A Polylogarithmic-Competitive Algorithm for the k-Server Problem

Nikhil Bansal [*]   Niv Buchbinder [†]   Aleksander Mądry [‡]   Joseph (Seffi) Naor [§]

Abstract

We give the first polylogarithmic-competitive randomized online 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. Our algorithm improves upon the deterministic $(2k-1)$-competitive algorithm of Koutsoupias and Papadimitriou [23] whenever n is sub-exponential in k.

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. [24] 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 [27] gave several k-competitive algorithms for paging, and showed that no deterministic algorithm can do better. This prompted Manasse et al. [24] 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. [19]. Improved bounds were obtained later on by [22, 9], though the ratio still remained exponential in k. A major breakthrough was achieved by Koutsoupias and Papadimitriou [23], 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 [14, 15].

Even though the aforementioned results are all deterministic, there is also a great deal of interest in randomized algorithms for the k-server problem. 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 [20, 25, 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 [12, 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 $2k - 1$ [23] 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 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 [26], the case of a metric corresponding to a binary HST with high separation [16], the case of $n = k + O(1)$ [7], as well as some other cases [17, 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 [13].

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. [16] for solving the k-server problem on hierarchically well-separated trees (HSTs). It is well known that

---

[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.

solving the problem on HSTs suffices, as any metric space can be embedded into a probability distribution over HSTs with low distortion [18].

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. [16] 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. [16], 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. Employing this relaxation makes 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. [16], 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 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 $O(\sigma \log_\sigma n)$ [18]. 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 [21]). More precisely, Blum et al. [7] defined a refined version of MTS on a uniform metric known as *unfair*-MTS and showed how an

---

[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.

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. [16] 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, $1 \leq i \leq d$, is w_i.[3] At time step t, the total number of available servers, $\kappa(t) \leq k$, 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 $0 \leq j \leq k-1$, the costs satisfy $h^t(j) \geq h^t(j+1)$. 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 $\theta \cdot (\text{Optcost} + \Delta \cdot g(\kappa))$;

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

where 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. [16] showed that a $(1 + \varepsilon, \beta)$-competitive online algorithm for the allocation problem on d points – provided ε is small enough and $\beta = O_\varepsilon(\text{polylog}(d, k))$ – 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 on d points. Let H be an σ-HST with depth ℓ. Then, for any $\varepsilon \leq 1$, 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 $\beta\ell$.

---

[3]Even though Coté et al. [16] 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.

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. The distribution of the servers on the leaves of the tree is determined in this manner, 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 $(1+\varepsilon)$ 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 the notion of a *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 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 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 in T.

It is easy to verify that the cost incurred by any randomized algorithm is at least as large as the cost incurred by its induced fractional view. Conversely, it turns out that the fractional view is not too lossy for a σ-HST (provided $\sigma > 5$). In particular, in Section 5.2 we show that for a σ-HST ($\sigma > 5$), 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

$$\sum_i w_i \sum_{j=1}^{k} \left| \sum_{j'<j} x_{i,j'}^t - \sum_{j'<j} x_{i,j'}^{t-1} \right|. \tag{1}$$

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, each term in the outermost sum can be seen as equal to the *earthmover distance* between the probability vectors $(x_{i,0}^{t-1}, \ldots, x_{i,k}^{t-1})$ and $(x_{i,0}^t, \ldots, x_{i,k}^t)$ with respect to a linear metric defined on $\{0, 1, \ldots, k\}$. 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 $x_{i,j}^{t-1}$ and $x_{i,j'}^t$ is $w_i \cdot |j - j'|$, since $|j - j'|$ is the change in number of servers resulting from sending this unit of flow. It is not hard to see[4] that in the case of a linear metric, the optimal solution to the transportation problem (up to a factor of 2) is captured by (1).

A gap instance for the fractional allocation problem. As 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 yield a randomized algorithm for its integral counterpart. We thus present an instance of the allocation problem for which the ratio between the cost of any integral solution and the cost of an optimal fractional solution is $\Omega(k)$. However, we stress that even though the fractional view fails to approximate the integral allocation problem, we are still able to use it to design a fractional (and, in turn, integral) solution to the k-server problem. In particular, we

---

[4]Using uncrossing arguments on the optimal transportation solution.

show in Section 4 that Theorem 2 holds even when we substitute the randomized algorithm for the allocation problem with the fractional algorithm.

Let us consider a uniform metric space over $d = 2$ points, and consider an instance of the allocation problem in which exactly $\kappa(t) = k$ servers are available at each time. Furthermore, at each odd time step $1, 3, 5, \ldots$, the cost vector $h = (1, 1, ..., 1, 0)$ arrives at location 1, and at each even time step $2, 4, 6, \ldots$, the vector $h' = (1, 0, 0, ..., 0)$ arrives at location 2.

We show that any integral solution to this instance of the allocation problem must incur a high cost, while there is an $\Omega(k)$ times cheaper solution in the fractional view.

Claim 3. *Any solution to the instance above incurs a cost of $\Omega(T)$ over T time steps.*

Proof. Observe that the hit cost can be avoided at location 1 only if it contains k servers, and it can be avoided at location 2 only if it contains at least one server. Thus, any algorithm that does not pay a hit cost of at least 1 during any two consecutive time steps, must move at least one server between locations 1 and 2, incurring a movement cost of at least 1, concluding that the cost is $\Omega(T)$. $\square$

Claim 4. *There is a solution in the fractional view of cost $O(T/k)$ over T time steps.*

Proof. Consider the following solution in the fractional view. At each time step t, let:

$$x_{1,0}^t = \frac{1}{k}, \qquad x_{1,k}^t = 1 - \frac{1}{k}, \qquad \text{and} \qquad x_{2,1}^t = 1.$$

Note that this solution satisfies all the constraints in the fractional view. Since location 2 always has a server, it never pays any hit cost. Moreover, location 1 has fewer than k servers with probability $1/k$, it thus incurs only a hit cost $1 \cdot x_{1,0} = 1/k$ at every odd time step. Also, as the solution does not change over time, the movement cost is 0. $\square$

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

From allocation to k-server problem. In Section 4 we show how the algorithm from Theorem 5 can be used to obtain a fractional k-server algorithm on a sufficiently well-separated weighted HST. In particular, we show that:

Theorem 6. *Let T be a weighted σ-HST of depth ℓ. If, for any $0 \leq \varepsilon \leq 1$, 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 Theorems 5 and 6 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 in Section 5. First,

Theorem 7. *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 8. *Let T be a σ-HST 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 [18] 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 8. As $\widetilde{T}$ has depth $\ell = O(\log n)$, it holds that $\sigma = \Theta(\ell \log(k\ell))$ and hence applying Theorem 6 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 7, 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:

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

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, $\mathrm{Alg}_M \leq \mathrm{Alg}_T$. Also, as Alg_T is $O(\log n \log(k \log n))$-competitive, it follows that:

$$\mathrm{Alg}_M \leq \mathrm{Alg}_T = O(\log n \log(k \log n)) \cdot c_T^* \leq O(\log n \log(k \log n)) \cdot c_T$$

where c_T^* is the optimum k-server cost on T (and hence $c_T^* \leq c_T$). Taking expectation with respect to μ above and using (2), the expected cost of our solution $\mathbb{E}_\mu[\mathrm{Alg}_M]$ satisfies:

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

which implies that the overall algorithm has a competitive ratio of

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

$\square$

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 $x_{i,j}^t$ denoting the probability that there are *exactly* j servers at location i. At each time t, the variables $x_{i,j}^t$ satisfy: (1) $\sum_j x_{i,j}^t = 1$, for each i; (2) $\sum_i \sum_j j x_{i,j}^t \leq \kappa(t)$.

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

$$y_{i,j}^t = \sum_{j'=0}^{j-1} x_{i,j'}^t, \qquad \text{for } i \in \{1, \ldots, d\}, \quad j \in \{1, 2, \ldots, k+1\}.$$

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

$$y_{i,j}^t \quad \in \quad [0,1] \tag{3}$$
$$y_{i,j-1}^t \quad \leq \quad y_{i,j}^t, \qquad y_{i,k+1} = 1, \qquad \forall i \in \{1, \ldots, d\}, \quad j \in \{2, \ldots, k+1\}, \tag{4}$$

there is always a unique setting of the variables $x_{i,j}^t$s that corresponds to the $y_{i,j}^t$s. Therefore, in what follows we make sure that the variables $y_{i,j}^t$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 $y_{i,j}^t$ as:

$$\sum_{i=1}^{d} \sum_{j=1}^{k} y_{i,j}^t \quad = \quad \sum_{i=1}^{d} \sum_{j=0}^{k} (k-j) x_{i,j}^t = k \sum_{i=1}^{d} \sum_{j=0}^{k} x_{i,j}^t - \sum_{i=1}^{d} \sum_{j=0}^{k} j x_{i,j}^t$$

$$= \quad kd - \left(\sum_{i=1}^{d} \sum_{j=0}^{k} j x_{i,j}^t \right) \geq kd - \kappa(t). \tag{5}$$

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 $h^t(j)$ is the hit cost incurred when serving the request using *exactly* j servers. We can express h^t as

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

The variables λ_j^t 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_j^t captures the marginal cost of serving the request with strictly less than j servers.[5] The hit cost incurred by a configuration $\overline{y}^t = \{y_{i,j}^t\}_{i,j}$ now has a simple formulation.

---

[5] We note that we can assume that λ_{k+1}^t 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 $\lambda_{k+1}^t = h^t(k) = 0$ carries over to the general case. Thus, from now on we remove λ_{k+1}^t and also $y_{i,k+1}$ (that is always 1) from our considerations and notation.

Let i^t denote the location on which the hit cost vector h^t appears, then the hit cost $\sum_{j=0}^{k-1} h^t(j) x_{i^t,j}^t$ can be expressed as

$$\sum_{j=1}^{k} \lambda_j^t \cdot \overline{y}_{i^t,j}^t.$$

Similarly, expression (1) for the movement cost from a configuration $\overline{y}^{t-1}$ to a configuration $\overline{y}^t$ becomes

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

3.1 Description of the Algorithm

In light of the above discussion it suffices to specify how state $\{y_{i,j}^{t-1}\}_{i,j}$ evolves to $\{y_{i,j}^t\}_{i,j}$ at time t 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 decrease of the quota $\kappa(t)$. Then, it proceeds with a *hit* stage, during which 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 governed by a set of differential equations. As it turns out, viewing the evolution of the server configuration this way allows us to both simplify the description and the analysis of the algorithm. The evolution of the fractional solution during the fix stage is parametrized by a time index τ that starts at 0 and grows until the number of servers is no more than $\kappa(t)$. The hit stage 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 $\overline{i}$ denote the location i^t that λ^t penalizes. The intermediate states of the algorithm are denoted by y^τ, $\tau \geq 0$, during the fix stage, and by y^η, $\eta \in [0,1]$, during the hit stage. At each time $\eta \in [0,1]$ (respectively, $\tau \geq 0$), the algorithm specifies the derivative $\frac{dy_{i,j}^\eta}{d\eta}$ of each variable $y_{i,j}^\eta$ (respectively, $\frac{dy_{i,j}^\tau}{d\tau}$ of each $y_{i,j}^\tau$). Denote by τ_e the final value that τ reaches during the fix stage. Eventually, each $y_{i,j}^t$ is defined as follows.

$$y_{i,j}^t = y_{i,j}^{t-1} + \int_{\tau=0}^{\tau_e} \frac{dy_{i,j}^\tau}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{dy_{i,j}^\eta}{d\eta} d\eta. \tag{6}$$

An important issue that needs to be addressed is proving that the differential equations specifying the derivatives at each step have a (unique) solution and thus the algorithm is well-defined. This proof turns out to be non-trivial in the case of the hit stage, since the derivative during this stage might change in a non-continuous manner. Nevertheless, as we will show, the process is still 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 cost vector λ^η that can vary with η. (During the first reading, the reader may assume that $\lambda^\eta = \lambda$ and skip the part below about blocks and go directly to the description of the fractional algorithm.)

We initialize $\lambda^0 = \lambda$. To define λ^η for $\eta > 0$, we need the notion of blocks.

Blocks: During the hit stage, for each $\eta \in [0,1]$, we maintain a partition of the index set $\{1, \ldots, k+1\}$ (for location $\bar{\imath}$) into *blocks* $B_1^\eta, B_2^\eta \ldots, B_\ell^\eta$. The collection of blocks is denoted by $\mathcal{D}^\eta$ and it satisfies the following properties.

1. $y_{\bar{\imath},j}^\eta$ 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 $1 \le r \le s$,

$$\sum_{j'=j}^{j+r-1} \frac{1}{r} \lambda_{\bar{\imath},j'} \le \sum_{j'=j}^{j+s-1} \frac{1}{s} \lambda_{\bar{\imath},j'}. \tag{7}$$

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 $B \in \mathcal{D}^\eta$, we set $\lambda(B) = (\sum_{j \in B} \lambda_{\bar{\imath},j})/|B|$, and then

$$\lambda_{i,j}^\eta \;=\; \begin{cases} \lambda(B) & \text{if } i = \bar{\imath} \text{ and } j \in B, \text{ for } B \in \mathcal{D}^\eta, \\ 0 & \text{otherwise.} \end{cases}$$

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

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

then B_p and B_{p+1} are merged and $\mathcal{D}^\eta$ is modified accordingly. Note that the condition $\lambda(B_p) \le \lambda(B_{p+1})$ guarantees that (7) is satisfied in the new block created by merging B_p and B_{p+1}. As we shall see later (Lemma 11), a crucial property of the evolution of $\mathcal{D}^\eta$ during the hit stage is that $y_{\bar{\imath},j}^\eta$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. It is parameterized by a parameter $\varepsilon > 0$ that will be fixed later.

Fractional Allocation Algorithm:

Set $\beta = \frac{\varepsilon}{1+k}, \alpha = \ln(1 + \frac{1}{\beta}) = \ln(1 + \frac{1+k}{\varepsilon})$.

Fix stage: Set $y^0 = y^{t-1}$.

For any $\tau \in [0, \infty)$, while $\sum_{i,j} y_{i,j}^\tau < kd - \kappa(t)$ (i.e., while the total volume of servers exceeds the quota) we increase each variable $y_{i,j}^\tau$ at a rate:

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

Denote by τ_e the termination time of the fix stage.

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 $\sum_{i,j} y_{i,j}^\eta = kd - \kappa(t)$, choose $N(\eta) \geq 0$ such that[b]:

$$\frac{dy_{i,j}^\eta}{d\eta} = \begin{cases} 0 & \text{if either } \left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) > 0 \text{ and } y_{i,j}^\eta = 1, \\ & \text{or } \left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) \leq 0 \text{ and } y_{i,j}^\eta = 0 \quad (9) \\ \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot \left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) & \text{otherwise} \end{cases}$$

  and

$$\sum_{i,j} \frac{dy_{i,j}^\eta}{d\eta} = 0.$$

- Otherwise (i.e., if $\sum_{i,j} y_{i,j}^\eta > kd - \kappa(t)$), set $N(\eta) = 0$, and define the derivatives of the variables as above.

Output: For each (i, j), return $y_{i,j}^t \triangleq y_{i,j}^{t-1} + \int_{\tau=0}^{\tau_e} \frac{dy_{i,j}^\tau}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{dy_{i,j}^\eta}{d\eta} d\eta.$

---

[a]Note that upon termination of the fix stage, $\sum_{i,j} y_{i,j}^{\tau_e} \geq kd - \kappa(t)$.

[b]As we show in Lemma 10, 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.

Dynamics of the fix stage: This is fairly straightforward. The algorithm simply increases all the variables $y_{i,j}^\tau$ 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 also be the case that the total number of servers used is already strictly smaller than the quota $\kappa(t)$ to begin with, e.g., if the server quota increases at time t. In this case, nothing is done during the fix stage. Notice that the rate of change of a variable $y_{i,j}^\tau$ is proportional to its value, which means that the change is governed by an exponential function. This kind of update rule is in line with previous algorithms for weighted paging [2, 3].

Dynamics of the hit stage: For simplicity, let us assume that during this stage we have that $0 < y_{i,j}^\eta < 1$, for all (i,j), and each $y_{i,j}^\eta$ is a strictly increasing function of j. That is,

$$0 < y_{i,1}^\eta < y_{i,2}^\eta < \ldots < y_{i,k}^\eta < 1, \tag{10}$$

for all locations i and $\eta \in [0,1]$.

Note that under this assumption 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 each variable $y_{i,j}^\eta$ is strictly between 0 and 1, its rate of change during the hit stage simplifies to:

$$\frac{dy_{i,j}^\eta}{d\eta} = \frac{1}{w_i} \left(y_{i,j}^\eta + \beta \right) \cdot (N(\eta) - \alpha\lambda_{i,j}), \tag{11}$$

with

$$N(\eta) = \begin{cases} 0 & \text{if } \sum_{i,j} y_{i,j}^\eta > kd - \kappa(t), \\[2ex] \frac{\sum_{i,j} \frac{1}{w_i}\left(y_{i,j}^\eta+\beta\right)\cdot\alpha\lambda_{i,j}}{\sum_{i,j} \frac{1}{w_i}\left(y_{i,j}^\eta+\beta\right)} & \text{otherwise (i.e. if } \sum_{i,j} y_{i,j}^\eta = kd - \kappa(t)). \end{cases} \tag{12}$$

(Note that the value of $N(\eta)$ in the second case of (12) is determined by the fact that $\sum_{i,j} \frac{dy_{i,j}^\eta}{d\eta}$ has to be zero.)

Let us study the dynamics given by (11) more carefully. Recall that $\lambda_{i,j} = 0$ for $i \neq \bar{i}$. First, if the number of servers used is below the quota, i.e. if $\sum_{i,j} y_{i,j}^\eta > kd - \kappa(t)$, the algorithm responds to the cost vector λ by simply increasing the number of servers at location $\bar{i}$ by decreasing each $y_{\bar{i},j}^\eta$ at a rate of $\frac{1}{w_{\bar{i}}}(y_{\bar{i},j}^\eta + \beta) \cdot \alpha\lambda_{\bar{i},j}^\eta$. To understand this better, it is instructive to consider the special case when $\lambda_{\bar{i},\bar{j}}^\eta = 1$ for some particular index $\bar{j}$ and is 0 otherwise (this corresponds to the hit cost vector h^η that incurs cost 1 if there are strictly fewer than $\bar{j}$ servers at $\bar{i}$ and cost 0 otherwise). In this case, the algorithm reduces $y_{\bar{i},\bar{j}}^\eta$ and keeps other $y_{\bar{i},j}^\eta$'s unchanged (in particular $y_{\bar{i},\bar{j}+1}^\eta$ and $y_{\bar{i},\bar{j}-1}^\eta$ remain unchanged). As $y_{\bar{i},\bar{j}+1}^\eta$ and $y_{\bar{i},\bar{j}-1}^\eta$ do not change while $y_{\bar{i},\bar{j}}$ decreases, this has the effect of increasing the probability mass $x_{\bar{i},\bar{j}}^\eta = y_{\bar{i},\bar{j}+1}^\eta - y_{\bar{i},\bar{j}}^\eta$ on $(\bar{i},\bar{j})$, and decreasing the probability mass $x_{\bar{i},\bar{j}-1}^\eta = y_{\bar{i},\bar{j}}^\eta - y_{\bar{i},\bar{j}-1}^\eta$ on $(\bar{i},\bar{j}-1)$[6] Moreover, note that the decrease in $x_{\bar{i},\bar{j}-1}^\eta$ is exactly equal to the increase in $x_{\bar{i},\bar{j}}^\eta$.

Now, let us consider the case when the number of servers used is exactly equal to the quota. Here, we also need to ensure that the quota is maintained. This is done by offsetting the increase in the number of servers at location $\bar{i}$ (as described by the dynamics in the previous paragraph), by decreasing the number of servers at all locations (including $\bar{i}$). This is precisely the purpose of the term $N(\eta)$ in (11). It increases $y_{i,j}^\eta$ (and hence decreases the number of servers) at a rate proportional to $\frac{1}{w_i}(y_{i,j}^\eta + \beta)$ (as in the fix stage). Note that as $\lambda_{i,j} = 0$ for $i \neq \bar{i}$, this update can only decrease the number of servers at locations $i \neq \bar{i}$. The overall number of servers at location $\bar{i}$ can only increase, but of course due to the redistribution of probability mass at $\bar{i}$, it may happen that the probability mass at some $(\bar{i},j)$ goes down.

Unfortunately, when assumption (10) does not hold, the simple dynamics described above may produce infeasible configurations. First, increasing or decreasing variables according to (11) does

---

[6]In the simplified discussion here we are implicitly assuming that $x_{\bar{i},\bar{j}-1}^\eta > 0$ by assuming that $y_{\bar{i},\bar{j}}^\eta > y_{\bar{i},\bar{j}-1}^\eta$.

not take into account that the variables need to stay in the range $[0, 1]$, and hence this may be violated. This happens if (i) a variable is equal to 0 and has a negative derivative, or (ii) when it is equal to 1 and has a positive derivative. To avoid this problem we need to deactivate such variables (by setting their derivative to be 0) when either one of these two cases occurs. Moreover, the above dynamics may also violate the monotonicity condition (4). To avoid this issue, we need to merge blocks and modify λ^η accordingly, as was previously described.

Now, the resulting algorithm does not produce infeasible configurations anymore. However, its dynamics is somewhat more involved. Before we discuss it, let us first provide a formal definition of an *inactive* coordinate or variable.

Definition 9. *During the fix stage, a coordinate (i, j) for which $y_{i,j}^\tau < 1$ is said to be* active *at time τ. Otherwise it is said to be* inactive. *During the hit stage, coordinate (i, j) (or variable $y_{i,j}^\eta$) is said to be* inactive *at time $\eta \in [0, 1]$ if either $\left(N(\eta) - \alpha\lambda_{i,j}^\eta \right) > 0$ and $y_{i,j}^\eta = 1$, or $\left(N(\eta) - \alpha\lambda_{i,j}^\eta \right) \leq 0$ and $y_{i,j}^\eta = 0$. Otherwise, coordinate (i, j) is said to be* active. *Denote by A^η (respectively A^τ) the set of active coordinates at time η (respectively τ).*

Now, by definition, during the hit stage at time η only the active variables might change. So, we can compactly rewrite the evolution of the variables during the hit stage given in (9) as

$$\frac{dy_{i,j}^\eta}{d\eta} = \begin{cases} 0 & \text{if } (i, j) \notin A^\eta, \\ \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot \left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) & \text{otherwise} \end{cases} \tag{13}$$

Furthermore, as we still require that $\sum_{i,j} \frac{dy_{i,j}^\eta}{d\eta} = \sum_{(i,j)\in A^\eta} \frac{dy_{i,j}^\eta}{d\eta} = 0$, if $\sum_{i,j} y_{i,j}^\eta > kd - \kappa(t)$, we have that $N(\eta)$ can be expressed as

$$N(\eta) = \begin{cases} 0 & \text{if } \sum_{i,j} y_{i,j}^\eta > kd - \kappa(t), \\ \frac{\sum_{(i,j)\in A^\eta} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)\cdot\alpha\lambda_{i,j}^\eta}{\sum_{(i,j)\in A^\eta} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)} & \text{otherwise (i.e. if } \sum_{i,j} y_{i,j}^\eta = kd - \kappa(t)). \end{cases} \tag{14}$$

In light of the above, one can see that the simple evolution of the variables in the special case of (10), as described by (11), is a special case of the general evolution in which all coordinates are being active and $\lambda^\eta = \lambda$ for all $\eta \in [0, 1]$. The reason why the analysis of the general process is more complicated is that the set of active coordinates (and the hit cost λ^η) can, in principle, change very abruptly between two values of η. Moreover, as stated, equation (14) and Definition 9, have a circular dependency. In particular, the value of $N(\eta)$ depends on the set A^η, but in turn the definition of A^η also depends on the value of $N(\eta)$. As a result, a priori it is not even clear that our algorithm is well defined. That is, a unique trajectory consistent with our local evolutionary rules indeed exists. We proceed to proving this now.

Well-definiteness of the algorithm. We start by addressing the above-mentioned issue of the circular dependency between the value of $N(\eta)$ and the set A^η. Note that it is not clear any more that there always exists a non-negative normalization factor $N(\eta)$ as required by our algorithm. As we prove in the next lemma, however, one can use a simple continuity argument to prove the existence of the desired normalization factor.

Lemma 10. *There exists a $N(\eta) \geq 0$ for which $\sum_{i,j} \frac{dy_{i,j}^\eta}{d\eta} = 0$, where the derivatives $\frac{dy_{i,j}^\eta}{d\eta}$ are as defined in the algorithm. Moreover, the set A^η of active coordinates is never empty.*

Proof. Fix any $\eta \in [0,1]$. Let us consider the function $f(s) = \sum_{i,j}(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s}$, i.e., $f(s)$ is the sum of all derivatives given by equation (9) for the case when $N(\eta)$ is equal to s.

Clearly, if $s = N(\eta) = 0$, then $(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s} \leq 0$ for each (i,j) and hence $f(s) \leq 0$. If $f(0) = 0$ then $N(\eta)$ satisfies the requirements. Note that in this case all coordinates that are non-zero are active. This set is non-empty as at the beginning of the hit stage the sum over all coordinates is at least $kd - \kappa(t) > 0$.

Thus, suppose that $f(0) < 0$. Let $\lambda_{\max} = \max_{i,j} \lambda_{i,j}^{\eta}$ be the largest entry in λ^{η}. Then, at $s = N(\eta) = \alpha\lambda_{\max}$ we have $(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s} \geq 0$ for each (i,j) and hence $f(s) \geq 0$.

Next, we claim that each derivative $(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s}$ is a continuous function of s. To this end, note that if $0 < y_{i,j}^{\eta} < 1$, then the function $(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s}$ is a linear function of s (and thus is continuous). For $y_{i,j}^{\eta} = 0$, the function $(dy_{i,j}^{\eta}/d\eta)|_{N(\eta)=s}$ is zero for $s \leq \alpha\lambda_{i,j}^{\eta}$, and then increases linearly for $s \geq \alpha\lambda_{i,j}^{\eta}$ – so, again, it is continuous. Similarly, for $y_{i,j}^{\eta} = 1$, $(\frac{dy_{i,j}^{\eta}}{d\eta})|_{N(\eta)=s}$ is negative if $s \leq \alpha\lambda_{i,j}^{\eta}$, and increases linearly until $s = \alpha \cdot \lambda_{i,j}^{\eta}$, and then remains zero.

Now, as each derivative is continuous, so is f. Thus, we know that by the intermediate value theorem the preimage $f^{-1}(0)$ in the interval $[0, \alpha\lambda_{\max}]$ is non-empty. Furthermore, as f is continuous and $f(s) < 0$ for $s = 0$, there exists a minimal s^* such that $f(s^*) = 0$ and $0 < s^* \leq \alpha\lambda_{\max}$.

We take $N(\eta) = s^*$ and claim that the corresponding set A^{η} is non-empty (which would prove the lemma). To see why it is the case, note that if there exists a coordinate $0 < y_{i,j}^{\eta} < 1$ then it is always active and we are done. Otherwise, let us consider A_0 (A_1) to be the set of (i,j) with $y_{i,j}^{\eta} = 0$ (resp. with $y_{i,j}^{\eta} = 1$). As the sum over all coordinates is at least $kd - \kappa(t) > 0$, the set A_1 is non-empty. Suppose for $s^* > 0$ all the coordinates in A_0, A_1 are inactive, then by definition for all $(i,j) \in A_1$, $s^* > \alpha\lambda_{i,j}^{\eta}$ (note the strict inequality), and for all coordinates in A_0, $s^* \leq \alpha\lambda_{i,j}^{\eta}$. This is a contradiction to the minimality of s^* as we could find a $0 < s' < s^*$, such that $f(s') = 0$. $\qquad \square$

Now, as our algorithm is defined via a set of differential equations indexed by τ and η, to prove that it is well-defined we need to show that there exists a unique solution to this set, and furthermore this solution is feasible for the allocation problem. To this end, we prove the following lemma, whose proof is in Appendix A.

Lemma 11. *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 $0 \leq \eta \leq 1$: $0 \leq y_{i,j}^{\eta}, y_{i,j}^{\tau} \leq 1$.*

- **Monotonicity:** *For each (i,j), $(j \leq k)$, $y_{i,j}^{\eta} \leq y_{i,j+1}^{\eta}$ and $y_{i,j}^{\tau} \leq y_{i,j+1}^{\tau}$.*

- **Quota:** *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, $\sum_{i,j} y_{i,j}^{\eta} \geq kd - \kappa(t)$ 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 (in fact, polynomial in k and d).*

16

3.2 Cost Accounting

In this section we prove some helpful properties that allow us to charge the algorithm and the optimal solution in a continuous fashion. This will simplify the potential function based analysis that we later perform. First, we deal with the charging of the hit cost, and then with the accounting of the movement cost.

Charging the hit cost. The issue we want to address here is that at a given time t the hit costs of the optimal solution and our algorithm depend only on the final states of both solutions. More precisely, if y^* is the optimal solution at time t, and $y = y^1$ is the final state of the algorithm at time t, then the hit cost of the optimal solution (respectively, of the algorithm) at time t is equal to $\lambda \cdot y^*$ (respectively, $\lambda \cdot y$). However, as our algorithm is described in a continuous fashion, it would be simpler to also have a way of accounting for the hit costs in a continuous and local fashion.

In particular, we would like to account for the hit cost of the optimal solution as:

$$\int_{\eta=0}^{1} \lambda^\eta \cdot y^* \cdot d\eta, \tag{15}$$

and for the hit cost of the algorithm as:

$$\int_{\eta=0}^{1} \lambda^\eta \cdot y^\eta d\eta. \tag{16}$$

Note that the above expressions can be interpreted as charging locally at every time $\eta \in [0,1]$ an infinitesimally small hit cost of $\lambda^\eta \cdot y^* d\eta$ (respectively, $\lambda^\eta \cdot y^\eta d\eta$) to the optimal solution (respectively, to the algorithm). Now, to make this accounting valid, we need to show that the above expressions can only overestimate the hit cost of our algorithm and underestimate the hit cost of the optimal solution. We prove that this is indeed the case in the following lemma.

Lemma 12. *The following inequalities hold:*

$$\int_{\eta=0}^{1} \lambda^\eta \cdot y^* \cdot d\eta \quad \leq \quad \lambda \cdot y^*, \tag{17}$$

$$\int_{\eta=0}^{1} \lambda^\eta \cdot y^\eta d\eta \quad \geq \quad \lambda \cdot y. \tag{18}$$

Proof. We first prove inequality (17). To this end, we show that for any non-decreasing vector $v = (v_1, v_2, \ldots)$, and any $0 \leq \eta_1 < \eta_2 \leq 1$, it holds that

$$\lambda^{\eta_1} \cdot v \geq \lambda^{\eta_2} \cdot v. \tag{19}$$

Note that as $\lambda = \lambda^0$ and y^* is feasible (and thus satisfies property (4)), taking v equal to y^* immediately gives Inequality (17).

By Lemma 11, we know that the only difference between λ^{η_1} and λ^{η_2} is that some of the blocks in $\mathcal{D}^{\eta_1}$ can be merged in $\mathcal{D}^{\eta_2}$. Therefore, it suffices to show that whenever two consecutive blocks B_1 and B_2 merge to form another block B, it holds that $\lambda(B_1)\sum_{i \in B_1} v_i + \lambda(B_2)\sum_{i \in B_2} v_i \geq \lambda(B)\sum_{i \in B} v_i$ for any hit cost vector λ.

17

Let $\ell_1 = |B_1|, \ell_2 = |B_2|$, and let $a_1 = (\sum_{i \in B_1} v_i)/\ell_1$, $a_2 = (\sum_{i \in B_2} v_i)/\ell_2$. Then, by the definition of $\lambda(B)$, the inequality above is equivalent to showing that

$$\lambda(B_1)\ell_1 a_1 + \lambda(B_2)\ell_2 a_2 \geq \left(\frac{\lambda(B_1)\ell_1 + \lambda(B_2)\ell_2}{\ell_1 + \ell_2} \right)(a_1\ell_1 + a_2\ell_2). \tag{20}$$

As v is increasing we have $a_1 \leq a_2$, and since B_1 and B_2 were merged, by (8) it must be that $\lambda(B_1) < \lambda(B_2)$. A direct calculation shows that (20) holds under these conditions.

Now, to prove that inequality (18) also holds, we prove that whenever the derivative of $\lambda^\eta \cdot y^\eta$ is defined, i.e., whenever neither A^η nor λ^η change (which is the case except for possibly finitely many points, cf. Lemma 11) we have that

$$\frac{d\left(\lambda^\eta \cdot y^\eta\right)}{d\eta} \leq 0. \tag{21}$$

That is, the state y^η evolves in a way that reduces the hit cost of the algorithm with respect to the corresponding hit cost vector.

To see how (18) follows from (21) we first note that (21) implies that

$$\lambda^\eta \cdot y^\eta \geq \lambda^1 \cdot y^1,$$

for any $\eta \in [0, 1]$. Now, we have for any block $B \in \mathcal{D}^1$, $y^1_{i,j} = y^1(B)$ for all $j \in B$, and thus

$$\lambda^0 \cdot y^1 = \sum_{B \in \mathcal{D}^1} \sum_{j \in B} \lambda^0_{i,j} y^1_{i,j} = \sum_{B \in \mathcal{D}^1} y^1(B) \sum_{j \in B} \lambda^0_{i,j} = \sum_{B \in \mathcal{D}^1} y^1(B)\lambda^1(B)|B| = \lambda^1 \cdot y^1,$$

where we recall that $\lambda^1(B) = (\sum_{j \in B} \lambda^\eta_{i,j})/|B| = (\sum_{j \in B} \lambda^0_{i,j})/|B|$.

So, we can conclude that

$$\int_{\eta=0}^{1} \lambda^\eta \cdot y^\eta d\eta \geq \int_{\eta=0}^{1} \lambda^1 \cdot y^1 d\eta = \int_{\eta=0}^{1} \lambda^0 \cdot y^1 d\eta = \lambda^0 \cdot y^1 = \lambda \cdot y.$$

In light of the above, it remains to prove (21). To this end, recall that when A^η and λ^η are fixed, the evolution of $y^\eta_{i,j}$s is described by Equation (13), thus the statement we need to prove is

$$\sum_{(i,j) \in A^\eta} \frac{\lambda^\eta_{i,j}}{w_i}\left(y^\eta_{i,j} + \beta\right) \cdot \left(N(\eta) - \alpha\lambda^\eta_{i,j}\right) \leq 0. \tag{22}$$

Plugging in the expression for $N(\eta)$ given by (14) and canceling α, we need to show that

$$\left(\sum_{(i,j) \in A^\eta} \frac{\lambda^\eta_{i,j}}{w_i}\left(y^\eta_{i,j} + \beta\right) \right)^2 \leq \left(\sum_{(i,j) \in A^\eta} \frac{(\lambda^\eta_{i,j})^2}{w_i}\left(y^\eta_{i,j} + \beta\right) \right) \cdot \left(\sum_{(i,j) \in A^\eta} \frac{1}{w_i}\left(y^\eta_{i,j} + \beta\right) \right).$$

Now, this inequality follows from the Cauchy-Schwarz inequality $(a \cdot b)^2 \leq |a|_2^2 |b|_2^2$, by taking a to be the vector with entries $a_{i,j} = \sqrt{\frac{(\lambda^\eta_{i,j})^2}{w_i}\left(y^\eta_{i,j} + \beta\right)}$, and b to be the vector with entries $b_{i,j} = \sqrt{\frac{1}{w_i}\left(y^\eta_{i,j} + \beta\right)}$. □

Charging the movement cost. We turn our attention to the accounting of the movement cost of our algorithm. Recall that the movement cost at time t is defined as

$$\sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} |y_{i,j}^t - y_{i,j}^{t-1}| \right). \tag{23}$$

We would like to approximate this expression by one which is simpler and more convenient to work with. First, instead of keeping track of both increases and decreases of the variables $y_{i,j}^t$ as in the above, we will account for the movement cost only through the increases of the variables $y_{i,j}^t$. That is, our bound for the movement cost is

$$\sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \max\{y_{i,j}^t - y_{i,j}^{t-1}, 0\} \right). \tag{24}$$

Note that we have for any coordinate (i, j) and $t \geq 1$

$$|y_{i,j}^t - y_{i,j}^{t-1}| \leq 2 \cdot \max\{y_{i,j}^t - y_{i,j}^{t-1}, 0\} + y_{i,j}^{t-1} - y_{i,j}^t,$$

and thus on any input sequence consisting of T requests, it is the case that

$$\sum_{t=1}^{T} \sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} |y_{i,j}^t - y_{i,j}^{t-1}| \right) \leq 2 \cdot \sum_{t=1}^{T} \sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \max\{y_{i,j}^t - y_{i,j}^{t-1}, 0\} \right) + \sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} y_{i,j}^0 - y_{i,j}^T \right).$$

Thus, accounting for the movement cost via expression (24) approximates the true movement cost (corresponding to expression (23)) up to a multiplicative factor of two and an additive factor of at most $\sum_{i=1}^{d} k w_i$, which depends only on the starting and final configuration, and is zero if the two configurations coincide. As we will see, for the sake of our competitive analysis, such an approximation suffices.

Next, similarly to the way we accounted for the hit cost described above, we wish to further simplify the charging of the movement cost and perform it in a continuous and local fashion. Namely, it is easy to see that the following quantity

$$\sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \int_{\tau=0}^{1} \frac{dy_{i,j}^\tau}{d\tau} d\tau + \int_{\eta=0}^{1} \max\{\frac{dy_{i,j}^\eta}{d\eta}, 0\} d\eta \right)$$

can only overestimate the movement cost given by (24). (Note that $\frac{dy_{i,j}^\tau}{d\tau}$ is always non-negative.) Furthermore, as the derivatives $\frac{dy_{i,j}^\eta}{d\eta}$ can only be positive if $(i, j) \in A^\eta$, we can write

$$\int_{\eta=0}^{1} \max\left\{ \frac{dy_{i,j}^\eta}{d\eta}, 0 \right\} d\eta \leq \int_{\eta=0}^{1} \frac{1}{w_i} \left(y_{i,j}^\eta + \beta \right) N(\eta) \cdot \mathbf{1}_{(i,j) \in A^\eta} d\eta,$$

where we used (13) and the fact that, trivially, $N(\eta) - \lambda_{i,j}^\eta \leq N(\eta)$. Thus, we can use the above in our final version of the estimate of the movement cost. The following claim summarizes the discussion.

19

Claim 13. *For any sequence of requests of length $T \geq 1$,*

$$\sum_{t=1}^{T} \sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} |y_{i,j}^{t} - y_{i,j}^{t-1}| \right) \leq$$

$$2 \cdot \sum_{t=1}^{T} \sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \int_{\tau=0}^{1} \frac{dy_{i,j}^{\tau}}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{1}{w_i} \left(y_{i,j}^{\eta} + \beta \right) N(\eta) \cdot \mathbf{1}_{(i,j) \in A^{\eta}} d\eta \right) + C',$$

where $C' \leq \sum_i k w_i$ depends only on the starting and final configuration of the algorithm, and $C' = 0$ if the two configurations coincide.

Thus, at time t, the movement cost of our algorithm is given by:

$$\sum_{i=1}^{d} w_i \left(\sum_{j=1}^{k} \int_{\tau=0}^{1} \frac{dy_{i,j}^{\tau}}{d\tau} d\tau + \int_{\eta=0}^{1} \frac{1}{w_i} \left(y_{i,j}^{\eta} + \beta \right) N(\eta) \cdot \mathbf{1}_{(i,j) \in A^{\eta}} d\eta \right). \tag{25}$$

3.3   Competitive Analysis

We are finally ready to bound the competitiveness of our algorithm. To this end, we prove the following theorem.

Theorem 14. *Consider an arbitrary instance of the 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 the following bounds:*

$$\begin{aligned}
\mathrm{H} &\leq (1 + \varepsilon) \left(\mathrm{Opt} + w_{\max} \cdot g(k) \right) + C, \\
\mathrm{M} &\leq O(\log(k/\varepsilon)) \cdot \left(\mathrm{Opt} + w_{\max} \cdot g(\kappa) \right).
\end{aligned}$$

Here, H and M denote the hit and movement costs of our fractional algorithm, and Opt denotes the sum of the total hit and movement 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 $C = 0$ if the two configurations coincide.

It is easy to see that Theorem 5 immediately follows from the above theorem.

To prove Theorem 14 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.

$$\mathrm{M}_t + \Delta\Phi_t^m \leq (1 + \varepsilon) \cdot \alpha \cdot \left(w_{\max} \cdot |\kappa(t) - \kappa(t-1)| + \mathrm{M}^*_t + \mathrm{H}^*_t \right), \tag{26}$$

$$\mathrm{H}_t + \Delta\Phi_t^h + \frac{1}{\alpha} \Delta\Phi_t^m \leq (1 + \varepsilon) \left(w_{\max} \cdot |\kappa(t) - \kappa(t-1)| + \mathrm{M}^*_t + \mathrm{H}^*_t \right). \tag{27}$$

Here, H_t (respectively H^*_t) and M_t (respectively M^*_t) denote the hit and movement costs incurred by the algorithm (respectively optimum) at time t. The quantities

$$\Delta\Phi_t^h := \Phi^h(\overline{y}^t, t) - \Phi^h(\overline{y}^{t-1}, t-1), \qquad \Delta\Phi_t^m := \Phi^m(\overline{y}^t, t) - \Phi^m(\overline{y}^{t-1}, t-1)$$

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

As we shall see, 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 14 will follow by summing up (26) and (27) over all times t. (Note that in our algorithm $\alpha = \log(1 + 1/\beta) = O(\log(k/\varepsilon))$.)

The potential functions. The potential function Φ^m is defined as follows.

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

Here, $\overline{y}^{*t}$ denotes the configuration of the optimum solution at time t. Note that if the configuration $\overline{y}^{*t}$ is integral and 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

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

So, 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), and 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

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

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

The proof plan. Our goal now is to show that Inequalities (26) and (27) always hold. For ease of analysis, we will consider the events at time t in three steps, and show that Inequalities (26) and (27) hold at each of these steps. The steps are the following:

1. The quota $\kappa(t)$ either increases, decreases, or stays unchanged, compared to $\kappa(t-1)$, and the optimal solution removes/adds servers accordingly.

2. The optimal solution moves some servers and its state changes from $\overline{y}^{*t-1}$ to $\overline{y}^{*t}$.

3. The online algorithm changes its state from $\overline{y}^{t-1}$ to $\overline{y}^t$: first, we analyze the fix stage then we analyze the hit stage. Also, while analyzing the hit stage, the hit costs of both the online algorithm and the optimal solution are accounted for.

The server quota increases/decreases and the optimum removes/adds servers. Note that the only quantities that can change in this step are the movement cost of the optimum and the potential Φ^m, thus it suffices to prove that Inequality (26) is preserved (in this case, (27) is identical to (26) scaled by $1/\alpha$). Now, there are two cases to consider, depending on how $\kappa(t)$ changes.

- Suppose $\kappa(t) < \kappa(t-1)$, then the optimum has to withdraw $|\kappa(t)-\kappa(t-1)|$ servers from some locations. That is, for $|\kappa(t)-\kappa(t-1)|$ locations (i,j), the corresponding variables $y_{i,j}^{*(t-1)}$ are set to 1, and as a result these locations start contributing to Φ^m. Clearly, each such location (i,j) increases Φ^m by at most

$$(1+\varepsilon)w_i \cdot \log\left(\frac{1+\beta}{y_{i,j}^{t-1}+\beta}\right) \le (1+\varepsilon)\ln\left(1+\frac{1}{\beta}\right) \cdot w_{\max} = (1+\varepsilon)\alpha \cdot w_{\max}.$$

  As a result, the total increase of Φ^m is at most $w_{\max} \cdot (1+\varepsilon)\alpha \cdot |\kappa(t)-\kappa(t-1)|$, and hence (26) holds even without accounting for the movement cost of the optimum.

- Suppose $\kappa(t) \ge \kappa(t-1)$, then the optimum can bring in $|\kappa(t)-\kappa(t-1)|$ servers, and as a result Φ^m can only decrease, as some terms $y_{i,j}^{*(t-1)}$ may change from 1 to 0, thus not contributing anymore to Φ^m. Hence, (26) also holds in this case.

The optimum moves its servers. Without loss of generality, it suffices to analyze the case in which the optimum moves exactly one server from location i to i'. (If multiple servers are moved, we can consider these moves one by one.) Also, as before, only Φ^m and the offline movement cost change, and hence it suffices to just show that (26) holds, and in particular that

$$\Delta\Phi_t^m \le (1+\varepsilon)\alpha M^*{}_t.$$

Suppose that location i had j servers prior to the movementand this number is reduced to $j-1$ (recall that by our convention we account only for the movement cost corresponding to withdrawal of servers). Then, the contribution of location i to Φ^m increases by precisely

$$w_i(1+\varepsilon) \cdot \log\left(\frac{1+\beta}{y_{i,j}^{t-1}+\beta}\right) \le w_i(1+\varepsilon) \cdot \alpha = (1+\varepsilon)\alpha M^*{}_t.$$

In contrast, increasing the number of servers at i' can only decrease Φ^m. Thus, we get that the desired inequalities hold in this case.

The online algorithm is executed. The case in which the online algorithm changes its distribution $\overline{y}^t$ is the most interesting, and we analyze Inequalities (26) and (27) separately. Recall that our online algorithm works in two steps: the fix stage and the hit stage, and hence we consider these separately. Moreover, as the evolution of the algorithm is described in a continuous manner, we will analyze Inequalities (26) and (27) in such a manner too.

The fix stage: proof of Inequality (26). To show that Inequality (26) holds during the fix stage it suffices to prove that for any τ,

$$\frac{dM_t}{d\tau} + \frac{d\Phi^m}{d\tau} \leq 0. \tag{28}$$

By definition of $\frac{dy_{i,j}^\tau}{d\tau}$ during the fix stage and our way of accounting for the movementcost (cf. Claim 13) we have

$$\frac{dM_t}{d\tau} = \sum_{(i,j)\in A^\tau} w_i \cdot \frac{1}{w_i} \left(y_{i,j}^\tau + \beta\right) = \sum_{(i,j)\in A^\tau} \left(y_{i,j}^\tau + \beta\right).$$

(Recall that A^τ is the set of active coordinates (i,j), i.e. those for which $y_{i,j} < 1$.) Also, it is easy to see that the change in the potential function Φ^m is

$$\frac{d\Phi^m}{d\tau} = -(1+\varepsilon) \sum_{(i,j)\in A^\tau} y_{i,j}^{*t}.$$

Next we need to prove the following claim

Claim 15. *Consider a subset A of the coordinates and two configurations $\overline{y}$ and $\overline{y}'$ with $\sum_{(i,j)\in A} y_{i,j}' \geq 1$, and $\sum_{i,j} y_{i,j}' \geq kd - k$, we have then that*

$$\sum_{(i,j)\in A} (y_{i,j} + \beta) - (1+\varepsilon) \sum_{(i,j)\in A} y_{i,j}' \leq \sum_{(i,j)\in A} y_{i,j} - \sum_{(i,j)\in A} y_{i,j}'.$$

Before we prove this claim, let us describe how (28) follows from it. If we set $\overline{y} = \overline{y}^\tau$, $\overline{y}' = \overline{y}^{*t}$, and $A = A^\tau$, then, clearly, $\sum_{i,j} y_{i,j}^{*t} = kd - \kappa(t) \geq kd - k$. Furthermore, since $(i,j) \notin A^\tau$ only if $y_{i,j}^\tau = 1$, and as we apply the fix stage only if $\sum_{i,j} y_{i,j}^\tau < kd - \kappa(t)$, we need to have $|A^\tau| > \kappa(t)$, and thus

$$\sum_{(i,j)\in A} y_{i,j}^{*t} \geq kd - \kappa(t) - |\overline{A^\tau}| \geq 1.$$

So, both requirements of the claim are satisfied and it follows that

$$\frac{dM_t}{d\tau} + \frac{d\Phi^m}{d\tau} = \sum_{(i,j)\in A^\tau} (y_{i,j}^\tau + \beta) - (1+\varepsilon) \sum_{(i,j)\in A^\tau} y_{i,j}^{*t} \leq \sum_{(i,j)\in A^\tau} y_{i,j}^\tau - \sum_{(i,j)\in A^\tau} y_{i,j}^{*t} \leq 0, \tag{29}$$

where the last inequality follows, as $\sum_{i,j} y_{i,j}^\tau < kd - \kappa(t) = \sum_{i,j} y_{i,j}^{*t}$, and $(i,j) \notin A^\tau$ only if $y_{i,j}^\tau = 1$ implies that $\sum_{(i,j)\in A^\tau} y_{i,j}^\tau \leq \sum_{(i,j)\in A^\tau} y_{i,j}^{*t}$.

Proof of Claim 15. As $\sum_{i,j} y_{i,j}' \geq kd - k$ and $\sum_{(i,j)\in A} y_{i,j}' \geq 1$, we have that

$$\frac{|A|}{k+1} - \sum_{(i,j)\in A} y_{i,j}' \leq \frac{|A|}{k+1} - \max\{|A| - k, 1\} \leq 0, \tag{30}$$

where the last inequality follows as $\frac{|A|}{k+1} < 1$ when $|A| \leq k$, and $\frac{|A|}{k+1} \leq |A| - k$ when $|A| \geq k+1$.

Now, the claim is proved by noticing that

$$
\sum_{(i,j)\in A} (y_{i,j} + \beta) - (1+\varepsilon) \sum_{(i,j)\in A} y'_{i,j} = \varepsilon \left(\frac{|A|}{k+1} - \sum_{(i,j)\in A} y'_{i,j} \right) + \left(\sum_{(i,j)\in A} y_{i,j} - \sum_{(i,j)\in A} y'_{i,j} \right)
$$
$$
\leq \sum_{(i,j)\in A} y_{i,j} - \sum_{(i,j)\in A} y'_{i,j},
$$

where the inequality follows by (30). $\qquad\square$

The fix stage, proof of Inequality (27)**.** To prove that Inequality (27) holds during the fix stage as well, we note that as we are accounting for the hit cost during the hit stage, currently there is no hit cost incurred, and hence we just need to show that

$$
\frac{d\Phi^h}{d\tau} + \frac{1}{\alpha} \frac{d\Phi^m}{d\tau} = 0. \tag{31}
$$

Observe that

$$
\frac{d\Phi^h}{d\tau} = \frac{1}{\alpha} \sum_{(i,j)\in A^\tau} w_i \frac{dy_{i,j}}{d\tau}
$$

which is exactly $1/\alpha$ times our accounting for the movement cost in the fix stage (cf. Claim 13). Therefore, (31) is identical to (29) (up to a scaling by $1/\alpha$), and hence follows from the above proof.

The hit stage, proof of Inequality (27)**.** Recall that both the optimal solution and the online algorithm incur a hit cost in this stage. We start by proving that Inequality (27) holds during this stage – later, we will analyze Inequality (26). We need to show that

$$
\frac{dH_t}{d\eta} + \frac{d\Phi_t^h}{d\eta} + \frac{1}{\alpha} \frac{d\Phi_t^m}{d\eta} \leq (1+\varepsilon) \frac{dH_t^*}{d\eta}. \tag{32}
$$

First, we note that by our way of accounting for the hit cost of the algorithm (cf. (16))

$$
\frac{dH_t}{d\eta} = \sum_{i,j} \lambda_{i,j}^\eta y_{i,j}^\eta \qquad \text{and} \qquad \frac{d\Phi_t^h}{d\eta} = \frac{1}{\alpha} \sum_{(i,j)\in A^\eta} \left(y_{i,j}^\eta + \beta \right) \cdot \left(N(\eta) - \alpha \lambda_{i,j}^\eta \right).
$$

Now, we have that

$$
\frac{dH_t}{d\eta} + \frac{d\Phi_t^h}{d\eta} = \sum_{i,j} \lambda_{i,j}^\eta y_{i,j}^\eta + \frac{1}{\alpha} \sum_{(i,j)\in A^\eta} \left(y_{i,j}^\eta + \beta \right) \cdot \left(N(\eta) - \alpha \lambda_{i,j}^\eta \right)
$$
$$
= \sum_{(i,j)\notin A^\eta} \lambda_{i,j}^\eta y_{i,j}^\eta + \frac{1}{\alpha} \sum_{(i,j)\in A^\eta} (y_{i,j}^\eta + \beta) \cdot N(\eta) - \sum_{(i,j)\in A^\eta} \beta \lambda_{i,j}^\eta
$$
$$
\leq \sum_{(i,j)\notin A^\eta} \lambda_{i,j}^\eta y_{i,j}^\eta + \frac{1}{\alpha} \sum_{(i,j)\in A^\eta} (y_{i,j}^\eta + \beta) \cdot N(\eta) = \sum_{(i,j)\notin A^\eta} \lambda_{i,j}^\eta y_{i,j}^\eta + \frac{1}{\alpha} \frac{dM_t}{d\eta},
$$

where in the last step we used the expression for our movement cost accounting (cf. Claim 13),

$$\frac{dM_t}{d\eta} = \sum_{(i,j)\in A^\eta} \left(y_{i,j}^\eta + \beta\right)\cdot N(\eta). \tag{33}$$

Thus, to establish (32), it suffices to show that

$$\sum_{(i,j)\notin A^\eta} \lambda_{i,j}^\eta y_{i,j}^\eta + \frac{1}{\alpha}\frac{dM_t}{d\eta} + \frac{1}{\alpha}\frac{d\Phi_t^m}{d\eta} \le (1+\varepsilon)\frac{dH_t^*}{d\eta}. \tag{34}$$

By our update rule for $y_{i,j}^\eta$, the derivative of Φ_t^m is

$$\frac{d\Phi_t^m}{d\eta} = -(1+\varepsilon)\sum_{(i,j)\in A^\eta} y_{i,j}^{*t}\cdot\left(N(\eta) - \alpha\lambda_{i,j}^\eta\right).$$

Finally, by our way of accounting for the hit cost of the optimal solution (cf. (15)) we have

$$\frac{dH^{*t}}{d\eta} = \sum_{i,j}\lambda_{i,j}^\eta y_{i,j}^{*t}.$$

Thus, after multiplying Inequality (34) by α and plugging in the above equalities, we need to prove that

$$\alpha\sum_{(i,j)\notin A^\eta}\lambda_{i,j}^\eta y_{i,j}^\eta + \sum_{(i,j)\in A^\eta}\left(y_{i,j}^\eta + \beta\right)\cdot N(\eta) - (1+\varepsilon)\sum_{(i,j)\in A^\eta}y_{i,j}^{*t}\cdot\left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) - (1+\varepsilon)\alpha\sum_{i,j}\lambda_{i,j}^\eta y_{i,j}^{*t} \le 0. \tag{35}$$

Let A' be the set of all coordinates that are either active or have $y_{i,j}^\eta = 0$. As any inactive coordinate has $y_{i,j}^\eta \in \{0,1\}$, we observe that:

- If $(i,j) \in A' \setminus A^\eta$, then it must be that $y_{i,j}^\eta = 0$ and $\alpha\lambda_{i,j}^\eta \ge N(\eta)$. This holds, as a coordinate for which $y_{i,j}^\eta = 0$ can be inactive only if $\frac{dy_{i,j}^\eta}{d\eta} = (y_{i,j}^\eta + \beta)(N(\eta) - \alpha\lambda) \le 0$.

- If $(i,j) \notin A'$ it must be that $y_{i,j}^\eta = 1$ and $\alpha\lambda_{i,j}^\eta < N(\eta)$. This holds, as a coordinate for which $y_{i,j}^\eta = 1$ can be inactive only if $\frac{dy_{i,j}^\eta}{d\eta} = (y_{i,j}^\eta + \beta)(N(\eta) - \alpha\lambda) > 0$.

Thus, using the above observations, we may rewrite (35) and get that

$$N(\eta)\cdot\left(\sum_{(i,j)\in A^\eta}\left(y_{i,j}^\eta + \beta\right) - (1+\varepsilon)\sum_{(i,j)\in A^\eta}y_{i,j}^{*t}\right) - (1+\varepsilon)\alpha\sum_{(i,j)\notin A^\eta}\lambda_{i,j}^\eta y_{i,j}^{*t} + \alpha\sum_{(i,j)\notin A^\eta}\lambda_{i,j}^\eta y_{i,j}^\eta$$

$$\le N(\eta)\cdot\left(\sum_{(i,j)\in A'}\left(y_{i,j}^\eta + \beta\right) - (1+\varepsilon)\sum_{(i,j)\in A'}y_{i,j}^{*t}\right) - (1+\varepsilon)\alpha\sum_{(i,j)\notin A'}\lambda_{i,j}^\eta y_{i,j}^{*t} + \alpha\sum_{(i,j)\notin A^\eta}\lambda_{i,j}^\eta y_{i,j}^\eta$$

$$\le N(\eta)\cdot\left(\sum_{(i,j)\in A'}\left(y_{i,j}^\eta + \beta\right) - (1+\varepsilon)\sum_{(i,j)\in A'}y_{i,j}^{*t}\right) + \alpha\sum_{(i,j)\notin A'}\lambda_{i,j}^\eta\left(y_{i,j}^\eta - y_{i,j}^{*t}\right)$$

$$\le N(\eta)\cdot\left(\sum_{(i,j)\in A'}\left(y_{i,j}^\eta + \beta\right) - (1+\varepsilon)\sum_{(i,j)\in A'}y_{i,j}^{*t} + \sum_{(i,j)\notin A'}\left(y_{i,j}^\eta - y_{i,j}^{*t}\right)\right).$$

The first inequality follows as $(i,j) \in A' \setminus A^\eta$, then it must be that $\alpha\lambda_{i,j}^\eta \geq N(\eta)$, and because $A^\eta \subseteq A'$. The second inequality follows since for all $(i,j) \in A' \setminus A^\eta$ it must be that $y_{i,j}^\eta = 0$. The third inequality follows as $\alpha\lambda_{i,j}^\eta < N(\eta)$ for $(i,j) \notin A'$, and by the observation that for each $(i,j) \notin A'$, $y_{i,j}^\eta = 1 \geq y_{i,j}^{*t}$.

If $N(\eta) = 0$, then the above expression equals 0, and we are done. Otherwise, if $N(\eta) > 0$, we get that

$$\sum_{(i,j)\in A'} \left(y_{i,j}^\eta + \beta \right) - (1 + \varepsilon) \sum_{(i,j)\in A'} y_{i,j}^{*t} + \sum_{(i,j)\notin A'} \left(y_{i,j}^\eta - y_{i,j}^{*t} \right)$$

$$\leq \sum_{(i,j)\in A'} y_{i,j}^\eta - \sum_{(i,j)\in A'} y_{i,j}^{*t} + \sum_{(i,j)\notin A'} \left(y_{i,j}^\eta - y_{i,j}^{*t} \right) \tag{36}$$

$$= \sum_{(i,j)} y_{i,j}^\eta - \sum_{(i,j)} y_{i,j}^{*t} = 0. \tag{37}$$

Inequality (36) follows from Claim 15 by the following arguments. ¿From Lemma 10 we know that the set A^η is non-empty. We claim that this implies that $\sum_{(i,j)\in A'} y_{i,j}^\eta$ has to be positive. Otherwise, all the active coordinates would have $y_{i,j}^\eta = 0$, and thus could only increase, contradicting the fact that when $N(\eta) > 0$, $\sum_{i,j} \frac{dy_{i,j}^\eta}{d\eta} = \sum_{(i,j)\in A^\eta} \frac{dy_{i,j}^\eta}{d\eta} = 0$. Moreover, as $N(\eta) > 0$, $\sum_{i,j} y_{i,j}^\eta$ is equal to $\sum_{i,j} y^{*t} = kd - \kappa(t)$ (and thus is integral). Now, as $y_{i,j}^\eta = 1$ if $(i,j) \notin A'$, we have that $\sum_{(i,j)\in A'} y_{i,j}^{*t} \geq \sum_{(i,j)\in A'} y_{i,j}^\eta$ and that $\sum_{(i,j)\in A'} y_{i,j}^\eta$ is integral as well. As this quantity is positive by the argument above, we get that

$$\sum_{(i,j)\in A'} y_{i,j}^{*t} \geq \sum_{(i,j)\in A'} y_{i,j}^\eta \geq 1. \tag{38}$$

Thus, we can use Claim 15 with $\bar{y} = \bar{y}^\eta$, $\bar{y}' = \bar{y}^{*t}$ and $A = A'$ (as all the requirements of this claim are satisfied). The last equality follows, since for $N(\eta) > 0$, we get from the algorithm, $\sum_{i,j} y_{i,j}^\eta = kd - \kappa(t) = \sum_{i,j} y_{i,j}^{*t}$.

The hit stage, proof of Inequality (26). To show that (26) holds, we need to show that

$$\frac{dM_t}{d\eta} + \frac{d\Phi_t^m}{d\eta} \leq (1 + \varepsilon)\alpha \cdot \frac{dH^{*t}}{d\eta}. \tag{39}$$

However, this follows directly by noting that the above is simply Inequality (34) after removing the first (non-negative) term and scaling by α.

4 Fractional k-server on Weighted HSTs

In this section, we show how the fractional allocation algorithm on a weighted star can be used as a building block to obtain a fractional k-server solution on a weighted HST. In particular, we prove the following.

Theorem 6. *Let T be a weighted σ-HST with depth ℓ. If, for any $0 \leq \varepsilon \leq 1$, 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 $\sigma = \Omega(\ell \log(k\ell))$.

To this end, we focus on a particular weighted σ-HST T and show how to construct a fractional k-server algorithm on it. Roughly speaking, the construction works as follows. Each internal node p of T will run a number of instances of the allocation problem which differ with respect to their quota patterns, but have the same hit cost vectors. These instances are maintained as a convex combination. The fractional solutions to the different instances, which we compute online using the fractional allocation algorithm, determine in a recursive manner how the servers available at each node are distributed among its children.

While this approach is similar to the approach of Coté et al. [16], the main difference here is that we can use the (much weaker) fractional allocation problem instead of using a randomized (integral) algorithm for the allocation problem.

Let us denote by r the root of our σ-HST T. Recall that for a node p of T, $T(p)$ denotes the subtree rooted at p, $W(p)$ is the length of the edge connecting p to its parent, and $w(p, i)$ denotes the length of the edge connecting p to its child p_i. By the definition of a weighted σ-HST, we have $W(p) \geq \sigma w(p, i)$ for all children i of p, unless p is either a leaf or the root.

Recall that the input to the fractional allocation problem running at node p consists of the quota pattern $\kappa = (\kappa(1), \kappa(2), \ldots)$ specifying the number of servers $\kappa(t)$ available at each time t, and the hit cost vectors h^t that arrive at each time t at location i^t. The output of an algorithm for the fractional allocation problem specifies a fractional solution $\bar{x}^t$ that provides a distribution on the number of servers at each location p_i, subject to the aggregate bound of $\kappa(t)$ on the (expected) number of servers.

Now, let us fix some instance ρ of the k-server problem on the leaves of T. Let $\rho = (\rho(1), \rho(2), \ldots)$ be the request sequence, where $\rho(t)$ denotes the leaf requested at time t.

Definition 16. *For a node p, integer j and time t, let $\mathrm{Optcost}(p, j \cdot \vec{1}, t)$ be the optimum cost for serving the k-server instance $\{\rho(1), \ldots, \rho(t)\} \cap T(p)$, i.e. the request sequence ρ restricted to the leaves of $T(p)$ until time t, subject to the constraint that exactly j servers are available.*

Remark: Optcost is well defined only with respect to an initial configuration, which we will always assume to be the initial starting positions of the servers at $t = 0$. Also, we use the notation $\mathrm{Optcost}(p, j \cdot \vec{1}, t)$, instead of just $\mathrm{Optcost}(p, j, t)$, as we will later extend the definition of Optcost to the case in which j can vary with time. For now, we only consider fixed j.

4.1 The algorithm

In this section we define the ensemble of allocation problems that will be running at each of the internal nodes of T. To do this, we have to define how the hit-cost vectors and the quota patterns are generated. Consider some internal node p. As mentioned earlier, each internal node p will run several instances of the allocation problem that are different with respect to their quota pattern. It will also hold a convex combination over these instances. All instances will have the same hit cost vector that will be defined later. We denote the convex combination over allocation instances on node p at time t by Λ_p^t. Λ_p^t is specified via the collection

$$\Lambda_p^t = \{(\lambda_{p,s}^t, \kappa_{p,s}^t, H_p^t)\}_s, \qquad \forall t, p, \sum_s \lambda_{p,s}^t = 1$$

Here $\lambda_{p,s}^t$ determines the fraction at time t given to the instance with quota pattern $\kappa_{p,s}^t$ (until time t), and $H_p^t = \{h_p^1, h_p^2, \ldots, h_p^t\}$ is the sequence of hit cost vectors that have appeared until time t. As we will see shortly, the hit cost vector will be the same for all instances $s \in \Lambda_p^t$ and therefore there is no subscript s to the hit cost. We will use the notation $s \in \Lambda_p^t$ to index the triples $(\lambda_{p,s}^t, \kappa_{p,s}^t, H_p^t)$ in Λ_p^t. As we shall see later, the convex combination Λ_p^t will be a refinement of the convex combination Λ_p^{t-1}, for every t.

To complete the description of the fractional k-server algorithm we need to define Λ_p^t for each node p and time t, and show how the fractional number of servers at the leaves of T is computed. We begin with defining how the hit costs h_p^t are generated for each node p.

Hit costs: Consider any internal node p. Let $p_1, \ldots, p_d$ be the children of p. For the allocation problems running at p, at time t we give the hit cost vector

$$h_{p_i}^t(j) = \text{Optcost}(p_i, j \cdot \vec{1}, t) - \text{Optcost}(p_i, j \cdot \vec{1}, t - 1).$$

As Coté et al. [16] prove, the cost vectors $h_{p_i}^t$ have the desired monotonicity property, i.e., $h_{p_i}^t(0) \geq h_{p_i}^t(1) \geq \ldots \geq h_{p_i}^t(k)$ for each i and time t. The following crucial observation follows directly from the definition of the k-server problem.

Observation 17. *Consider subtree $T(p)$ and request $\rho(t)$. If $\rho(t) \in T(p_i)$, then*

1. *$h_p^t(i, 0) = \infty$. (This follows since any 0-server solution is infeasible for any instance with one or more requests, or equivalently incurs infinite cost.).*

2. *$h_p^t(i', j) = 0$ for all $i' \neq i$ and for all j. (This follows simply since the request is not in the sub-tree of $p_{i'}$ for $i' \neq i$.).*

This completes the description of the cost vectors of node p. We next define the quota patterns $\kappa_{p,s}(t)$ for the various allocation instances running at node p.

Quota patterns: The quota patterns are determined recursively in a top down manner over the tree T (and inductively over time) by the fractional solutions of the allocation instances that are generated at each node. To specify how these patterns evolve, we describe below a procedure for updating both the quota patterns $\kappa_{p,s}^t$ and the convex combination $\lambda_{p,s}^t$, associated with the allocation instances maintained.

Base case:

1. At the root r of the tree T there is a single allocation instance running with a quota of k at all times. That is, Λ_r^t consists of a single allocation instance (with fraction 1), hit costs as described above, and $\kappa = k \cdot \vec{1}$.

2. For any internal node $p \in T$ and time $t = 0$, Λ_p^0 consists of a single allocation instance (with fraction 1). The quota pattern $\kappa_{p,s}$ for this single instance s, until time $t = 0$, is simply the number of servers present initially at the leaves of subtree $T(p)$. Moreover, there is no hit cost thus far.

The inductive step: Consider time t. We describe the procedure to obtain Λ_p^t from Λ_p^{t-1} in a top down manner on the tree as follows. As the base case, recall that Λ_r^t has already been determined for all t. Arguing inductively top down on the tree, suppose that Λ_p^t has already been determined. Then, for the children $p_1, \ldots, p_d$ of p, we determine $\Lambda_{p_i}^t$ as follows.

Consider the allocation instances that are executed at node p. Let $\{x_{i,j,s}^t\}_{i,j,s}$ be the fractional solutions generated (by the allocation instances) at time t. The algorithm will maintain the following consistency between the quota for servers available at p_i and what the allocation problems running at the parent p determine. In particular,

$$\text{(Consistency)} \qquad \sum_{s \in \Lambda_{p_i}^t \;|\; \kappa_s^t(t)=j} \lambda_s^t = \sum_{s \in \Lambda_p^t} \lambda_s^t x_{i,j,s} \triangleq x_{i,j}^t, \tag{40}$$

Also, for each child p_i it should maintain

$$\text{(Convex combination)} \qquad \sum_{s \in \Lambda_{p_i}^t} \lambda_s^t = 1. \tag{41}$$

Suppose that $x_{i,j}^{t-1}$ changes to $x_{i,j}^t$ due to the execution of the allocation instances $s \in \Lambda_p^t$ at time t. We show how to update $\Lambda_{p_i}^t$ from $\Lambda_{p_i}^{t-1}$ such that it remains consistent with (40) and (41). This update will be done in a natural (and cost-efficient) way.

Consider first the cost paid by the convex combination of the allocation instances running at node p. The cost is

$$\sum_s \sum_i w(p,i) \lambda_s^t \sum_{j=1}^k \left| \sum_{\ell<j} \left(x_{i,\ell,s}^t - x_{i,\ell,s}^{t-1} \right) \right| \geq \sum_i w(p,i) \sum_{j=1}^k \left| \sum_s \lambda_s^t \sum_{\ell<j} \left(x_{i,\ell,s}^t - x_{i,\ell,s}^{t-1} \right) \right|, \tag{42}$$

where the inequality follows, since for any non-negative numbers p_i, $\sum_i p_i|a_i| \geq |\sum_i p_i a_i|$.

We note that the change from $x_{i,j}^{t-1}$ to $x_{i,j}^t$ can be decomposed into a collection of elementary moves in which $\pm\delta(i,j)$ units of mass are removed from $x_{i,j}$ and put on $x_{i,j\pm1}$, such that the total fractional movement cost remains the same. Thus, we can assume without loss of generality that $x_{i,j}^t$ and $x_{i,j}^{t-1}$ differ by an elementary move.

Consider an elementary move where $x_{i,j}^t = x_{i,j}^{t-1} - \delta$ and $x_{i,j-1}^t = x_{i,j-1}^t + \delta$ (all other types of elementary moves are handled analogously). To implement this, we choose an arbitrary δ measure of allocation problems $s \in \Lambda_{p_i}^{t-1}$ with $\kappa_s(t-1) = j$ and set $\kappa_s(t)$ to $j-1$. For all other κ_s, we set $\kappa_s(t) = \kappa_s(t-1)$. After all entries are updated by applying the elementary moves, $\kappa_s(t)$ is determined.

It is clear that this update rule maintains both (40) and (41). This completes the procedure for obtaining Λ_i^t from Λ_i^{t-1}.

Obtaining the fractional k-server solution: To complete the description of our algorithm we should describe how to determine the fractional number of servers at each leaf q at each time t. This is determined in a natural way using the following observation. Consider a leaf q and let p be its parent. Then,

$$z(q,t) := \sum_{s \in \Lambda_p^t} \lambda_s^t \sum_j j \cdot x_{q,j,s}^t$$

29

is the number of servers at q at time t. Here, $x^t_{q,j,s}$ is the probability of having j servers at q at time t, when the fractional allocation algorithm is applied to the allocation instance $s \in \Lambda^t_p$.

4.2 Feasibility

We first note that our fractional k-server solution is feasible since it satisfies the following.

Lemma 18. *Whenever there is a k-server request $\rho(t)$, then there is at least one server unit at the location $\rho(t)$, i.e. $z_{\rho(t),t} \geq 1$. This holds provided the total cost incurred by the allocation problems is finite.*

Proof. The lemma follows by the way the hit costs are generated. Suppose leaf q is requested at time t, and q is the i-th child of its parent p. Then, by observation 17 (part 1), the hit cost entry $h^t(i, 0)$ for every allocation instance running at p is ∞. Thus, if the total cost of the allocation problems is finite, it must be that for each $s \in \Lambda^t_p$, the algorithm ensures that $x^t_{q,0,s} = 0$. Since, $\sum_j x^t_{q,j,s} = 1$ for all s, and $\sum_{s \in \Lambda^t_p} \lambda_s = 1$, it follows that $z(q,t) = \sum_{s \in \Lambda^t_p} \lambda^t_s \sum_j j \cdot x^t_{q,j,s} \geq 1$. □

Remark: Lemma 18 assumes that the total cost of the allocation problems is finite. Later on we show that the cost is in fact bounded by at most a polylogarithmic factor from the optimal k-server cost, and hence finite.

4.3 Performance analysis

We first show that the cost of the fractional k-server solution we generate (at the leaves of the tree) is at most the total convex combination cost of the allocation instances running on T. For a node p (not necessarily a leaf) in T, let $z(p,t)$ denote the total (fractional) number of servers at time t at the leaves of the subtree $T(p)$. The cost of the k-server solution is

$$\sum_t \sum_p W(p) |z(p,t) - z(p, t-1)|.$$

Lemma 19. *The movement cost incurred by the fractional k-server solution is no more than the total movement cost incurred by the convex combination of the allocation instances running on internal nodes of T.*

Proof. First, we claim that $z(p,t) = \sum_{s \in \Lambda^t_p} \lambda^t_s \kappa^t_s(t)$. This follows from the consistency relation (40) we maintain, and our procedure for generating Λ^t_p from Λ^{t-1}_p. In particular, $\kappa^t_s(t)$ is the number of servers available for the fractional allocation instance s running at p. Since the solution produced by the fractional allocation algorithm on this instance satisfies $\sum_i \sum_j j \cdot x^t_{i,j,s} = \kappa^t_s(t)$,[7] this implies that

$$\sum_{s \in \Lambda^t_p} \lambda^t_s \kappa^t_s(t) = \sum_{s \in \Lambda^t_p} \lambda^t_s \sum_i \sum_j j \cdot x^t_{i,j,s} = \sum_i \sum_j \sum_{s' \in \Lambda^t_{p_i} | \kappa^t_{s'}(t)=j} j\lambda^t_{s'} = \sum_i \sum_{s' \in \Lambda^t_{p_i}} \lambda^t_{s'} \kappa^t_{s'}(t).$$

---

[7]Note that when we designed the fractional allocation algorithm in Section 3, we allowed it to deploy less servers than the current quota. As a result, when applying this algorithm here we could sometimes have $\sum_i \sum_j j \cdot x^t_{i,j,s} < \kappa^t_s(t)$. To see that our analysis is still valid in this case, it suffices to consider a modified version of the tree T. In this version, each non-leaf node p of T would have a dummy leaf i_p (that will never be requested in our input sequence) added as its child and set the length $w(p, i_p)$ of the corresponding edge to 0. Now, we would just make each instance of the fractional allocation run at each such p deposit any unused quota of servers at the leaf i_p. Note that as $w(p, i_p) = 0$, this depositing would not incur any additional movement cost and that the modified tree would still be a weighted σ-HST.

The second equality above follows from (40). Applying this iteratively, and noting that $z(q,t)$ for a leaf q is simply $\sum_{s' \in \Lambda_q^t} \lambda_{s'}^t \kappa_{s'}^t(t)$, it follows that $z(p,t) = \sum_{s \in \Lambda_p^t} \lambda_s^t \kappa_s^t(t)$.

Suppose that $\delta = z(p,t) - z(p,t-1) > 0$ server units are removed from the subtree $T(p)$ (the case when $\delta < 0$ is analogous). Let p' be the parent of p (note that $p \neq r$, since $z(r,t) = k$ for all t). As $z(p,t) = \sum_{s \in \Lambda_{p'}^t} \lambda_s^t \sum_j j \cdot x_{p,j,s}^t$, it follows that the allocation algorithm running on the instances $s \in \Lambda_{p'}^t$ will incur a movement cost of at least

$$
\begin{aligned}
W(p) \sum_{s \in \Lambda_{p'}^t} \lambda_s^t \sum_{j=1}^k \left| \sum_{\ell < j} (x_{p,\ell,s}^t - x_{p,\ell,s}^{t-1}) \right| &\geq W(p) \cdot \left| \sum_{s \in \Lambda_{p'}^t} \lambda_s^t \sum_{j=1}^k \sum_{\ell < j} \left(x_{p,\ell,s}^t - x_{p,\ell,s}^{t-1} \right) \right| \\
&= W(p) \cdot \left| \sum_{s \in \Lambda_{p'}^t} \lambda_s^t \sum_{j=0}^k (k-j)(x_{p,j,s}^t - x_{p,j,s}^{t-1}) \right| \\
&= W(p) \cdot \left| \sum_{s \in \Lambda_{p'}^t} \lambda_s^t \sum_{j=0}^k (-j)(x_{p,j,s}^t - x_{p,j,s}^{t-1}) \right| \\
&= W(p) \cdot |z(p,t) - z(p,t-1)|,
\end{aligned}
$$

where we used the fact that $k \cdot \sum_{j=0}^k x_{p,j,s}^t = k = k \cdot \sum_{j=0}^k x_{p,j,s}^{t-1}$. $\qquad\square$

Given Lemma 18 and 19 above, it suffices to consider the total movement cost incurred by the allocations instances running on the tree T and compare it with the optimum k-server cost. This will be our goal in the following. We begin by defining a notion of optimum k-server cost on a weighted σ-HST T when k varies over time.

Definition 20. *Let $T(p)$ be the subtree rooted at p, and let κ be a quota pattern. We define $\mathrm{Optcost}(p,\kappa,t)$ as the optimum cost of serving the request sequence $\rho \cap T(p)$ until time step t subject to the constraint that $\kappa(t')$ servers are available at each time t', for $1 \leq t' \leq t$.*

We should be precise about the meaning of a k-server solution on T in the case $\kappa(t)$ can vary. First, at any time t' there should be one server unit at the requested location $\rho(t')$. The cost of the solution is the total movement cost of the servers. The servers are always located on the leaves of T. At time t, when the number of servers changes from $\kappa(t-1)$ to $\kappa(t)$, we will require that $\kappa(t) - \kappa(t-1)$ servers enter (or leave, if $\kappa(t) < \kappa(t-1)$) from the root of T.

For a vector κ, let us define $g(\kappa,t) = \sum_{t'=1}^t |\kappa(t') - \kappa(t'-1)|$. The following is a simple but very useful fact about Optcost, that we will need.

Lemma 21. *Let p be an internal node in T with children $p_1, \ldots, p_d$. For any k-server request sequence ρ on the leaves of T and any quota pattern vector κ, the following recurrence holds.*

$$
\mathrm{Optcost}(p,\kappa,t) = \min_{\kappa_1,\ldots,\kappa_d : \sum_{i=1}^d \kappa_i = \kappa} \left(\sum_{i=1}^d \mathrm{Optcost}(p_i,\kappa_i,t) + w(p,i) \sum_{i=1}^d g(\kappa_i,t) \right). \tag{43}
$$

Here, in the base case in which p is a leaf, define $\mathrm{Optcost}(p,\kappa,t) = \infty$ if there is some time $t' \leq t$ such that $\rho(t') = p$ and $\kappa(t') = 0$. Otherwise, if $\kappa(t') \geq 1$, whenever $\rho(t') = p$, define $\mathrm{Optcost}(p,\kappa,t) = 0$.

Proof. The condition $\sum_{i=1}^{d} \kappa_i = \kappa$ ensures consistency between the number of servers in $T(p)$ and its subtrees $T(p_i)$. The term $\text{Optcost}(p_i, \kappa_i, t)$ measures the cost of serving the requests within $T(p_i)$ and $g(\kappa_i, t)$ measures the cost of servers leaving or entering subtree $T(p_i)$. $\qquad\square$

Next, we need the following key lemma that relates $\text{Optcost}(p, \kappa, t)$ to our procedure for generating hit costs at node p.

Lemma 22. *Let p be a non-root node of T. Given a quota pattern κ for $T(p)$, let $\text{Optcost}(p, \kappa, t)$ be as defined above. Then,*

$$\left| \text{Optcost}(p, \kappa, t) - \sum_{t'=1}^{t} h_p^{t'}(\kappa(t')) \right| \leq 2 \cdot \frac{1}{\sigma - 1} \cdot W(p) \cdot g(\kappa, t), \tag{44}$$

where $h_p^{t'}(j) = \text{Optcost}(p, j \cdot 1, t') - \text{Optcost}(p, j \cdot 1, t' - 1)$ denotes the incremental cost of the optimal k-server solution for $T(p)$ with exactly j servers.

This lemma (for the case of HSTs) is implicit in the work of Coté et al. [16]. For completeness, and since we need the extension to weighted σ-HSTs, we give a proof of Lemma 22 in Appendix B.

We are now ready to prove the following theorem.

Theorem 23. *Let T be a weighted σ-HST with $\sigma > 9$, depth ℓ, and diameter Δ. Let ρ be a k-server request sequence on T, and κ be the quota pattern. Consider the total movement cost incurred by the convex combination of the allocation instances running on nodes of T (based on the algorithm described in Section 4.1). This cost is no more than*

$$\beta_\ell(\text{Optcost}(r, \kappa, \infty) + \Delta \cdot g(\kappa, \infty)),$$

where β_ℓ satisfies the recurrence $\beta_\ell = \gamma \beta_{\ell-1} + O(\log(k/\varepsilon))$, and $\beta_0 = 1$. Here, ϵ is any constant for which the fractional allocation algorithm is $(1 + \varepsilon, O(\log(k/\varepsilon)))$-competitive, and

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

We first show how Theorem 23 implies Theorem 6. The recurrence $\beta_\ell \leq \gamma \beta_{\ell-1} + O(\log(k/\varepsilon))$ in Theorem 23, together with $\beta_0 = 1$, implies that

$$\beta_\ell = O(\log(k/\varepsilon))\left(\frac{\gamma^{l+1} - 1}{\gamma - 1}\right).$$

Choosing $\varepsilon = 1/(4\ell)$, and provided $\sigma = \Omega(\varepsilon^{-1}\log(k/\varepsilon)) = \Theta(\ell \log(k\ell))$, we get that $\gamma \leq (1 + \frac{1}{2\ell})$, and hence

$$\beta_\ell = O(\ell \log(k/\varepsilon)) = O(\ell \log(k\ell)).$$

As $g(k \cdot \vec{1}, \infty) = 0$, this implies an $O(\ell \log(k\ell))$ guarantee for a weighted σ-HST of depth ℓ, provided $\sigma = \Omega(\ell \log(k\ell))$. We now prove Theorem 23.

Proof. (Theorem 23): We prove by induction on the depth of the tree.

Base case: the theorem is clearly true for $\ell = 0$ (i.e. a single point space).

32

Inductive step: suppose the theorem is true for weighted σ-HSTs of depth $\ell - 1$, and let T be a weighted σ-HST of depth ℓ, rooted at r. Let w_i be the distance to the i-th child of r, and let $w = \max_i w_i$. Given κ, consider some optimal solution for T that achieves value $\mathrm{Optcost}(r, \kappa, \infty)$. We also denote the total cost $\mathrm{Optcost}(r, \kappa, \infty)$ by $\mathrm{Optcost}(r, \kappa)$, and $g(\kappa, \infty)$ by $g(\kappa)$. Let κ_i^* be optimal vectors for the children p_i of r corresponding to this solution. Since κ_i^* determines $\mathrm{Optcost}(r, \kappa)$, by (43) we have

$$\mathrm{Optcost}(r, \kappa) = \sum_i \left(\mathrm{Optcost}(p_i, \kappa_i^*) + w_i \cdot g(\kappa_i^*) \right).$$

By (44), for each child i, $\mathrm{Optcost}(p_i, \kappa_i^*) \geq \mathrm{Hitcost}(i, \kappa_i^*) - 2w_i \cdot g(\kappa_i^*)/(\sigma - 1)$, where we denote $\mathrm{Hitcost}(i, \kappa') = \sum_t h_i^t(\kappa'(t))$. Thus,

$$\mathrm{Optcost}(r, \kappa) \geq \sum_i \left(\mathrm{Hitcost}(i, \kappa_i^*) + w_i \cdot \left(1 - \frac{2}{\sigma - 1} \right) g(\kappa_i^*) \right).$$

Multiplying throughout by $(\sigma - 1)/(\sigma - 3)$, which is at most $1 + 3/\sigma$ (as $\sigma \geq 9$), implies

$$\sum_i (\mathrm{Hitcost}(i, \kappa_i^*) + w_i \cdot g(\kappa_i^*)) \leq \frac{\sigma - 1}{\sigma - 3} \cdot \mathrm{Optcost}(r, \kappa) \leq \left(1 + \frac{3}{\sigma} \right) \cdot \mathrm{Optcost}(r, \kappa). \qquad (45)$$

Consider the fractional allocation instance A running at r in our algorithm and let $\{x_{i,j}^t\}_{i,j}$ be its solution at time step t. (Since there is only a single quota pattern $\kappa(r)$ at the root, we assume that there is only one instance running, and keep it notationally convenient.) By Theorem 14 and by (45), the hit cost incurred by A satisfies

$$
\begin{aligned}
\sum_t \sum_{i=1}^d \sum_{j=0}^k x_{i,j}^t h_i^t(j) &\leq (1 + \varepsilon) \left(\sum_{i=1}^d \mathrm{Hitcost}(i, \kappa_i^*) + \sum_{i=1}^d w_i \cdot g(\kappa_i^*) + w \cdot g(\kappa) \right) \\
&\leq (1 + \varepsilon) \left(1 + \frac{3}{\sigma} \right) \mathrm{Optcost}(r, \kappa) + (1 + \varepsilon) w \cdot g(\kappa),
\end{aligned}
\qquad (46)
$$

and the movement cost satisfies

$$
\begin{aligned}
\sum_t \sum_{i=1}^d w_i \sum_j |y_{i,j}^t - y_{i,j}^{t-1}| &\leq O(\log(k/\varepsilon)) \left(\sum_{i=1}^d \mathrm{Hitcost}(i, \kappa_i^*) + \sum_{i=1}^d w_i \cdot g(\kappa_i^*) + w \cdot g(\kappa) \right) \\
&\leq O(\log(k/\varepsilon))(\mathrm{Optcost}(r, \kappa) + w \cdot g(\kappa)).
\end{aligned}
\qquad (47)
$$

Now, recall that each of the children $p_1, \ldots, p_d$ of r is running a convex combination on allocation instances Λ_i^t, the quota pattern of which is determined by $\{x_{i,j}^t\}_{i,j}$. So, the hit costs and movement costs of A (i.e. left hand sides of (46) and (47)) can be expressed alternately as follows. Since the quota patterns at p_i maintain the invariant (40) throughout the algorithm, the total hit cost accumulated by A can be expressed as

$$
\begin{aligned}
\sum_t \sum_{i=1}^d \sum_{j=0}^k x_{i,j}^t h_i^t(j) &= \sum_t \sum_{i=1}^d \sum_{j=0}^k \sum_{s \in \Lambda_i^t, \kappa_s^t(t)=j} \lambda_s^t h_i^t(j) \\
&= \sum_{i=1}^d \sum_{s \in \Lambda_i^\infty} \lambda_s^\infty \mathrm{Hitcost}(i, \kappa_s^\infty).
\end{aligned}
\qquad (48)
$$

33

Henceforth, we use Λ_i and κ_s to denote Λ_i^∞ and κ_s^∞. By our cost preserving procedure for updating κ_s^t when $x_{i,j}^{t-1}$ changes to $x_{i,j}^t$, the movement cost incurred by A can be expressed as

$$
\begin{aligned}
\sum_t \sum_{i=1}^d w_i \sum_j |y_{i,j}^t - y_{i,j}^{t-1}|) &= \sum_{i=1}^d w_i \sum_t \sum_{s \in \Lambda_i^t} \lambda_s^t |\kappa_s^t(t) - \kappa_s^t(t-1)| \\
&= \sum_{i=1}^d w_i \sum_{s \in \Lambda_i} \lambda_s g(\kappa_s). \quad (49)
\end{aligned}
$$

Let us now consider the overall movement cost incurred by the convex combination of the allocation instances. This is equal to the movement cost for A (running at the root) plus the sum movement costs incurred within $p_1, \ldots, p_d$. By the induction hypothesis, the movement cost for each of these recursive algorithms that are run on subtrees $T(p_i)$ with quota pattern κ_s is at most

$$
\beta_{\ell-1} \left(\text{Optcost}(p_i, \kappa_s) + \frac{w_i}{\sigma - 1} \cdot g(\kappa_i) \right).
$$

Thus, the total recursive cost is at most

$$
\begin{aligned}
&\sum_{i=1}^d \sum_{s \in \Lambda_i} \lambda_s \beta_{\ell-1} \left(\text{Optcost}(p_i, \kappa_s) + \frac{w_i}{\sigma - 1} \cdot g(\kappa_s) \right) \\
&\leq \sum_{i=1}^d \sum_{s \in \Lambda_i} \lambda_s \beta_{\ell-1} \left(\text{Hitcost}(i, \kappa_s) + \frac{3w_i}{\sigma - 1} \cdot g(\kappa_s) \right) \quad (50) \\
&\leq (1 + \varepsilon) \beta_{\ell-1} \left(\left(1 + \frac{3}{\sigma} \right) \text{Optcost}(r, \kappa) + w \cdot g(\kappa) \right) + \sum_{i=1}^d \sum_{s \in \Lambda_i} \lambda_s \beta_{\ell-1} \frac{3w_i}{\sigma - 1} \cdot g(\kappa_s). \quad (51)
\end{aligned}
$$

Here, (50) follows as $\text{Optcost}(p_i, \kappa_s) \leq \text{Hitcost}(i, \kappa_s) + 2w_i/(\sigma - 1)$ by (44), and (51) follows from (48) and (46). Now, the total cost of movement of servers across the subtrees $p_1, \ldots, p_d$ is

$$
\sum_{i=1}^d w_i \sum_{s \in \Lambda_i} \lambda_s g(\kappa_s). \quad (52)
$$

Adding up the costs from (51) and (52), the total cost incurred by the algorithm is at most

$$
(1 + \varepsilon) \beta_{\ell-1} \left(\left(1 + \frac{3}{\sigma} \right) \text{Optcost}(r, \kappa) + w \cdot g(\kappa) \right) + \sum_{i=1}^d \sum_{s \in \Lambda_i} \lambda_s \left(1 + \frac{3\beta_{\ell-1}}{\sigma - 1} \right) w_i \cdot g(\kappa_s). \quad (53)
$$

By (49) and (47) we have

$$
\sum_{i=1}^d w_i \sum_{s \in \Lambda_i} \lambda_s g(\kappa_s) \leq O(\log(k/\varepsilon)) \left(\text{Optcost}(r, \kappa) + w \cdot g(\kappa) \right). \quad (54)
$$

Plugging (54) into (53) implies that the total cost is at most

$$
\left(\beta_{\ell-1} \cdot \gamma + O\left(\log(k/\varepsilon) \right) \right) \left(\text{Optcost}(r, \kappa) + w \cdot g(\kappa) \right),
$$

34

where

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

Thus, the claimed result follows. □

5   Weighted HSTs and Online Rounding

In this section, we show how one can embed a σ-HST into a small depth weighted σ-HST with constant distortion, i.e., we prove Theorem 8. Also, we present an online rounding procedure for the fractional k-server problem on an (un-weighted) σ-HST, that is, we establish Theorem 7.

5.1   Embedding σ-HSTs into Weighted σ-HSTs:

Theorem 8. *Let T be a σ-HST with n leaves, but possibly arbitrary depth. Then, T can be transformed into a weighted σ-HST $\widetilde{T}$ such that: the leaves of $\widetilde{T}$ and T are identical, $\widetilde{T}$ has depth $O(\log n)$, and any leaf to leaf distance in T is distorted in $\widetilde{T}$ by a factor of at most $2\sigma/(\sigma - 1)$.*

Proof. For a given rooted tree T', we say that it is *balanced* if: (1) there is no child p of the root such that the subtree $T'(p)$ (rooted at p) contains more than half of the nodes of T', and (2) each subtree $T'(p')$, rooted at a child p' of the root, is balanced as well. It is easy to see that if a balanced T' has n leaves then its depth is $O(\log n)$.

We now present a procedure that contracts some of the edges of T and yields a weighted σ-HST T' such that: (a) T' is balanced, and (b) for any leaf-to-leaf path in T, at least one (out of two) of the longest edges on this path has not been contracted in T'.

The procedure works as follows. Let r be the root of T and $p_1, \ldots, p_d$ be its children. We first recursively transform each of the trees $T(p_i)$ rooted at p_i. Next, we check if there is a child p_i such that $T(p_i)$ contains more than half of the nodes of T. (Note that there can be at most one such child.) If not, then T (with modified $T(p_1), \ldots, T(p_d)$) is the desired T'. Otherwise, T' is the tree T with the edge that connects p_i to r contracted.

It is easy to see that the tree T' obtained by the above procedure is indeed balanced and also the lengths of the edges on any root-to-leaf part decrease at rate of at least σ. Thus, T' is a weighted σ-HST and (a) holds. Now, to prove (b), let us inductively assume that it holds for all the transformed subtrees $T(p_i)$. We prove it for the transformed T. Consider a leaf-to-leaf path in T. If the path is contained entirely in one of the subtrees $T(p_i)$, then we are done by our inductive assumption. Otherwise, the path contains two edges incident to the root r. As T is a σ-HST, these two edges must be the longest ones on this path. Thus, as the procedure could contract only one of them, (b) follows as well.

Now, clearly, taking $\widetilde{T}$ to be T' satisfies the first two desired properties of $\widetilde{T}$, as stated in the theorem. To prove that the last one holds too, we note that the length of any leaf-to-leaf path can only decrease in $\widetilde{T}$ (as we only contract edges). However, as we retain at least one of the longest edges, we have that the worst case distortion it incurs is at most:

$$\frac{2}{\sigma^\ell} \cdot \sum_{j=0}^{\ell} \sigma^j \le 2 \cdot \sum_{j=0}^{\ell} \frac{1}{\sigma^j} \le \frac{2\sigma}{\sigma - 1},$$

where σ^ℓ is the length of the longest edge on the path. The theorem thus follows. □

35

5.2 Rounding the Fractional k-server Solution Online

We now show how to obtain an online randomized (integral) k-server algorithm from a *fractional k-server algorithm*, when the underlying metric corresponds to a σ-HST T. The competitiveness of the obtained algorithm will only be an $O(1)$ factor worse than the competitiveness of the fractional algorithm, provided $\sigma > 5$. The rounding procedure builds on ideas in [11] which were developed in the context of the finely competitive paging problem, and extends those ideas from a uniform metric to HSTs.

Let $1, \ldots, n$ denote the leaves of the σ-HST T. Recall that the fractional solution to the k-server problem specifies at each time t the probability x_i^t of having a server at leaf i. The variables x_i^t satisfy $\sum_i x_i^t = k$.[8] When these probabilities change at each time step t, the movement cost paid by the fractional algorithm is equal to the earthmover distance between the distribution $\overline{x}^{t-1}$ and $\overline{x}^t$. In contrast, an execution of a randomized algorithm can be viewed as an evolution of a distribution over k-tuples of leaves. (There is no point in having more than one server at a leaf.) To make this more precise, let us define a *configuration* to be a subset C of $\{1, \ldots, n\}$ of size exactly k. The state S^t at a given time t of a randomized k-server algorithm is specified by a probability distribution μ_{S^t} on the configurations, where $\mu_S(C)$ denotes the probability mass of configuration C in state S.

Now, we say that a state S is *consistent* with a fractional state $\overline{x}$ if,

$$\text{(consistency)} \quad \text{for each } i \in [n] \qquad \sum_{C:i\in C} \mu_S(C) = x_i, \tag{55}$$

i.e., if the marginal probabilities of the state S coincide with $\overline{x}$.

We therefore see that in order to round a fractional algorithm to a randomized algorithm, we need to devise a way of maintaining (in an online manner) a sequence of states $S^0, S^1, \ldots$ that are always consistent with the corresponding states $\overline{x}^0, \overline{x}^1, \ldots$ of the fractional algorithm, and such that the cost of the maintenance is within an $O(1)$ factor of the movement cost of the fractional algorithm. More precisely, our goal is to establish the following result.

Theorem 24. *Let T be a σ-HST with n leaves, $\sigma > 5$, and let $\overline{x}^0, \overline{x}^1, \ldots$ be a sequence of states of a fractional k-server algorithm. There is an online procedure that maintains a sequence of randomized k-server states $S^0, S^1, \ldots$ with the following properties:*

1. *At any time t, the state S^t is consistent with the fractional state $\overline{x}^t$.*

2. *If the fractional state changes from $\overline{x}^{t-1}$ to $\overline{x}^t$ at time t, incurring a movement cost of c_t, then the state S^{t-1} can be modified to a state S^t while incurring a cost of $O(c_t)$.*

The key idea in bounding the maintenance cost of our rounding in Theorem 7 is to require that the states S^t that we produce are not only consistent with $\overline{x}^t$, but also each configuration in the support of S^t does not deviate much from the fractional state x^t. To this end, we introduce the following additional property for k-server states.

For a node p of T and a fractional state $\overline{x}$, let $x_p = \sum_{i\in T(p)} x_i$ be the fractional amount of servers that $\overline{x}$ has on the leaves of the subtree $T(p)$ rooted at p. Also, let $n_p(C) = C \cap T(p)$ denote the number of servers in configuration C on leaves of $T(p)$. We say that a configuration C is *balanced* with respect to $\overline{x}$ if $n_p(C) \in \{\lfloor x_p \rfloor, \lceil x_p \rceil\}$ for every node p. Now, we say that a k-server state S

---

[8]Note that, in principle, the definition of a fractional solution allows us to have $\sum_i x_i^t$ to be strictly smaller than k. However, as the starting configuration $\overline{x}^0$ has exactly k servers, it is easy to modify our fractional solution so that it always has exactly k servers, while not increasing its movement cost.

is balanced with respect to $\overline{x}$ if every configuration in its support (i.e., with non-zero probability mass) is balanced with respect to $\overline{x}$. That is, for all p and C for which $\mu_S(C) > 0$,

$$\lfloor x_p \rfloor \leq n_p(C) \leq \lceil x_p \rceil. \tag{56}$$

Now, our approach to making the states we work with balanced is facilitated by the following definition. Let $\overline{x}$ be a fractional state and let S be some k-server state consistent with $\overline{x}$ (but S might be not balanced with respect to $\overline{x}$). We define the *balance gap*, $G(S, \overline{x})$ of S, (with respect to $\overline{x}$) to be:

$$G(S, \overline{x}) = \sum_p W(p) \sum_{C \in S} \mu_S(C) \min\left(|n_p(C) - \lfloor x_p \rfloor|, |n_p(C) - \lceil x_p \rceil|\right). \tag{57}$$

Here, $W(p)$ denotes the length of the edge from p to its parent. Clearly, when S is balanced with respect to $\overline{x}$, its balance gap is zero. Intuitively, the balance gap measures the distance of the state S from being balanced. This intuition is made concrete by the following lemma.

Lemma 25. *Let $\overline{x}$ be a fractional state and let S be a k-server state on the leaves of a σ-HST T, with $\sigma > 5$, which is consistent with $\overline{x}$ (but not necessarily balanced with respect to it). Then, S can be converted to another state S' which is both consistent and balanced with respect to $\overline{x}$, while incurring a cost of $O(G(S, \overline{x}))$.*

We will prove Lemma 25 later. First we show show how Theorem 24 follows from it.

Proof of Theorem 24. Consider a fractional state $\overline{x}$ that changes to some other fractional state $\overline{x}'$, and let S be a k-server state which is both consistent and balanced with respect to $\overline{x}$. As S^0 (the state initially at time $t = 0$) is consistent and balanced with respect to $\overline{x}^0$, it is easy to see that to establish the theorem, it suffices to show that for any $\overline{x}, \overline{x}'$, and S as above, there is a k-server state S' which is consistent and balanced with respect to $\overline{x}'$, and such that the cost of changing the state S to state S' is within $O(1)$ factor of the cost of changing the state $\overline{x}$ to $\overline{x}'$. Furthermore, it is enough to restrict oneself to the case in which $\overline{x}'$ is obtained from $\overline{x}$ by applying an elementary move, i.e. x_i, for some leaf i, is increased by δ and $x_{i'}$, for some other leaf i', is decreased by δ, where δ can be chosen to be infinitesimally small.

In light of this, we can focus on proving the existence of such S'. To this end, let p denote the least common ancestor of i and i'. Note that in this case the fractional cost of changing $\overline{x}$ to $\overline{x}'$ is at least $2\delta w(p)$, where $w(p) = W(p)/\sigma$ is the length of the edges from p to its children.

Consider now the following transformation of the state S. First, we add the leaf i to a probability mass of δ on arbitrarily chosen configurations in S that do not contain i already. Next, we remove i' from some probability mass δ of configurations containing i'. (Note that the existence of sufficient mass of each type of configurations follows from the consistency property of S with $\overline{x}$.) Let $\widetilde{S}$ be the resulting state.

Before proceeding, we note that as δ can be taken to be arbitrarily small, we can restrict our discussion to the case in which i is added to a mass of δ of a particular configuration C, and i' is removed from a mass of δ of a particular configuration C'.

Now, to continue the proof, we observe that $\widetilde{S}$ is consistent with $\overline{x}'$. However, the modified configurations C and C' that $\widetilde{S}$ contains are not legal anymore as they do not consist of exactly k leaves. Also, $\widetilde{S}$ might be unbalanced with respect to $\overline{x}'$.

We show how we can modify $\widetilde{S}$ to fix these shortcomings. To this end, we note that as C contains i, and it was balanced with respect to $\overline{x}$ prior to adding i, it must hold now that $n_p(C) \geq \lfloor x_p \rfloor + 1$. Similarly, for C', it must hold that $n_p(C') \leq \lfloor x_p \rfloor < \lfloor x_p \rfloor + 1 = n_p(C)$. Thus, by the pigeon hole principle, there must exist a leaf j of $T(p)$ which is contained in C, but not in C'. Let us modify $\widetilde{S}$ by removing j from C and adding it to C'. Clearly, this makes all the configurations in $\widetilde{S}$ legal again, keeps $\widetilde{S}$ consistent with $\overline{x}'$, and the total movement cost corresponding to this modification (due to deleting i, adding i', and swapping j) is at most $4\delta w(p)\sigma/(\sigma - 1) = O(\delta w(p))$, for $\sigma > 5$, which is within an $O(1)$ factor of the cost of changing $\overline{x}$ to $\overline{x}'$.

Unfortunately, $\widetilde{S}$ might still be unbalanced with respect to $\overline{x}'$. To bound the imbalance, let us first consider the case in which $\lfloor x_q \rfloor = \lfloor x_q' \rfloor$ and $\lceil x_q \rceil = \lceil x_q' \rceil$ for all nodes q. This implies that all the configurations in $\widetilde{S}$ other than the modified configurations C and C' are already balanced with respect to $\overline{x}'$ as they were balanced with respect to $\overline{x}$. Now, we note that $x_q \neq x_q'$ only for nodes q that are on the path between i and i' (but excluding p). Similarly, $n_q(C)$ and $n_q(C')$ could change only for nodes on the path from p to i, i', or j (but, again, excluding p). Therefore, as both C and C' were initially balanced with respect to $\overline{x}$, we can conclude that the total imbalance gap $G(\widetilde{S}, \overline{x}) = G(\widetilde{S}, \overline{x}')$ of $\widetilde{S}$ after our modifications is at most:

$$3 \cdot 2\delta w(p) \left(1 + \frac{1}{\sigma} + \frac{1}{\sigma^2} + \dots \right) = O(\delta w(p)).$$

Thus, by applying Lemma 25 to $\widetilde{S}$, we obtain a state S' that is consistent and balanced with respect to $\overline{x}'$ and the cost of this procedure is $O(\delta w(p))$, as desired.

Now, it remains to deal with the case in which either $\lfloor x_q \rfloor \neq \lfloor x_q' \rfloor$, or $\lceil x_q \rceil \neq \lceil x_q' \rceil$, for some nodes q. To this end, we note that by taking δ to be small enough (but non-zero), we can ensure that for each q for which at least one of these two inequalities holds, it must be the case that either x_q or x_q' is an integer. In the former case, we have that for all the configurations C'' in S that have non-zero mass, $n_q(C'') = x_q = \lfloor x_q \rfloor = \lceil x_q \rceil$ and thus $\lfloor x_q' \rfloor \leq n_q(C'') \leq \lceil x_q' \rceil$. In the latter case, as $|x_q - x_q'| \leq \delta$ and for every relevant configuration C'' in S, $\lfloor x_q \rfloor \leq n_q(C'') \leq \lceil x_q \rceil$, the total probability mass of configurations C'' in S, such that $n_q(C'') = \lfloor x_q \rfloor < \lfloor x_q' \rfloor = x_q'$ or $n_q(C'') = \lceil x_q \rceil > \lceil x_q' \rceil = x_q'$, can be at most δ.

As a result, we see that the total probability mass of configurations in $\widetilde{S}$ that are not balanced with respect to $\overline{x}'$ is at most 3δ (the contribution of 2δ comes from the modified configurations C and C'). Thus, by calculating the imbalance gap similarly to what we did before, we can show that $G(\widetilde{S}, \overline{x}')$ is $O(\delta w(p))$, and once again use Lemma 25 to obtain the desired S'. This concludes the proof of the theorem. $\qquad\square$

It remains to prove Lemma 25.

Proof of Lemma 25. Let us call a node p in our σ-HST T *imbalanced* if

$$\sum_{C \in S} \mu_S(C) \min(|n_p(C) - \lfloor x_p \rfloor|, |n_p(C) - \lceil x_p \rceil|) > 0.$$

If no node is imbalanced, then clearly $G(S, x) = 0$, and we are already done, so we assume that this is not the case. Let p be an imbalanced nodes which is at the highest level of T (breaking ties arbitrarily). We note that p cannot be the root r of T, as each configuration has exactly k servers, and $x_r = k$.

Consider now a configuration C for which $\mu_S(C) > 0$ and $n_p(C) \notin \{\lfloor x_p \rfloor, \lceil x_p \rceil\}$ and let us assume that $n_p(C) < \lfloor x_p \rfloor$ (the other case can be treated similarly). As S is consistent with $\overline{x}$, $\sum_{C''} \mu_S(C'') n_p(C'') = x_p$, and so there must be some other configuration C' with $\mu_S(C') > 0$ such that $n_p(C') \geq \lfloor x_p \rfloor + 1$. So, in particular, we have $n_p(C') - n_p(C) \geq 2$.

Now, let $\widetilde{p}$ denote the parent of p in T (recall that p is not the root). As we choose p to be an imbalanced node at the highest possible level, $\widetilde{p}$ must be balanced, and hence $|n_{\widetilde{p}}(C') - n_{\widetilde{p}}(C)| \leq 1$. But, since $n_p(C') - n_p(C) \geq 2$, it implies the existence of some other child p', $p' \neq p$, of $\widetilde{p}$ such that $n_{p'}(C') < n_{p'}(C)$.

Therefore, by the pigeon hole principle, there must exist a leaf i in the subtree $T(p)$ rooted at p which is contained in C', but not in C. Similarly, C must contain a leaf i' in $T(p')$ which is not contained in C'. Let $\delta = \min(\mu_S(C), \mu_S(C'))$ (note that $\delta > 0$). Consider a modification of S in which we take any arbitrary probability mass δ of configurations C and replace i' by i in them. Next, we take any arbitrary probability mass δ of configurations C' and replace i by i' in them.

Let us summarize the properties satisfied by S after this modification. First, the state remains consistent with the fractional solution $\overline{x}$, because the marginals of the leaves i and i' have not changed. Second, since neither $n_{\widetilde{p}}(C)$ nor $n_{\widetilde{p}}(C')$ have changed, $\widetilde{p}$ remains balanced. Moreover, the only nodes for which the imbalance could have changed are on the path from i to p and i' to p'. Third, replacing i' with i (in a δ measure of C) increases $n_p(C)$ by 1 for these configurations, and replacing i with i', leaves the quantity $n_p(C')$ to be of value at least $\lfloor x_p \rfloor$. Together, this implies that the imbalance of p decreases by at least δ. Finally, as $n_{p'}(C) > n_{p'}(C')$ before the modification, the imbalance $\sum_{C'' \in S} \mu_S(C'') \min(|n_{p'}(C'') - \lfloor x_{p'} \rfloor|, |n_{p'}(C'') - \lceil x_{p'} \rceil|)$ of p' can only decrease.

Now, if we analyze the change in the imbalance gap of S, in the worst case, the imbalance of every node from p to i (excluding p) could have increased by 2δ due to the addition of i in C or removal of i from C'. Similarly, the imbalance of every node from p' to i' (excluding p') could have increased by up to 2δ. Together with the above observations, this implies that the imbalance gap of S decreases by at least

$$W(p)\delta - 4\delta w(p)\left(1 + \frac{1}{\sigma} + \frac{1}{\sigma^2} + \ldots\right) = W(p)\delta\left(\frac{\sigma - 5}{\sigma - 1}\right) = \Omega(W(p)\delta),$$

where the last inequality uses the fact that $\sigma > 5$.

On the other hand, as both i and i' lie in $T(\widetilde{p})$, the movement cost incurred in the above procedure is at most $4\delta w(\widetilde{p})(\sigma/(\sigma-1)) = 4\delta W(p)\frac{\sigma}{(\sigma-1)}$, which is within $O(1)$ factor of the reduction in the imbalance gap. The lemma follows by applying the above steps repeatedly until the imbalance gap reaches zero. $\square$

References

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

[2] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. A primal-dual randomized algorithm for weighted paging. In *FOCS'07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 507–517, 2007.

[3] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *SODA'10: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, 2010.

[4] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Unfair metrical task systems on hsts and applications. In *ICALP'10: Proceedings of the 37th International Colloquium on Automata, Languages and Programming*, 2010.

[5] Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *FOCS'96: Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 184–193, 1996.

[6] Yair Bartal. On approximating arbitrary metrices by tree metrics. In *STOC'98: Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 161–168, 1998.

[7] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *STOC'97: Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 711–719, 1997.

[8] Yair Bartal, Béla Bollobás, and Manor Mendel. A ramsy-type theorem for metric spaces and its applications for metrical task systems and related problems. In *FOCS'01: Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science*, pages 396–405, 2001.

[9] Yair Bartal and Eddie Grove. The harmonic k-server algorithm is competitive. *Journal of the ACM*, 47(1):1–15, 2000.

[10] Yair Bartal, Nathan Linial, Manor Mendel, and Assaf Naor. On metric ramsey-type phenomena. In *STOC'03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 463–472, 2003.

[11] Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *FOCS'99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, page 450, 1999.

[12] Avrim Blum, Howard J. Karloff, Yuval Rabani, and Michael E. Saks. A decomposition theorem and bounds for randomized server problems. In *FOCS'92: Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 197–207, 1992.

[13] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

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

[15] M. Chrobak and L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.

[16] A. Coté, A. Meyerson, and L. Poplawski. Randomized an optimal on-line algorithm for k-server on hierarchical binary trees. In *STOC'08: Proceedings of the 40th Annual ACM Symposium on Theory of Computing*, pages 227–234, 2008.

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

[18] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *STOC'03: Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 448–455, 2003.

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

[20] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, 1991.

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

[22] Edward F. Grove. The harmonic online k-server algorithm is competitive. In *STOC'91: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 260–266, 1991.

[23] Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Journal of the ACM*, 42(5):971–983, 1995.

[24] M. Manasse, L.A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.

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

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

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

A Proof of Lemma 11

Consider first the simpler case of the fix stage. Here the variables evolve according to

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

As the derivative of $y_{i,j}^{\tau}$ only depends on $y_{i,j}^{\tau}$ and is continuous, the function $y_{i,j}^{\tau}$ is well defined. As the derivative is 0 when $y_{i,j}^{\tau} = 1$, and non-negative otherwise, it ensures that $y_{i,j}^{\tau}$ always stays in the range $[0, 1]$. Finally, the monotonicity property holds here, as it holds initially when $\tau = 0$, and whenever $y_{i,j} \leq y_{i,j'}$ for some $j < j'$, we have that $dy_{i,j}^{\tau}/d\tau \leq dy_{i,j'}^{\tau}/d\tau$, unless $y_{i,j'}^{\tau} = 1$, in which case monotonicity holds trivially.

We now consider the hit stage. Recall that configuration $y_{i,j}^\eta$, for each (i,j), evolves according to Equation (13), which we reproduce here for convenience:

$$\frac{dy_{i,j}^\eta}{d\eta} = \begin{cases} 0 & \text{if } (i,j) \notin A^\eta, \\ \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot \left(N(\eta) - \alpha\lambda_{i,j}^\eta\right) & \text{otherwise.} \end{cases} \tag{58}$$

In the above, the set A^η denotes the active coordinates at time η (cf. Definition 9) and the normalization factor $N(\eta)$ can be expressed as (cf. (14))

$$N(\eta) = \begin{cases} 0 & \text{if } \sum_{i,j} y_{i,j}^\eta > kd - \kappa(t), \\ \frac{\sum_{(i,j)\in A^\eta} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot \alpha\lambda_{i,j}^\eta}{\sum_{(i,j)\in A^\eta} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)} & \text{otherwise (i.e. if } \sum_{i,j} y_{i,j}^\eta = kd - \kappa(t)). \end{cases} \tag{59}$$

First, we show that during the hit stage blocks never split. To this end, we note that when two blocks merge, their y-values are identical, and since we also modify λ^η to be identical for these blocks, all the variables contained in the merged block evolve in the same way from that point on, as desired. (Note that we do not assume here that the trajectory $\overline{y}^\eta$ is well defined and unique, we just argue that any trajectory compatible with our definition of derivatives cannot split blocks.)

Now, we proceed to analyzing the properties of the evolution described by Equations (58) and (59). As a first step, let us prove the following claim that will be helpful later.

Claim 26. *Consider a feasible configuration $\overline{y}^{\eta'}$, for $\eta' \in [0,1]$, a subset of coordinates A, and a hit cost vector λ. Define for any $\eta \geq \eta'$,*

$$N_A(\eta) = \frac{\sum_{(i,j)\in A} u_{i,j}^\eta \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A} u_{i,j}^\eta},$$

where $u_{i,j}^\eta = \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)$. Now, if we make the configuration $\overline{y}^{\eta'}$ evolve according to

$$\frac{dy_{i,j}^\eta}{d\eta} = \begin{cases} 0 & \text{if } (i,j) \notin A, \\ \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot (N_A(\eta) - \alpha\lambda_{i,j}) & \text{otherwise} \end{cases},$$

then we have that $N_A(\eta)$ does not increase, i.e., $N_A(\eta) \leq N_A(\eta')$ for any $\eta \geq \eta'$.

Proof. Let us fix a $\eta \geq \eta'$ and denote $\eta^+ = \eta + d\eta$. We will prove that $N_A(\eta^+) \leq N_A(\eta)$, which, in turn, implies our claim. To this end, note that one can view $N_A(\eta)$ as a weighted average, over all coordinates in A, of the value of $\alpha\lambda_{i,j}$, where $u_{i,j}^\eta$ is the weight which we attribute to coordinate (i,j) at time η.

Now, the key observation is that the way the $y_{i,j}$s evolve implies that $y_{i,j}^{\eta^+} \geq y_{i,j}^\eta$ if $\alpha\lambda_{i,j} \leq N_A(\eta)$, and $y_{i,j}^{\eta^+} \leq y_{i,j}^\eta$ if $\alpha\lambda_{i,j} \geq N_A(\eta)$. So, as the weights $u_{i,j}^\eta$ are directly proportional to $y_{i,j}^\eta$, we can conclude that during our evolution, the weights of coordinates that have a value of $\alpha\lambda_{i,j}$ above the average value $N_A(\eta)$ (i.e. $u_{i,j}^{\eta^+} \leq u_{i,j}^\eta$ in this case) can only decrease, and the weights of coordinates that have a value of $\alpha\lambda_{i,j}$ which is at most the average (i.e., we have $u_{i,j}^{\eta^+} \geq u_{i,j}^\eta$ for such (i,j)) can only increase. As a result, we can express $N_A(\eta^+)$ as

$$N_A(\eta^+) = \frac{\sum_{(i,j)\in A} u_{i,j}^{\eta^+} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A} u_{i,j}^{\eta^+}} = \frac{\sum_{(i,j)\in A} u_{i,j}^\eta \cdot \alpha\lambda_{i,j} + \sum_{(i,j)\in A} \alpha\lambda_{i,j}\Delta_{i,j}}{\sum_{(i,j)\in A} u_{i,j}^\eta + \sum_{(i,j)\in A} \Delta_{i,j}}, \tag{60}$$

where $\Delta_{i,j} = u_{i,j}^{\eta^+} - u_{i,j}^{\eta}$ and, by our discussion above, we have that $\Delta_{i,j} \geq 0$ if $\alpha\lambda_{i,j} \leq N_A(\eta)$, and $\Delta_{i,j} \leq 0$ otherwise.

Now, if the right hand side of (60) is at most $N_A(\eta)$, then we are done. Otherwise, we must have that

$$\frac{\sum_{(i,j)\in A} u_{i,j}^{\eta} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A} u_{i,j}^{\eta}} > N_A(\eta), \tag{61}$$

as it is easy to check that if $\frac{a+d\cdot t}{b+t} > c$, then also $\frac{a}{b} > c$, as long as t and d are such that $t \geq 0$ if $d \leq c$, and $t \leq 0$ otherwise.

But, the left hand side of (61) is by definition equal to $N_A(\eta)$. So, the obtained contradiction implies that indeed $N_A(\eta^+) \leq N_A(\eta)$, and the claim follows. $\qquad\square$

Now, observe that the set A^η, as well as $N(\eta)$, depend only on the state $\overline{y}^\eta$ and the hit cost vector λ^η. As a result, both A^η and $N(\eta)$ can, in principle, vary drastically between points of time. However, as we show in the following claim, "locally" they tend to behave in a regular manner.

Claim 27. *For any $\eta' \in [0,1)$ and feasible configuration $\overline{y}^{\eta'}$, there exists an η'', $\eta' < \eta'' \leq 1$, such that:*

(a) $A^\eta = A^{\eta'}$ *for each $\eta \in [\eta', \eta'')$, i.e., the set A^η of active coordinates does not change for $\eta < \eta''$;*

(b) $\lambda^\eta = \lambda^{\eta'}$ *for any $\eta \in [\eta', \eta'')$, i.e., there are no block merges until time η'';*

(c) *the sum $\sum_{i,j} y_{i,j}^\eta$ is bigger than $kd - \kappa(t)$ for all $\eta < \eta''$, unless it was already equal to $kd - \kappa(t)$ at time η';*

(d) *the configuration $\overline{y}^{\eta''}$ is well defined and feasible for $\eta \in [\eta', \eta'']$ (note that this interval contains η'').*

Furthermore, one can assume η'' is maximal, that is, unless η'' is already equal to 1, there exists no larger value of η'' for which the above holds.

For future reference, we call η'', defined as in the above claim, the *horizon* of $\overline{y}^{\eta'}$.

Proof of Claim 27. Let us define a new process for the evolution of $y_{i,j}^\eta$ in a suitably small neighborhood of η' as follows:

$$\frac{dy_{i,j}^\eta}{d\eta} = = \begin{cases} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right) \cdot \left(\tilde{N}(\eta) - \alpha\lambda_{i,j}^{\eta'}\right) & \text{if } (i,j) \in A^{\eta'} \\ 0 & \text{otherwise} \end{cases} \tag{62}$$

where

$$\tilde{N}(\eta) = \begin{cases} 0 & \text{if } \sum_{i,j} y_{i,j}^{\eta'} > kd - \kappa(t), \\ \frac{\sum_{(i,j)\in A^{\eta'}} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)\alpha\lambda_{i,j}^{\eta'}}{\sum_{(i,j)\in A^{\eta'}} \frac{1}{w_i}\left(y_{i,j}^\eta + \beta\right)} & \text{otherwise (i.e. if } \sum_{i,j} y_{i,j}^{\eta'} = kd - \kappa(t)). \end{cases} \tag{63}$$

While the new process looks similar to the original one, there are some crucial differences. First, we do not care if a variable $y_{i,j}^\eta$ either exceeds 1 or becomes negative, i.e., the active set is not updated as η progresses and remains the set $A^{\eta'}$. Second, the cost vector is not updated and it remains $\lambda^{\eta'}$,

i.e., blocks are not merged and the monotonicity requirement (4) are ignored. Third, the value of $\tilde{N}(\eta)$ is not changed once the value of $\sum_{i,j} y_{i,j}^\eta$ hits the quota. If $\sum_{i,j} y_{i,j}^{\eta'} > kd - \kappa(t)$ at time η', $\tilde{N}(\eta)$ will be always 0. As a result, the latter sum might become less than $kd - \kappa(t)$ at some point.

Now, the key observation that makes this new process useful to us is that it is identical to our original process, as long as the evolving configuration is still feasible, and no coordinate becomes active and changes in the original process. More precisely, the trajectories of the two processes coincide as long as in the configuration $\overline{y}^\eta$ evolving with respect to this new process:

(1) $y_{i,j}^\eta$ is in $[0,1]$ for each $(i,j) \in A^{\eta'}$. (Otherwise, a coordinate (i,j) violating this condition would have already become inactive in the original process.), or

(2) $\tilde{N}(\eta) - \alpha \lambda_{i,j}^{\eta'}$ is non-negative (respectively non-positive) for $(i,j) \notin A^{\eta'}$ with $y_{i,j}^{\eta'} = 1$ (respectively $y_{i,j}^{\eta'} = 0$). (Otherwise, a coordinate (i,j) violating this condition would have already become active and would have changed in the original process.), or

(3) the monotonicity condition holds, i.e., $y_{i,j}^\eta \leq y_{i,j+1}^\eta$ for each (i,j). (Otherwise, a block merge would have already happened in the original process.), or

(4) the server quota is obeyed, i.e., $\sum_{i,j} y_{i,j}^\eta \geq kd - \kappa(t)$. (Otherwise, $N(\eta)$ would have already changed so as to guarantee that the number of servers stays at the quota.)

Let η^ℓ for $\ell \in \{1,2,3,4\}$ be the last $\eta \geq \eta'$ for which the ℓ-th condition is satisfied in the interval $[\eta', \eta]$. Also, let $\overline{\eta}^0$ (respectively $\overline{\eta}^1$) be the first time $\eta \geq \eta'$ in which $y_{i,j}^\eta = 0$ (respectively, $y_{i,j}^\eta = 1$) for $(i,j) \in A^{\eta'}$ (respectively $(i,j) \notin A^{\eta'}$), and $\tilde{N}(\eta) - \alpha \lambda_{i,j}^{\eta'} = 0$. As in the new process $\tilde{N}(\eta)$ – and thus all the derivatives and the trajectory of $\overline{y}^\eta$ – are continuous and bounded in the our interval of interest[9], such maximal η^ℓs and $\overline{\eta}^0, \overline{\eta}^1$ always exists. Let us take $\eta'' = \min\{\min_\ell \eta^\ell, \overline{\eta}^0, \overline{\eta}^1, 1\}$. Note that this implies that $\overline{y}^\eta$ is well defined and feasible for any $\eta \in [\eta', \eta'']$ – so, condition (d) is satisfied. Furthermore, it is not hard to verify that all of the conditions (a)-(c) hold for such η'', and that this $\eta'' \leq 1$ is indeed maximal with respect to satisfying these conditions. So, if we manage to prove that also $\eta'' > \eta'$, our claim will follow.

To this end, observe that by applying Claim 26 with $A = A^{\eta'}$ and $\lambda = \lambda^{\eta'}$, we see that $\tilde{N}(\eta)$ can only decrease for $\eta \geq \eta'$. Therefore, all the derivatives (62) in this process can only decrease too. This implies, in particular, that if some variable stops increasing (i.e., its derivative becomes non-positive) at some point, it will never increase again.

Now, note that if $(i,j) \in A^{\eta'}$ then

- if $y_{i,j}^{\eta'} \in (0,1)$, then, as the derivatives are bounded, it must be the case that $y_{i,j}^\eta \in (0,1)$ until some time $\overline{\eta} > \eta'$;

- if $y_{i,j}^{\eta'} = 0$, then its derivative $\tilde{N}(\eta') - \alpha \lambda_{i,j}^{\eta'}$ at time η' has to be strictly positive (otherwise, (i,j) would be inactive at time η') and thus $y_{i,j}^\eta$ (respectively its derivative) has to stay non-negative (respectively positive) until some time $\overline{\eta} > \eta'$ too;

---

[9]More precisely, $\tilde{N}(\eta)$ is continuous and bounded, as long as each $y_{i,j}^\eta$ is bounded away from $-\beta$. However, we are interested in only analyzing the process as long as all the variables stay non-negative, so for the sake of our analysis, $\tilde{N}(\eta)$ is indeed continuous and bounded.

- finally, if $y_{i,j}^{\eta'} = 1$, then its derivative $\tilde{N}(\eta') - \alpha\lambda_{i,j}^{\eta'}$ at time η' has to be non-positive (as otherwise, $(i,j) \notin A^{\eta'}$). However, by our discussion above, it means that neither this derivative, nor $y_{i,j}^{\eta}$, will ever increase again.

Thus, we can infer from the above that both η^1 and $\overline{\eta}^0$ are strictly larger than η'.

Next, let us focus on some $(i,j) \notin A^{\eta'}$. We have that

- if $y_{i,j}^{\eta'} = 0$, then $\tilde{N}(\eta') - \alpha\lambda_{i,j}^{\eta'}$ has to be non-positive (otherwise, (i,j) would be active at time η') and as $\tilde{N}(\eta)$ never increases, $\tilde{N}(\eta') - \alpha\lambda_{i,j}^{\eta'}$ will never become positive;

- if $y_{i,j}^{\eta'} = 1$, then $\tilde{N}(\eta') - \alpha\lambda_{i,j}^{\eta'}$ at time η' has to be strictly positive (as otherwise, $(i,j) \in A^{\eta'}$). So, this quantity has to remain positive for some time $\overline{\eta} > \eta'$.

Thus, we see that both η^2 and $\overline{\eta}^1$ are also strictly larger than η'.

Observe now that if $y_{i,j}^{\eta'} = y_{i,j+1}^{\eta'}$ for some $(i,j),(i,j+1) \in A^{\eta'}$, we must have that $\lambda_{i,j}^{\eta'} \geq \lambda_{i,j+1}^{\eta'}$. Otherwise, the blocks to which (i,j) and $(i,j+1)$ belong would have been merged at time η'. This means that the derivative of $y_{i,j}^{\eta}$ is always bounded from above by the derivative of $y_{i,j+1}^{\eta}$. Thus, such a pair of coordinates will never violate the monotonicity property. On the other hand, if $y_{i,j}^{\eta} < y_{i,j+1}^{\eta}$, then, as the derivatives are bounded, there always exists $\overline{\eta} > \eta'$ for which this strict inequality still holds (and thus monotonicity is not violated). Hence, we get that $\eta^3 > \eta'$.

Finally, to see that $\eta^4 > \eta'$ as well, we note that if $\sum_{i,j} y_{i,j}^{\eta'} = kd - \kappa(t)$, then, by design, it will remain equal henceforth. If, however, the latter sum is larger than $kd - \kappa(t)$ at time η', then it would still remain so for some $\overline{\eta} > \eta'$.

Thus, indeed we have $\eta'' = \min\{\min_\ell \eta^\ell, \overline{\eta}^0, \overline{\eta}^1, 1\} > \eta'$, concluding the proof of the claim. $\square$

In light of the above claim, one can consider obtaining a feasible configuration $\overline{y}^1$ from the starting (feasible) configuration $\overline{y}^0$ by simply gluing together the trajectories corresponding to the horizons. More precisely, one could start with $\eta_0 = 0$, and for each η_s, with $s \geq 0$ and $\eta_s < 1$, define η_{s+1} to be the horizon of $\overline{y}^{\eta_s}$. Note that, as we start with the feasible configuration $\overline{y}^0$, Claim 27 implies that all $\overline{y}^{\eta_s}$ are well defined and feasible too.

Now, the only reason why the above approach might not end up giving us the desired feasible configuration $\overline{y}^1$ is that, a priori, it is not clear whether the sequence $\{\eta_s\}_s$ ever reaches 1. That is, even though we know that $\eta_0 = 0$ and $\eta_{s+1} > \eta_s$, it might still be possible that this sequence converges without ever reaching 1, and thus there is no s with $\eta_s = 1$.

In order to rule out this possibility, we will prove that the total number of horizons is always finite. Observe that each horizon can be associated with at least one of the following *events*: (a) the set of active coordinates changes, or (b) a block merge occurs, or (c) the number of servers hits the quota. Thus, it suffices to show that the total number of such events is bounded.

To this end, let us note that in our evolution, once $\sum_{i,j} y_{i,j}^{\eta}$ becomes equal to $kd - \kappa(t)$, $N(\eta)$ is chosen so that $\sum_{i,j} dy_{i,j}^{\eta}/d\eta = 0$. So, once we hit the quota we stay there throughout the rest of the hit stage. Hence, there can be at most one event of type (c). Also, as we have already argued, during our evolution blocks never split once they are formed, and thus the total number of block merges (i.e., events of type (b)) can be at most k.

It remains to bound the number of events of type (a), i.e., the ones corresponding to variables becoming active/inactive. For notational convenience, let us say that a coordinate (i,j) 0-*inactivates* (respectively, 1-*inactivates*) at time η_s, for some $s \geq 1$, if $N(\eta_s) \leq \alpha\lambda_{i,j}^{\eta_s}$ (respectively,

$N(\eta_s) > \alpha\lambda_{i,j}^{\eta_s})$ and $y_{i,j}^{\eta_s} = 0$ (respectively, $y_{i,j}^{\eta_s} = 1$), but (i,j) was active at time η_{s-1}. We prove the following claim.

Claim 28. $N(\eta)$ *can increase only at a horizon, i.e., for any $s \geq 0$ with $\eta_s < 1$, $N(\eta_s) \geq N(\eta)$ for $\eta \in [\eta_s, \eta_{s+1})$. Furthermore, if $N(\eta)$ indeed increases at time η_{s+1}, then at η_{s+1} we have an occurrence of either a block merge, or the quota is hit, or a 1-inactivation of some (i,j) with $y_{i,j}^{\eta_s} < 1$.*

Proof. First, consider the case where at time η_s the number of servers is still below the quota. By (59), it means that $N(\eta) = 0 = N(\eta_s)$, for $\eta \in (\eta_s, \eta_{s+1})$, and $N(\eta_{s+1}) = 0$ unless the quota is hit at time η_{s+1}. So, the claim follows in this case, and in the rest of the proof we can assume that the number of servers is already at the quota at time η_s.

First, we prove that $N(\eta_s) \geq N(\eta)$ for $\eta \in [\eta_s, \eta_{s+1})$, i.e., the first part of the claim. Let us fix some $\eta \in [\eta_s, \eta_{s+1})$. Note that by the definition of the horizon, we have that $A^\eta = A^{\eta_s}$ and $\lambda^\eta = \lambda^{\eta_s}$. So, for our purposes, it suffices to show that whenever $A^\eta = A^{\eta^+} = A$, and $\lambda^\eta = \lambda^{\eta^+} = \lambda$, for $\eta^+ = \eta + d\eta$, we have that $N(\eta) \geq N(\eta^+)$. This follows immediately from Claim 26.

Now, to prove the second part of the claim, let us assume that none of the events mentioned in the statement of the claim occurred at time η_{s+1}, otherwise we are already done. So, we have, in particular, that $\lambda_{i,j}^{\eta_s} = \lambda_{i,j}^{\eta_{s+1}} = \lambda_{i,j}$ for each (i,j). This implies that if there is an (i,j) with $y_{i,j}^{\eta_{s+1}} = y_{i,j}^{\eta_s} = 1$ that becomes active at time η_{s+1}, then by Definition 9 it must be the case that

$$N(\eta_s) > \alpha\lambda_{i,j}^{\eta_s} = \alpha\lambda_{i,j}^{\eta_{s+1}} \geq N(\eta_{s+1}).$$

So, in this case $N(\eta_s) \geq N(\eta_{s+1})$, and thus we can restrict ourselves to the scenario in which the only coordinates (i,j) that become active at time η_{s+1} have $y_{i,j}^{\eta_{s+1}} = 0$. As a result, we have

$$A^{\eta_{s+1}} = (A^{\eta_s} \setminus (A_-^0 \cup A_-^1)) \cup A_+,$$

where A_-^0 is the set of coordinates (i,j) such that (i,j) 0-inactivates at time η_{s+1}, A_-^1 contains (i,j)-s which 1-inactivate at that time and $y_{i,j}^{\eta_s} = 1$, and A_+ is the set of (i,j)-s that become active at time η_{s+1} with $y_{i,j}^{\eta_{s+1}} = 0$.

Now, observe that if some $(i,j) \in A_-^1$, then we need to have $\alpha\lambda_{i,j} \geq N(\eta_s)$. Otherwise, (i,j) would have already been inactive at time η_s. Furthermore, we actually need to have $\alpha\lambda_{i,j} = N(\eta_s)$, as otherwise the derivative of $y_{i,j}^\eta$ would be negative in the interval $[\eta_s, \eta_{s+1})$, contradicting the fact that $y_{i,j}^{\eta_{s+1}} = 1$. (Recall that we have already proved that $N(\eta)$ – and thus all the derivatives – do not increase in the interval $[\eta_s, \eta_{s+1})$.)

So, by the above, and Definition 9, we can conclude that

$$\alpha\lambda_{i,j} \geq N(\eta_{s+1}) \qquad \text{for each} \quad (i,j) \in A_-^0, \tag{64}$$

$$\alpha\lambda_{i,j} = N(\eta_s) \qquad \text{for each} \quad (i,j) \in A_-^1, \tag{65}$$

$$\alpha\lambda_{i,j} < N(\eta_{s+1}) \qquad \text{for each} \quad (i,j) \in A_+. \tag{66}$$

On the other hand, we can express $N(\eta_{s+1})$ as the weighted average of $\alpha\lambda_{i,j}$s over the set $A^{\eta_{s+1}}$ (cf. Claim 26), i.e. we have

$$N(\eta_{s+1}) = \frac{\sum_{(i,j)\in A^{\eta_{s+1}}} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A^{\eta_{s+1}}} u_{i,j}^{\eta_{s+1}}},$$

46

where $u_{i,j}^{\eta} = \frac{1}{w_i}\left(y_{i,j}^{\eta} + \beta\right)$.

Now, as $A^{\eta_{s+1}} = (A^{\eta_s} \setminus (A_-^0 \cup A_-^1)) \cup A_+$, we can utilize conditions (64) and (66) to bound $N(\eta_{p+1})$ from above by a corresponding weighted average of $\alpha\lambda_{i,j}$-s over the set $A^{\eta_s} \setminus A_-^1$. In particular, we have

$$
\begin{aligned}
N(\eta_{s+1}) &= \frac{\sum_{(i,j)\in A^{\eta_{s+1}}} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A^{\eta_{s+1}}} u_{i,j}^{\eta_{s+1}}} = \frac{\sum_{(i,j)\in((A^{\eta_s}\setminus(A_-^0\cup A_-^1))\cup A_+)} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in((A^{\eta_s}\setminus(A_-^0\cup A_-^1))\cup A_+)} u_{i,j}^{\eta_{s+1}}} \\
&\leq \frac{\sum_{(i,j)\in(A^{\eta_s}\setminus A_-^1)} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in(A^{\eta_s}\setminus A_-^1)} u_{i,j}^{\eta_{s+1}}},
\end{aligned}
\tag{67}
$$

where the last inequality follows as for any $a, b > 0$, $\frac{a}{b} \leq \frac{a+c_1t_1-c_2t_2}{b+t_1-t_2}$, whenever $c_1 \geq \frac{a}{b}$, $c_2 \leq \frac{a}{b}$, and $t_1 \geq 0$, $b > t_2 \geq 0$.

If the last expression in (67) is at most $N(\eta_s)$, then we are already done. So, let us assume, for the sake of contradiction, that it is strictly larger than $N(\eta_s)$. In this case, by (66), we need to have that also

$$
\frac{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_{s+1}}} > N(\eta_s),
\tag{68}
$$

as for any $a, b, c, t > 0$, if $\frac{a}{b} > c$, then also $\frac{a+ct}{b+t} > c$. However, by applying Claim 26 with $A = A^{\eta_s}$, we have that

$$
\frac{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_{s+1}} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_{s+1}}} \leq \frac{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_s} \cdot \alpha\lambda_{i,j}}{\sum_{(i,j)\in A^{\eta_s}} u_{i,j}^{\eta_s}} = N(\eta_s),
$$

contradicting (68), and thus proving that indeed $N(\eta_s) \geq N(\eta_{s+1})$. The proof of the claim is now concluded. $\qquad\square$

Now, we are ready to bound the number of events of type (a). To show that the number of these events is finite, it suffices to show that the number of 0-inactivations and 1-inactivations is finite. Also, observe that during the period in which the number of servers is below the quota, by definition, we have $N(\eta) = 0$, and thus variables can only decrease. As a result, coordinates can only 0-inactivate in that period, and once they become inactive they stay that way. Hence, we have at most kd such events.

In light of the above, we can focus on analyzing the events after reaching the quota. Note that in this case we can assume that $N(\eta) > 0$. (If $N(\eta) = 0$, then all derivatives are equal to zero, and the desired bounds trivially follow.) As we have that $\lambda_{i,j}^{\eta}$ is always zero when $i \neq \bar{i}$, $N(\eta) > 0$ implies that coordinates (i,j) with $i \neq \bar{i}$ can only increase, and once they 1-inactivate they stay inactive. As a consequence, it suffices to show that the number of 0-inactivations and 1-inactivations is finite for all coordinates (i,j) with $i = \bar{i}$. In order to do so, we prove the following claim.

Claim 29. *The total number of 1-inactivations of coordinates $(\bar{i}, j)$ is finite.*

Proof. We will prove the claim first for $j = k$ and then consider consecutive j-s in decreasing order. As a result, our task is to prove for a given j, that $(\bar{i}, j)$ 1-inactivates a finite number of times, provided that the number of 1-inactivations is finite for all coordinates $(\bar{i}, j')$ with $j < j' \leq k$.

To this end, we argue that whenever there are two consecutive 1-inactivations of some coordinate $(\bar{i}, j)$ – the first one at time $\eta_{s'}$, and the second one at time $\eta_{s''}$ – then in the interval $[\eta_{s'}, \eta_{s''}]$ we

have either a block merge, or the quota is hit, or a 1-activation of a coordinate (i', j') that has either $i' \neq \bar{i}$ or $j' > j$. As we know that the number of occurrences of each of these events is finite, we get the desired proof.

To establish the above, let $\eta_{\bar{s}}$ for $s' < \bar{s} < s''$ be the time in which $(\bar{i}, j)$ is activated between the two 1-inactivations. Observe that as $N(\eta_{\bar{s}}) \leq \alpha\lambda_{\bar{i},j}$ and $N(\eta_{s''}) > \alpha\lambda_{\bar{i},j}$, there is a time η_{s^*} with $\bar{s} < s^* \leq s''$ in which $N(\eta)$ increases above $\alpha\lambda_{\bar{i},j}$. (Recall that by Claim 28 we know that $N(\eta)$ can increase only at horizons.) Without loss of generality we take s^* to be the first $s > \bar{s}$ corresponding to such an increase.

Now, in light of Claim 28, we know that $N(\eta)$ increases at time η_{s^*}. Thus, to conclude our proof it suffices to show that if we have a coordinate (i', j') that 1-inactivates at time η_{s^*} and $y_{i',j'}^{\eta_{s^*}-1} < 1$, then we cannot have $i' = \bar{i}$ and $j' \leq j$.

We consider two cases here. The first one corresponds to $s^* < s''$. In this case we have $y_{\bar{i},j}^{\eta_{s^*}} < 1$, as otherwise $(\bar{i}, j)$ would be 1-inactivated already at time s^*, instead of s''. However, by the monotonicity property (4), we have that $y_{\bar{i},j''}^{\eta_{s^*}} \leq y_{\bar{i},j}^{\eta_{s^*}}$ for all $j'' \leq j$. So, if $i' = \bar{i}$, then (i', j') cannot 1-inactivate at time η_{s^*} if $j' \leq j$, and the claim follows.

Consider now the remaining case of $s^* = s''$. If we have $i' = \bar{i}$ and $j' \leq j$, then we must have $\lambda_{\bar{i},j'} \geq \lambda_{\bar{i},j}$. Otherwise, condition (8) for block merge would trigger at time $\eta_{s''}$. As a result, by (58), we know that in the interval of our interest the derivatives of $y_{\bar{i},j'}^{\eta}$ are bounded from above by the derivatives of $y_{\bar{i},j}^{\eta}$.

Furthermore, we have that the derivative of $y_{\bar{i},j}^{\eta}$ is always non-positive for $\eta \in [\eta_{\bar{s}}, \eta_{s''})$. This is so, as by the definition of s^*, $N(\eta) \leq \alpha\lambda_{\bar{i},j}$ for $\eta \in [\eta_{\bar{s}}, \eta_{s^*}) = [\eta_{\bar{s}}, \eta_{s''})$. As a consequence, we must have both $y_{\bar{i},j}^{\eta}$ and $y_{\bar{i},j'}^{\eta}$ to be equal to 1 for all $\eta \in [\eta_{\bar{s}}, \eta_{s''})$, as otherwise these variables would not be able to reach 1 at time $\eta_{s''}$. This, however, contradicts the fact that $y_{i',j'}^{\eta_{s^*}-1}$ has to be strictly smaller than 1, as $s^* - 1 = s'' - 1 \geq \bar{s}$. Thus, we cannot have $i' = \bar{i}$ and $j' \leq j$ and our claim follows. $\qquad\square$

Finally, it remains to bound the number of 0-activations of coordinates $(\bar{i}, j)$. We do this by simply noting that if there are two consecutive 0-activations of some coordinate $(\bar{i}, j)$, then $N(\eta)$ has to increase at least once between these two events. But, by Claim 27, it means that one of the events, (whose total number is already bounded), would also occur in this period. Therefore, the number of 0-activations is also finite and we can conclude the proof of Lemma 11.

B Proof of Lemma 22

Here we prove Lemma 22. As mentioned earlier, the proof is implicit in the work of [16], and we make it explicit here for completeness. We begin with some notation, and state another result that we need.

Let M be an arbitrary metric space. Let $C[0]$ denote the configuration specifying the initial location of the k-servers. We assume that the servers are labeled, so for every $k' \leq k$, the first k' entries of $C[0]$ specify the location of the first k' servers. Let ρ be some fixed k-server request sequence. Let $\mathrm{Opt}(k', X)$ denote the optimum cost of serving ρ with k' servers on M, starting in $C[0]$ and ending in configuration X (for notational ease, we are suppressing the dependence on $\rho, M, C[0]$ here). Let $\mathrm{Opt}(k') = \min_X \mathrm{Opt}(k', X)$, denote the minimum cost of server ρ starting in $C[0]$.

Lemma 30 ([16], Corollary 2). *Let ρ be some fixed request sequence and $C[0]$ be some fixed initial configuration. For any $k_1, k_2 \in [k]$, given any state X on k_1 locations, there exists another state Y such that*

1. *$|X \cap Y| = \min(|X|, |Y|)$, i.e. Y overlaps with X as much as possible, and*

2. *$\mathrm{Opt}(k_2, Y) \leq \mathrm{Opt}(k_2) + \mathrm{Opt}(k_1, X) - \mathrm{Opt}(k_1)$. That is, the excess cost incurred for an optimum k_2-server solution to end in Y, is no more than the excess cost incurred for the optimum k_1-server to end in X.*

This lemma and its proof can be found in [16] (Corollary 2).

Let T be a weighted σ-HST. Again, for notational convenience, let us drop ρ, $C[0]$, and the underlying metric T from the notation (these remain the same, and dropping them will not cause any confusion). Given a quota pattern κ, recall the definition of $\mathrm{Optcost}(\kappa, t)$ as the optimum cost of serving ρ until time t with quota pattern κ. We also use $\mathrm{Optcost}(\kappa) = \mathrm{Optcost}(\kappa, \infty)$ to denote the optimum cost of serving the entire sequence ρ. As previously, let us define $h^t(\kappa) = \mathrm{Optcost}(\kappa(t) \cdot \vec{1}, t) - \mathrm{Optcost}(\kappa(t) \cdot \vec{1}, t - 1)$ and $g(\kappa) = \sum_{t \geq 1} |\kappa(t) - \kappa(t - 1)|$. Let D denote the diameter of T.

We will prove the following, which is the same as Lemma 22. (In that notation, note that $\Delta \leq W(p)/(\sigma - 1)$ for $T(p)$.)

Theorem 31.
$$\sum_t h^t(\kappa) - \Delta \cdot g(\kappa) \leq \mathrm{Optcost}(\kappa) \leq \sum_t h^t(\kappa) + \Delta \cdot g(\kappa).$$

Proof. We do an induction on the value of $g(\kappa)$. In the base case, when $g(\kappa) = 0$, the vector κ is constant throughout, say $\kappa = k \cdot \vec{1}$. In this case, the claimed result holds trivially as the sum over h^t telescopes and we obtain

$$\sum_t h^t(\kappa) = \sum_t (\mathrm{Optcost}(k \cdot \vec{1}, t) - \mathrm{Optcost}(k \cdot \vec{1}, t - 1)) = \mathrm{Optcost}(\kappa).$$

So, let κ be such that $g(\kappa) > 0$. Let τ be the earliest time when $\kappa(\tau) \neq \kappa(\tau + 1)$. Define a new quota pattern κ' as

$$\kappa'(t) = \begin{cases} \kappa(\tau + 1) & \text{if } t \leq \tau \\ \kappa(t) & \text{if } t > \tau. \end{cases}$$

Note that both κ and κ' are constant for $t \leq \tau$. Also, $g(\kappa') = g(\kappa) - |\kappa(\tau + 1) - \kappa(\tau)| < g(\kappa)$, and hence we can inductively assume that the claimed result holds for κ'.

We first show that $\mathrm{Optcost}(\kappa) \geq \sum_t h^t(\kappa) - D \cdot g(\kappa)$. Fix some solution that attains cost $\mathrm{Optcost}(\kappa)$ and let $C[t]$ denote its configuration at time t. Applying Lemma 30 with $X = C(\tau)$ and $k_1 = \kappa(\tau)$ and $k_2 = \kappa'(\tau)$, there is some configuration Y satisfying

$$\mathrm{Optcost}(k_2 \cdot \vec{1}, \tau, Y) \leq \mathrm{Optcost}(k_2 \cdot \vec{1}, \tau) + \mathrm{Optcost}(k_1 \cdot \vec{1}, \tau, X) - \mathrm{Optcost}(k_1 \cdot \vec{1}, \tau) \quad (69)$$
$$|X \cap Y| = \min(k_1, k_2). \quad (70)$$

We construct a solution S' corresponding to κ' as follows: Until time τ, S' follows the solution $\mathrm{Optcost}(k_2 \cdot \vec{1}, \tau, Y)$. Then, after serving the request at $t = \tau$, it switches to state $C[\tau] = X$, and henceforth for $t > \tau$ sets its the configurations $C'[t] = C[t]$. Now

$$\mathrm{cost}(S') = \mathrm{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + c(Y, X) + Q,$$

49

where $c(Y, X)$ is the cost of moving from state Y to X, and Q is the contribution of the solution Optcost(κ) starting from time τ and state X (recall that $C[\tau] = X$).

It is easily checked that the solution S' constructed above is feasible for quota pattern κ'. As the optimum solution for κ' can only be better, cost$(S') \geq$ Optcost(κ') and since Optcost$(\kappa') \geq \sum_t h^t(\kappa') - D \cdot g(\kappa')$ by the inductive hypothesis, it follows that

$$\sum_t h^t(\kappa') - D \cdot g(\kappa') \leq \text{cost}(S') = \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + c(Y, X) + Q. \tag{71}$$

By (70), $|Y \cap X| = \min(k_1, k_2)$ and hence $c(Y, X) \leq D|k_2 - k_1| = D|\kappa(\tau + 1) - \kappa(\tau)|$. Thus $c(Y, X) + D \cdot g(\kappa') \leq D \cdot g(\kappa)$, and hence (71) implies that

$$Q \geq \sum_t h^t(\kappa') - D \cdot g(\kappa) - \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y). \tag{72}$$

On the other hand, as $X = C[\tau]$ we have

$$\text{Optcost}(\kappa) = \text{Optcost}(\kappa, \tau, X) + Q = \text{Optcost}(k_1 \cdot \vec{1}, \tau, X) + Q.$$

Thus by (72),

$$
\begin{aligned}
\text{Optcost}(\kappa) \;\geq\;& \text{Optcost}(k_1 \cdot \vec{1}, \tau, X) - \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + \sum_t h^t(\kappa') - D \cdot g(\kappa) \\
\geq\;& \text{Optcost}(k_1 \cdot \vec{1}, \tau) - \text{Optcost}(k_2 \cdot \vec{1}, \tau) + \sum_t h^t(\kappa') - D \cdot g(\kappa) \qquad (73) \\
=\;& \sum_t h^t(\kappa) - D \cdot g(\kappa), \qquad (74)
\end{aligned}
$$

implying the desired lower bound. Here (73) follows from (69), and (74) follows from (73) since

$$
\begin{aligned}
\sum_{t=1}^{\tau} h^t(\kappa) &= \text{Optcost}(k_1 \cdot \vec{1}, \tau), \\
\sum_{t=1}^{\tau} h^t(\kappa') &= \text{Optcost}(k_2 \cdot \vec{1}, \tau),
\end{aligned}
$$

and $\sum_t h^t(\kappa) - \sum_t h^t(\kappa') = \sum_{t=1}^{\tau} h^t(\kappa) - \sum_{t=1}^{\tau} h^t(\kappa')$, since κ, and κ' are the same for any $t > \tau$.

We now show show the upper bound on Optcost(κ). The proof is similar to the one above. Let κ' be defined as previously. Let $\{C'[t]\}_t$ denote the configurations for some fixed solution that has value Optcost(κ'). Applying Lemma 30 with $X = C'[\tau]$, $k_1 = \kappa'(\tau)$ and $k_2 = \kappa(\tau)$, we obtain a configuration Y satisfying

$$
\begin{aligned}
\text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) \;\leq\;& \text{Optcost}(k_2 \cdot \vec{1}, \tau) + \text{Optcost}(k_1 \cdot \vec{1}, \tau, X) - \text{Optcost}(k_1 \cdot \vec{1}, \tau) \quad (75) \\
|X \cap Y| \;=\;& \min(k_1, k_2). \qquad (76)
\end{aligned}
$$

Consider the following solution S corresponding to κ: Until time τ, S mimics the solution Optcost$(k_2 \cdot \vec{1}, \tau, Y)$. Then, after serving the request at $t = \tau$, it switches to state $C'[\tau] = X$, and henceforth for $t > \tau$ sets its the configurations $C[t] = C'[t]$. Now

$$\text{cost}(S) = \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + c(Y, X) + Q,$$

where Q is the cost of solution $\text{Optcost}(\kappa')$ incurred from time τ starting at state Y. Again S is feasible for quota κ, and hence $\text{Optcost}(\kappa) \leq \text{cost}(S)$. By definition of X,

$$\text{Optcost}(\kappa') = \text{Optcost}(k_1 \cdot \vec{1}, \tau, X) + Q.$$

Thus,

$$\text{Optcost}(\kappa) \leq \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + c(Y, X) + \text{Optcost}(\kappa') - \text{Optcost}(k_1 \cdot \vec{1}, \tau, X).$$

As $\text{Optcost}(\kappa') \leq \sum_t h^t(\kappa') + D \cdot g(\kappa')$ by the inductive hypothesis and $c(Y, X) + D \cdot g(\kappa') \leq D \cdot g(\kappa)$, we obtain

$$
\begin{aligned}
\text{Optcost}(\kappa) \quad &\leq \quad \text{Optcost}(k_2 \cdot \vec{1}, \tau, Y) + \sum_t h^t(\kappa') + D \cdot g(\kappa) - \text{Optcost}(k_1 \cdot \vec{1}, \tau, X) && (77) \\
&\leq \quad \text{Optcost}(k_2 \cdot \vec{1}, \tau) - \text{Optcost}(k_1 \cdot \vec{1}, \tau) + \sum_t h^t(\kappa') - D \cdot g(\kappa) && (78) \\
&= \quad \sum_t h^t(\kappa) - D \cdot g(\kappa), && (79)
\end{aligned}
$$

implying the desired inequality. Here (78) follows from (75), and (79) follows by noting that

$$\sum_t h^t(\kappa') - \sum_t h^t(\kappa) = \sum_{t=1}^{\tau} h^t(\kappa') - \sum_{t=1}^{\tau} h^t(\kappa) = \text{Optcost}(k_1 \cdot \vec{1}, \tau) - \text{Optcost}(k_2 \cdot \vec{1}, \tau).$$

$\square$