2013 IEEE 54th Annual Symposium on Foundations of Computer Science

On Randomized Memoryless Algorithms for the Weighted k-server Problem

Ashish Chiplunkar*
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
ashishc@cse.iitb.ac.in

Sundar Vishwanathan
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Mumbai, India
sundar@cse.iitb.ac.in

***Abstract*—The weighted k-server problem is a generalization of the k-server problem in which the cost of moving a server of weight β_i through a distance d is $\beta_i \cdot d$. The weighted server problem on uniform spaces models caching where caches have different write costs. We prove tight bounds on the performance of randomized memoryless algorithms for this problem on uniform metric spaces. We prove that there is an α_k competitive memoryless algorithm for this problem, where $\alpha_k = \alpha_{k-1}^2 + 3\alpha_{k-1} + 1$; $\alpha_1 = 1$. On the other hand, we also prove a lower bound result, which is a strong evidence to our conjecture, that no randomized memoryless algorithm can have competitive ratio better than α_k.**

To prove the upper bound of α_k, we develop a framework to bound from above the competitive ratio of any randomized memoryless algorithm for this problem. The key technical contribution is a method for working with potential functions defined implicitly as the solution of a linear system. The result is robust in the sense that a small change in the probabilities used by the algorithm results in a small change in the upper bound on the competitive ratio. The above result has two important implications. Firstly this yields an α_k-competitive memoryless algorithm for the weighted k-server problem on uniform spaces. This is the first competitive algorithm for $k > 2$ which is memoryless. For $k = 2$, our algorithm agrees with the one given by Chrobak and Sgall [1]. Secondly, this helps us prove that the Harmonic algorithm, which chooses probabilities in inverse proportion to weights, has a competitive ratio of $k\alpha_k$.

The only known competitive algorithm for every k before this work is a carefully crafted deterministic algorithm due to Fiat and Ricklin [2]. Their algorithm uses memory crucially and their bound on competitive ratio more than 2^{4^k}. Our algorithm is not only memoryless, but also has a considerably improved competitive ratio of $\alpha_k < 1.6^{2^k}$. Further, the derandomization technique by Ben-David et al. [3] implies that there exists a deterministic algorithm for this problem with competitive ratio $\alpha_k^2 < 2.56^{2^k}$.

***Keywords*-Weighted k-server; Memoryless Randomized Algorithms; Competitive Ratio**

I. Introduction

The k-server problem of Manasse et al. [4] is, arguably, the most extensively studied problem in the online setting. The large body of research around this problem is summarized in the beautiful survey by Koutsoupias [5]. In this problem, k servers occupy points in a metric space. An adversary presents a sequence of requests, each of which is a point in the metric space. To serve the current request, the algorithm has to move one of the servers to the requested point, incurring a cost equal to the distance traveled by the server. In the online model, an algorithm has to serve the current request before the next request is revealed. A (randomized) online algorithm is said to be c-*competitive* against an adversary if it produces a solution, whose (expected) cost is at most c times the cost of the solution produced by the adversary.

A generalization of the k-server problem proposed by Fiat and Ricklin [2], called the weighted k-server problem, associates a weight with each server. The cost incurred in moving a server is equal to the product of its weight and the distance traveled. Introducing weights adds a new dimension to the k-server problem and presents new challenges. While a $(2k-1)$-competitive algorithm is known for the k-server problem [6], the only competitive algorithms known for the weighted k-server problem are for uniform spaces [2], and for $k = 2$ [7]. On uniform spaces, this problem models caching with different types of caches, each having a different write cost. Fiat and Ricklin [2] point out the practical significance of such caches, in order to optimize both the overall write time as well as the chip area occupied by the cache.

A randomized algorithm for the weighted k-server problem is said to be *memoryless*, if its behaviour on getting a request is completely determined by the pairwise distances between the k points occupied by its servers and the requested point. In other words, a memoryless algorithm for the weighted k-server problem with a given set of weights is specified by a function, which maps the $\binom{k+1}{2}$ distances to a probability distribution on the servers. In particular, on uniform metric spaces a memoryless algorithm is completely specified by a probability distribution p on the servers, where p_i is the probability by which the i^{th} server is shifted to the requested point, if that point is not already occupied by some server. The Harmonic algorithm is a memoryless algorithm, which moves the servers with probabilities inversely proportional to their weights.

For online problems modeling certain practical problems like caching, it is imperative that decisions are taken

*Supported by Microsoft Research India Travel Grant

instantaneously. Ideally, we would like the algorithm to be memoryless. For the k-server problem, the Harmonic algorithm is known to be $O(k2^k)$-competitive on any metric space [8], [9]. Additionally, Coppersmith et al. [10] proved that on *resistive* metric spaces there exists a k-competitive memoryless algorithm, in which the probabilities of moving the servers are determined by the *resistive inverse* of the metric space. It hence came as a surprise when Chrobak and Sgall [1] proved that no memoryless algorithm with a finite competitive ratio exists, even for the weighted 2-server problem on the line metric (which is resistive). Among other nice results in the same paper, Chrobak and Sgall [1] give the only known competitive memoryless algorithm for uniform spaces: a 5-competitive algorithm for 2 servers, and they also prove that this is optimally competitive. We generalize this result and prove the following theorem.

Theorem 1: For every k, there exists an α_k-competitive memoryless algorithm for the weighted k-server problem on uniform metric spaces against an online adaptive adversary, where $\alpha_1 = 1$ and $\alpha_k = \alpha_{k-1}^2 + 3\alpha_{k-1} + 1$ for $k > 1$.

In order to establish Theorem 1, we prove a more general result. Given server weights $\beta = (\beta_1, \ldots, \beta_k)$, and a probability distribution p on the servers used by an algorithm, we derive an upper bound $\tilde{\alpha}(\beta, p)$ on the competitive ratio, as a function of β and p. Given β, we use this result to identify a probability distribution p such that the competitive ratio is at most α_k. As a by-product of this more general result, we also see that the Harmonic algorithm is $(k\alpha_k)$-competitive for any β against an online adaptive adversary. For $k = 2$ we get $\alpha_2 = 5$ and our result matches that of Chrobak and Sgall [1].

The main difficulty in analyzing algorithms for this problem stems from the inability to describe suitable potential functions explicitly. Analogous to [11], we formulate a set of linear inequalities that the potentials must satisfy, where the co-efficients involved in the inequalities depend on the probabilities and the weights. We then show that the point, at which a certain subset of the linear inequalities is tight, is feasible. Our work indicates that the potentials given by this point are complicated rational functions of the probabilities and weights, and describing them seems hopeless, even for $k = 4$. Our key technical contribution is a framework to work with potential functions defined implicitly, as the solution of a linear system.

Theorem 1 also has the following consequence. Together with the derandomization result by Ben-David et al. [3], it implies the existence of a deterministic algorithm for the weighted k-server problem on uniform spaces, with competitive ratio α_k^2. It can be easily proved that $\alpha_k < 1.6^{2^k}$ and thus, we have an upper bound of 2.56^{2^k}, significantly better than the earlier bound on the deterministic competitive ratio by Fiat and Ricklin [2], which was more than 2^{4^k}. The best lower bound known is $(k+1)!/2$, also due to [2], and this can be improved to $(k+1)! - 1$, by a more careful argument.

We conjecture that the ratio α_k is the best possible for a memoryless algorithm against an online adaptive adversary. Towards proving this conjecture, we prove that the upper bound of $\tilde{\alpha}(\beta, p)$ is tight. Specifically, we prove that if the *separation* $\min_i \beta_{i+1}/\beta_i$ between the weights is sufficiently large, then there exists an online adaptive adversary which forces the algorithm using the probability distribution p, to perform almost $\tilde{\alpha}(\beta, p)$ times worse than itself. (Theorem 3 in Section IV.) It is interesting to note that we leverage the machinery to prove the upper bound, to prove this lower bound; we use the same potentials in a different avatar. We believe that with a suitable choice of weights β of arbitrarily large separation $\tilde{\alpha}(\beta, p)$ can be forced to be arbitrarily close to α_k.

II. Preliminaries and Techniques

Let $\beta = (\beta_1, \ldots, \beta_k)$ be the weights of the servers, in an instance of the weighted k-server problem. Consider a memoryless algorithm that, in response to a request on a point not already occupied by a server, moves the i^{th} server with probability p_i. We derive an upper bound on the competitive ratio as a function of β and $p = (p_1, \ldots, p_k)$. Note that whenever a point not occupied by the algorithm's servers is requested, the expected cost incurred by the algorithm is $\sum_{j=1}^{k} p_j \beta_j$.

A. Potential functions

In this paper we focus on competitive ratios against an online adaptive adversary [3], who observes the behaviour of the algorithm on the previous requests, generates the next request, and immediately serves it. The traditional method to analyze an online algorithm is to associate a *potential* with each *state*, determined by the positions of the adversary's and algorithm's servers, such that

1) When the adversary moves, the increase in the potential is at most α times the cost incurred by it.
2) When the algorithm moves, the decrease in the potential is at least as much as the cost incurred by the algorithm.

We think of each request being first served by the adversary and then by the algorithm. A standard telescoping argument implies that the competitive ratio is then bounded from above by α.

In our case, we define the states as follows. At any point of time, let a_i (resp. s_i) denote the position of the adversary's (resp. algorithm's) i^{th} server. We identify our state with the set $S = \{i \mid a_i = s_i\} \subseteq [k]$. We denote by ϕ_S the potential we associate with state S. We assume, without loss of generality, that the adversary never requests a point occupied by one of algorithm's servers, and that the adversary moves its servers only to serve requests. Suppose that at some point of time the state is S, and the adversary moves the i^{th} server, incurring a cost β_i. If $i \notin S$ the state does not change, while if $i \in S$ the state changes to $S \setminus \{i\}$. In order to prove α-competitiveness, it is sufficient to have potentials satisfying

$$\phi_{S\setminus\{i\}} - \phi_S \leq \beta_i \cdot \alpha \text{ for every } S \text{ and } i \in S \tag{1}$$

Now suppose that the state is S and it is the algorithm's turn to serve the request. The request must be a_i for some $i \notin S$. If the algorithm moves its i^{th} server, the new state is $S \cup \{i\}$. This happens with probability p_i, and the decrease in potential is $\phi_S - \phi_{S\cup\{i\}}$. Else, if the algorithm moves its j^{th} server for some $j \in S$, the new state is $S \setminus \{j\}$. This happens with probability p_j, and the decrease in potential is $\phi_S - \phi_{S\setminus\{j\}}$. Finally, if the algorithm moves its j^{th} server for some $j \notin S$ and $j \neq i$, there is no change in S, and hence the potential. We want the expected decrease in potential to be at least the expected cost incurred by the algorithm. Thus we need,

$$p_i(\phi_S - \phi_{S\cup\{i\}}) - \sum_{j\in S} p_j(\phi_{S\setminus\{j\}} - \phi_S) \geq \sum_{j=1}^{k} p_j\beta_j \tag{2}$$

for every S and $i \notin S$.

B. *A Linear Program and a choice of an Extreme Point*

Among the set of potentials ϕ_S for each $S \subseteq [k]$ satisfying (2), we wish to pick one to minimize α. The conditions (2) define a polyhedron in $\mathbb{R}^{2^k}$. Note that the right hand side of each constraint in (2) is constant. We assume that $\phi_\emptyset$, the potential of the empty set, is 0.

To simplify calculations and to facilitate an inductive approach, with foresight, we introduce a change of variables. Let us replace the variables $(\phi_S)_{S\subseteq[k]}$ by the variables $(\varphi_S)_{S\subseteq[k]}$ such that $\phi_S = -(\sum_{j=1}^{k} p_j\beta_j)\varphi_S$. With this substitution, we have the following optimization problem.

Minimize α subject to

For every S and $i \in S$:

$$\left(\sum_{j=1}^{k} p_j\beta_j\right) \frac{\varphi_S - \varphi_{S\setminus\{i\}}}{\beta_i} \leq \alpha \tag{3}$$

For every S and $i \notin S$

$$p_i\left(\varphi_{S\cup\{i\}} - \varphi_S\right) - \sum_{j\in S} p_j\left(\varphi_S - \varphi_{S\setminus\{j\}}\right) \geq 1 \tag{4}$$

$$\varphi_\emptyset = 0 \tag{5}$$

Note that the objective value of this problem does not change when all the p_j's are scaled by a positive constant, and the feasible space gets merely scaled by the inverse of that constant. Henceforward, for the convenience of discussion, we will relax the condition $\sum_{j=1}^{k} p_j = 1$. We will assume that $p_1, \ldots, p_k$ are some numbers chosen by the algorithm, as functions of $\beta_1, \ldots, \beta_k$, and that the algorithm moves the i^{th} server with probability $p_i/(\sum_{j=1}^{k} p_j)$.

Our task is to establish the existence of one feasible point of the linear program given by (3), (4), (5) with the required bound on the competitive ratio. Recall that the linear programming theory says that the optimum must occur at an extreme point. We guess a subset of 2^k linearly independent constraints among (4) which will be satisfied with equality. This forms the implicit description of the potentials. Assume without loss of generality, that $p_1 \geq \cdots \geq p_k$. Let $\varphi_S = \varphi_S(p)$ for all S, be the solution of the following linear system of equations.

$$p_i\left(\varphi_{S\cup\{i\}} - \varphi_S\right) - \sum_{j\in S} p_j\left(\varphi_S - \varphi_{S\setminus\{j\}}\right) = 1 \tag{6}$$

for every $S \neq [k]$ and i: smallest integer not in S, and

$$\varphi_\emptyset = 0 \tag{7}$$

and let $\alpha(p) = \left(\sum_{j=1}^{k} p_j\beta_j\right) \max_{S,i\in S} \frac{\varphi_S - \varphi_{S\setminus\{i\}}}{\beta_i}$. We need to prove that the point $(\varphi_S(p))_S$ is feasible, that is, it satisfies all the remaining constraints in (4). We will then prove an upper bound on the value of the objective function at this point.

C. *Checking Feasibility: The Gauss-Seidel Trick*

The straightforward way to check feasibility is to find explicitly what each $\varphi_S(\cdot)$ is, substitute, and check whether the constraints (4) are satisfied for every p. However, these functions tend to be more and more complicated as k grows, and it is hopeless to find the closed form expressions, even when $k = 4$. We therefore resort to the following indirect way.

Suppose we want to prove that the solution $x^* \in \mathbb{R}^n$ of the system $Ax = b$ satisfies $c^\top x^* \leq d$. The Gauss-Seidel iterative procedure in numerical computation to compute x^* is as follows. Write A as $L_* + U$ where L_* consists of the diagonal and the lower triangular part of A, and U consists of the upper triangular part. Choose an initial point x^0, and for i going from 1 to ∞ calculate $x^i = L_*^{-1}(b - Ux^{i-1})$. In other words, in every iteration the j^{th} coordinate is computed using the j^{th} equality in the system $Ax = b$, and the latest values of other coordinates. The coordinates are computed in a fixed order in all iterations. Under certain sufficiency conditions on A (which imply that L^* is invertible), the sequence (x^i) converges to x^*.

Our technique to prove $c^\top x^* \leq d$ is as follows. With suitable choice of x^0, we prove that $c^\top x^0 \leq d$, and that for all i, $c^\top x^{i-1} \leq d$ implies $c^\top x^i \leq d$. This proves that the entire sequence (x^i) satisfies the constraint $c^\top x \leq d$. Further, since the constraint defines a closed subset of $\mathbb{R}^n$, the limit point x^* also satisfies the constraint. We will call this trick the Gauss-Seidel trick of feasibility checking.

Two sufficient conditions for the Gauss-Seidel iterations to converge are that the system be strictly diagonally dominated, or irreducibly diagonally dominated.[1] In our case the

[1]http://en.wikipedia.org/wiki/Gauss-Seidel_method

system given by (6) and (7) is diagonally dominated, but it is neither strictly diagonally dominated, nor irreducibly diagonally dominated. Hence the Gauss-Seidel trick does not apply directly. To deal with this and other minor technical issues we will define φ_S inductively, using certain other functions f_S of the probabilities. For a fixed probability distribution, these functions f_S will be the solutions of a strictly diagonally dominated linear system. We will use the Gauss-Seidel trick to prove certain properties of these functions, which imply the conditions (4).

III. Proof Sketch for the Upper Bound

A. Defining the Potentials

We now formally define the function $\varphi_S : (\mathbb{R}_{>0})^t \longrightarrow \mathbb{R}$, for each finite subset $S \subseteq \mathbb{N}$, with arity t equal to the largest element present in S. Throughout this paper, if g is a function of arity t and x is a (possibly infinite) sequence of positive real numbers with more than t elements, we use $g(x)$ to denote g applied to the first t numbers in x, that is, $g(x_1, \ldots, x_t)$. For this section, p will be a sequence of positive real numbers. For a finite $S \subseteq \mathbb{N}$ and $i \in S$ we denote the quantity $p_i(\varphi_S(p) - \varphi_{S\setminus\{i\}}(p))$ by $\mathbf{I}_S^{S\setminus\{i\}}(p)$. Throughout this paper we will imagine that given a p, $\varphi_S(p)$ is the electric potential applied at S, S and $S \setminus \{i\}$ are connected by the conductance p_i, and therefore $\mathbf{I}_S^{S\setminus\{i\}}(p)$ is the current flowing from S to $S \setminus \{i\}$.[2] Note that the arity of the function $\mathbf{I}_S^{S\setminus\{i\}}$ is same as that of φ_S, that is, equal to the largest element in S.

We define φ_S by induction on the largest element in S. $\varphi_\emptyset(p) = 0$ for all p. Having defined φ_S with arity at most $k-1$ for each $S \subseteq [k-1]$, we define

$$\varphi_S(p) = \varphi_{S\setminus\{k\}}(p) + \frac{f_S(p)}{p_k} \tag{8}$$

for $S \subseteq [k]$ such that $k \in S$, where the functions f_S satisfy

$$\left(p_i + \sum_{j\in S} p_j\right) f_S(p) = p_i f_{S\cup\{i\}}(p) + \sum_{j\in S\setminus\{k\}} p_j f_{S\setminus\{j\}}(p) \tag{9}$$

for $S \neq [k]$ where $i < k$ is the smallest element not in S; and

$$f_{[k]}(p) = 1 + \sum_{j=1}^{k-1} \mathbf{I}_{[k-1]}^{[k-1]\setminus\{j\}}(p) \tag{10}$$

Note that the system given by the equations (9) and (10) is strictly diagonally dominant. This guarantees not only that the system has a unique solution, but also that the Gauss-Seidel procedure converges to that solution when started from any point.

By definition, $f_S(p)$ depends only on $p_1, \ldots, p_k$, and $f_{[k]}(p)$ depends only on $p_1, \ldots, p_{k-1}$. Hence the arity of φ_S, for each $S \subseteq [k]$ containing k, is k as promised. Note that $\mathbf{I}_S^{S\setminus\{k\}}(p) = f_S(p)$. Further, for $i \in S$, $i < k$ we have

$$\begin{aligned}
\mathbf{I}_S^{S\setminus\{i\}}(p) &= p_i\left(\varphi_S(p) - \varphi_{S\setminus\{i\}}(p)\right) \\
&= p_i\left\{\left(\varphi_{S\setminus\{k\}}(p) + \frac{f_S(p)}{p_k}\right)\right. \\
&\qquad \left. - \left(\varphi_{S\setminus\{i,k\}}(p) + \frac{f_{S\setminus\{i\}}(p)}{p_k}\right)\right\} \\
&= p_i\left(\varphi_{S\setminus\{k\}}(p) - \varphi_{S\setminus\{i,k\}}(p)\right) \\
&\quad + \frac{p_i}{p_k}\left(f_S(p) - f_{S\setminus\{i\}}(p)\right)
\end{aligned}$$

Thus,

$$\mathbf{I}_S^{S\setminus\{i\}}(p) = \mathbf{I}_{S\setminus\{k\}}^{S\setminus\{i,k\}}(p) + \frac{p_i}{p_k}\left(f_S(p) - f_{S\setminus\{i\}}(p)\right) \tag{11}$$

Note that for any constant $c > 0$, $\varphi_S(cp) = \varphi_S(p)/c$, whereas $\mathbf{I}_S^{S\setminus\{i\}}(p)$ remains invariant under scaling of p, for any S and $i \in S$.

We begin by proving that the potentials $\varphi_S(p)$ satisfy the relations given by (6).

Lemma 1: For any p, any finite $S \subseteq \mathbb{N}$ let i be the smallest element not in S. Then

$$p_i\left(\varphi_{S\cup\{i\}}(p) - \varphi_S(p)\right) = 1 + \sum_{j\in S} p_j\left(\varphi_S(p) - \varphi_{S\setminus\{j\}}(p)\right)$$

or equivalently

$$\mathbf{I}_{S\cup\{i\}}^S(p) = 1 + \sum_{j\in S} \mathbf{I}_S^{S\setminus\{j\}}(p)$$

Proof: If $S = [k-1]$ for some k, then $i = k$ and we have $\mathbf{I}_{S\cup\{i\}}^S(p) = f_{[k]}(p) = 1 + \sum_{j=1}^{k-1} \mathbf{I}_{[k-1]}^{[k-1]\setminus\{j\}}(p)$. Here the second equality is exactly equation (10). We now induct on the largest element k in S. The previous case covers the base case of $S = \emptyset$. Now for the inductive step we can assume $S \neq [k]$ and hence $i < k$. We have from equation (11)

$$\mathbf{I}_{S\cup\{i\}}^S(p) = \mathbf{I}_{S\cup\{i\}\setminus\{k\}}^{S\setminus\{k\}}(p) + \frac{p_i}{p_k}\left(f_{S\cup\{i\}}(p) - f_S(p)\right) \tag{12}$$

But i is also the smallest element not in $S \setminus \{k\}$, and by induction hypothesis we have

$$\mathbf{I}_{S\cup\{i\}\setminus\{k\}}^{S\setminus\{k\}}(p) = 1 + \sum_{j\in S\setminus\{k\}} \mathbf{I}_{S\setminus\{k\}}^{S\setminus\{j,k\}}(p)$$

Further, from equation (9) we have

$$\begin{aligned}
p_i\left(f_{S\cup\{i\}}(p) - f_S(p)\right) &= \sum_{j\in S\setminus\{k\}} p_j\left(f_S(p) - f_{S\setminus\{j\}}(p)\right) \\
&\quad + p_k f_S(p)
\end{aligned}$$

[2]Note that the potentials and currents satisfy the Kirchhoff's voltage law but not the current law.

Substituting in equation (12) and again using equation (11) we get

$$\begin{aligned}\mathbf{I}^S_{S\cup\{i\}}(p) &= 1+\sum_{j\in S\setminus\{k\}}\Bigg[\mathbf{I}^{S\setminus\{j,k\}}_{S\setminus\{k\}}(p)\\ &\quad +\frac{p_j}{p_k}(f_S(p)-f_{S\setminus\{j\}}(p))\Bigg]+f_S(p)\\ &= 1+\left[\sum_{j\in S\setminus\{k\}}\mathbf{I}^{S\setminus\{j\}}_S(p)\right]+\mathbf{I}^{S\setminus\{k\}}_S(p)\\ &= 1+\sum_{j\in S}\mathbf{I}^{S\setminus\{j\}}_S(p)\end{aligned}$$

■

B. Proving Feasibility

Towards proving Theorem 1, our first goal is to prove that the potentials satisfy the constraints given by (4). We prove this by first proving suitable inequalities involving f_S and then using induction and (11). The inequalities involving f_S that we need are given by the following claim.

Lemma 2: Suppose p is a non-increasing sequence. Then for any set $S\subseteq[k]$ such that $k\in S$ and for $i,i'\notin S$, $i<i'<k$, we have

$$p_i\left(f_{S\cup\{i\}}(p)-f_S(p)\right)\le p_{i'}\left(f_{S\cup\{i'\}}(p)-f_S(p)\right)$$

Here we use the Gauss-Seidel trick, where we prove that the claim is true after every iteration of the Gauss-Seidel procedure, when started from an appropriately chosen point. This claim enables us to prove the following current monotonicity property.

Lemma 3 (Monotonicity of currents): Suppose p is a non-increasing sequence. Then for any finite $S\subseteq\mathbb{N}$ and for $i,j\notin S$, $i\le j$, we have

$$\mathbf{I}^S_{S\cup\{i\}}(p)\le\mathbf{I}^S_{S\cup\{j\}}(p)$$

We defer the proofs of the above two claims to the full version [12]. The following feasibility lemma, which states that the constraints (4) are satisfied, is immediate from Lemmas 1 and 3.

Lemma 4 (Feasibility): Suppose p is a non-increasing sequence. Then for any finite $S\subseteq\mathbb{N}$ and $i\notin S$

$$p_i\left(\varphi_{S\cup\{i\}}(p)-\varphi_S(p)\right)\ge 1+\sum_{j\in S}p_j\left(\varphi_S(p)-\varphi_{S\setminus\{j\}}(p)\right)$$

or equivalently

$$\mathbf{I}^S_{S\cup\{i\}}(p)\ge 1+\sum_{j\in S}\mathbf{I}^{S\setminus\{j\}}_S(p)$$

with equality if i is the smallest element not in S.

C. Bounding the Objective Function

We need to bound the quantity $\left(\sum_{j=1}^k p_j\beta_j\right)\max_{S,i\in S}\frac{\varphi_S(p)-\varphi_{S\setminus\{i\}}(p)}{\beta_i}$ from above. Towards this end, the key property we need is that as a set-function $\varphi_S(p)$ is supermodular.

Lemma 5 (Supermodularity): For every non-increasing p, finite S and $i,j\notin S$, we have

$$\varphi_{S\cup\{i\}}(p)+\varphi_{S\cup\{j\}}(p)\le\varphi_{S\cup\{i,j\}}(p)+\varphi_S(p)$$

Thus, for any fixed p, the function mapping a set S to $\varphi_S(p)$ is supermodular, and we have

$$\begin{aligned}\mathbf{I}^{S'}_{S'\cup\{i\}}(p) &= p_i\left(\varphi_{S'\cup\{i\}}(p)-\varphi_{S'}(p)\right)\\ &\le\\ p_i\left(\varphi_{S\cup\{i\}}(p)-\varphi_S(p)\right) &= \mathbf{I}^S_{S\cup\{i\}}(p)\end{aligned}$$

whenever $S'\subseteq S$ and $i\notin S$.

We defer the proof to the full version [12]. This lemma enables us to prove that each current is bounded from above by a constant independent of p. These constants are defined as follows.

Definition 1: For each finite $S\subseteq\mathbb{N}$ the constant C_S is defined as follows. $C_\emptyset=1$ and if $S\neq\emptyset$ then $C_S=1+\sum_{j\in S}C_{S\setminus\{j\}\cup[j-1]}$. For each $k\in\mathbb{N}$, α_k is defined as $\alpha_k=\alpha_{k-1}^2+3\alpha_{k-1}+1$; $\alpha_0=0$.

Consider the following order on finite subsets of $\mathbb{N}$, which we call the co-lex order. We say that a set S precedes T in this order if there exists $i\in T\setminus S$ such that S and T agree on membership of integers greater than i. Note that the constants C_S have been defined by induction on S according to this order. We will also use this order to induct on the finite subsets of $\mathbb{N}$, in the subsequent proofs.

The bounds on the currents are given by the following lemma.

Lemma 6 (Boundedness of currents): For every finite set $S\subseteq\mathbb{N}$ and for all non-increasing p, $\mathbf{I}^S_{S\cup\{i\}}(p)\le C_S$, where i is the smallest element not in S.

Proof: We prove by induction on the position of S in the co-lex order. For the base case, when $S=\emptyset$ and $i=1$, we have $\mathbf{I}^{\emptyset}_{\{1\}}(p)=f_{\{1\}}(p)=1=C_\emptyset$ for all p, by equation (10).

For the inductive case, assume that the claim holds for all finite subsets of $\mathbb{N}$ which precede a set S in the co-lex order. Let i be the smallest element not in S. Then by Lemma 4 we have

$$\begin{aligned}\mathbf{I}^S_{S\cup\{i\}}(p) &= 1+\sum_{j\in S}\mathbf{I}^{S\setminus\{j\}}_S(p)\\ &\le 1+\sum_{j\in S}\mathbf{I}^{S\setminus\{j\}\cup[j-1]}_{S\cup[j-1]}(p)\\ &\le 1+\sum_{j\in S}C_{S\setminus\{j\}\cup[j-1]}\\ &= C_S\end{aligned}$$

where the first inequality is due to supermodularity (Lemma 5), and the second is by induction hypothesis, since the smallest element not in $S \setminus \{j\} \cup [j-1]$ is j. Note that $S \setminus \{j\} \cup [j-1]$ precedes S in the co-lex order, for all $j \in S$. ■

Our final ingredient in the proof of Theorem 1 is the following lemma relating the quantities from Definition 1.

Lemma 7: The following hold for every $k \in \mathbb{N}$.

1) If k is the largest element in $S \subseteq \mathbb{N}$, then

$$C_S = (\alpha_{k-1} + 2) C_{S\setminus\{k\}}$$

2) $\alpha_k = \sum_{j=1}^{k} C_{[k]\setminus\{j\}} = C_{[k]} - 1$.
3) $C_{[k]\setminus\{1\}} > C_{[k]\setminus\{2\}} > \cdots > C_{[k]\setminus\{k-1\}} > C_{[k]\setminus\{k\}}$.

Proof: We prove the claims by induction on k, and then (for part (1)) induction on the position of S in the co-lex order. Parts (2) and (3) are easily verified for $k = 1$, and so is part (1) for $S = \{1\}$.

Let $k > 1$. The first set in the co-lex order having k as the largest element in $\{k\}$. For $S = \{k\}$ we have from Definition 1 and part (2) of induction hypothesis

$$C_{\{k\}} = 1 + C_{[k-1]} = 2 + \alpha_{k-1} = (\alpha_{k-1} + 2) C_{\emptyset}$$

For an arbitrary S having k as the largest element, we have from Definition 1 and part (2) of induction hypothesis

$$\begin{aligned} C_S &= 1 + \sum_{j\in S} C_{S\setminus\{j\}\cup[j-1]} \\ &= 1 + C_{S\setminus\{k\}\cup[k-1]} + \sum_{j\in S\setminus\{k\}} C_{S\setminus\{j\}\cup[j-1]} \end{aligned}$$

By part (2) of induction hypothesis

$$C_{S\setminus\{k\}\cup[k-1]} = C_{[k-1]} = \alpha_{k-1} + 1$$

Furthermore, since $S \setminus \{j\} \cup [j-1]$ precedes S in the co-lex order, we have

$$C_{S\setminus\{j\}\cup[j-1]} = (\alpha_{k-1} + 2) C_{S\setminus\{j,k\}\cup[j-1]}$$

Thus,

$$\begin{aligned} C_S &= 2 + \alpha_{k-1} + (\alpha_{k-1} + 2) \sum_{j\in S\setminus\{k\}} C_{S\setminus\{j,k\}\cup[j-1]} \\ &= (\alpha_{k-1} + 2) \left(1 + \sum_{j\in S\setminus\{k\}} C_{S\setminus\{j,k\}\cup[j-1]} \right) \\ &= (\alpha_{k-1} + 2) C_{S\setminus[k]} \end{aligned}$$

This proves part (1). In particular, we have

$$C_{[k]} = (\alpha_{k-1} + 2) C_{[k-1]}$$

Again by part (2) of induction hypothesis and Definition 1

$$\begin{aligned} C_{[k]} &= (\alpha_{k-1} + 2)(\alpha_{k-1} + 1) \\ &= \alpha_{k-1}^2 + 3\alpha_{k-1} + 2 \\ &= \alpha_k + 1 \end{aligned}$$

This proves part (2). Finally, for any $i < j < k$ we have

$$\begin{aligned} C_{[k]\setminus\{i\}} &= (\alpha_{k-1} + 2) C_{[k-1]\setminus\{i\}} \\ &> (\alpha_{k-1} + 2) C_{[k-1]\setminus\{j\}} \\ &= C_{[k]\setminus\{j\}} \end{aligned}$$

where the inequality is by part (3) of induction hypothesis. Further,

$$\begin{aligned} C_{[k]\setminus\{j\}} &= (\alpha_{k-1} + 2) C_{[k-1]\setminus\{j\}} \\ &> \alpha_{k-1} + 1 \\ &= C_{[k-1]} \\ &= C_{[k]\setminus\{k\}} \end{aligned}$$

This proves (3). ■

D. Proof of Theorem 1

We now show how the above lemmas imply Theorem 1. First, we prove an upper bound on the competitive ratio achieved by the probability distribution $p = (p_1, \ldots, p_k)$ when the weights are $\beta_1, \ldots, \beta_k$.

Theorem 2: Consider an instance of the weighted k-server problem with weights $\beta_1, \ldots, \beta_k$, and a randomized memoryless algorithm which moves the i^{th} server with a probability p_i, where $p_1 \geq \cdots \geq p_k$. Then the competitive ratio of this algorithm against an adaptive online adversary is at most $\tilde{\alpha}(\beta, p)$, where

$$\tilde{\alpha}(\beta, p) = \left(\sum_{j=1}^{k} p_j \beta_j \right) \max_{i\in[k]} \frac{\mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p)}{p_i \beta_i}$$

Proof: Lemma 4 assures that the constraints (4) hold. To satisfy the set of constraints given by (3), we choose

$$\alpha = \left(\sum_{j=1}^{k} p_j \beta_j \right) \max_{S\subseteq[k], i\in S} \frac{\varphi_S(p) - \varphi_{S\setminus\{i\}}(p)}{\beta_i}$$

Due to the supermodularity property from Lemma 5, it is sufficient to take the maximum with $S = [k]$ and over all i. Thus we have

$$\begin{aligned} \alpha &= \left(\sum_{j=1}^{k} p_j \beta_j \right) \max_{i\in[k]} \frac{\varphi_{[k]}(p) - \varphi_{[k]\setminus\{i\}}(p)}{\beta_i} \\ &= \left(\sum_{j=1}^{k} p_j \beta_j \right) \max_{i\in[k]} \frac{\mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p)}{p_i \beta_i} \end{aligned}$$

■

With Theorem 2 in place we are ready to prove Theorem 1.

Proof of Theorem 1: Let $\beta_1, \ldots, \beta_k$ be the weights of the servers, and assume $\beta_1 \leq \cdots \leq \beta_k$ without loss of generality. The required memoryless algorithm behaves as follows. Let $p_i = C_{[k]\setminus\{i\}}/\beta_i$ for all i. On receiving a request which is not covered by any server, the algorithm serves it with the i^{th} server, with probability p_i/P, where $P = \sum_{j=1}^{k} p_j$. By part (3) of Lemma 7 and our assumption: $\beta_1 \leq \cdots \leq \beta_k$, we have $p_1 \geq \ldots \geq p_k$. Thus we can apply Theorem 2 and hence, the competitive ratio of our algorithm is at most

$$\begin{aligned}\tilde{\alpha}(\beta, p) &= \left(\sum_{j=1}^{k} p_j\beta_j\right) \max_{i\in[k]} \frac{\mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p)}{p_i\beta_i} \\ &= \left(\sum_{j=1}^{k} C_{[k]\setminus\{j\}}\right) \max_{i\in[k]} \frac{\mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p)}{C_{[k]\setminus\{i\}}} \\ &\leq \alpha_k\end{aligned}$$

where the last inequality follows from part (2) of Lemma 7, and Lemma 6. Note that since the currents are invariant under scaling of p, so is $\tilde{\alpha}(\beta, p)$, and hence P can be ignored. ■

Corollary 1 (to Theorem 2): The Harmonic algorithm for the weighted k-server problem on uniform spaces has a competitive ratio of $k\alpha_k$ against an online adaptive adversary.

Proof: The probabilities for the Harmonic algorithm are given by $p_i = (1/\beta_i)/\sum_{j=1}^{k}(1/\beta_j)$ and therefore $1/(p_i\beta_i) = \sum_{j=1}^{k}(1/\beta_j)$ for all i, and $\sum_{j=1}^{k} p_j\beta_j = k/\sum_{j=1}^{k}(1/\beta_j)$. By Theorem 2, the competitive ratio is given by

$$\begin{aligned}\alpha &= \left(\sum_{j=1}^{k} p_j\beta_j\right) \max_{i\in[k]} \frac{\mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p)}{p_i\beta_i} = k \cdot \max_{i\in[k]} \mathbf{I}_{[k]}^{[k]\setminus\{i\}}(p) \\ &\leq k \cdot \max_{i\in[k]} C_{[k]\setminus\{i\}} \leq k \cdot \sum_{i=1}^{k} C_{[k]\setminus\{i\}} = k\alpha_k\end{aligned}$$

■

IV. A Lower Bound Result

We believe that it is not possible to improve the upper bound of α_k on the competitive ratio of for randomized memoryless algorithms for the weighted k-server problem on uniform spaces. This conjecture is formally stated as follows.

Conjecture 1: For any $\varepsilon > 0$ there exist weights $\beta_1, \ldots, \beta_k$ and an adversarial strategy which forces any randomized memoryless algorithm to perform at least $(1 - \varepsilon)\alpha_k$ times worse than the adversary.

As a first step towards proving Conjecture 1, we prove that Theorem 2 is essentially tight. For an algorithm which uses probabilities p_i where $p = (p_1, \ldots, p_k)$ when the weights are $\beta = (\beta_1, \ldots, \beta_k)$, we prove a lower bound on the competitive ratio, which goes arbitrarily close to $\tilde{\alpha}(\beta, p)$ as the separation between the weights grows unbounded.

Theorem 3: Let $\beta_1 \leq \cdots \leq \beta_k$ be the weights of the servers in an instance of the weighted k-server problem on uniform metric spaces. Consider a randomized memoryless algorithm which moves the i^{th} server with probability p_i whenever there is no server on the requested point. Then the competitive ratio of this algorithm is at least $\tilde{\alpha}(\beta, p)/(1 + s\tilde{\alpha}(\beta, p))$, where $s = \max_{1\leq i<k} \beta_i/\beta_{i+1}$.

Proof: Consider the uniform metric space with $2k$ points. We may assume that at any point of time, the algorithm's servers occupy k distinct points. We will also ensure that the adversary's servers occupy k distinct points.

As before, let at any point of time, let a_i (resp. s_i) denote the position of the adversary's (resp. algorithm's) i^{th} server. The adversary always ensures that $a_i \neq s_j$ for any $i < j$. In order to maintain this invariant, whenever the j^{th} server of the algorithm is shifted to a_i for $j > i$, the adversary shifts its i^{th} moves to a point not occupied by any other (adversary's as well as algorithm's) server.

The adversary generates the request sequence as follows. If there exists an i such that $a_i \neq s_i$, the adversary finds the smallest such i and requests a_i. By the choice of i, for all $j < i$ we have $a_j = s_j$ and hence $a_i \neq s_j$. Further the invariant ensures that $a_i \neq s_j$ for any $j > i$. We already have $a_i \neq s_i$. Thus, a_i is not occupied by any of the algorithm's servers.

On the other hand, if $a_i = s_i$ for all i, the adversary behaves as follows. Suppose $i = t$ achieves the maximum in the expression for $\tilde{\alpha}(\beta, p)$. That is,

$$\tilde{\alpha}(\beta, p) = \left(\sum_{j=1}^{k} p_j\beta_j\right) \cdot \frac{\mathbf{I}_{[k]}^{[k]\setminus\{t\}}(p)}{p_t\beta_t}$$

The adversary requests a point not occupied by any server, and shifts its t^{th} server to that point.

The adversary pays only in the following two cases - when it shifts its i^{th} server out of a point because the j^{th} server of the algorithm arrives at the same point, for some $j > i$, and when it shifts its t^{th} server because $a_i = s_i$ for all i. We will denote the total cost incurred in the former case by ADV' and the total cost in the latter case by ADV. Suppose the total cost incurred by the algorithm is ALG. In the former case, the algorithm incurs a cost of β_j just before the adversary pays β_i, where $\beta_j \geq \beta_i/s$. Thus $ALG \geq ADV'/s$. For the latter case, we prove $ALG \geq \tilde{\alpha}(\beta, p)ADV$ (up to an additive constant).

As before, at any point of time let $S = \{i \mid a_i = s_i\} \subseteq [k]$, and S will denote the state of the system. We will again assign a potential to each state, but this time we will ensure the following.

1) When the adversary moves its t^{th} server, the increase in potential is **at least** $\tilde{\alpha}(\beta, p) \cdot \beta_t$.
2) When the algorithm is moves a server, the expected decrease in potential is **at most** the expected cost incurred by the algorithm.

Note that when the adversary moves its servers to ensure $a_i \neq s_j$ for all $i < j$, the state remains the same. Thus, the above two statements imply $ALG \geq \tilde{\alpha}(\beta,p)ADV$ (up to an additive constant). Interestingly, the potentials that we assign to the states here are same as those that we assigned in the proof of the upper bound. That is, $\phi_S = -(\sum_{j=1}^{k} p_j\beta_j)\varphi_S(p)$. Note however, that we have not made any assumption about whether p is a non-decreasing sequence.

Consider the situation when the adversary incurs a cost of β_t, due to shifting of the t^{th} server because $a_i = s_i$ for all i, that is, the state is $[k]$. The state after the move is $[k] \setminus \{t\}$ and the change in potential is

$$\begin{aligned}
\phi_{[k]\setminus\{t\}} - \phi_{[k]} &= \left(\sum_{j=1}^{k} p_j\beta_j\right)\left(\varphi_{[k]}(p) - \varphi_{[k]\setminus\{t\}}(p)\right) \\
&= \left(\sum_{j=1}^{k} p_j\beta_j\right) \cdot \frac{\mathbf{I}_{[k]}^{[k]\setminus\{t\}}(p)}{p_t} \\
&= \tilde{\alpha}(\beta,p)\cdot\beta_t
\end{aligned}$$

Now consider the algorithm's move, in response to a request given while the system is in state $S \subsetneq [k]$. Let i be the smallest element not in S. As discussed before, the next request is a_i, and this point is not occupied by any of the algorithm's servers. Hence the algorithm must incur a cost $\sum_{j=1}^{k} p_j\beta_j$ in expectation. The expected change in potential will be

$$\begin{aligned}
& p_i\left(\phi_{S\cup\{i\}} - \phi_S\right) + \sum_{j\in S} p_j\left(\phi_{S\setminus\{j\}} - \phi_S\right) \\
=\; & -\left(\sum_{j=1}^{k} p_j\beta_j\right)\Bigg\{ p_i\left(\varphi_{S\cup\{i\}}(p) - \varphi_S(p)\right) \\
& -\sum_{j\in S} p_j\left(\varphi_S(p) - \varphi_{S\setminus\{j\}}(p)\right)\Bigg\} \\
=\; & -\sum_{j=1}^{k} p_j\beta_j
\end{aligned}$$

where the last equality is by Lemma 1. Thus we have proved $ALG \geq \tilde{\alpha}(\beta,p)ADV$ (up to an additive constant). Therefore, the competitive ratio is at least

$$\begin{aligned}
\frac{ALG}{ADV + ADV'} &= \left(\frac{ADV}{ALG} + \frac{ADV'}{ALG}\right)^{-1} \\
&\geq \left(\frac{1}{\tilde{\alpha}(\beta,p)} + s\right)^{-1} \\
&= \frac{\tilde{\alpha}(\beta,p)}{1 + s\tilde{\alpha}(\beta,p)}
\end{aligned}$$

■

Now proving Conjecture 1 reduces to proving that $\frac{\tilde{\alpha}(\beta,p)}{1+s\tilde{\alpha}(\beta,p)}$ can be forced to be arbitrarily close to α_k. Formally this is stated as follows.

Conjecture 2: For any k and any $\varepsilon > 0$ there exists $\beta = (\beta_1,\ldots,\beta_k)$ with $\beta_1 \leq \cdots \leq \beta_k$ such that for all $p = (p_1,\ldots,p_k)$ we have

$$\frac{\tilde{\alpha}(\beta,p)}{1 + s\tilde{\alpha}(\beta,p)} \geq (1-\varepsilon)\alpha_k$$

where $s = \max_{1\leq i<k} \beta_i/\beta_{i+1}$.

V. Concluding Remarks

We have proved that there exists a competitive memoryless algorithm for the weighted k-server problem on uniform metric spaces. This is in contrast with the line metric, which does not admit a competitive memoryless algorithm, even with 2 servers. The competitive ratio α_k that we establish, is given by $\alpha_k = \alpha_{k-1}^2 + 3\alpha_{k-1} + 1$. We can bound α_k as follows.

$$\alpha_k + 2 = (\alpha_{k-1}+2)^2 - \alpha_{k-1} - 1 < (\alpha_{k-1}+2)^2$$

Therefore

$$\alpha_k + 2 < (\alpha_t + 2)^{2^{k-t}} = \left[(\alpha_t+2)^{2^{-t}}\right]^{2^k}$$

for any $t < k$. For $t = 4$ one can verify that $(\alpha_t+2)^{2^{-t}} < 1.6$, and hence $\alpha_k < 1.6^{2^k}$, as promised in the introduction.

We believe that α_k is the best possible competitive ratio achievable by memoryless algorithms for the weighted k-server problem on uniform metric spaces. This is implied by Conjecture 2. Furthermore, we can show that Conjecture 2 is true if Lemma 6 is tight for certain currents. Proving this seems to require a deeper understanding of the behaviour of currents as functions of probability distribution.

The immediate increment to our results would perhaps be to determine whether there exists a competitive memoryless algorithm for the weighted server problem on star metrics. This problem translates to having a weight β_i for the i^{th} cache location and a cost c_t with each page t; the overall cost of replacing page t by page t' at the i^{th} cache location being $\beta_i(c_t + c_{t'})$. We believe that it should be possible to prove an unbounded lower bound for this problem, on the lines of the lower bound proof by [1] for 2 servers on the line metric.

We improve the upper bound on the deterministic competitive ratio by [2] for the weighted server problem on uniform metrics. However, our bound is still doubly exponential, whereas the lower bound is only exponential in the number of servers. It would be interesting to reduce this gap. The prime candidate for improving the upper bound is perhaps the (generalized) work function algorithm, which has been proved to be optimally competitive for the weighted 2-server problem on uniform metrics [1], and which is the best known algorithm for several other problems [6], [13].

The introduction of different costs for replacements at different cache locations seems to make the caching problem notoriously hard. This is certified by the fact that attempts to develop algorithms, better than the one by Fiat and Ricklin

[2], have given negligible success even with $k = 3$. For $k = 2$, Sitters [7] has shown that the generalized work function algorithm is competitive for the generalized server problem on arbitrary metrics, which subsumes the weighted 2-server problem. He has also expressed a possibility of the non-existence of a competitive algorithm, for $k > 2$. All this is in a striking contrast with the problem of weighted caching, where the pages (points) have costs instead of cache locations (servers). For the weighted caching problem, k-competitive deterministic and $O(\log k)$-competitive randomized algorithms have been discovered [14], [15], [16], [17], [18], [19], even when the pages have different sizes, matching the respective lower bounds.

Acknowledgment

The authors would like to thank Nikhil Bansal for pointing them to some important work on the weighted k-server problem.

References

[1] M. Chrobak and J. Sgall, "The weighted 2-server problem," *Theoretical Computer Science*, vol. 324, no. 2-3, pp. 289–312, 2004.

[2] A. Fiat and M. Ricklin, "Competitive algorithms for the weighted server problem," *Theoretical Computer Science*, vol. 130, no. 1, pp. 85–99, 1994.

[3] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos, and A. Wigderson, "On the power of randomization in on-line algorithms," *Algorithmica*, vol. 11, no. 1, pp. 2–14, 1994.

[4] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, "Competitive algorithms for on-line problems," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing*. ACM, 1988, pp. 322–333.

[5] E. Koutsoupias, "The k-server problem," *Computer Science Review*, vol. 3, no. 2, pp. 105–118, 2009.

[6] E. Koutsoupias and C. H. Papadimitriou, "On the k-server conjecture," *Journal of the ACM*, vol. 42, no. 5, pp. 971–983, 1995.

[7] R. Sitters, "The generalized work function algorithm is competitive for the generalized 2-server problem," *CoRR*, vol. abs/1110.6600, 2011.

[8] E. F. Grove, "The harmonic online k-server algorithm is competitive," in *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*. ACM, 1991, pp. 260–266.

[9] Y. Bartal and E. Grove, "The harmonic k-server algorithm is competitive," *Journal of the ACM*, vol. 47, no. 1, pp. 1–15, 2000.

[10] D. Coppersmith, P. Doyle, P. Raghavan, and M. Snir, "Random walks on weighted graphs and applications to online algorithms," *J. ACM*, vol. 40, no. 3, pp. 421–453, 1993.

[11] Y. Bartal, M. Chrobak, and L. L. Larmore, "A randomized algorithm for two servers on the line (extended abstract)," in *ESA*, ser. Lecture Notes in Computer Science, vol. 1461. Springer, 1998, pp. 247–258.

[12] A. Chiplunkar and S. Vishwanathan, "On the competitiveness of randomized memoryless algorithms for the weighted k-server problem," *CoRR*, vol. abs/1301.0123, 2013.

[13] W. R. Burley, "Traversing layered graphs using the work function algorithm," *Journal of Algorithms*, vol. 20, no. 3, pp. 479–511, 1996.

[14] M. Chrobak, H. J. Karloff, T. H. Payne, and S. Vishwanathan, "New results on server problems," in *SODA*, 1990, pp. 291–300.

[15] M. S. Manasse, L. A. McGeoch, and D. D. Sleator, "Competitive algorithms for server problems," *J. Algorithms*, vol. 11, no. 2, pp. 208–230, 1990.

[16] N. E. Young, "The k-server dual and loose competitiveness for paging," *Algorithmica*, vol. 11, no. 6, pp. 525–541, 1994.

[17] ——, "On-line file caching," in *SODA*, 1998, pp. 82–86.

[18] N. Bansal, N. Buchbinder, and J. Naor, "A primal-dual randomized algorithm for weighted paging," *J. ACM*, vol. 59, no. 4, p. 19, 2012.

[19] A. Adamaszek, A. Czumaj, M. Englert, and H. Räcke, "An $O(\log k)$-competitive algorithm for generalized caching," in *SODA*. SIAM, 2012, pp. 1681–1689.