

Unfair Metrical Task Systems on HSTs and Applications

Nikhil Bansal *

Niv Buchbinder †

Joseph (Seffi) Naor ‡

Abstract

We consider a problem introduced by Cote et al [7] as a very promising approach for obtaining poly-logarithmic competitive randomized algorithms for the k -server problem. Solving this problem on an arbitrary uniform metric would imply a poly-logarithmic guarantee for the k -server problem. Cote et al solved this problem for the space of two points, which enabled them to obtain the first poly-logarithmic competitive algorithm for well-separated *binary* Hierarchically Separated Trees (HSTs). In this work, we extend their result to an arbitrary number of points. While our result is not strong enough to obtain poly-logarithmic guarantees for k -server, it still implies the following new results:

1. An $O(d\ell \log^2 k \log(k\ell))$ -competitive algorithm for k -server on d -ary HSTs with height ℓ provided $\alpha = \Omega(d\ell \log^2 k \log(k\ell))$. Prior to our work a poly-logarithmic guarantee was known only for binary HSTs (i.e. $d = 2$) provided $\alpha = \Omega(\ell)$ [7], and it was unclear how to obtain such guarantees even for the case of $d = 3$.
2. An $O(\log n \cdot \exp(O(\sqrt{\log \log k \log n})))$ -competitive algorithm for k -server on n uniformly spaced points on a line. This substantially improves upon the prior guarantee of $O(\min(k, n^{2/3}))$ for this metric [8]. More generally, our result implies $O(\log n \cdot \exp(O(\sqrt{\log \log k \log \Delta})))$ -competitive algorithm for k -server on a line of length Δ with n points.

These results are based on obtaining a refined guarantee for the Unfair Metrical Task Systems problem on an HST. Prior to our work, such a guarantee was only known for the case of the uniform metric. Our algorithm and analysis are based on an enhancement of the primal-dual approach for online algorithms, which could be of independent interest.

*IBM T. J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: nikhil@us.ibm.com

†Microsoft Research, Cambridge, MA. E-mail: nivbuchb@microsoft.com

‡Technion, Haifa, Israel. E-mail: naor@cs.technion.ac.il

1 Introduction

The k -server problem is one of the most central and well studied problems in competitive analysis and is considered by many to be the “holy grail” problem in the field. In the k -server problem, there is a distance function d defined over an n -point metric space and k servers located at 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 it is served by moving a server to the requested point. The cost is defined to be the distance traveled by the server. Thus, the goal of an online algorithm is to minimize the total sum of the distances traveled by the servers so as to serve a given sequence of requests. The k -server problem can model many problems, and one of the most widely studied problems is *paging*. In this problem 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 evicting some other page) at a cost of one unit. Paging is the special case of k -server on a uniform metric. In their seminal paper on competitive analysis, Sleator and Tarjan [15] gave k -competitive algorithms for paging, and also showed that this is the best possible for any deterministic algorithm.

The k -server problem in its full generality was first posed by Manasse et al. [13]. They conjectured that there is a k -competitive online algorithm in an arbitrary metric space and for any value of k . This is called the *k -server conjecture*. Fiat et al. [9] were the first to prove that there exists an online algorithm for any metric space with competitive ratio which depends only on k . Following a sequence of results, a major breakthrough was achieved by Koutsoupias and Papadimitriou [12] who showed that the work function algorithm is $2k - 1$ competitive. We note that for special metrics such as the uniform metric, lines, and more generally trees [6], a competitive factor of k is known for the k -server problem.

The competitive ratio for k -server seems to substantially improve if randomization is allowed. For example, Fiat et al. [10] gave a randomized $2H_k$ -competitive algorithm for paging, where H_k is the k -th Harmonic number. They also showed that any randomized algorithm is at least H_k -competitive. In fact, this lower bound applies to any metric space. The *randomized k -server conjecture* states that there is an $O(\log k)$ -competitive randomized algorithm for the k -server problem in an arbitrary metric space against an *oblivious* adversary. However, despite much interest, the randomized case remains poorly understood and $o(k)$ upper bounds are known for very few special cases. In addition to paging, poly-logarithmic competitive algorithms are known only for general metrics on $n = k + O(1)$ points, for certain well-separated spaces [14], and for weighted paging [2]. Even for the seemingly simple case of n uniformly spaced points on a line, the best known result is an $O(n^{2/3})$ -competitive algorithm [8], which is $o(k)$ -competitive for $n = o(k^{3/2})$. Given our current lack of understanding, a major breakthrough would be to even resolve a weaker variant of the randomized k -server conjecture – obtaining an $\text{polylog}(k, n, \Delta)$ -competitive algorithm for general metrics, where Δ is the diameter. Since a general metric space can be embedded into a probability distribution over Hierarchically Separated Trees (HSTs) with logarithmic distortion of the distances, it suffices to consider the k -server problem on HSTs to obtain the latter bound.

1.1 Metrical Task System and Allocation Problems

The *metrical task system problem* (MTS) was introduced by Borodin et al. [4] as a generalization of various online problems. In this problem there is a machine, or server, which can be in any one of n states $1, 2, \dots, n$, and a metric d which defines the cost of moving between states. At each time step t a new task appears and needs to be served. The new task is associated with a cost vector $c_t = (c_t(1), \dots, c_t(n))$ denoting the processing cost in each of the states. To serve the task, the machine is allowed to move to any other state from its current state. Assuming the machine is currently in state i and it moves to state j , then the

cost of serving the task is $d_{ij} + c_t(j)$. The goal is to minimize the total cost. Borodin et al. [4] gave a tight deterministic $2n - 1$ competitive algorithm for any metric, and a $\Theta(\log n)$ -competitive randomized algorithm for the uniform metric.

Later on, in a breakthrough paper, Bartal et al. [3] obtained the first polylogarithmic competitive randomized algorithm for general metrics by recursively solving the problem on HSTs. To this end, [3] defined a more general problem called the *unfair* MTS problem on a uniform metric. It is similar to MTS, but has an extra parameter $r \geq 1$. Given a vector c_t , the online algorithm pays $r \cdot c_t(i)$ in state i , while the offline algorithm pays $c_t(i)$. The movement costs are the same for both offline and online. Using techniques from online learning, [3] gave an $r + O(\log n)$ -competitive algorithm for unfair MTS. (In contrast, “standard” algorithms for MTS can only give a guarantee of $O(r \log n)$ on the competitive factor.) The idea is that each state in the unfair MTS problem represents a subtree of an HST and r is going to be the competitive ratio achieved within this subtree. Applying the algorithm of [3] recursively on an HST, intuitively, $r + O(\log n)$ becomes the parameter of the cost ratio for the next level of the HST (and so on), thus implying an $O(\ell \log n)$ -competitive algorithm for an HST of depth ℓ . Bartal et al. [3] further refined their approach to remove dependence on the diameter of the HST. Currently, the best known guarantee for the MTS problem on an HST is $O(\log n \log \log n)$, which is almost optimal, due to Fiat and Mendel [11].

Recently, Coté et al. [7] gave the first completely formal approach to solving the k -server problem on HSTs, continuing the successful line of work on the MTS problem that approached the problem by recursively solving it on a uniform metric. The problem defined by Coté et al. [7] on uniform metrics is called the *allocation problem* by [1]. Coté et al. [7] were able to solve the allocation problem only on a metric space with two points, allowing them to obtain a $O(\log \Delta)$ -competitive algorithm for well separated *binary* HSTs. Given the relative simplicity of uniform metrics, such an approach seems quite promising; we note that it has already been applied very successfully to the related (but easier) Metrical Task Systems (MTS) problem.

The allocation problem is defined as follows. There is a uniform metric on n points and there are up to k available servers. At time step t , the total number of available servers $k(t) \leq k$ is specified, and a request arrives at some point i_t . The request is specified by a $(k + 1)$ -dimensional vector $\vec{h}^t = (h^t(0), h^t(1), \dots, h^t(k))$, where $h^t(j)$ denotes the cost when serving the request using j servers. Upon receiving a request, the algorithm may choose to move additional servers to the requested point and then serve it. The cost is divided into two parts. The *Move-Cost* incurred for moving the servers, and the *Hit-Cost*, $h^t(j)$, determined by the cost vector. The goal is to minimize the total cost. In addition, 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 less resources costs more.

Denote by Optcost the optimal cost of an instance of the allocation problem. Coté et al. [7] showed that if there is an online algorithm that incurs a hit cost of $(1 + \varepsilon)\text{Optcost}$ and a move cost of Optcost times a polylogarithmic factor, then there is a polylogarithmic competitive algorithm for the k -server problem on general metrics. More generally, the next theorem implicitly follows from their work.

Theorem 1.1 ([7, 1]). *Suppose there is a $(1 + \varepsilon, \beta(\varepsilon))$ -competitive algorithm for the allocation problem on a uniform metric on d points. Let H be a d -ary HST with depth ℓ and parameter α . Then, for any $\varepsilon \leq 1$, there is a*

$$\beta(\varepsilon)\gamma^{\ell+1}/(\gamma - 1)$$

competitive algorithm for k -server on H , where

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

Recently, Bansal et al. [1] showed how to obtain a $(1 + \epsilon, \ln(k/\epsilon))$ -algorithm for the allocation problem with an additional convexity restriction on the requests. Although this restriction does not hold directly in the reduction of [7], Bansal et al. [1] conjectured that this property still holds in some aggregate way that allows the reduction. However, it is left as a major open question whether a similar result holds for the (original) allocation problem of [7] on a uniform metric with an arbitrary number of points.

1.2 Our Results

We build on the approach of Coté et al. [7] and design an algorithm for the allocation problem on any uniform metric with a competitive factor that depends on the number of points d . We show that this result implies a non-trivial sublinear competitive ratio for a line metric. Our algorithm for the allocation problem is based on a new algorithm for the metrical task system problem with refined guarantees. Specifically, we show how to solve the allocation problem on d nodes with the following guarantee.

Theorem 1.2. *There exists an algorithm that is $O(1+\epsilon, O(d \log^2 k \log(k/\epsilon)))$ -competitive for the allocation problem on d nodes.*

Exploring the meaning of such a result in the k -server context, let us define an (ℓ, d, α) -HST as an HST with height ℓ , maximum degree d , and stretch factor α . Given such an HST and combining the above result with Theorem 1.1 we get the following.

Theorem 1.3. *Given an (ℓ, d, α) -HST metric, then for any $\epsilon \leq 1$ there exists an online algorithm for the k -server problem with competitive factor:*

$$O \left(\min(\alpha, \beta(\epsilon)/\epsilon) \cdot \left(1 + \epsilon + O\left(\frac{d \log^2 k \log(k/\epsilon)}{\alpha}\right) \right)^{\ell+1} \right).$$

Choosing $\epsilon = \frac{1}{\ell}$ and $\alpha = \Omega(\ell d \log^2 k \log(k\ell))$, the algorithm is $O(\ell d \log^2 k \log(k\ell))$ -competitive.

Applying Theorem 1.3 along with standard embedding results for equally spaced points on the line we get the following Theorem*.

Theorem 1.4. *There is an online algorithm for the k -server problem on the line with competitive ratio:*

$$O \left(\alpha^2 \log n \cdot \left(1 + \epsilon + O(\log^2 k \log(k/\epsilon)) \right)^{\frac{\log n}{\log \alpha} + 1} \right).$$

Setting $\epsilon = 1$ and $\alpha = \exp(O(\sqrt{\log \log k \log n}))$, the competitive ratio is $O(\log n \cdot \exp(O(\sqrt{\log \log k \log n})))$.

This is the first online algorithm with a sublinear competitive ratio for the line, providing a strong evidence that randomization does help for this kind of metric.

1.2.1 Techniques

In order to prove Theorem 1.2 we design here a new MTS algorithm with refined guarantees which is interesting on its own. Specifically, the allocation problem on d points can be cast as an MTS problem on an HST with $(k+1)^d$ nodes, $O(\log k)$ height, and $O(\log k)$ stretch factor. We remark that this reduction does not use the fact that the hit costs in the allocation problem are monotonic, and so our result applies to a more general setting. We then design a general result for the MTS problem on an HST of height ℓ with n points. We obtain the following result.

*For a general line on n points and length Δ our guarantee is $O(\log n \cdot \exp(O(\sqrt{\log \log k \log \Delta})))$.

Theorem 1.5. *For any $\gamma < 1$, there is an online algorithm for the MTS problem on an HST of height ℓ such that:*

- *Its movement cost is bounded by $8\ell \ln(1 + n/\gamma)$ times the optimal cost.*
- *Its service cost is bounded by $(1 + \gamma)$ times the optimal cost.*

Although this result does not compete with the best algorithm known for the MTS problem on an HST in terms of movement cost, it has the refined guarantee on the service cost that allows us to obtain all the above interesting results. Furthermore, we believe that the techniques used to achieve this new result are interesting on their own, and should eventually lead to an optimal algorithm for the MTS problem on an HST. As opposed to previous approaches for solving the MTS on an HST that are based on sophisticated composition of tailored algorithms for the uniform metric, our algorithm is a “one shot algorithm”. Our design is based on an application of the primal-dual framework for designing online algorithms developed by Buchbinder and Naor [5]. Specifically, we cast the (offline) MTS problem on an HST as a minimization linear program. We then obtain the dual maximization problem that is used to bound the cost of our online solution. The main (and only) decision we make in our algorithm is about a function that relates the probability of being inside a sub-tree of the HST to a corresponding dual variable (of the same sub-tree). After making this binding decision, the algorithm is in a sense dictated to us by *consistency constraints* posed on the probabilities, and by a natural targeting of keeping the dual constraints tight. This makes our algorithm simple to understand and also might enable an optimization of the function we use that will tighten the competitive ratio. Our analysis is based on solving a set of linear equations that relates the change of probabilities in all sub-trees of the HST when getting a new request. We note that several technical ideas are needed in extending the unfair MTS guarantee from uniform metric to HSTs. In general, it is not clear at all how to achieve this. For example, we can obtain poly-logarithmic guarantees for (even the unfair version) of the k -server problem on the uniform metric, but we do not know how to extend this to even a height two HST. In fact, being able to do this for k -server would imply a poly-logarithmic guarantee for k -server.

2 Preliminaries

Let us formally define a Hierarchically well-Separated Tree (HST) with stretch factor α . We denote the leaves of an HST by $1, \dots, n$ and use i to index them. In an HST all leaves have the same depth, denoted by ℓ (where we use the convention that a star has depth 1). The root is denoted by r and it is at level 0. Let $\ell(v)$ denote the level of node v . Then, the distance of v to its parent is $\alpha^{\ell - \ell(v)}$ and so the diameter of the HST is $O(\alpha^{\ell - 1})$. Let T_v denote the set of leaves in the subtree rooted at node v . For a leaf i , let (i, j) denote the j -th ancestor of i . That is $(i, 0)$ is i itself, $(i, 1)$ is the parent of i , and so on. Note that (i, ℓ) is the root r for any leaf i . Let $p(v)$ be the parent node of v and let $C(v)$ be the set of children of v . The number of children of v is denoted by $|C(v)|$; if v is a leaf then we use the convention $|C(v)| = 1$.

We study here the metrical task system (MTS) problem on a metric \mathcal{M} defined by an HST. By a standard transformation, we can assume that the cost vector at any time has the form $c \cdot e_i$, where c is arbitrarily small and $e_i = (0, 0, \dots, 1, 0, \dots)$ has 1 in the i th position. Hence, we use i_t to denote the location with non-zero cost at time t . Our algorithm keeps a fractional solution in which each leaf i has mass $y_{i,t}$ at time t . Let $y_{v,t}$ denote the total mass in the subtree rooted at v at time t . Then, by definition $y_{v,t} = \sum_{w \in C(v)} y_{w,t} = \sum_{i \in T_v} y_{i,t}$. Note that this is consistent with the definition for leaves.

2.1 LP Formulation

We now present our linear programming formulation of the MTS problem on an HST. Let $y_{i,t}$ be the amount of mass the algorithm maintains in leaf i at time t (we will always use i to index the leaves). It is also convenient to define $y_{v,t} = \sum_{i \in T_v} y_{i,t}$, though we will use the variables $y_{i,t}$ corresponding to leaves in the LP. Let $z_{v,t}$ denote the amount of probability mass in leaves of T_v that decreases at time t . Note that there is no need to define a variable $z_{r,t}$ (for the root) and without loss of generality it suffices to charge only for removing mass from a subtree (and not for introducing more mass to a subtree). The linear program is as follows.

$$(P) \quad \min \quad \sum_{v,t} \alpha^{\ell-\ell(v)} z_{v,t} + \sum_t c \cdot y_{i_t,t}$$

$$\text{For any time } t: \quad \sum_i y_{i,t} = 1 \quad (1)$$

$$\text{For any time } t \text{ and subtree } T_v \text{ (} v \neq r \text{):} \quad z_{v,t} \geq \sum_{i \in T_v} (y_{i,t-1} - y_{i,t}) \quad (2)$$

It is easy to verify that the formulation is valid. We now define the dual program. We associate variables a_t and $b_{v,t}$ with the above constraints. For notational convenience, let $\Delta b_{v,t+1} = b_{v,t+1} - b_{v,t}$. If v is the j -th parent of leaf i , we will use the notation v and (i,j) interchangeably (especially if we need to specify the relation of v to i).

$$(D) \quad \max \quad \sum_t a_t$$

$$\text{For any time } t \text{ and leaf } i \neq i_t: \quad a_t - \sum_{j=0}^{\ell-1} \Delta b_{(i,j),t+1} \leq 0 \quad (3)$$

$$\text{For any time } t \text{ and leaf } i_t: \quad a_t - \sum_{j=0}^{\ell-1} \Delta b_{(i_t,j),t+1} \leq c \quad (4)$$

$$\text{For any time } t \text{ and subtree } T_v: \quad b_{v,t} \leq \alpha^{\ell-\ell(v)} \quad (5)$$

Equivalently, the last constraint can be written as $b_{(i,j),t} \leq \alpha^j$ for any time t , leaf i , and index j .

3 The MTS Algorithm

The algorithm for the MTS problem on an HST is based on a two-step approach. First, we show how to maintain online a fractional solution, then we show how to transform the fractional online algorithm into a randomized algorithm. The randomized algorithm is going to maintain a primal solution (a distribution on states) and a corresponding dual solution. The following invariant, defining a relation between the value of the dual variables and the primal variables, is maintained throughout the execution of the algorithm. Specifically, the value of variable $y_{v,t}$ (amount of mass in subtree T_v) is a function of the dual variable $b_{v,t+1}$:

$$y_{v,t} \triangleq f(b_{v,t+1}) \triangleq \frac{\gamma \cdot |C(v)|}{n} \left(\exp \left(\frac{\ln(1+n/\gamma) b_{v,t+1}}{\alpha^{\ell-\ell(v)}} \right) - 1 \right). \quad (6)$$

Let $\gamma < 1$ be some arbitrary constant. Recall that if v is a leaf then $|C(v)| = 1$. Before continuing with the description of the algorithm, we derive several properties that are a consequence of the invariant. These properties are somewhat technical, and the reader may wish to go to section 3.2 and refer back to this part as necessary.

3.1 Properties of the function f

The first property is a simple observation.

Observation 3.1. *If, during the execution of the algorithm, $0 \leq y_{i,t} \leq 1$, then $0 \leq b_{v,t+1} \leq \alpha^{\ell-\ell(v)}$. Moreover $y_{v,t} = 0$ whenever $b_{v,t+1} = 0$.*

The observation follows by simple properties of the function f . A second simple property is about the derivative of the function f .

$$\frac{dy_{v,t}}{db_{v,t+1}} = \frac{\ln(1+n/\gamma)}{\alpha^{\ell-\ell(v)}} \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right). \quad (7)$$

We next examine additional properties of f that are going to be crucial to the analysis. Due to lack of space the proofs appear in Appendix A. The first property is *consistency*. Since, at all times, $y_{v,t} = \sum_{w \in C(v)} y_{w,t}$, it means that when increasing or decreasing $y_{v,t}$, it is also necessary to increase and decrease the children of v at a certain rate. The following lemma states the connection between these rates.

Lemma 3.2 (Consistency). *Suppose we increase variable $b_{v,t+1}$ for node v at rate r . Then:*

$$\frac{1}{\alpha} \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right) \cdot \frac{db_{v,t+1}}{dr} = \sum_{w \in C(v)} \frac{db_{w,t+1}}{dr} \cdot \left(y_{w,t} + \frac{\gamma \cdot |C(w)|}{n} \right)$$

We need the following two special cases of Lemma A.1. The first case happens when $b_{v,t+1}$ is increased at rate r , and we need to increase all the children of v at the same rate. From Lemma A.1, we get the following lemma.

Lemma 3.3 (Consistency: Equal Rates). *Let v be a node. Suppose we increase (or decrease) the probability mass in subtree T_v by increasing (decreasing) $b_{v,t+1}$ at rate r . Then, consistency is maintained for each child $w \in C(v)$ by setting for each child w :*

$$\frac{db_{w,t+1}}{dr} = \frac{1}{\alpha} \cdot \frac{db_{v,t+1}}{dr}.$$

Applying Lemma A.2 recursively, we get the following corollary.

Corollary 3.4. *Let v be a node, $\ell(v) = j$, and suppose that $b_{v,t+1}$ is increased (decreased) at rate r . Consider a path p from leaf $i \in T_v$ to v . The sum of the derivatives of $b_{v',t}$ of nodes $v' \in p$ is:*

$$\frac{db_{v,t+1}}{dr} \cdot \delta(j),$$

where $\delta(j) \triangleq \left(1 + \frac{1}{\alpha} + \frac{1}{\alpha^2} + \dots + \frac{1}{\alpha^{j-1}}\right) = (1 + O(1/\alpha))$.

The second special case of Lemma A.1 that we need is the following. Let v be a node and let w_1 be the first child of v . Suppose we increase (decrease) $b_{w_1,t+1}$ at some rate and then use the same rate for increasing (decreasing) $b_{w',t+1}$, for each child $w' \in C(v)$, $w' \neq w_1$. Also, suppose that $b_{v,t+1}$ is not changed. Then the following should hold to maintain consistency.

Lemma 3.5 (Consistency Case 2). *Let v be a node, and let w_1, \dots, w_m be its children. Suppose we increase (decrease) w_1 at rate r and also increase w_2 to w_m at the same rate r . Then, for $i \geq 2$,*

$$\frac{db_{w_i,t+1}}{dr} = \frac{db_{w_{i+1},t+1}}{dr} \triangleq \frac{db_{w',t+1}}{dr}.$$

If we want to keep the amount y_v unchanged then:

$$\frac{db_{w',t+1}}{dr} = \frac{\left(y_{w_1,t} + \frac{\gamma \cdot |C(w_1)|}{n}\right)}{\left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n}\right)} \cdot \left(-\frac{db_{w_1,t+1}}{dr} + \frac{db_{w',t+1}}{dr}\right).$$

3.2 The Algorithm

We are now ready to describe our algorithm. At any time t , the algorithm maintains a distribution on the leaves $y_{i,t}$. We describe how this distribution is updated upon arrival of a new request. Suppose, without loss of generality, that the request at time t arrives at leaf 1 with cost $(c, 0, 0, \dots, 0)$. The algorithm maintains the following three invariants:

1. All dual constraints of type (3) on leaves $2, 3, \dots, n$ are tight.
2. Either $y_{1,t} = 0$, or the dual constraint (4) on leaf 1 is tight. .
3. For each node v , $y_v = \sum_{w \in C(v)} y_w$.

The high level idea of the algorithm is very simple. Since the cost is non-zero only at leaf 1, we will move some probability mass of out of leaf 1, and distribute it to various other leaves. It will in fact turn out that the invariants mentioned above will completely determine how much mass is moved to which leaf (there is no freedom available to our algorithm!). So, this essentially specifies the algorithm. Below, we calculate these quantities explicitly.

Initially, Invariant (3) holds (by induction on the steps of the algorithm). Suppose that the value of the dual variables remains unchanged, then all the dual constraints for leaves $2, 3, \dots, n$ remain valid, and thus Invariant (1) holds. If also $y_{1,t} = 0$, then Invariant (2) holds and we are done. In this case the primal cost is zero and also the dual profit is zero. However, if $y_{1,t} > 0$, then Invariant (2) does not hold.

To make Invariant (2) hold as well, we start increasing a_t at rate 1. However, doing so violates the dual constraints on leaves $2, 3, \dots, n$, forcing us to raise other dual variables to keep these constraints tight. Raising these variables may also violate consistency (Invariant (3)), and thus leading to the update of other dual variables. This process results in transferring mass from leaf 1 to leaves $2, 3, \dots, n$, and also the constraint on leaf 1 becomes more tightened. We stop the updating process when either the constraint on leaf 1 is tight or when $y_{1,t} = 0$. We next describe this process more formally.

Consider the root, and let 1_j , $1 \leq j \leq \ell$, denote the level j node that contains leaf 1 (1_ℓ is the leaf itself). At each step we would like to keep all dual constraints tight, as well as consistent. Our starting point is increasing a_t at rate 1. We would like to determine the increase/decrease rate of all dual variables in the tree with respect to a_t . Let us use the following notation. For $1 \leq j \leq \ell$, let $\Delta b_j \triangleq -\frac{db_{1_j,t+1}}{da_t}$ be the rate in which $b_{1_j,t+1}$ is **decreasing** with respect to a_t . For $1 \leq j \leq \ell$, let $\Delta b'_j \triangleq \frac{db_{w,t+1}}{da_t}$ be the rate in which the siblings of 1_j are **increasing** with respect to a_t .

We first take care of the siblings of 1_1 . Since a_t is growing at rate 1, we need to compensate by increasing the dual variables on the path from the root to these leaves. By Corollary 3.4 we can use the following rate for $\Delta b'_1$:

$$\delta(1) \cdot \Delta b'_1 = 1$$

This takes care of the dual constraints for all these leaves. However, doing so violates consistency in the root since we increase the mass on these leaves. To keep consistency by lemma A.3 we must set Δb_1 such that:

$$\Delta b'_1 = (\Delta b_1 + \Delta b'_1) \cdot \frac{\left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n}\right)}{(1 + \gamma)}$$

We next take care of the siblings of node 1_2 . Their dual constraint grows at rate $1 + \delta(1)\Delta b_1$. To compensate for this we use corollary 3.4 again and get that we can set $\Delta b'_2$ to be such that:

$$\delta(2) \cdot \Delta b'_2 = 1 + \delta(1)\Delta b_1.$$

We remark that in the final ratio in which $b_{w,t+1}$ grows for the siblings of 1_2 (i.e., $\frac{db_{w,t+1}}{da_t}$) is going to be a sum of the increases caused in this level and the decrease due to the the change in Δb_1 in the previous level (that caused $\Delta b'_2$ to change as well). Increasing $\Delta b'_2$ violates consistency in 1_1 , so we need to fix this using Δb_2 . We keep using this process to update all dual variables until we get to leaf 1. Although, the process is described here recursively the real algorithm simply solves the following set of linear equations to get the correct rate in which the dual variables should grow in order to keep Invariant (1) and Invariant (3). It then uses these rates to increase the variables until Invariant (2) is satisfied.

Applying this recursive view we get the following set of constraints.

Constraints for keeping the dual constraints tight:

$$\begin{aligned} \delta(1) \cdot \Delta b'_1 &= 1 \\ \delta(2) \cdot \Delta b'_2 &= 1 + \delta(1)\Delta b_1 \\ &\dots \\ \delta(\ell) \cdot \Delta b'_\ell &= 1 + \sum_{j=1}^{\ell-1} \delta(j)\Delta b_j. \end{aligned}$$

Constraints for maintaining consistency:

$$\begin{aligned} \Delta b'_1 &= (\Delta b_1 + \Delta b'_1) \cdot \frac{\left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n}\right)}{(1 + \gamma)} \\ \Delta b'_2 &= (\Delta b_2 + \Delta b'_2) \cdot \frac{\left(y_{1_2,t} + \frac{\gamma \cdot |C(2)|}{n}\right)}{\left(1 + \frac{\gamma \cdot |C(1)|}{n}\right)} \\ &\dots \\ \Delta b'_\ell &= (\Delta b_\ell + \Delta b'_\ell) \cdot \frac{\left(y_{w_{1_\ell},t} + \frac{\gamma \cdot |C(\ell)|}{n}\right)}{\left(y_{w_{1_{\ell-1}},t} + \frac{\gamma \cdot |C(\ell-1)|}{n}\right)}. \end{aligned}$$

Due to lack of space the full analysis appears in Appendix B. We just state here the main steps. First, solving the set of equations we get that:

$$\delta(j)\Delta b'_j = \frac{(1 + \gamma)}{\left(y_{w_{1,j-1}} + \frac{\gamma \cdot |C(j-1)|}{n}\right)}.$$

Using this solution we prove our main theorem:

Theorem 3.6. *For any $\gamma < 1$, the algorithm has the following properties:*

- *Its movement cost is bounded by $8\ell \ln(1 + n/\gamma)$ times the optimal cost.*
- *Its service cost is bounded by $(1 + \gamma)$ times the optimal cost.*

The full analysis appears in Appendix B. Bounding the service cost is done by applying Invariant (2) and integrating over the service cost. The movement cost in each layer of the HST is bounded by $O(\log n/\gamma)$ which implies our results.

3.3 Transforming the fractional solution into a randomized algorithm

There is a standard reduction from fractional MTS to a randomized algorithm for MTS. In fact, we do not incur any loss either in the hit-cost or in the move-cost. See [2] for more details.

4 Applications of the MTS algorithm

In this section we will describe applications of our method to several problems.

4.1 The Allocation Problem

The allocation problem on d points and k servers can be viewed as an MTS problem on $O((k + 1)^d)$ states. Here there is a state for each possible way of distributing up to k servers among the d points. There is natural metric on these states (equal to half of the hamming distance between vectors corresponding to two states). Note that the diameter (the ratio of the maximum to the minimum distance between any two distinct points) of this space is k .

One issue in the allocation problem is that the number of available servers $k(t)$ can change with time. This can be handled as follows: Suppose we have Allocation problem on d points. Now imagine that there are $d + 1$ points and the number of servers is fixed at k . The number of servers at point $d + 1$ is supposed to be at least $k - k(t)$. We can enforce this via MTS, by giving infinite cost in the cost vector for states that have less than $k - k(t)$ servers at $d + 1$. Thus the number of states in MTS is at most $O((k + 1)^{d+1})$.

We embed the metric space of the MTS into a probability distribution over dominating HSTs. Note that the embedding only affects the move costs. Moreover, the distortion is at most $O(\log \Delta) = O(\log k)$, where Δ is the diameter of the space. The depth of the HST is $\ell = O(\log k)$. Our result in section 3 implies an $(1 + \epsilon, O(\ell \log(n/\epsilon)))$ competitive algorithm for MTS on an HST with depth ℓ . By the application the Allocation problem on d points in which $n = O(k^d)$ this implies an $O(1 + \epsilon, O(\ell \log \Delta \log(n/\epsilon))) = O(1 + \epsilon, O(d \log^2 k \log(k/\epsilon)))$ competitive algorithm for the Allocation Problem on d points.

Theorem 4.1. *There exists an $O(1 + \epsilon, O(d \log^2 k \log(k/\epsilon)))$ -competitive algorithm for the allocation problem of degree d .*

4.2 Application to k -Server on d -ary HSTs

We use the following connection between k -server and the Allocation Problem. This result is implicit in [7], the quantitative version below can be found in [1].

Theorem 4.2 ([7, 1]). *Suppose there is a $(1 + \epsilon, \beta(\epsilon))$ -competitive algorithm for the allocation problem on a uniform metric on d points. Let H be a d -ary HST with depth ℓ and parameter α . Then, there for any $\epsilon \leq 1$, there is a*

$$\beta(\epsilon)\gamma^{\ell+1}/(\gamma - 1)$$

competitive algorithm for k -server on H , where

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

By Theorem 4.1 we have $\beta(\epsilon) = O(d \log^2 k \log(k/\epsilon))$. Thus we obtain that,

Theorem 4.3. *Let T be a d -ary HST with depth ℓ and parameter α , then for any $\epsilon \leq 1$, there exists a competitive algorithm for the HST with competitive factor:*

$$O\left(\min(\alpha, \beta(\epsilon)/\epsilon) \cdot \left(1 + \epsilon + O\left(\frac{d \log^2 k \log(k/\epsilon)}{\alpha}\right)\right)^{\ell+1}\right)$$

Choosing $\epsilon = \frac{1}{\ell}$ and $\alpha = \Omega(\ell d \log^2 k \log(k\ell))$ the algorithm is $O(\ell d \log^2 k \log(k\ell))$ -competitive.

4.3 The k -server Problem on Equally Spaced Points on the Line

A well known (folklore) result for embedding a line into an HST is the following.

Lemma 4.4. *For any $\alpha \geq 2$ the line metric may be embedded into an $(\ell = \log \Delta / \log \alpha, d = O(\alpha), \alpha)$ -HST, with distortion of $\alpha \log n$.*

(For example this can be done by recursively splitting a line of diameter Δ randomly into d pieces of expected diameter $O(\Delta/d)$.)

For a line of length Δ and in particular for equally spaced line ($\Delta = n$). Using standard techniques and plugging corollary 1.3 we get the following:

Theorem 4.5. *There exist an algorithm for the k -server on the line with competitive ratio:*

$$O\left(\alpha^2 \log n \cdot \left(1 + \epsilon + O(\log^2 k \log(k/\epsilon))\right)^{\frac{\log n}{\log \alpha} + 1}\right).$$

In particular, choosing $\epsilon = 1$ and $\alpha = \exp(O(\sqrt{\log \log k \log \Delta}))$ we get an algorithm for the line with competitive ratio:

$$O\left(\log n \cdot \exp\left(\sqrt{O(\log \log k \log \Delta)}\right)\right).$$

References

- [1] Nikhil Bansal, Niv Buchbinder, and Joseph (Seffi) Naor. Towards the randomized k -server conjecture: A primal-dual approach. In *Manuscript*.
- [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] Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proceedings of the 29th Annual ACM Symposium on Theory of computing*, pages 711–719, 1997.
- [4] Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.
- [5] N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *13th Annual European Symposium on Algorithms*, 2005.
- [6] M. Chrobak and L. Larmore. An optimal on-line algorithm for k -servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.
- [7] A. Coté, A. Meyerson, and L. Poplawski. Randomized k -server on hierarchical binary trees. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 227–234, 2008.
- [8] 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.
- [9] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k -server algorithms. *Journal of Computer and System Sciences*, 48(3):410–428, 1994.
- [10] Amos Fiat, Richard M. Karp, Michael Luby, Lyle A. McGeoch, Daniel Dominic Sleator, and Neal E. Young. Competitive paging algorithms. *J. Algorithms*, 12(4):685–699, 1991.
- [11] Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM J. Comput.*, 32(6):1403–1422, 2003.
- [12] Elias Koutsoupias and Christos H. Papadimitriou. On the k -server conjecture. *J. ACM*, 42(5):971–983, 1995.
- [13] M. Manasse, L.A. McGeoch, and D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230, 1990.
- [14] 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.
- [15] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

A Proofs of the properties of the function f

We restate here the Lemmas of Section 3.1 that examine additional properties of the function f along with their proofs. The first property is *consistency*. Since, at all times, $y_{v,t} = \sum_{w \in C(v)} y_{w,t}$, it means that when increasing or decreasing $y_{v,t}$, it is also necessary to increase and decrease the children of v at a certain rate. The following lemma states the connection between these rates.

Lemma A.1 (Consistency). *Suppose we increase variable $b_{v,t+1}$ for node v at rate r . Then:*

$$\frac{1}{\alpha} \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right) \cdot \frac{db_{v,t+1}}{dr} = \sum_{w \in C(v)} \frac{db_{w,t+1}}{dr} \cdot \left(y_{w,t} + \frac{\gamma \cdot |C(w)|}{n} \right)$$

Proof. We need to maintain $y_{v,t} = \sum_{w \in C(v)} y_{w,t}$. Thus, we take the derivative of both sides and get:

$$\frac{dy_{v,t}}{db_{v,t+1}} \cdot \frac{db_{v,t+1}}{dr} = \sum_{w \in C(v)} \frac{dy_{w,t}}{db_{w,t+1}} \cdot \frac{db_{w,t+1}}{dr}$$

Plugging in Equation (7), we get:

$$\frac{\ln(1 + n/\gamma)}{\alpha^{\ell - \ell(v)}} \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right) = \sum_{w \in C(v)} \frac{db_{w,t+1}}{dr} \cdot \frac{\ln(1 + n/\gamma)}{\alpha^{\ell - \ell(w)}} \left(y_{w,t} + \frac{\gamma \cdot |C(w)|}{n} \right)$$

Simplifying, we get the desired bound. □

We need the following two special cases of Lemma A.1. The first case happens when $b_{v,t+1}$ is increased at rate r , and we need to increase all the children of v at the same rate. From Lemma A.1, we get the following lemma.

Lemma A.2 (Consistency: Equal Rates). *Let v be a node. Suppose we increase (or decrease) the probability mass in subtree T_v by increasing (decreasing) $b_{v,t+1}$ at rate r . Then, consistency is maintained for each child $w \in C(v)$ by setting for each child w :*

$$\frac{db_{w,t+1}}{dr} = \frac{1}{\alpha} \cdot \frac{db_{v,t+1}}{dr}.$$

Proof. By Lemma A.1, if we increase all $w \in C(v)$ at the same rate, we get that:

$$\begin{aligned} \frac{1}{\alpha} \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right) \cdot \frac{db_{v,t+1}}{dr} &= \frac{db_{w,t+1}}{dr} \cdot \sum_{w \in C(v)} \left(y_{w,t} + \frac{\gamma \cdot |C(w)|}{n} \right) \\ &= \frac{db_{w,t+1}}{dr} \cdot \left(y_{v,t} + \frac{\gamma \cdot |C(v)|}{n} \right). \end{aligned}$$

Simplifying, we get the desired bound. □

The second special case of Lemma A.1 that we need is the following. Let v be a node and let w_1 be the first child of v . Suppose we increase (decrease) $b_{w_1,t+1}$ at some rate and then use the same rate for increasing (decreasing) $b_{w',t+1}$, for each child $w' \in C(v)$, $w' \neq w_1$. Also, suppose that $b_{v,t+1}$ is not changed. Then the following should hold to maintain consistency.

Lemma A.3 (Consistency Case 2). *Let v be a node, and let w_1, \dots, w_m be its children. Suppose we increase (decrease) w_1 at rate r and also increase w_2 to w_m at the same rate r . Then, for $i \geq 2$,*

$$\frac{db_{w_i, t+1}}{dr} = \frac{db_{w_{i+1}, t+1}}{dr} \triangleq \frac{db_{w', t+1}}{dr}.$$

If we want to keep the amount y_v unchanged then:

$$\frac{db_{w', t+1}}{dr} = \frac{\left(y_{w_1, t} + \frac{\gamma \cdot |C(w_1)|}{n}\right)}{\left(y_{v, t} + \frac{\gamma \cdot |C(v)|}{n}\right)} \cdot \left(-\frac{db_{w_1, t+1}}{dr} + \frac{db_{w', t+1}}{dr}\right).$$

Proof. In this case we get from Lemma A.1 that:

$$\begin{aligned} 0 &= \frac{db_{w_1, t+1}}{dr} \cdot \left(y_{w_1, t} + \frac{\gamma \cdot |C(w_1)|}{n}\right) + \sum_{w \in C(v) \setminus \{w_1\}} \frac{db_{w, t+1}}{dr} \cdot \left(y_{w, t} + \frac{\gamma \cdot |C(w)|}{n}\right) \\ &= \left(\frac{db_{w_1, t+1}}{dr} - \frac{db_{w', t+1}}{dr}\right) \cdot \left(y_{w_1, t} + \frac{\gamma \cdot |C(w_1)|}{n}\right) + \frac{db_{w', t+1}}{dr} \cdot \sum_{w \in C(v)} \left(y_{w, t} + \frac{\gamma \cdot |C(w)|}{n}\right) \\ &= \left(\frac{db_{w_1, t+1}}{dr} - \frac{db_{w', t+1}}{dr}\right) \cdot \left(y_{w_1, t} + \frac{\gamma \cdot |C(w_1)|}{n}\right) + \frac{db_{w', t+1}}{dr} \cdot \left(y_{v, t} + \frac{\gamma \cdot |C(v)|}{n}\right) \end{aligned}$$

□

B Analysis of the algorithm

We state here the full analysis of our algorithm proving Theorem 3.6.

Theorem B.1. *For any $\gamma < 1$, the algorithm has the following properties:*

- *Its movement cost is bounded by $8\ell \ln(1 + n/\gamma)$ the optimal cost.*
- *Its service cost is bounded by $1 + \gamma$ the optimal cost.*

For completeness we state here again the constraints of the algorithm.

Constraints for keeping the dual constraints tight:

$$\begin{aligned} \delta(1) \cdot \Delta b'_1 &= 1 \\ \delta(2) \cdot \Delta b'_2 &= 1 + \delta(1)\Delta b_1 \\ &\dots \\ \delta(\ell) \cdot \Delta b'_\ell &= 1 + \sum_{j=1}^{\ell-1} \delta(j)\Delta b_j \end{aligned}$$

Constraints for keeping consistency:

$$\begin{aligned}\Delta b'_1 &= (\Delta b_1 + \Delta b'_1) \cdot \frac{\left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n}\right)}{(1 + \gamma)} \\ \Delta b'_2 &= (\Delta b_2 + \Delta b'_2) \cdot \frac{\left(y_{1_2,t} + \frac{\gamma \cdot |C(2)|}{n}\right)}{\left(1 + \frac{\gamma \cdot |C(1)|}{n}\right)} \\ \dots & \\ \Delta b'_\ell &= (\Delta b_\ell + \Delta b'_\ell) \cdot \frac{\left(y_{w_{1_\ell},t} + \frac{\gamma \cdot |C(\ell)|}{n}\right)}{\left(y_{w_{1_{\ell-1}},t} + \frac{\gamma \cdot |C(\ell-1)|}{n}\right)}\end{aligned}$$

Solving the Set of Equations: First, by subtracting the i th constraint that keeps the dual tight from the $i + 1$ constraint we get.

$$\begin{aligned}\delta(1) \cdot \Delta b'_1 &= 1 \\ \delta(2) \cdot \Delta b'_2 &= \delta(1)(\Delta b_1 + \Delta b'_1) \\ \dots & \\ \delta(\ell) \cdot \Delta b'_\ell &= \delta(\ell - 1)(\Delta b_{\ell-1} + \Delta b'_{\ell-1}).\end{aligned}$$

Next, by plugging the consistency constraints in this set of equations, we get a recursive definition of b'_j .

$$\delta(j)\Delta b'_j = \delta(1)\Delta b'_{j-1} \frac{\left(y_{w_{1_{j-2}},t} + \frac{\gamma \cdot |C(j-2)|}{n}\right)}{\left(y_{w_{1_{j-1}},t} + \frac{\gamma \cdot |C(j-1)|}{n}\right)}.$$

Solving the recursion we get.

$$\delta(j)\Delta b'_j = \frac{(1 + \gamma)}{\left(y_{w_{1_{j-1}},t} + \frac{\gamma \cdot |C(j-1)|}{n}\right)}.$$

B.1 Finalizing the analysis

By the construction of the algorithm we maintain a feasible dual solution at all times. When we increase a_t the derivative of the dual profit is 1. We would like to bound the derivative in the primal cost.

Bounding the service cost: The service cost is simply cy_1 , where y_1 is the amount of mass we have in leaf 1 after satisfying invariant (2). If $y_1 = 0$ then we have no service cost. Otherwise, we know that the constraint for leaf 1 is tight. Let a_f be the final value of a_t . Since the dual constraint is tight we get that:

$$c = \int_{a_t=0}^{a_t=a_f} \left(1 + \sum_{j=1}^{\ell} \delta(j)\Delta b_j\right) da_t$$

So the hitting cost is simply,

$$cy_1 = y_1 \cdot \int_{a_t=0}^{a_t=a_f} \left(1 + \sum_{j=1}^{\ell} \delta(j) \Delta b_j \right) da_t$$

Taking the derivative of both sides, we get that the derivative of the service cost is $y_1 \cdot \left(1 + \sum_{j=1}^{\ell} \delta(j) \Delta b_j \right)$. Using the last dual constraint and the last consistency constraint we get that:

$$\begin{aligned} y_1 \cdot \left(1 + \sum_{j=1}^{\ell} \delta(j) \Delta b_j \right) &= y_1 \cdot \delta(\ell) \cdot (\Delta b'_\ell + \Delta b_\ell) = y_1 \cdot \delta(\ell) \Delta b'_\ell \frac{\left(y_{w_{1_{\ell-1},t}} + \frac{\gamma \cdot |C(\ell-1)|}{n} \right)}{\left(y_{w_{1_\ell,t}} + \frac{\gamma \cdot |C(\ell)|}{n} \right)} \\ &= y_1 \frac{(1 + \gamma)}{\left(y_1 + \frac{\gamma \cdot |C(\ell)|}{n} \right)} \leq 1 + \gamma \end{aligned}$$

Note that we assume that y_1 is not changing during the algorithm, but even if we assume it does change then we should put y'_1 in the denominator. Since $y'_1 > y_1$ we get the desired result.

Bounding the movement cost: Let $C_j = 2(1 + \alpha + \dots + \alpha^{\ell-j})$. We bound the amount of mass that we transfer due to the change in $\Delta b'_i$ in each level j ($1 \leq j \leq \ell$). In level j the algorithm incurs the following movement cost:

$$\begin{aligned} C_j \sum_{w \in C(1_{j-1}) \setminus \{1_j\}} \frac{dy_w}{db'_j} \cdot \frac{db'_j}{da_t} &= C_j \Delta b'_j \cdot \sum_{w \in C(1_{j-1}) \setminus \{1_j\}} \frac{\ln(1 + n/\gamma)}{\alpha^{\ell-j}} \left(y_{w,t} + \frac{\gamma \cdot |C(j)|}{n} \right) \\ &= 4 \ln(1 + n/\gamma) \cdot \Delta b'_j \cdot \sum_{w \in C(1_{j-1}) \setminus \{1_j\}} \left(y_{w,t} + \frac{\gamma \cdot |C(j)|}{n} \right) \\ &\leq 4 \ln(1 + n/\gamma) \cdot \Delta b'_j \cdot \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right) \\ &= 4 \ln(1 + n/\gamma) \cdot \frac{1 + \gamma}{\delta(j)} \leq 8 \ln(1 + n/\gamma) \end{aligned}$$