is generalized in a straight forward way to an integral instead of a sum. It is well known [10, Sec. 9.1.1] that any continuous time algorithm can be transformed to a discrete time algorithm without increasing the total cost. On the other hand, since the continuous time model is a relaxation of the discrete model it is clear that the optimal cost can only decrease.

As a first step towards obtaining a competitive online algorithm for the MTS problem we define a new MTS model and show that on a star metric the cost of an optimal solution can only change by a constant factor. The high level idea of the new model is to cancel the transition cost incurred due to state change and pay only for serving the requests. To balance, we restrict the algorithm and allow it to change its state only if certain conditions are fulfilled. For each state we partition the time interval into phases. We permit the solutions to leave each state i only at the end of a phase (of state i). The first phase of each state starts at time $t = 0$. Phase p of state i starts at time $t_{p-1}(i)$ and ends at the earliest time $t_p(i)$ for which the accumulated cost of service at state i in the interval $[t_{p-1}(i), t_p(i)]$ is exactly $d(i)$.

We are now ready to describe the new MTS model in its full generality. An online algorithm is allowed to leave state i only at the end of a phase (of state i). The algorithm does not pay any transition cost when moving from one state to another. If the algorithm is in state i during phase p then it pays a cost $d(i)$. The algorithm pays the full cost of the phase even if it was in state i only during part of the phase p. This can happen if the algorithm moves to state i from i' in the middle of the pth phase of i (and at the end of a phase of state i'). Given a set of requests $\bar{\sigma}$, let $\mathrm{OPT}_n(\bar{\sigma})$ be the minimum offline cost of serving the set of requests in the new MTS model. Let $\mathrm{OPT}_o(\bar{\sigma})$ be the minimum offline cost of serving the set of requests in the standard MTS model. In the following we state two relatively easy lemmas. Due to space constraints we omit the proofs.

Lemma 5.1. *Let $\bar{\sigma}$ be a set of requests. Any solution S to $\bar{\sigma}$ in the standard MTS model with cost C can be transformed into a legal solution S' in the new MTS model with cost at most $2C$. In particular,* $\mathrm{OPT}_n(\bar{\sigma}) \le 2\mathrm{OPT}_o(\bar{\sigma})$.

Lemma 5.2. *Let $\bar{\sigma}$ be a set of requests. Any solution S to $\bar{\sigma}$ in the new MTS model with cost C is a legal solution S' in the standard MTS model with cost at most $2C$.*

From Lemmas 5.1 and 5.2 a c competitive algorithm in the new MTS model implies a $4c$ competitive algorithm in the standard MTS model, and hence it suffices to consider the new MTS model.

5.1 The Algorithm

We next describe a simple linear programming formulation for the offline problem in the new MTS model. Our online algorithm will generate a fractional solution to this linear program. We later show how to transform this fractional solution to a randomized integral solution. Let $x(i,p)$ be an indicator to the event that the solution is in state i during the pth phase. We relax the solution and allow the algorithm to be at time t in several states as long as the sum of the fractions of the states is at least 1. (The latter constraint is valid since our objective function is minimization.). Let n_i be the number of phases of state i. The linear program is then the following:

$$(P) \qquad \min \qquad \sum_{1=1}^{N} \sum_{p=1}^{n_i} d(i) x(i,p)$$

$$\text{For any time } t\text{:} \sum_{i=1}^{N} \sum_{p \mid t \in [t_{p-1}(i), t_p(i)]} x(i,p) \ge 1 \qquad (12)$$

It may seem that the linear program contains an unbounded number of constraints. However, it is easy to see that we need only to consider times t which are the end of a phase for some state. It can also be easily verified that given an instance of the MTS problem, any feasible solution in the new MTS model defines a feasible solution to (P) with the same cost. We also observe that a feasible solution to (P) defines a (fractional) solution which is feasible in the new MTS model with the same cost. We should be a bit more careful in the online case, where the constraints of (P) are revealed one-by-one. Upon arrival of a constraint, the algorithm finds a feasible assignment to the (primal) variables that satisfies the constraint. Consider variable $x(i,p)$. In the offline case, we can assume without loss of generality that the value of $x(i,p)$ is determined at the beginning of phase p of state i. However, this is not necessarily true in the online case; thus, we restrict our attention to solutions that assign values to $x(i,p)$ forming a monotonically non-decreasing sequence.

The dual program has a variable $y(t)$ for each constraint of (P), and we demand that the total sum of the variables $y(t)$, taken over a phase p of state i, is at most $d(i)$. Note that the primal and dual linear programs forms a covering packing pair. The dual program is the following:

$$(D) \qquad \max \sum_{t=1}^{T} y(t)$$

$$\text{For state } i \text{ and phase } p: \quad \sum_{t\in[t_{p-1}(i),t_p(i)]} y(t) \le d(i) \quad (13)$$

We now describe an online primal-dual algorithm for the MTS problem which is a special case of the online covering-packing algorithm described in [12]. Therefore, the analysis of [12] already implies an $O(\log N)$-competitive factor for the algorithm. Since the algorithm described in [12] is more general (and complicated), we provide here for completeness the algorithm and a proof of the competitive factor.

At each time period t (when the phase of some state ends),

- Initiate $y(t) \leftarrow 0$ and $x(i,p) \leftarrow 0$, if the p-th phase begins for i at t.
- While $\sum_{i=1}^{N} \sum_{p \mid t\in[t_{p-1}(i),t_p(i)]} x(i,p) < 1$:
  1. Increase $y(t)$ continuously.
  2. Increase each variable $x(i,p)$ by the following increment function:

$$x(i,p) \leftarrow \frac{1}{N}\left[\exp\left(\frac{\log(1+N)}{d(i)} \sum_{t\in[t_{p-1}(i),t_p(i)]} y(t)\right) - 1\right]$$

Notice that each variable $x(i,p)$ starts from zero and it always increases during the algorithm. Its value is 1 when the corresponding dual constraint is tight.

Theorem 5.3. *The algorithm is $O(\log N)$-competitive.*

Proof. Let $X(t)$ and $Y(t)$ be the values of the primal and dual solutions, respectively, at time t. We prove the following claims:

1. During time t, $\frac{dX(t)}{dy(t)} \le 2\log(1+N)\frac{dY(t)}{dy(t)}$.
2. The primal solution produced is feasible.
3. The dual solution produced is feasible.

The theorem now directly follows from weak LP duality.

Proof of (1): Simple calculations give us:

$$\begin{aligned} \frac{dX(t)}{dy(t)} &= \sum_{i=1}^{N} \sum_{p \mid t\in[t_{p-1}(i),t_p(i)]} d(i)\cdot\frac{dx(i,p)}{dy(t)} \\ &= \log(1+N)\sum_{i=1}^{N} \sum_{p \mid t\in[t_{p-1}(i),t_p(i)]} \left(x(i,p)+\frac{1}{N}\right) \\ &\le 2\log(1+N) = 2\log(1+N)\frac{dY(t)}{dy(t)}. \end{aligned} \quad (14)$$

Where Inequality (14) follows since $\sum_{i=1}^{N} \sum_{p \mid t\in[t_{p-1}(i),t_p(i)]} x(i,p) < 1$ and also the number of variables in any primal constraint is N.

Proof of (2): This claim is trivial since we increase the primal variables until the current primal constraint becomes feasible. We never decrease any $x(i,p)$, so all previous constraints remain feasible.

Proof of (3): Consider the dual constraint of $x(i,p)$. Since each $x(i,p)$ is at most 1 (when it is 1 then all the constraints it belongs to are satisfied), we get:

$$x(i,p) = \frac{1}{N}\left[\exp\left(\frac{\log(1+N)}{d(i)} \sum_{t\in[t_{p-1}(i),t_p(i)]} y(t)\right) - 1\right] \le 1.$$

Simplifying, we get that for each state i and phase p: $\sum_{t\in[t_{p-1}(i),t_p(i)]} y(t) \le d(i)$. □

Rounding the fractional solution. Rounding the fractional solution is simple. The algorithm maintains the invariant that it is in state i at time t (in phase p) with probability equal to $x(i,p)$. Suppose at the end of a phase p in state i, $x(i,p) = a$ and the value of a is distributed among the states of the system (including i) by the fractional solution. Let a_j the increase of the fraction associated with state j at that point of time. As $\sum_j a_j = a$, if the algorithm with was in state i at the end on phase p, it moves to state j with probability a_j/a. It is easy to verify that the expected cost of the algorithm is exactly the cost of the fractional solution.

6 Conclusion

Our main result in this work is a randomized $O(\log k)$-competitive algorithm for the weighted paging problem. We believe that the methods used in this work, and especially the primal-dual approach which is a fundamental method for (offline) optimization problems, are very general and may lead eventually to a sub-linear algorithm for the much more challenging k-server problem.

We have used the techniques in the paper, along with several additional ideas, to obtain poly-logarithmic competitive algorithms for a generalized model of paging in which pages have both arbitrary weights and sizes.

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] S. Albers. Online algorithms: A survey. *Mathematical Programming*, 97: 3–26, 2003.

[3] A. Bar-Noy, R. Bar-Yehuda, A. Freund, J. Naor, and B. Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5), pp. 1069–1090, 2001.

[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. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science*, 2001, 396–405.

[5] Y. Bartal, A. Blum, C. Burch and A. Tomkins. A polylog(n)-Competitive algorithm for metrical task systems. In *Proceedings of the 29th Annual ACM Symposium on Theory of computing*, 1997, 711–719.

[6] Y. Bartal, N. Linial, M. Mendel and A. Naor. On metric ramsey-type phenomena. In *Proceedings of the 35th Annual ACM Symposium on Theory of computing*, 2003, 463–472.

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

[8] A. Blum, C. Burch and A. Kalai. Finely-competitive paging. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999, 450–458.

[9] 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.

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

[11] A. Borodin and N. Linial and M. Saks, An optimal online algorithm for metrical task systems. In *Proceedings of the 19th Annual ACM Symosium on Theory of computing*, 1987, 373–382.

[12] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proceedings of the 13th Annual European Symposium on Algorithms*, 2005, 689–701.

[13] N. Buchbinder and J. Naor. Improved bounds for online routing and packing via a primal-dual approach. In *Proceedings of the 47th Annual Symposium on Foundations of Computer Science*, 2006, 293–304.

[14] N. Buchbinder, K. Jain, and J. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 13th Annual European Symposium on Algorithms*, 2007.

[15] E. Cohen and H. Kaplan. LP-based analysis of greedy-dual-size. In *Proceedings of the 10th Annual ACM-SIAM symposium on Discrete algorithms*, 1999, 879–880.

[16] C. Chekuri, S. Khanna, J. Naor and L. Zosin. A linear programming formulation and approximation algorithms for the metric labeling problem. SIAM J. Discrete Math. 18(3), 608-625, 2004.

[17] 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.

[18] B. Csaba and S. Lodha. A randomized on-line algorithm for the k-server problem on a line. *Random Structures and Algorithms*, 29(1), 82–104, 2006.

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

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

[21] 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.

[22] Sandy Irani. Page replacement with multi-size pages and applications to Web caching. In *Proceedings of the 29th Annual ACM Symposium on Theory of computing*, 1997, 701–710.

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

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

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

[26] Mark S. Manasse and Lyle A. McGeoch and Daniel D. Sleator. Competitive algorithms for on-line problems. In *Proceedings of the 20th Annual ACM Symposium on Theory of computing*, 1997, 322–333.

[27] Mark S. Manasse and Lyle A. McGeoch and Daniel D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2), 208–230,1990 .

[28] S. S. Seiden. A general decomposition theorem for the k-server problem. *Information and Computation*, 174(2), 193–202, 2002.

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

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

[31] 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.