Metrical Task Systems and the k-Server Problem on HSTs

Nikhil Bansal[1], Niv Buchbinder[2], and Joseph (Seffi) Naor[3,⋆]

[1] IBM T. J. Watson Research Center, Yorktown Heights, NY 10598.
[2] Microsoft Research, Cambridge, MA.
[3] Computer Science Dept., Technion, Haifa, Israel.

Abstract. We consider the randomized k-server problem, and give improved results for various metric spaces. In particular, we extend a recent result of Coté et al [15] for well-separated *binary* Hierarchically Separated Trees (HSTs) to well-separated d-ary HSTs for poly-logarithmic values of d. One application of this result is an $\exp(O(\sqrt{\log\log k \log n}))$-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 [16].
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 a uniform metric [5, 7, 18]. Our results are based on the primal-dual approach for online algorithms. Previous primal-dual approaches in the context of k-server and MTS [2, 4, 3] worked only for uniform or weighted star metrics, and the main technical contribution here is to extend many of these techniques to work *directly* on HSTs.

1 Introduction

The k-server problem is a central and well studied problem in competitive analysis of online problems and is considered by many to be the "holy grail" problem in the field. The k-server problem is defined as follows. 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 the most widely studied of these is *paging* with all its variants [27]. Paging is the special case of k-server on a uniform metric. In their seminal paper on competitive analysis, Sleator and Tarjan [27] 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. [24], who conjectured that a k-competitive online algorithm exists in any metric space and for

⋆ This work was partly supported by ISF grant 1366/07. Part of this work was done while visiting Microsoft Research, Redmond, WA.

any value of k. This is called the *k-server conjecture*. After a long line of work, a major breakthrough was achieved by Koutsoupias and Papadimitriou [23] who gave a $2k-1$ competitive algorithm. We note that for special metrics such as the uniform metric, line metric [13], and more generally trees [14], a competitive factor of k is known for the k-server problem. The reader is referred to [9] for an excellent survey of various results and history of the problem.

However, despite much interest, randomized algorithms for k-server remain poorly understood. No lower bound better than $\Omega(\log k)$ is known for any metric space, and it is widely believed that an $O(\log k)$-competitive randomized algorithm exists for every metric space against an *oblivious* adversary. This is called the *randomized k-server conjecture*. Yet, better than k-competitive algorithms are known only for few special cases [17, 5, 2, 26, 16, 15]. The most well-studied of these cases is paging and its variants, for which several $\Theta(\log k)$-competitive algorithms are known [17, 1, 2, 21, 20, 8][†]. However, not much is essentially known beyond the uniform metric. Even for the seemingly simple case of n uniformly spaced points on a line, which is perhaps the simplest "non-uniform like" metric, only an $O(n^{2/3})$-competitive algorithm [16] is known, which is $o(k)$-competitive only for $n = o(k^{3/2})$.

Given our current level of understanding, resolving a weaker variant of the randomized k-server conjecture – obtaining a polylog(k, n, Δ)-competitive algorithm for general metrics, where Δ is the diameter of the metric, would be a major breakthrough.[‡] Since a general metric space can be embedded into a probability distribution over hierarchically well-separated trees (HSTs) with logarithmic distortion of the distances, it suffices to consider the k-server problem on HSTs to obtain the latter bound. This approach seems particularly promising, as randomized k-server is relatively well understood for uniform metrics, and algorithms on an HST can often be obtained by recursively applying a suitable algorithm on uniform metrics. This has previously been done in many other settings, e.g., metrical task systems and metric labeling [5, 18, 22].

Along these lines, Coté et al. [15] recently gave the first[§] completely formal framework to solving the k-server problem on HSTs by defining a related problem, called the *allocation problem* by [3], on uniform metrics. In the allocation problem (defined formally in Section 2), upon each request, two types of cost are incurred: the *hit-cost* and the *move-cost*. Coté et al. [15] showed that designing an online algorithm that, given any $\epsilon > 0$, has a hit-cost of at most $(1 + \epsilon)$ times the total optimum cost and a move-cost of at most poly-logarithmic times the total optimum cost, would imply a poly-logarithmic competitive algorithm for the k-server problem (see Theorem 5 below for a formal statement). Note the asymmetry between the guarantees required for the hit-cost and move-cost. It is crucial that the hit-cost factor is $(1 + \epsilon)$, as the hit-cost factor multiplies across the levels of the HST, while the move-cost factor only adds up. In a very interesting result, Coté et al. [15] were able to obtain such a guarantee for the allocation problem on a metric space with *two* points. Using their framework, this guarantee for the allocation problem implies an $O(\log \Delta)$-competitive algorithm for *binary*

[†] Due lack to space here, we refer the reader to [9] for an excellent survey of paging and other results for randomized k-server.

[‡] In fact, even an $o(k)$-competitive algorithm would be extremely interesting.

[§] Previously, related (but simpler) problems such as finely competitive paging were studied [8].

HSTs. However, being limited to two points, this result is not general enough to imply any non-trivial guarantees for more general metrics.

Thus, the big outstanding question is to extend the result of Coté et al. [15] to a uniform metric on an arbitrary number of points. However, as already pointed out by [15], it is not even clear how to obtain such a guarantee for a space with three points. In particular, their algorithm crucially uses the fact that the allocation problem on two points looks like a metrical task system problem (defined later) on a line with a special cost structure.

1.1 Our Results

In this work we extend the result of Coté et al. [15] for the allocation problem on any uniform metric on d points. Specifically, we show the following result:

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

Exploring the meaning of such a result in the k-server context, let us define an (ℓ, p, α)-HST as an HST with height ℓ, maximum degree p, and stretch factor α. Given such an HST and combining the above result with Theorem 5 we get the following.

Theorem 2. *Given an* (ℓ, p, α)*-HST metric, there exists an online algorithm for the* k*-server problem with competitive factor:*

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

In particular, if $\alpha = \Omega\left(\ell p\log^2 k\log(k\ell)\right)$ *(i.e. the HST is well-separated), then the algorithm has a competitive ratio of* $O\left(\ell p\log^2 k\log(k\ell)\right)$.

While our result applies to any number of points d, our competitive ratio with respect to the move-cost linearly varies with d, and hence our result is not strong enough to obtain poly-logarithmic guarantees for general metrics. (For this, one needs a guarantee which is poly-logarithmic in d.) Still, our result has useful consequences. Applying Theorem 2, along with standard embedding results for equally spaced points on the line, we get the following Theorem.

Theorem 3. *There is a* $\exp\left(O(\sqrt{\log\log k\log n})\right)$*-competitive algorithm for* k*-server on* n *equally spaced points on a line. In general, for an arbitrary* n*-point line with diameter* Δ*, our algorithm is* $\exp\left(O(\sqrt{\log\log k\log\Delta})\right)$*-competitive.*

This is the first online algorithm with a sublinear competitive ratio for the line, providing strong evidence that randomization does help for this kind of metric. The line metric is particularly interesting since it is one of the simplest metrics that is essentially different from the uniform metric. Historically, new insights gained from the line case have often been useful for more general metrics. For example, the double coverage algorithm was first discovered for a line [13], and then extended to tree metrics [14]. Obtaining a better online algorithm, even for a line or a circle, is already stated as a major open problem in [9, Problem 11.1] .

Techniques and Comparison with Previous Approaches

Refined Guarantee for Metrical Task Systems: In order to prove Theorem 1, we design here a new algorithm for the *metrical task system* problem (MTS) with refined guarantees, which are interesting on their own. MTS was introduced by Borodin et al. [10] 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 $1, 2, \ldots, n$, and a metric d defining 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 $c_t = (c_t(1), \ldots, c_t(n))$ 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, the cost of serving the task is its processing cost $c_t(j)$ plus the movement cost d_{ij}. The goal is to minimize the total cost.

We obtain the following refined guarantee for MTS on an HST.

Theorem 4. *Consider the MTS problem on an HST of height ℓ and n points. For any $0 < \gamma < 1$, there is an online algorithm achieving the following bounds:*

- *Its service cost is bounded by $(1 + \gamma)$ times the optimal cost.*
- *Its movement cost is bounded by $O(\log(\frac{n}{\gamma}) \cdot \min\{\ell, \log(\frac{n}{\gamma})\})$ times the optimal cost.*

While poly-logarithmic guarantees for randomized MTS on HSTs (and hence general metrics) are already known [5, 7, 18], the key difference here is that the competitive ratio with respect to the service cost is only $(1 + \gamma)$. Previously, such a result was only known for uniform metrics [5, 7, 19], where this guarantee was referred to as the *unfair* MTS guarantee. Thus, Theorem 4 can be viewed as an extension of the unfair MTS result of [5, 7, 19] to HSTs. Such an extension is crucial to obtain meaningful results for the allocation problem (see Section 4.1 for more details about this reduction). In fact, obtaining such a result was already proposed as an approach to attacking the allocation problem [25] (see e.g. slide 45).

As we explain below, it is quite unclear whether Theorem 4 can be obtained using previous techniques. Next, we explain the techniques we use here.

Previous Results on Randomized MTS: In a breakthrough paper, Bartal et al. [5] obtained the first polylogarithmic competitive randomized algorithm for HSTs (and hence general metrics) by recursively solving a stronger variant of the MTS problem on uniform metrics. This stronger variant is the unfair MTS problem that, given any $\gamma > 0$, pays a movement cost of $O(\log(\frac{n}{\gamma}))$ times the optimum, but has service cost only $(1+\gamma)$ times the optimum. In particular, previous algorithms obtaining a polylogarithmic competitive ratio on HSTs, achieved it by recursively "combining unfair MTS algorithms" on uniform metrics.

A natural question is whether the approach of recursively combining an unfair MTS algorithm for uniform metrics can be used to obtain an unfair MTS algorithm for the entire HST. However, it is not clear how to use previous techniques toward this task [6]. In particular, any approach based on "combining unfair MTS" loses right away at the second level of the HST the distinction between the service costs and movements costs. This happens since the cost vector for the unfair MTS algorithm running at a node at a higher level must necessarily use the total cost incurred at the lower level (i.e. the sum of service costs plus movement costs incurred at the lower level). Thus, it is not clear how to even track the service costs and movement costs for HSTs with 2 or more levels.

Our Techniques: We completely move away from recursively combining unfair MTS algorithms on uniform metrics. In particular, our algorithm is a "one shot algorithm" that works directly on the whole metric, and is based on linear programming techniques. Specifically, we cast the (offline) MTS problem on an HST as a minimization linear program. We then obtain the dual maximization problem which is used to bound the cost of our online solution. When a new cost vector arrives, a set of equations is solved to compute the amount of "flow" it should move from each leaf in the HST to other leaves. This flow solution is dictated by a single decision implied by a "connecting function" between primal and dual variables.

At a high level, our approach follows the general primal-dual framework for designing online algorithms developed by Buchbinder and Naor [11, 12, 2, 4, 3]. However, the main contribution here is that we extend many of the previous techniques to work for HSTs. In particular, all previous results and techniques [2, 4, 3] hold only for uniform metrics or weighted star metrics[¶]. These metrics seem substantially simpler than HSTs, and in general it is not at all clear how to extend these results from uniform metrics to HSTs. For example, several ways are known to obtain poly-logarithmic guarantees for k-server on the uniform metric (paging), but extending these results to HSTs would prove the weaker randomized k-server conjecture!

2 Preliminaries

Allocation 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 $\boldsymbol{h}^t = (h^t(0), h^t(1), \ldots, h^t(k))$, 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 $\mathrm{Optcost}$ the optimal cost of an instance of the allocation problem. Coté et al. [15] showed that if there is an online algorithm that incurs a hit cost of $(1{+}\varepsilon)\mathrm{Optcost}$ and a move cost of $\beta(\epsilon)\mathrm{Optcost}$, where $\beta(\cdot)$ is a polylogarithmic function, 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.

[¶] In related work, [3] shows how an unfair MTS algorithm on uniform metrics (i.e. the result of [5]) can be obtained using the primal-dual framework. It also solves the finely competitive paging problem [8], and the allocation problem assuming the costs satisfy a certain convexity assumption.

Theorem 5 ([15, 3]). *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, for any $\epsilon \leq 1$, there is a $\beta(\epsilon)\gamma^{\ell+1}/(\gamma-1)$-competitive algorithm for the k-server problem on H, where*

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

Hierarchically well-separated trees: We formally define a hierarchically well-separated tree (HST) with stretch factor α and fix some notation. We denote the leaves of an HST by $1, \ldots, n$ 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 r 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 $|C(v)| = 1$.

LP Formulation for MTS on an HST: We study the 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 $e_i = (0, 0, \ldots, 1, 0, \ldots)$ has 1 in the i-th position. Hence, we use i_t to denote the location with non-zero cost at time t.

We now present our linear programming formulation of the MTS problem on an HST. 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. The variables will be $y_{i,t}$ for each leaf i and time t. Let $z_{v,t}$ denote the decrease in the total probability mass in leaves of T_v 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.

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

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

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

Clearly, the formulation is valid. The first constraint says that the total probability mass on the leaves is exactly 1. The second set of constraints captures the movement cost (i.e. the cost of decreasing the total probability mass in a subtree) at each internal node.

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).

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

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

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

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

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

We view the dual as follows. With each node v in the tree there is a variable b_v that varies as a function of time, but is bounded by a fixed constant depending on the level of v (constraint (5)). The dual profit can only be obtained by increasing a_t, and hence we try to increase these variables as much as possible over time. At time t, when a request arrives at leaf i_t, constraints (3) require that for every leaf $i \neq i_t$, the sum of b_v values along the path from i to the root must increase so as to offset a_t. Thus, they force the b_v variables to increase as a_t is raised. However, as the b_v's are bounded from above, this process cannot go on for too long, thus preventing the growth of the dual profit. The rescue comes from the slack in the corresponding constraint for leaf i_t (constraint (4)), allowing us to decrease the appropriate b_v variables as requests arrive over time.

3 The Metrical Task System 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 show how to transform the fractional online algorithm into a randomized algorithm (the second step is actually trivial). The randomized algorithm is going to maintain a primal solution (a probability distribution on states) and a corresponding dual solution. The high level idea of the algorithm is very simple. It maintains a (carefully chosen) relation between the primal and dual variables. When a cost arrives at leaf i_t at time t, we update the dual variables subject to the dual constraints ((3)-(5)). The relation between the primal and the dual variables determines how much probability mass should be moved out of leaf i_t and how it should be distributed among other leaves.

The relation between the dual and primal variables is that the value of variable $y_{v,t}$ (amount of mass in subtree T_v) is a function of the dual variable $b_{v,t+1}$:

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

This relation is maintained throughout the execution of the algorithm. Let $0 < \gamma < 1$ be an arbitrary constant. Recall that if v is a leaf then $|C(v)| = 1$. Before continuing with the description of the algorithm, we should derive several properties from the invariant. These properties are somewhat technical and we omit them due to space limitations.

3.1 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 $(c, 0, 0, ..., 0)$. The high level idea of the algorithm is very simple. Since the cost is non-zero only at leaf 1, we should move some probability mass out of leaf 1, and distribute it to various other leaves. To determine the exact amount at each leaf, the algorithm simply should keep the following invariants:

MTS Conceptual Algorithm: When a request arrives at time t at leaf 1 keep the following invariants:

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

It turns out that the conceptual algorithm above completely determines how much mass we should move from leaf 1 to any other leaf in the sub-tree. So, essentially, the restriction on keeping the dual constraints tight and keeping consistency (i.e., mass in each sub-tree is equal to the mass in its children) completely determines the algorithm!

For the rest of this section we perform some calculations to give an explicit description of the algorithm, specifying the exact rate at which we should move mass in the tree. Having obtained these rates, we will bound the total movement cost and the total service cost by relating it to the growth of the dual profit.

We first give an intuitive description of the process. When the cost arrives at leaf 1 at time t, if $y_{1,t} = 0$ then the primal cost is zero and the dual profit is also zero. Moreover, nothing changes and all the invariants continue to hold. Thus, we consider the case that $y_{1,t}$. We start increasing a_t at rate 1. At each step we would like to keep all dual constraints tight, as well as consistent. However, raising a_t violates the dual constraints on leaves $2, 3, \ldots, n$, forcing us to raise other dual variables to keep these constraints tight. Raising these variables may also violate consistency (Invariant (3)), and thus lead to the update of other dual variables. This process results in the transfer of mass from leaf 1 to leaves $2, 3, \ldots, n$, and also the constraint on leaf 1 becomes tighter. We stop the updating process when either the constraint on leaf 1 is tight, or when $y_{1,t} = 0$. We next consider this process more formally.

Consider the root, and let 1_j, $1 \leq j \leq \ell$, denote the node at level j containing leaf 1 (1_ℓ is the leaf itself). Since mass moves out of each subtree containing 1, variable $b_{v,t+1}$ decreases for each $v = 1_j$. For every other node v in the tree not containing 1, the probability mass increases. Any node v not containing 1 must be a sibling of some unique node 1_j. By symmetry, all siblings v of a node 1_j must increase $b_{v,t+1}$ at the same rate (it can be easily checked that this is indeed necessary to keep consistency). We wish 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 at which $b_{1_j,t+1}$ is *decreasing* with respect to a_t. For $1 \le j \le \ell$, let $\Delta b'_j \triangleq \frac{db_{w,t+1}}{da_t}$ be the rate at which the siblings of 1_j are *increasing* with respect to a_t.

We can derive the quantities Δb_j and $\Delta b'_j$ in a top-down fashion as follows (the exact arguments are omitted due to lack of space). We first consider the siblings of 1_1 (i.e. children of root other than 1_1). Let v be one of these siblings. If we raise $b_{v,t+1}$ by $\Delta b'_1$, by the consistency requirement, the sum of $\Delta b'$ on any path from v to a leaf in T_v must be $\delta(1) \cdot \Delta b'_1$. As a_t is growing at rate 1, keeping the dual constraint (3) tight for leaves in T_v requires that

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

This takes care of the dual constraints for these leaves. Now, this increase of mass must come by decreasing the mass in T_{1_1} since the total probability mass is 1. To keep consistency of the root we should set Δb_1 so that:

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

We repeat the argument for siblings of node 1_2. Let v be a sibling. Consider a path from a leaf in T_v to the root. Their dual constraint (3) already grows at rate $1+\delta(1)\Delta b_1$. This must be compensated by increasing $b_{v,t+1}$, and by consistency all the variables $b_{w,t+1}$ for $w \in T_v$. Therefore, $\Delta b'_2$ has to be set so that:

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

Again, this additional increase of mass must come from an additional decreasing mass in T_{1_2}. To keep consistency of 1_1 we must set Δb_2 so that:

$$\Delta b'_2 = (\Delta b_2 + \Delta b'_2) \cdot \left(y_{1_2,t} + \frac{\gamma \cdot |C(2)|}{n} \right) / \left(y_{1_1,t} + \frac{\gamma \cdot |C(1)|}{n} \right).$$

Continuing on, we obtain a system of linear equations. Due to lack of space, the set of constraints fully defining our algorithm is omitted. Solving the equations we get:

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

$$\Delta b_j = \frac{1+\gamma}{\delta(j)} \cdot \left(\left(y_{1_j,t} + \frac{\gamma \cdot |C(j)|}{n} \right)^{-1} - \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1} \right).$$

This fully defines our explicit MTS algorithm in terms of derivatives. To implement it, the algorithm simply does a binary search for the correct value of a_t defining the correct flow.

MTS Explicit Algorithm: When a request arrives at time t at leaf 1 keep the following invariants:

1. While the dual constraint of leaf 1 is not tight and $y_{1,t} > 0$:
2. Increase a_t with rate 1.
3. Decrease each b_{1_j} with rate:

$$\frac{db_{1_j,t+1}}{da_t} = \frac{1+\gamma}{\delta(j)} \cdot \left[\left(y_{1_j,t} + \frac{\gamma \cdot |C(j)|}{n} \right)^{-1} - \left(y_{1_{j-1},t} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1} \right].$$

4. For each sibling w of 1_j increase $b_{w,t+1}$ with rate:

$$\frac{db_{w,t+1}}{da_t} = \frac{(1+\gamma)}{\delta(j)} \left(y_{1_{j-1}} + \frac{\gamma \cdot |C(j-1)|}{n} \right)^{-1}.$$

5. For every node w, recursively, top to bottom, if the variable of the parent of w, $b_{v,t+1}$ is increasing/decreasing with rate c, then decrease/increase $b_{w,t+1}$ with rate c/α.

Based on these rates, we can now calculate the movement cost and the service cost as a function of the dual profit. This allows us to show our main result, Theorem 4. The proof is omitted.

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. There is a state for each possible way of distributing (up to) k servers among the d points. There is a natural metric on these states (equal to half the hamming distance between vectors corresponding to two states). The diameter (ratio of maximum to 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 an instance of the 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 setting an infinite cost to states having 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 problem 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 $\ell = O(\log k)$. Theorem 4 implies an $(1+\epsilon, O(\ell \log(n/\epsilon)))$ competitive algorithm for MTS on an HST with depth ℓ. By the application to an allocation problem on d points, in which $n = O(k^d)$ and adding an addition factor $\log(\Delta) = O(\log k)$ due to distortion in the movement costs, we obtain that

Theorem 6. *There exists an* $(1+\epsilon, O(d\log^2 k \log(k/\epsilon)))$*-competitive algorithm for the allocation problem of degree d.*

4.2 Application to k-Server on d-ary HSTs

We use theorem 5 that relates the k-server problem on HSTs to the allocation problem. By Theorem 6 we have $\beta(\epsilon) = O(d\log^2 k \log(k/\epsilon)))$. Thus we obtain:

Theorem 7. *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:*

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

Choosing $\epsilon = \frac{1}{\ell}$, if $\alpha = \Omega\left(\ell d\log^2 k \log(k\ell)\right)$, the algorithm is $O\left(\ell d \log^2 k \log(k\ell)\right)$-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 1. *For any $\alpha \geq 2$, the line metric can be embedded into an $(\ell = \log \Delta/\log\alpha, d = O(\alpha), \alpha)$-HST, with distortion of $\alpha \log \Delta$.*

Apply Theorem 7 with $\epsilon = 1$ and $\alpha = \exp(O(\sqrt{\log\log k \log \Delta}))$ (which turn out to optimal choices of parameters), yielding:

Theorem 8. *There exists an* $\exp\left(\sqrt{O(\log\log k \log \Delta)}\right)$*-competitive algorithm for the the k-server problem on the line.*

References

1. Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theory 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 *Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 507–517, 2007.
3. Nikhil Bansal, Niv Buchbinder, and Seffi Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *to appear in ACM-SIAM Symposium on Discrete Algorithms (SODA 2010)*, 2010.
4. Nikhil Bansal, Buchbinder Niv, and Naor Joseph (Seffi). Randomized competitive algorithms for generalized caching. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 235–244, 2008.

[‖] For example, this can be done by recursively splitting a line of diameter Δ in random into d pieces of expected diameter $O(\Delta/d)$.

5. 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.
6. Avrim Blum. Personal communication.
7. Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Machine Learning*, 39(1):35–58, 2000.
8. Avrim Blum, Carl Burch, and Adam Kalai. Finely-competitive paging. In *IEEE Symposium on Foundations of Computer Science*, pages 450–458, 1999.
9. Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
10. Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *Journal of the ACM*, 39(4):745–763, 1992.
11. N. Buchbinder and J. Naor. Online primal-dual algorithms for covering and packing problems. In *Proceedings of the 13th Annual European Symposium on Algorithms*, 2005.
12. N. Buchbinder and J. Naor. The design of competitive online algorithms via a primal-dual approach. *Foundations and Trends in Theoretical Computer Science*, 3(2-3):93–263, 2009.
13. M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. *SIAM Journal on Discrete Math*, 4(2):172–181, 1991.
14. M. Chrobak and L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM Journal on Computing*, 20(1):144–148, 1991.
15. A. Coté, A. Meyerson, and L. Poplawski. Randomized k-server on hierarchical binary trees. In *STOC '08: Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 227–234, 2008.
16. 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.
17. 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.
18. Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.
19. Amos Fiat and Manor Mendel. Better algorithms for unfair metrical task systems and applications. *SIAM Journal on Computing*, 32(6):1403–1422, 2003.
20. S. Irani. Randomized weighted caching with two page weights. *Algorithmica*, 32(4):624–640, 2002.
21. Sandy Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 701–710, 1997.
22. Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. *J. ACM*, 49(5):616–639, 2002.
23. Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *Jornal of the ACM*, 42(5):971–983, 1995.
24. Mark S. Manasse, Lyle A. McGeoch, and Daniel D. Sleator. Competitive algorithms for server problems. *Journal of Algorithms*, 11(2):208–230, 1990.
25. Adam Meyerson. http://www.cs.ucla.edu/ awm/talks/kserver.ppt.
26. Steven S. Seiden. A general decomposition theorem for the k-server problem. In *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.