A Polylogarithmic-Competitive Algorithm for the k-Server Problem (Extended Abstract)

Nikhil Bansal
IBM T. J. Watson
NY 10598 USA
bansal@gmail.com

Niv Buchbinder*
Comp. Science Dept.
Open Univ. of Israel
niv.buchbinder@gmail.com

Aleksander Mądry+
Microsoft Research
Cambridge, MA, USA
madry@mit.edu

Joseph (Seffi) Naor*
Comp. Science Dept.
Technion, Israel
naor@cs.technion.ac.il

Abstract— We give the first polylogarithmic-competitive randomized algorithm for the k-server problem on an arbitrary finite metric space. In particular, our algorithm achieves a competitive ratio of $\widetilde{O}(\log^3 n \log^2 k)$ for any metric space on n points. This improves upon the $(2k-1)$-competitive algorithm of Koutsoupias and Papadimitriou [22] whenever n is sub-exponential in k.

Keywords-k-server problem; randomized algorithms; competitive analysis;

1. Introduction

The k-server problem is one of the most fundamental and extensively studied problems in online computation. Suppose there is an n-point metric space and k servers are located at some of the points of the metric space. At each time step, an online algorithm is given a request at one of the points of the metric space, and this request is served by moving a server to the requested point (if there is no server there already). The cost of serving a request is defined to be the distance traveled by the server. Given a sequence of requests, the task is to devise an online strategy minimizing the sum of the costs of serving the requests.

The k-server problem was originally proposed by Manasse et al. [23] as a broad generalization of various online problems. The most well studied problem among them is the paging (also known as caching) problem, in which there is a cache that can hold up to k pages out of a universe of n pages. At each time step a page is requested; if the page is already in the cache then no cost is incurred, otherwise it must be brought into the cache (possibly causing an eviction of some other page) at a cost of one unit. It is easy to see that the paging problem is equivalent to the k-server problem on a uniform metric space, and already in their seminal paper on competitive analysis, Sleator and Tarjan [26] gave several k-competitive algorithms for paging, and showed that no deterministic algorithm can do better. This prompted Manasse et al. [23] to state a far-reaching conjecture that a similar result holds for an arbitrary metric. More precisely, they conjectured that there is a k-competitive online algorithm for the k-server problem on any metric space and for any value of k. This conjecture is known as as the *k-server conjecture*.

At the time that the k-server conjecture was stated, an online algorithm with competitive ratio that depends only on k was not known. It was first obtained by Fiat et al. [18]. Improved bounds were obtained later on by [21], [9], though the ratio still remained exponential in k. A major breakthrough was achieved by Koutsoupias and Papadimitriou [22], who showed that so-called *work function algorithm* is $(2k-1)$-competitive. This result is almost optimal, since we know that any deterministic algorithm has to be at least k-competitive. We note that a tight competitive factor of k is only known for special metrics such as the uniform metric, line metric, and – more generally – trees [13], [14].

Even though the aforementioned results are all deterministic, there is also a great deal of interest in randomized algorithms for k-server. This is motivated primarily by the fact that randomized online algorithms (i.e., algorithms working against an oblivious adversary) tend to have much better performance than their deterministic counterparts. For example, for the paging problem, several $O(\log k)$-competitive algorithms are known [19], [24], [1], [2], as well as a lower bound of $\Omega(\log k)$ on the competitive ratio.

Unfortunately, our understanding of the k-server problem when randomization is allowed is much weaker than in the deterministic case. Despite much work [11], [8], [10], no better lower bound than $\Omega(\log k)$ is known on competitive factors in the randomized setting. Conversely, no better upper bound, other than the *deterministic* guarantee of $2k-1$ [22] mentioned above, is known for general metrics. Thus, an exponential gap still remains between the known lower and upper bounds.

Given the lack of any lower bounds better than $\Omega(\log k)$, it is widely believed that there is an $O(\log k)$-competitive randomized algorithm for the k-server problem on every metric space against an oblivious adversary - this belief is captured by the so-called *randomized k-server conjecture*. Unfortunately, besides the previously-mentioned $O(\log k)$-competitive algorithm for the case of a uniform metric, even when we allow the competitiveness to depend on other

* Supported by ISF grant 954/11 and BSF grant 2010426.
+ Research done while at the Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA, USA, and partially supported by NSF grant CCF-0829878 and by ONR grant N00014-11-1-0053.

parameters of the metric, such as the number of points n, or the diameter Δ, non-trivial guarantees are known only for very few special cases. For example, the case of a well-separated metric [25], the case of a metric corresponding to a binary HST with high separation [15], the case of $n = k + O(1)$ [7], as well as some other cases [16], [3], [4]. For the weighted paging problem[1], [2] gave an $O(\log k)$-competitive algorithm (see also [3]) which is based on the online primal-dual approach. However, no non-trivial guarantees are known even for very simple extensions of the uniform metric, e.g., two-level HSTs with high separation.

For a more in-depth treatment of the extensive literature on both paging and the k-server problem, we suggest [12].

1.1. Our Result

We give the first polylogarithmic competitive algorithm for the k-server problem on a general metric with a finite number of points n. More precisely, our main result is the following.

Theorem 1. *There is a randomized algorithm for the k-server problem that achieves a competitive ratio of $O(\log^2 k \log^3 n \log\log n) = \widetilde{O}(\log^2 k \log^3 n)$ on any metric space on n points.*

The starting point of our algorithm is the recent approach proposed by Coté et al. [15] for solving the k-server problem on hierarchically well-separated trees (HSTs). It is well known that solving the problem on HSTs suffices, as any metric space can be embedded into a probability distribution over HSTs with low distortion [17].

More precisely, Coté et al. defined a problem on uniform metrics which we call the *allocation problem*. They showed that an online randomized algorithm for the allocation problem that provides certain refined competitive guarantees can be used as a building block to recursively solve the k-server problem on an HST, provided the HST is sufficiently well-separated. Roughly speaking, in their construction, each internal node of the HST runs an instance of the allocation problem that determines how to distribute the available servers among its children nodes. Starting from the root, which has k servers, the recursive calls to the allocation problem determine the number of servers at each leaf of the HST, giving a valid k-server solution. The guarantee of this k-server solution depends on both the guarantees for the allocation problem, as well as the depth of the HST (i.e., the number of levels of recursion). The guarantees obtained by Coté et al. [15] for the allocation problem on a metric space with two points allowed them to obtain an algorithm for the k-server problem on a sufficiently well-separated *binary* HST having a competitive ratio that is polylogarithmic in k, n, and the diameter Δ of the underlying metric space. Unfortunately, the fact that the HST has to be binary as well as have a sufficiently good separation severely restricts the metric spaces to which this algorithm can be applied.

Given the result of Coté et al. [15], a natural approach to establishing our result is coming up with a randomized algorithm having the required refined guarantees for the allocation problem on an arbitrary number of points. However, it is unclear to us how to obtain such an algorithm. Instead, we pursue a more refined approach to solving the k-server problem via the allocation problem. By doing so we are able to bypass the need for a "true" randomized algorithm for the allocation problem and are able to work with a (much) weaker formulation. More precisely, our result consists of three main parts.

(1): We show that instead of obtaining a randomized algorithm for the allocation problem, it suffices to obtain an algorithm for a certain *fractional* relaxation of it. By employing this relaxation, we make the task of designing such a fractional allocation algorithm easier than designing the version of the allocation problem that was considered earlier. Next, building upon the arguments in Coté et al. [15], we show that a sufficiently good online algorithm for this fractional allocation problem can be used as a building block to obtain a good *fractional* solution to the k-server problem on an HST. Finally, by proving that such a fractional k-server solution can be rounded in an online randomized manner, while losing only an $O(1)$ factor in the competitive ratio, we get a reduction of the k-server problem to our version of the fractional allocation problem.

An interesting feature of this reduction is that our fractional relaxation is too weak to give anything better than an $O(k)$ guarantee for the (integral) allocation problem, since there are instances on which any integral solution must pay $\Omega(k)$ times the fractional cost. Therefore, it is somewhat surprising that even though our relaxation is unable to provide any reasonable algorithm for the (integral) allocation problem, it suffices to give a good guarantee for the (integral) k-server problem.

(2): As the next step, we design an online algorithm for the fractional allocation problem with the refined guarantees required in the above reduction. Our techniques here are inspired by the ideas developed recently in the context of the caching with costs problem [3] and weighted paging [2]. However, while these previous algorithms were designed and described using the online primal-dual framework, our algorithm is combinatorial. To analyze the performance we employ a novel potential function approach.

By plugging the algorithm for the fractional allocation problem into the above reduction, we get a (roughly) $O(\ell \log(k\ell))$-competitive algorithm for the k-server problem on an HST of depth ℓ, provided that the HST is sufficiently well-separated.

(3): Finally, we note that the competitive guarantee provided by the above k-server algorithm depends on the

[1]In weighted paging, arbitrary weights are associated with fetching the pages into the cache. This problem corresponds to the k-server problem on a weighted star.

depth ℓ of the HST we are working with and, as ℓ can be $\Omega(\log \Delta)$, this guarantee can be polylogarithmic in Δ. Therefore, as Δ can be $2^{\Omega(n)}$, this would lead to competitiveness that is even polynomial in n.[2] To deal with this issue, we define a weighted version of an HST in which the edge lengths on any root-to-leaf path still decrease at (at least) an exponential rate, but the lengths of the edges from a given node to its children could be non-uniform. We prove that any HST can be transformed to a weighted HST of depth $\ell = O(\log n)$ while incurring only an $O(1)$ distortion in leaf-to-leaf distances. We then show that our previous ideas can be applied to weighted HSTs as well. In particular, our online fractional allocation algorithm is actually developed for a weighted star metric (instead of a uniform one), and, as we show, it can be employed in our reduction to obtain a fractional k-server algorithm on a weighted HST. The fractional k-server algorithm can again be rounded to a randomized algorithm with only an $O(1)$ factor loss. Since ℓ is now $O(\log n)$ and thus does not depend on Δ, it gives us an overall guarantee which is polylogarithmic only in n and k.

In Section 2 we describe the above ideas more formally and also give an overview of the paper.

1.2. Preliminaries

We provide definitions and concepts that will be needed in the paper.

Hierarchically well-separated trees: Hierarchical well-separated trees (HST-s), introduced by Bartal [5], [6], is a metric embedding technique in which a general metric is embedded into a probability distribution defined over a set of structured trees (the HST-s). The points of the metric are mapped onto the leaves of the HST, while internal tree nodes represent clusters. The distances along a root-leaf path form a geometric sequence, and this factor is called the *stretch* of the HST. An HST with stretch σ is called a σ-HST. The embedding guarantees that the distance between any pair of vertices in the metric can only increase in an HST, and the expected blowup of each distance, known as the *distortion*, is bounded. It is well known that any metric on n points can be embedded into a distribution over σ-HSTs with distortion $O(\sigma \log_\sigma n)$ [17]. This approach of embedding into HSTs is particularly useful for problems (both offline and online) which seem hard on a general metric, but can be solved fairly easily on trees (or HSTs).

Due to the special structure of HSTs, the task of solving problems on them can sometimes be reduced to the task of solving a more general (and thus harder) problem, but on a uniform metric. For example, this approach was used to obtain the first polylogarithmic guarantees for the *metrical task systems* problem (MTS) by [7] (later further refined by [20]). More precisely, Blum et al. [7] defined a refined version of MTS on a uniform metric known as *unfair*-MTS and showed how an algorithm with a certain refined guarantee for it can be used recursively to obtain an algorithm for MTS on an HST. This approach is especially appealing in the context of the k-server problem, as this problem on a uniform metric (i.e. paging) is well- understood. This motivated Coté et al. [15] to define a problem on a uniform metric, that we call the allocation problem, and show how a good algorithm for it can be used to design good k-server algorithms on HSTs. This problem is defined as follows.

The allocation problem: Suppose that a metric on d points is defined by a weighted star in which the distance from the center to each point i, $1 \le i \le d$, is w_i.[3] At time step t, the total number of available servers, $\kappa(t) \le k$, is specified, and we call the vector $\kappa = (\kappa(1), \kappa(2), \ldots)$ the *quota pattern*. A request arrives at a point i^t and it is specified by a $(k+1)$-dimensional vector $\vec{h}^t = (h^t(0), h^t(1), \ldots, h^t(k))$, where $h^t(j)$ denotes the cost of serving the request using j servers. The cost vectors at any time are guaranteed to satisfy the following *monotonicity* property: for any $0 \le j \le k-1$, the costs satisfy $h^t(j) \ge h^t(j+1)$. That is, serving a request with more servers cannot increase the cost. Upon receiving a request, the algorithm may choose to move additional servers from other locations to the requested point and then serve it. The cost is divided into two parts. The *movement cost* incurred for moving the servers, and the *hit cost* determined by the cost vector and the number of servers at location i^t.

In this paper, we will be interested in designing algorithms for (a fractional version of) this problem that provide a certain refined competitive guarantee. Namely, we say that an online algorithm for the allocation problem is (θ, γ)-*competitive* if it incurs:

- a hit cost of at most $\theta \cdot (\mathrm{Optcost} + \Delta \cdot g(\kappa))$;
- a movement cost of at most $\gamma \cdot (\mathrm{Optcost} + \Delta \cdot g(\kappa))$,

where $\mathrm{Optcost}$ is the total cost (i.e. hit cost plus movement cost) of an optimal solution to a given instance of the allocation problem, $g(\kappa) := \sum_t |\kappa(t) - \kappa(t-1)|$ is the total variation of the server quota pattern, and Δ is the diameter of the underlying metric space.

From allocation to k-server: Coté et al. [15] showed that a $(1+\varepsilon, \beta)$-competitive online algorithm for the allocation problem on d points – provided ϵ is small enough and $\beta = O_\varepsilon(\mathrm{polylog}(d,k))$ – can be used to obtain a polylogarithmic competitive algorithm for the k-server problem on general metrics. In particular, the next theorem follows from their work and is stated explicitly in [3].

Theorem 2. *Suppose there is a $(1+\varepsilon, \beta)$-competitive algorithm for the allocation problem on a uniform metric*

[2] To see an example when this is the case, one could consider a metric space corresponding to n points on a line that are spaced at geometrically increasing distances.

[3] Even though Coté et al. [15] considered the allocation problem on a uniform metric, we find it useful to work with the more general weighted-star metric version of this problem.

on d points. Let H be an σ-HST with depth ℓ. Then, for any $\epsilon \leq 1$, there is an $O(\beta\gamma^{\ell+1}/(\gamma-1))$-competitive algorithm for the k-server problem on H, where

$$\gamma = (1+\varepsilon)\left(1+\frac{3}{\sigma}\right) + O\left(\frac{\beta}{\sigma}\right).$$

Setting $\varepsilon = 1/\ell$, this gives an $O(\beta\ell)$-competitive algorithm on σ-HSTs, provided the HST separation parameter σ is at least $\beta\ell$.

At a high level, the k-server algorithm in Theorem 2 is obtained as follows. Each internal node p in the HST runs an instance of the allocation problem on the uniform metric formed by its children. In this instance, the cost vectors appearing at a child i are guided by the evolution of the cost of the optimal solution to the instance of the k-server problem restricted to the leaves of the subtree that is rooted at i. Furthermore, the quota patterns for each of the allocation problem instances is determined recursively. The root of the tree has a fixed server quota of k, and the quota corresponding to a non-root node i is specified by the number of servers that are allocated to i by the instance of the allocation problem run at the parent of i. In this manner, the distribution of the servers on the leaves of the tree is determined, thus leading to a solution to the k-server problem. The overall guarantee in Theorem 2 follows roughly by showing that the hit cost guarantee of $(1+\epsilon)$ multiplies at each level of the recursion, while the movement cost guarantee of β adds up.

Weighted HSTs: Note that the guarantee in Theorem 2 depends on ℓ, the depth of the σ-HST which in general is $\Theta(\log_\sigma \Delta)$. To avoid the resulting dependence on the diameter Δ that can be as large as $2^{\Omega(n)}$, we introduce a notion of *weighted* σ-HST. A weighted σ-HST is a tree having the property that for any node p, which is not the root or a leaf, the distance from p to its parent is at least σ times the distance from p to any of its children. Thus, unlike an HST, distances from p to its children can be non-uniform. The crucial property of weighted HSTs that we will show later is that any σ-HST T with $O(n)$ nodes can be embedded into a weighted σ-HST with depth $O(\log n)$, such that the distance between any pair of leaves of T is distorted by a factor of at most $2\sigma/(\sigma-1)$ (which is $O(1)$ if, say, $\sigma \geq 2$). Reducing the depth from $O(\log \Delta)$ to $O(\log n)$ allows us to replace the factor of $\log \Delta$ by $\log n$ in the bound on the competitive factor we get for the k-server problem.

Fractional view of randomized algorithms: The relation between randomized algorithms and their corresponding fractional views is an important theme in our paper. By definition, a randomized algorithm is completely specified by the probability distribution over the configurations (deterministic states) at each time step of the algorithm. However, working explicitly with such distributions is usually very cumbersome and complex, and it is often simpler to work with a *fractional view* of it. In a fractional view, the algorithm only keeps track of the marginal distributions on certain quantities, and specifies how these marginals evolve with time. Note that there are multiple ways to define a fractional view (depending on which marginals are tracked). For example, for the k-server problem on an HST, the fractional view might simply correspond to specifying the probability p_i of having a server at leaf i (instead of specifying the entire distribution on the k-tuples of possible server locations). Clearly, the fractional view is a lossy representation of the actual randomized algorithm. However, in many cases (though not always), a fractional view can be converted back to a randomized algorithm with only a small loss. We now describe the fractional views we will employ for the two main problems considered in this paper.

Fractional view of the k-server problem on an HST: Let T be a σ-HST. For a node $j \in T$, let $T(j)$ be the set of leaves in the subtree rooted at j. In the fractional view, at each time step t, the probability of having a server at leaf i, denoted by p_i^t, is specified. Upon getting a request at leaf i at time t, a fractional algorithm must ensure that $p_i^t = 1$. Let the expected number of servers at time t at the leaves of $T(j)$ be denoted by $k^t(j) = \sum_{i \in T(j)} p_i^t$. Clearly, the movement cost incurred at time t is $\sum_{j \in T} W(j)|k^t(j) - k^{t-1}(j)|$, where $W(j)$ is the distance from j to its parent.

It can be easily verified that for any randomized algorithm, the cost incurred by it is at least as large as that incurred by its induced fractional view. On the other hand, it turns out that the fractional view is not too lossy for a σ-HST (provided $\sigma > 5$). In particular, we show that for a σ-HST with $\sigma > 5$, an online algorithm for the k-server problem in the fractional view above can be converted into an online randomized algorithm while losing only an $O(1)$ factor in the competitive ratio.

The fractional allocation problem: For the allocation problem we consider the following fractional view. For each location $i \in [d]$, and all possible number of servers $j \in \{0, \ldots, k\}$, there is a variable $x_{i,j}^t$ denoting the probability of having *exactly* j servers at location i at time t. For each time t, the variables $x_{i,j}^t$ must satisfy the following constraints.

1) For each location i, the variables $x_{i,j}^t$ specify a probability distribution, i.e., $\sum_j x_{i,j}^t = 1$ and each $x_{i,j}^t$ is non-negative.
2) The number of servers used is at most $\kappa(t)$, the number of available servers. That is,

$$\sum_i \sum_j j \cdot x_{i,j}^t \leq \kappa(t).$$

At time step t, when cost vector h^t arrives at location i^t, and possibly $\kappa(t)$ changes, the algorithm can change its distribution from $\overline{x}^{t-1}$ to $\overline{x}^t$ incurring a hit cost of $\sum_j h^t(j) x_{i^t,j}^t$. The movement cost incurred is defined to

be

$$\sum_i w_i \sum_{j=1}^{k} \left| \sum_{j'<j} x^t_{i,j'} - \sum_{j'<j} x^{t-1}_{i,j'} \right|. \quad (1)$$

Remark: Note that when our configurations are integral, this quantity is exactly the cost of moving the servers from configuration $\overline{x}^{t-1}$ to configuration $\overline{x}^t$. In the fractional case, the move cost incurred is defined to be the *earthmover distance* between the probability vectors x^{t-1} and x^t with respect to a linear metric defined on $\{1, 2, \ldots, k\}$. (The linear distance between j and j' is $|j - j'|$.) The earthmover distance is the optimal solution to a transportation problem in which x^{t-1} is the supply vector, x^t is the demand vector, and the cost of sending one unit of flow between $x^{t-1}_{i,j}$ and $x^t_{i,j'}$ is $w_i \cdot |j - j'|$ (since $|j - j'|$ is the change in number of servers resulting from sending this unit of flow).

A gap instance for the fractional allocation problem: As we mentioned earlier, unlike the fractional view of the k-server problem presented above, the fractional view of the allocation problem turns out to be too weak to obtain a randomized algorithm for its integral version. In particular, in the full version we present an instance of the allocation problem such that the discrepancy between the cost of any integral solution to it and the cost of a certain fractional solution is $\Omega(k)$.

However, even though the fractional view fails to capture the integral allocation problem accurately, we still are able to use it to design a fractional (and, in turn, integral) solution to the k-server problem. In particular, we show that Theorem 2 holds even when we substitute the randomized algorithm for the allocation problem with the fractional algorithm.

2. Overview of Our Approach

In this section we give a formal description of our results, outline how they are organized, and discuss how they fit together so as to obtain our main result.

Fractional allocation algorithm: In Section 3 we consider the fractional allocation problem on a weighted star, and prove the following theorem.

Theorem 3. *For any $\varepsilon > 0$, there exists a* fractional *$(1 + \varepsilon, O(\log(k/\epsilon)))$-competitive allocation algorithm on a weighted star metric.*

From allocation to k-server problem: In the next step we show how the algorithm from Theorem 3 can be used to obtain a fractional k-server algorithm on a sufficiently well-separated weighted HST. In particular, we show that:

Theorem 4. *Let T be a weighted σ-HST with depth ℓ. If for any $0 \le \varepsilon \le 1$ there exists a $(1 + \varepsilon, \log(k/\varepsilon))$-competitive algorithm for the fractional allocation problem on a weighted star, then there is an $O(\ell \log(k\ell))$-competitive algorithm for the fractional k-server problem on T, provided $\sigma = \Omega(\ell \log(k\ell))$.*

Putting it all together: We now show how to use Theorem 4 and Theorem 3 to prove our k-server guarantee for general metrics, i.e., to prove Theorem 1.

To this end, we need two more results that we prove. First,

Theorem 5. *Let T be a σ-HST with $\sigma > 5$. Then any online fractional k-server algorithm on T can be converted into a randomized k-server algorithm on T with an $O(1)$ factor loss in the competitive ratio.*

Note that the above result gives a rounding procedure only for HSTs (and not weighted HSTs). To relate HSTs to weighted HSTs, we show the following.

Theorem 6. *Let T be a σ-HST T with n leaves, but possibly arbitrary depth. Then T can be transformed into a weighted σ-HST $\widetilde{T}$ such that: $\widetilde{T}$ has depth $O(\log n)$, the leaves of $\widetilde{T}$ and T are identical, and any leaf to leaf distance in T is distorted by a factor of at most $2\sigma/(\sigma - 1)$ in $\widetilde{T}$.*

Given the above results, we can present the proof of our main theorem.

Proof of Theorem 1: Our algorithm proceeds as follows. First, we use the standard technique [17] to embed the input (arbitrary) metric M into a distribution μ over σ-HSTs with stretch $\sigma = \Theta(\log n \log(k \log n))$. This incurs a distortion of $O(\sigma \log_\sigma n)$ and the resulting HSTs have depth $O(\log_\sigma \Delta)$, where Δ is the diameter of M.

Next, we pick a random HST T according to the distribution μ, and transform T into $\widetilde{T}$ using Theorem 6. As $\widetilde{T}$ has depth $\ell = O(\log n)$, it holds that $\sigma = \Theta(\ell \log(k\ell))$ and hence applying Theorem 4 to $\widetilde{T}$ gives an $O(\ell \log(k\ell)) = O(\log n \log(k \log n))$-competitive fractional k-server algorithm on $\widetilde{T}$. Since the leaves of T and $\widetilde{T}$ are identical, and the distances only have $O(1)$ distortion, the fractional k-server solution on $\widetilde{T}$ induces an $O(\log n \log(k \log n))$-competitive fractional k-server solution on T. By Theorem 5, this gives an $O(\log n \log(k \log n))$-competitive randomized k-server algorithm on T.

We now relate the optimum k-server cost on M to the optimum on T. Let Opt^*_M denote the optimum k-server solution on M, and let c_T denote the cost of this solution on T. Since the expected distortion of distances in our ensemble of HSTs is small, we have:

$$\mathbb{E}_\mu[c_T] = O(\sigma \log_\sigma n) \cdot \mathrm{Opt}^*_M. \quad (2)$$

Let Alg_T denote the cost of the solution produced by the online algorithm on T, and let Alg_M denote the cost of this solution on the metric M. As the pairwise distances in T are at least the distances in M, $\mathrm{Alg}_M \le \mathrm{Alg}_T$. Also, as Alg_T is $O(\log n \log(k \log n))$-competitive, it follows that:

$$\begin{aligned} \mathrm{Alg}_M &\le \mathrm{Alg}_T = O(\log n \log(k \log n)) \cdot c^*_T \\ &\le O(\log n \log(k \log n)) \cdot c_T \end{aligned}$$

where c^*_T is the optimum k-server cost on T (and hence $c^*_T \le c_T$). Taking expectation with respect to μ above

and using (2), the expected cost of our solution $\mathbb{E}_\mu[\mathrm{Alg}_M]$ satisfies:

$$\begin{aligned}\mathbb{E}_\mu[\mathrm{Alg}_M] &= O(\log n \log(k \log n)) \cdot \mathbb{E}_\mu[c_T] \\ &= O(\sigma \log_\sigma n) \cdot O(\log n \log(k \log n)) \cdot \mathrm{Opt}^*_M,\end{aligned}$$

which implies that the overall algorithm has a competitive ratio of

$$\begin{aligned}& O\left(\sigma\left(\frac{\log n}{\log \sigma}\right)\right) \cdot O\left(\log n \log(k \log n)\right) \\ &= O\left(\frac{\log^3 n (\log(k \log n))^2}{\log \log n}\right) \\ &= O\left(\log^2 k \log^3 n \log \log n\right).\end{aligned}$$

■

Due to lack of space we only provide the description of our algorithm for the fractional allocation problem, as well as, an overview of its analysis. The rest of the details and proofs, including the proofs of the theorems cited above, can be found in the full version of the paper.

3. The Fractional Allocation Problem

Consider a metric corresponding to a weighted star on d leaves (also called *locations*) $1, \ldots, d$, where w_i is the distance from leaf i to the root. Let us fix a sequence of cost vectors $h^0, h^1, \ldots$ and a server quota pattern $\kappa = (\kappa(1), \kappa(2), \ldots)$, where $\kappa(t)$ is the number of servers available at time t, and $\kappa(t) \leq k$ for all times t.

Recall that in the fractional allocation problem the state at each time t is described by non-negative variables $x^t_{i,j}$ denoting the probability that there are *exactly* j servers at location i. At each time t, the variables $x^t_{i,j}$ satisfy: (1) $\sum_j x^t_{i,j} = 1$, for each i; (2) $\sum_i \sum_j j x^t_{i,j} \leq \kappa(t)$.

As we shall see, when describing and analyzing our algorithm for the fractional allocation problem, it will be easier to work with variables $y^t_{i,j}$, defined as

$$y^t_{i,j} = \sum_{j'=0}^{j-1} x^t_{i,j'}, \quad \text{for } i \in \{1, \ldots, d\}, \quad j \in \{1, 2, \ldots, k+1\}.$$

I.e., $y^t_{i,j}$ is the probability that at time t we have *less* than j servers at location i. Clearly, for every i, as long as:

$$y^t_{i,j} \in [0,1] \qquad y_{i,k+1} = 1 \tag{3}$$

$$y^t_{i,j-1} \leq y^t_{i,j}, \quad \forall i \in \{1, \ldots, d\}, j \in \{2, \ldots, k\}, \tag{4}$$

there is always a unique setting of the variables $x^t_{i,j}$s that corresponds to the $y^t_{i,j}$s. Therefore, in what follows we make sure that the variables $y^t_{i,j}$s generated by our algorithm satisfy the above two conditions.

The condition that at most $\kappa(t)$ servers are available at each time t can be expressed in terms of $y^t_{i,j}$ as:

$$\begin{aligned}\sum_{i=1}^{d}\sum_{j=1}^{k} y^t_{i,j} &= \sum_{i=1}^{d}\sum_{j=0}^{k}(k-j)x^t_{i,j} \\ &= k\sum_{i=1}^{d}\sum_{j=0}^{k} x^t_{i,j} - \sum_{i=1}^{d}\sum_{j=0}^{k} j x^t_{i,j} \\ &= kd - \left(\sum_{i=1}^{d}\sum_{j=0}^{k} j x^t_{i,j}\right) \geq kd - \kappa(t).\end{aligned} \tag{5}$$

Let us now focus on a particular cost vector $h^t = (h^t(0), h^t(1), \ldots, h^t(k))$ corresponding to time step t. Recall that $h(j)^t$ is the hit cost incurred when serving the request using *exactly* j servers. We can express h^t as

$$\lambda^t_j = \begin{cases} h(j-1) - h(j) & j = 1, 2, \ldots, k \\ h(k) & j = k+1 \end{cases}$$

The variables λ^t_j are non-negative as the hit costs are non-increasing in j, i.e., $h^t(0) \geq h^t(1) \geq \ldots \geq h^t(k)$. Intuitively, λ^t_j captures the marginal cost of serving the request with strictly less than j servers.[4] The hit cost incurred by a configuration $\overline{y}^t = \{y^t_{i,j}\}_{i,j}$ now has a simple formulation. Let i^t denote the location on which the hit cost vector h^t appears, then the hit cost $\sum_{j=0}^{k-1} h^t(j) x^t_{i^t,j}$ can be expressed as

$$\sum_{j=1}^{k} \lambda^t_j \cdot \overline{y}^t_{i^t,j}.$$

Similarly, our formula (1) for the move cost from a configuration $\overline{y}^{t-1}$ to a configuration $\overline{y}^t$ can be conveniently expressed as

$$\sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \left|y^t_{i,j} - y^{t-1}_{i,j}\right|\right).$$

3.1. Description of the Algorithm

In the light of the above discussion, all one needs to do to describe the algorithm is to specify how state $\{y^{t-1}_{i,j}\}_{i,j}$ evolves to $\{y^t_{i,j}\}_{i,j}$ upon arrival of cost vector h^t and server quota $\kappa(t)$. Our algorithm performs this evolution in two stages. First, it executes a Fix stage in which the number of servers is decreased so as to comply with a possible decrease of the quota $\kappa(t)$. Then, a Hit stage proceeds. During this stage the (fractional) configuration of the servers is modified to react to the cost vector h^t. We describe the dynamics of both stages as a continuous process. As it turns out,

[4] We note that we can assume that λ^t_{k+1} is always 0. Otherwise, as any valid algorithm (including the optimal one) always has at most k servers at a given location, any competitive analysis established for the case $\lambda^t_{k+1} = h^t(k) = 0$ carries over to the general case. Thus, from now on we remove λ^t_{k+1} and also $y_{i,k+1}$ (that is always 1) from our considerations and notation.

viewing the evolution of the configurations through that lens allows us to simplify the description and the analysis of the algorithm. More precisely, this evolution during the Fix stage is parametrized by a time index τ that starts initially at 0 and grows until the number of servers is no more than $\kappa(t)$. The Hit stage, on the other hand, is parametrized by a time index η that starts initially at 0 and ends at 1.

For the sake of simplicity, let us drop the index t from our notation since it does not play any role in our analysis. We denote the configuration at time $t-1$ by y^0 and the configuration at time t by y^1. Let λ denote the hit cost vector λ^t and let $\bar{i}$ denote the location i^t that λ^t penalizes. The intermediate states of the algorithm are denoted by y^τ, for $\tau \geq 0$, during the Fix stage, and by y^η, for $\eta \in [0,1]$, during the Hit stage. At each time $\eta \in [0,1]$ ($\tau \geq 0$), the algorithm specifies the derivative $\frac{dy^\eta_{i,j}}{d\eta}$ of each $y^\eta_{i,j}$ (Respectively, $\frac{dy^\tau_{i,j}}{d\tau}$ of each $y^\tau_{i,j}$). Denote by τ_e the final value that τ reaches during the Fix stage. Eventually, each $y^t_{i,j}$ is defined as follows.

$$y^t_{i,j} = y^{t-1}_{i,j} + \int_{\tau=0}^{\tau_e} \frac{dy^\tau_{i,j}}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{dy^\eta_{i,j}}{d\eta} d\eta. \qquad (6)$$

Note that one of the issues that needs to be address in this context is proving that the differential equations that specify the derivatives at each step have a (unique) solution and thus the algorithm is well-defined. This turns out to be non-trivial in the case of Hit stage since the derivative during this stage might change in an non-continuous manner. Nevertheless, as we will show, the process is indeed well-defined.

Another technical detail is that during the Hit stage, in intermediate times $\eta \in [0,1]$, we will not work with the hit cost vector λ directly, but rather with a modified version λ^η of it that can vary with η. (During the first reading, the reader may assume that $\lambda^\eta = \lambda$ and skip directly to the description of the fractional algorithm.)

To define λ^η for $\eta \in [0,1]$, we need a notion of blocks.

Blocks: During the Hit stage, for each $\eta \in [0,1]$, we maintain a partition of the index set $\{1,\ldots,k+1\}$ (that corresponds to location $\bar{i}$) into *blocks* $B^\eta_1, B^\eta_2 \ldots, B^\eta_\ell$. This collection of blocks is denoted by $\mathcal{D}^\eta$ and we make sure it satisfies the following properties.

1) $y^\eta_{\bar{i},j}$ is identical for all indices j within any block $B \in \mathcal{D}^\eta$. For future reference, let us denote by $y^\eta(B)$ this common value for all $j \in B$.
2) For any block $B = \{j,\ldots,j+s-1\}$ of length s in $\mathcal{D}^\eta$, it holds that for every $1 \leq r \leq s$,

$$\sum_{j'=j}^{j+r-1} \frac{1}{r}\lambda_{j'} \leq \sum_{j'=j}^{j+s-1} \frac{1}{s}\lambda_{j'}. \qquad (7)$$

That is, the average value of the λ's in any prefix of a block is no more than the average of the entire block.

We define λ^η to be the cost vector obtained by averaging λ over the blocks in $\mathcal{D}^\eta$. That is, for each $B \in \mathcal{D}^\eta$, $\lambda(B) = (\sum_{j\in B}\lambda_j)/|B|$. For each $j \in B$, $\lambda^\eta(j) = \lambda(B)$.

Now, in our algorithm, we take the initial partitioning $\mathcal{D}^0$ of blocks to be a trivial one, i.e., a one in which each index j forms its own block. (Note that we thus have $\lambda^0 = \lambda$.) Next, blocks are updated as η increases. We have that for any $\eta \geq 0$ if two consecutive blocks $B_p, B_{p+1} \in \mathcal{D}^\eta$ satisfy:

$$y^\eta(B_p) = y^\eta(B_{p+1}) \quad \text{and} \quad \lambda(B_p) \leq \lambda(B_{p+1}), \qquad (8)$$

then B_p and B_{p+1} are merged and $\mathcal{D}^\eta$ is modified accordingly. Note that the condition $\lambda(B_p) \leq \lambda(B_{p+1})$ guarantees that (7) is satisfied in the block created by merging of B_p and B_{p+1}. As we shall see later (Lemma 9), a crucial property of the evolution of $\mathcal{D}^\eta$ during the Hit stage is that $y^\eta_{i,j}$s are updated in a way that guarantees that a block never splits once it is formed.

The algorithm: We are now ready to state our algorithm. This algorithm is parametrized by a certain parameter $\epsilon > 0$ that will be fixed later.

Fractional Allocation Algorithm:
Set $\beta = \frac{\epsilon}{1+k}, \alpha = \ln(1+\frac{1}{\beta}) = \ln(1+\frac{1+k}{\epsilon})$.
Fix stage: Set $y^0 = y^{t-1}$.
For any $\tau \in [0,\infty)$, while $\sum_{(i,j)} y^\tau_{i,j} < kd - \kappa(t)$ (i.e., while the total volume of servers exceeds the quota) we increase each variable $y^\tau_{i,j}$ at a rate:

$$\frac{dy^\tau_{i,j}}{d\tau} = \begin{cases} \frac{1}{w_i}\left(y^\tau_{i,j}+\beta\right) & y^\tau_{i,j} < 1 \\ 0 & y^\tau_{i,j} = 1 \end{cases}$$

Hit stage: Set y^0 to be the state obtained at the end of the Fix stage[a]. Define the following update rule for any $\eta \in [0,1]$:

- If $\sum_{(i,j)} y^\eta_{i,j} = kd - \kappa(t)$, choose $N(\eta) \geq 0$ such that[b]:
  - $\frac{dy^\eta_{i,j}}{d\eta} = 0$ if $(N(\eta) - \alpha\lambda^\eta_{i,j}) > 0$ and $y^\eta_{i,j} = 1$, or $\left(N(\eta) - \alpha\lambda^\eta_{i,j}\right) \leq 0$ and $y^\eta_{i,j} = 0$;
  - $\frac{dy^\eta_{i,j}}{d\eta} = \frac{1}{w_i}(y^\eta_{i,j}+\beta)\cdot(N(\eta)-\alpha\lambda^\eta_{i,j})$, otherwise.
- Else (i.e., if $\sum_{(i,j)} y^\eta_{i,j} > kd-\kappa(t)$), set $N(\eta) = 0$, and define the derivatives of the variables as above.

Output: For each (i,j), return $y^t_{i,j} \triangleq y^{t-1}_{i,j} + \int_{\tau=0}^{\tau_e} \frac{dy^\tau_{i,j}}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{dy^\eta_{i,j}}{d\eta} d\eta$.

[a] Note that upon termination of the fix stage, $\sum_{(i,j)} y^{\tau_e}_{i,j} \geq kd - \kappa(t)$.
[b] As we show in Lemma 7, there is always a way of choosing $N(\eta)$ such that the desired conditions are satisfied.

High-level intuition: Before proving correctness and analyzing the performance of the above algorithm, we provide some intuition on the dynamics underlying it.

The dynamics of the fix stage are fairly straightforward. During this stage the algorithm simply increases all variables $y^{\tau}_{i,j}$ that are strictly less than 1 (which decreases the total number of servers), until the quota $\kappa(t)$ on the number of servers is met. (We note that it may be the case that the total number of servers is strictly smaller than $\kappa(t)$, e.g., if the server quota increases at time t.) Notice that the rate of change of a variable $y^{\tau}_{i,j}$ is proportional to its value, which means that the change is governed by an exponential function. This kind of update rule is consistent with previous algorithms for weighted paging [2], [3].

Next, let us consider the hit stage. For simplicity, let as assume that during this stage we have that:

$$0 < y^{\eta}_{i,1} < y^{\eta}_{i,2} < \ldots < y^{\eta}_{i,k} < 1, \tag{9}$$

for all locations i and $\eta \in [0,1]$. That is, $0 < y^{\eta}_{i,j} < 1$ for all (i,j) and each $y^{\eta}_{i,j}$ is a strictly increasing function of j.

Note that given this assumption holds, condition (8) will never trigger. As a result, no blocks are merged and we have $\lambda^{\eta} = \lambda$ for all $\eta \in [0,1]$. Furthermore, as in this case $y^{\eta}_{i,j}$ is strictly between 0 and 1, the rate of change, $\frac{dy^{\eta}_{i,j}}{d\eta}$, of each variable $y^{\eta}_{i,j}$ during the hit stage simplifies to

$$\frac{dy^{\eta}_{i,j}}{d\eta} = \frac{1}{w_i}\left(y^{\eta}_{i,j} + \beta\right) \cdot \left(N(\eta) - \alpha\lambda_{i,j}\right), \tag{10}$$

with

$$N(\eta) \quad = \quad \begin{cases} 0 & \text{if } \sum_{(i,j)} y^{\eta}_{i,j} > kd - \kappa(t), \\ \frac{\sum_{(i,j)} \frac{1}{w_i}\left(y^{\eta}_{i,j}+\beta\right)\cdot\alpha\lambda_{i,j}}{\sum_{(i,j)} \frac{1}{w_i}\left(y^{\eta}_{i,j}+\beta\right)} & \text{otherwise.} \end{cases} \tag{11}$$

(Note that the value of $N(\eta)$ in the second case of (11) is determined by the fact that $\sum_{(i,j)} \frac{dy^{\eta}_{i,j}}{d\eta}$ has to be zero.)

Now, to interpret the dynamics emerging from (10), we note that $\lambda_{i,j} = 0$ if $i \neq \bar{i}$. Thus, during the hit stage, if our number of servers is still below the quota, we respond to the cost vector λ by simply increasing the number of servers at location $\bar{i}$ (i.e., each $y^{\eta}_{\bar{i},j}$ decreases at a rate of $\frac{1}{w_{\bar{i}}}(y^{\eta}_{\bar{i},j} + \beta) \cdot \alpha\lambda^{\eta}_{\bar{i},j}$). This, in principle, leads to a reduction of the hit cost incurred.

Otherwise, if the number of servers is exactly at the quota, the dynamics described by (10) correspond to offsetting the increase of the number of servers at location $\bar{i}$ by decreasing the number of servers from all locations (including $\bar{i}$). As in the fix stage, the decrease is at a rate proportional to $\frac{1}{w_i}(y^{\tau}_{i,j} + \beta)$ and it is normalized by $N(\eta)$. (Note that as $\lambda_{i,j} = 0$ for $i \neq \bar{i}$, the net effect of this process is decreasing the number of servers from locations different from $\bar{i}$ and redistribution of the number of servers at $\bar{i}$.[5])

Unfortunately, once assumption (9) does not hold, the above simple dynamics may produce infeasible configurations. For example, when increasing or decreasing variables according to (10), they may hit the boundary of the feasible interval $[0,1]$ and then leave it. This happens in the case of variables that are either equal to 0 and have a negative derivative, or are equal to 1 and have a positive derivative. To avoid this problem we need to deactivate such variables (by setting their derivative to be zero) when one of these two cases occurs. Furthermore, the above dynamics may also violate the monotonicity condition (4). Thus, to avoid the latter issues, we need to merge blocks and modify λ^{η} accordingly, as was previously described. Now, the algorithm does not produce infeasible configurations anymore. However, its dynamics are much more involved now. In fact, a sizable part of the technical discussion of our algorithm is devoted to proving that the algorithm is well defined (cf. the full version of the paper).

Well-definiteness of the algorithm: We start by addressing the fact that now, due to our way of ensuring that the variables are in the interval $[0,1]$, it is not clear any more that there always exists a normalization factor $N(\eta)$ as required in the algorithm (in particular, the explicit formula for $N(\eta)$ presented in (11) is not valid anymore). It is possible to show, however, that a standard continuity argument can be used to prove that such a $N(\eta) \geq 0$ indeed always exists. This is captured by following lemma whose proof appears in the full version of the paper.

Lemma 7. *There exists a unique $N(\eta) \geq 0$ for which $\sum_{(i,j)} \frac{dy^{\eta}_{i,j}}{d\eta} = 0$, where $\frac{dy^{\eta}_{i,j}}{d\eta}$ as defined in the algorithm, and at least one of $\frac{dy^{\eta}_{i,j}}{d\eta}$ is non-zero.*

We now give a formal definition of an *inactive* location or variable.

Definition 8. *During the Hit stage, location (i,j) (or $y^{\eta}_{i,j}$) is said to be* inactive *at time $\eta \in [0,1]$ if either $\left(N(\eta) - \alpha\lambda^{\eta}_{i,j}\right) > 0$ and $y^{\eta}_{i,j} = 1$, or $\left(N(\eta) - \alpha\lambda^{\eta}_{i,j}\right) \leq 0$ and $y^{\eta}_{i,j} = 0$. Otherwise, location (i,j) is said to be* active *at η. During the Fix stage, all locations (i,j) for which $y^{\tau}_{i,j} < 1$ are said to be* active *at time τ. Let A^{η} (respectively A^{τ}) denote the set of active locations at time η (respectively τ).*

Since our algorithm is defined via a set of differential equations indexed by τ and η, to establish that it is well defined, we need to show that there exists a unique solution to theme, and furthermore that this solution is feasible for the allocation problem. To this end, the following lemma is proved in the full version of the paper.

Lemma 9. *There exists a unique solution y^{τ} and y^{η}, defined on the intervals $\tau \geq 0$, $\eta \in [0,1]$, to the set of differential equations defined by the algorithm. Furthermore, the solution satisfies the following properties:*

- **Boundary:** *For each (i,j), and for all $0 \leq \tau$, and $0 \leq \eta \leq 1$: $0 \leq y^{\eta}_{i,j}, y^{\tau}_{i,j} \leq 1$.*

[5] Observe that even though the total number of servers at location $\bar{i}$ always increases, it might still happen that some particular variable $y^{\eta}_{\bar{i},j}$ grows.

- **Monotonicity:** *For each* (i,j), $(j \leq k)$, $y^{\eta}_{i,j} \leq y^{\eta}_{i,j+1}$ *and* $y^{\tau}_{i,j} \leq y^{\tau}_{i,j+1}$.
- **Feasibility:** *The expected number (volume) of servers at the end of the Fix stage and at any* $\eta \in [0,1]$ *does not exceed* $\kappa(t)$. *That is,* $\sum_{(i,j)} y^{\eta}_{i,j} \geq kd - \kappa(t)$ *for all* $\eta \in [0,1]$.
- **Blocks:** *During the Hit stage, Blocks can only merge (and they never split).*
- **Discontinuity:** *The total number of times* $\eta \in [0,1]$ *that each location* (i,j) *changes its status from active to inactive, as well as, the number of discontinuity points of* $N(\eta)$ *as a function of* η, *is finite.*

3.2. Accounting

In this section we describe a way of dealing with charging of the hit cost and the move cost that allows us to analyze the performance of the algorithm and of the optimal solution in a continuous fashion.

Charging the hit cost: Let y^* be the optimal solution at time t. The hit cost at time t of the optimal solution is $\lambda \cdot y^*$, where λ corresponds to the hit cost at time t. Rather, in our analysis we would like to charge the hit cost of the optimal solution as $\int_{\eta=0}^{1} \lambda^{\eta} \cdot y^* \cdot d\eta$. Similarly, our algorithm actually pays a hit cost of $\lambda \cdot y^1$ (i.e. the cost is incurred for the final state at time t). However, we would like to account the hit cost in our analysis as $\int_{\eta=0}^{1} \lambda^{\eta} \cdot y^{\eta} d\eta$.

To facilitate this accounting, we prove in the full version of the paper the following lemma.

Lemma 10. *We have*

$$\int_{\eta=0}^{1} \lambda(h^{\eta}) \cdot y^* \cdot d\eta \quad \leq \quad \lambda(h) \cdot y^* \tag{12}$$

$$\int_{\eta=0}^{1} \lambda(h^{\eta}) \cdot y^{\eta} d\eta \quad \geq \quad \lambda(h) \cdot y^1 \tag{13}$$

Roughly speaking, this lemma shows that our way of accounting of hit cost can only overcharge the hit cost of the online algorithm and can only undercharge the hit cost of the optimal solution.

Charging the move cost: With some manipulation (see the full version of the paper) we can charge the movement cost of the optimal solution for increasing $y^*_{i,j}$ (and not for decreasing), also we may charge the online algorithm as follows.

Claim 11. *For any time step* t, *we account the move cost during the fix stage as* $\sum_{(i,j)} w_i \int_{\tau} \frac{dy^{\tau}_{i,j}}{d\tau} d\tau$ *and as* $\sum_{(i,j)} w_i \cdot \int_{\eta=0}^{1} \left(y^{\eta}_{i,j} + \beta\right) N(\eta) \cdot \mathbf{1}_{(i,j) \in A^{\eta}} d\eta$ *during the Hit stage.*

3.3. Competitive Analysis

To bound the competitiveness of the algorithm in the full version of the paper we prove the following theorem.

Theorem 12. *Consider an arbitrary instance of allocation problem with cost vectors* $h^1, h^2, \ldots$, *a starting configuration* $\overline{y}^0$ *and a quota pattern* $\kappa = (\kappa(1), \kappa(2), \ldots)$. *For any* $0 \leq \varepsilon \leq 1$, *we have:*

$$\begin{aligned} \mathrm{H} &\leq (1+\varepsilon)\left(\mathrm{Opt} + w_{\max} \cdot g(k)\right) + C \qquad \textit{and,} \\ \mathrm{M} &\leq (1+\epsilon)\alpha \cdot \left(\mathrm{Opt} + w_{\max} \cdot g(\kappa)\right). \end{aligned}$$

Here, H *and* M *denote the hit and move costs of our fractional algorithm, and* Opt *denotes the sum of the total hit and move costs of a fixed* integral *optimum solution to the allocation problem instance. Let* $g(\kappa) := \sum_t |\kappa(t) - \kappa(t-1)|$, *and denote by* $w_{\max} = \max_i w_i$ *the diameter of our metric space. Let* C *be a quantity that depends only on the start and final configurations of the online algorithm, and it is zero if these two configurations are the same.*

To prove Theorem 12 we employ a potential function approach. Namely, we define potentials $\Phi^h(\overline{y}, t)$ and $\Phi^m(\overline{y}, t)$ that depend on the state $\overline{y}$ of the online algorithm and on the state of some arbitrary fixed integral optimum solution at time t. Then we show that the following inequalities are satisfied at each time step t.

$$\begin{aligned} &\mathrm{M}_t + \Delta\Phi^m_t \qquad (14) \\ &\leq (1+\epsilon) \cdot \alpha \cdot \left(w_{\max} \cdot |\kappa(t) - \kappa(t-1)| + \mathrm{M^*}_t + \mathrm{H^*}_t\right) \\ &\mathrm{H}_t + \Delta\Phi^h_t + \frac{1}{\alpha}\Delta\Phi^m_t \qquad (15) \\ &\leq (1+\epsilon)\left(w_{\max} \cdot |\kappa(t) - \kappa(t-1)| + \mathrm{M^*}_t + \mathrm{H^*}_t\right) \end{aligned}$$

Here, H_t (resp. $\mathrm{H^*}_t$) and M_t (resp. $\mathrm{M^*}_t$) denote the hit and move cost incurred by the algorithm (resp. optimum) at time t. The quantities

$$\begin{aligned} \Delta\Phi^h_t &:= \Phi^h(\overline{y}^t, t) - \Phi^h(\overline{y}^{t-1}, t-1) \\ \Delta\Phi^m_t &:= \Phi^m(\overline{y}^t, t) - \Phi^m(\overline{y}^{t-1}, t-1) \end{aligned}$$

denote the change in the potentials Φ^h and Φ^m at time step t.

It will be the case that $\Phi^m(\overline{y}, t) \geq 0$ and $\Phi^m(\overline{y}^0, 0) = 0$. Moreover, both $\Phi^m(\overline{y}, t)$ and $\Phi^h(\overline{y}, t)$ will be bounded by some universal constant C, independent of the length of the request sequence. Thus, Theorem 12 will follow by summing up (14) and (15) over all times t.

The Potential Functions: The potential function Φ^m is defined as follows.

$$\Phi^m(\overline{y}, t) \quad := \quad (1+\epsilon) \cdot \sum_i w_i \left(\sum_j y^{*t}_{i,j} \cdot \log\left(\frac{1+\beta}{y^{\eta}_{i,j} + \beta}\right)\right).$$

Here, $y^{*t}_{i,j} = 1$ if the optimum has fewer than j servers at i, and otherwise $y^{*t}_{i,j} = 0$ (note that $y^{*t}_{i,j} \in \{0,1\}$, as the optimum can be assumed to be integral). In other words, if the optimum has k^*_i servers at location i at time t, then the

contribution of location i to $\Phi^m(\overline{y}^t, t)$ is

$$w_i \left(\sum_{j > k_i^*} \log \left(\frac{1+\beta}{y_{i,j}^t + \beta} \right) \right).$$

Roughly speaking, $\Phi^m(\overline{y}^t, t)$ accounts for the excess servers in the online configuration $\overline{y}^t$ at location i compared to the optimum solution. For example, suppose that $\overline{y}^t$ has k_i servers at location i, i.e., $y_{i,j}^t = 0$ for $j \leq k_i$, for some $k_i > k_i^*$, and $y_{i,j}^t = 1$ otherwise. Then the contribution of location i to $\Phi^m(\overline{y}^t, t)$ is $O(w_i(k_i - k_i^*) \log k)$. Intuitively, the offline adversary can penalize the online algorithm for "wasting" $k_i - k_i^*$ servers at i, by giving cost vectors at the other locations (where the optimum has more servers), by making it pay a larger hit cost. The potential $\Phi^m(\overline{y}^t, t)$ will be used to offset this additional hit cost in such situations.

Next, we define the potential Φ^h to be

$$\Phi^h(\overline{y}, t) \quad := \quad \frac{1}{\alpha}(\sum_{(i,j)} w_i \cdot y_{i,j}^t),$$

It is easily verified that these potentials are bounded. Moreover $\Phi^m(\overline{y}^0, 0) = 0$, as both offline and online are assumed to have the same initial configuration.

The overview of the proof: Now, Theorem 12 is established by dividing the events at time t into the following four parts, and show that Inequalities (14) and (15) hold in each of them for every t.

1) The quota $\kappa(t)$ either increases, decreases, or stays unchanged, in comparison with $\kappa(t-1)$, and the optimum removes/adds servers accordingly.
2) The optimum moves some servers and its state changes from $\overline{y}^{*t-1}$ to $\overline{y}^{*t}$.
3) The fix stage of the online algorithm.
4) The hit costs of both the online algorithm and the optimum.

We refer the reader to the full version of the paper for the details of the proof.

REFERENCES

[1] D. Achlioptas, M. Chrobak, and J. Noga, "Competitive analysis of randomized paging algorithms," *Theor. Comput. Sci.*, vol. 234, no. 1-2, pp. 203–218, 2000.

[2] N. Bansal, N. Buchbinder, and J. S. Naor, "A primal-dual randomized algorithm for weighted paging," in *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, 2007, pp. 507–517.

[3] ——, "Towards the randomized k-server conjecture: A primal-dual approach," in *SODA*, 2010.

[4] ——, "Unfair metrical task systems on hsts and applications," in *ICALP*, 2010.

[5] Y. Bartal, "Probabilistic approximations of metric spaces and its algorithmic applications," in *IEEE Symposium on Foundations of Computer Science*, 1996, pp. 184–193.

[6] ——, "On approximating arbitrary metrices by tree metrics," in *STOC*, 1998, pp. 161–168.

[7] 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, pp. 711–719.

[8] Y. Bartal, B. Bollobás, and M. Mendel, "A ramsy-type theorem for metric spaces and its applications for metrical task systems and related problems," in *FOCS*, 2001, pp. 396–405.

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

[10] Y. Bartal, N. Linial, M. Mendel, and A. Naor, "On metric ramsey-type phenomena," in *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 463–472.

[11] A. Blum, H. J. Karloff, Y. Rabani, and M. E. Saks, "A decomposition theorem and bounds for randomized server problems," in *FOCS*, 1992, pp. 197–207.

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

[13] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan, "New results on server problems," *SIAM J. Discret. Math.*, vol. 4, no. 2, pp. 172–181, 1991.

[14] M. Chrobak and L. Larmore, "An optimal on-line algorithm for k-servers on trees," *SIAM Journal on Computing*, vol. 20, no. 1, pp. 144–?148, 1991.

[15] A. Coté, A. Meyerson, and L. Poplawski, "Randomized k-server on hierarchical binary trees," in *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, 2008, pp. 227–234.

[16] B. Csaba and S. Lodha, "A randomized on-line algorithm for the k-server problem on a line," *Random Structures and Algorithms*, vol. 29, no. 1, pp. 82–104, 2006.

[17] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics," in *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, 2003, pp. 448–455.

[18] A. Fiat, Y. Rabani, and Y. Ravid, "Competitive k-server algorithms," *Journal of Computer and System Sciences*, vol. 48, no. 3, pp. 410–428, 1994.

[19] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, "Competitive paging algorithms," *J. Algorithms*, vol. 12, no. 4, pp. 685–699, 1991.

[20] A. Fiat and M. Mendel, "Better algorithms for unfair metrical task systems and applications," *SIAM J. Comput.*, vol. 32, no. 6, pp. 1403–1422, 2003.

[21] E. F. Grove, "The harmonic online k-server algorithm is competitive," in *STOC*, 1991, pp. 260–266.

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

[23] M. Manasse, L. McGeoch, and D. Sleator, "Competitive algorithms for server problems," *Journal of Algorithms*, vol. 11, pp. 208–230, 1990.

[24] L. A. McGeoch and D. D. Sleator, "A strongly competitive randomized paging algorithm," *Algorithmica*, vol. 6, no. 6, pp. 816–825, 1991.

[25] S. S. Seiden, "A general decomposition theorem for the k-server problem," in *ESA '01: Proceedings of the 9th Annual European Symposium on Algorithms*, 2001, pp. 86–97.

[26] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Commun. ACM*, vol. 28, no. 2, pp. 202–208, 1985.