A polylog(n)-competitive algorithm for metrical task systems

Yair Bartal* Avrim Blum† Carl Burch‡ Andrew Tomkins§

October 7, 1997

Abstract

We present a randomized on-line algorithm for the Metrical Task System problem that achieves a competitive ratio of $O(\log^6 n)$ against an oblivious adversary, on any metric space. This is the first algorithm to achieve a sublinear competitive ratio for all metric spaces. Our algorithm uses a recent result of Bartal [Bar96] that an arbitrary metric space can be probabilistically approximated by a set of metric spaces called "k-hierarchical well-separated trees" (k-HST's). Indeed, the main technical result of this paper is an $O(\log^2 n)$-competitive algorithm for $\Omega(\log^2 n)$-HST spaces. This, combined with the result of [Bar96], yields the general bound.

Note that for the k-server problem on metric spaces of $k + c$ points our result implies a competitive ratio of $O(c^6 \log^6 k)$.

1 Introduction

The Metrical Task System (MTS) problem, introduced by Borodin, Linial, and Saks [BLS92], can be stated as follows. Consider a machine that can be in one of n *states* or *configurations*. This machine is given a sequence of *tasks*, where each task has an associated *cost vector* specifying the cost of performing the task in each state of the machine. There is also a distance metric among the machine's states specifying the cost of moving from one configuration to another. Given a new task, an algorithm chooses either to process the task in the current state (paying the amount specified in the cost vector) or to move to a new state and process the task there (paying both the movement cost and the amount specified in the cost vector for the new state). The problem is to decide on-line what action to take.

A natural way to measure the performance of an on-line algorithm for this problem is the *competitive ratio*. Let $c_A(\sigma)$ represent the cost incurred by an algorithm A on task sequence σ. An on-line algorithm A has a competitive ratio of r if, for some a, for all task sequences σ,

$$c_A(\sigma) \leq r \cdot c_{OPT}(\sigma) + a$$

where OPT represents the optimal off-line algorithm. That is, $c_{OPT}(\sigma)$ is the optimal cost, in hindsight, of processing task sequence σ.

*U C Berkeley and International Computer Science Institute (ICSI), Berkeley. E-mail: `yairb@icsi.berkeley.edu`.
Research supported in part by the Rothschild Postdoctoral Fellowship and by the National Science Foundation operating grants CCR-9304722 and NCR-9416101.

†Carnegie Mellon University. E-mail: `avrim+@cs.cmu.edu`. Supported in part by NSF National Young Investigator grant CCR-9357793 and a Sloan Foundation Research Fellowship.

‡Carnegie Mellon University. E-mail: `cburch+@cmu.edu`. Supported in part by a National Science Foundation Graduate Fellowship.

§Carnegie Mellon University. E-mail: `andrewt+@cs.cmu.edu`.

For randomized algorithms, we replace the cost to A by its expected cost; this is sometimes called the "oblivious adversary" measure since the tasks can be viewed as generated by an adversary that produces the task sequence before any of A's random choices. Specifically, we say that randomized algorithm A has competitive ratio r if, for some a, for all σ,

$$\mathbf{E}[c_A(\sigma)] \leq r \cdot c_{OPT}(\sigma) + a$$

Borodin, Linial, and Saks [BLS92] present a deterministic on-line MTS algorithm that for any metric space achieves a competitive ratio of $2n-1$ and prove that this is optimal for deterministic algorithms (in a strong sense: namely, there is no metric space for which a deterministic algorithm can guarantee a better ratio). They also show that for the special case of the uniform metric space, one can achieve a competitive ratio of $O(\log n)$ with randomization. Several papers have since presented randomized algorithms for other special metric spaces, such as an $O(\log n)$-competitive algorithm for "highly unbalanced" spaces [BKRS92], and an $O(2^{\sqrt{\log n \log\log n}})$-competitive algorithm for equally-spaced points on the line [BRS91, FK90]. Irani and Seiden [IS95] present a randomized algorithm that for arbitrary spaces achieves a competitive ratio of roughly $1.58n$.

Two types of lower bounds are known for randomized MTS algorithms. For certain specific metric spaces such as the uniform space [BLS92], and the superincreasing space [KRR91], there are $\Omega(\log n)$ bounds on the competitive ratio of any randomized on-line algorithm. More generally, a weaker lower bound of $\Omega(\log\log n)$ [KRR91], subsequently improved to $\Omega(\sqrt{\log n/\log\log n})$ [BKRS92], applies to every metric space. A long-standing conjecture maintains that the correct answer is $\Theta(\log n)$: that is, an on-line algorithm exists with competitive ratio $O(\log n)$ for *every* metric space, and there is *no* metric space for which one can guarantee $o(\log n)$.

1.1 The k-HST approximation

Recently, Bartal [Bar96] made important progress by reducing the problem on general metric spaces to the problem on one particular type of space. He defines the following notion.

Definition 1 (k-HST) *A k-hierarchical well-separated tree (k-HST) is a rooted tree with the following properties.*

- *The edge lengths on any path from the root to a leaf decrease by a factor of at least k.*
- *For any node, the lengths of the edges to its children are all equal.*

The metric space induced by a k-HST has one point for each leaf of the tree, with distances given by the tree's path lengths.

For example, a metric space consisting of three "clusters" where the distance between any two points in the same cluster is 1 and the distance between any two points in different clusters is $k+1$ would be a k-HST space. If we add a new point at distance d from all the rest, the metric space remains a k-HST space so long as $d \geq k^2 + k/2 + 1/2$.

Bartal shows how to approximate an arbitrary metric space M by a probability distribution over a set of k-HST spaces. (The reader may find it easier to think of this as a randomized hierarchical decomposition of the metric space M.) This approximation results in the following theorem.

Theorem 1 ([Bar96]) *Suppose there is an algorithm whose competitive ratio is r on any metric space induced by an n-leaf k-HST. Then there is a randomized algorithm for general n-node metric spaces that achieves a competitive ratio of*

$$O(rk \log n \log_k \min\{n, \Delta\})$$

where Δ is the diameter of the metric space (assuming the minimum non-zero distance in the space is 1).

Theorem 1 invites developing strategies for a k-HST. Bartal [Bar96] does this using a variant of the the marking algorithm [FKL+91] and achieves a competitive ratio of $2^{O(\sqrt{\log(\log n+\Delta) \log\log n})}$ for general metric spaces. For metric spaces of $\mathrm{poly}(n)$ diameter (such as shortest-path metrics on unweighted graphs), the ratio is sublinear in n; this is the first sublinear bound known for such spaces.

Our paper improves on this bound in two ways. First, we bring the ratio into the polylogarithmic range, and, second, we remove dependence on Δ. Specifically, we achieve a competitive ratio of $O(\log^6 n/\log\log n)$ on all metric spaces.

Our presentation begins in Section 2 with a description of an on-line problem that will be useful in recursively constructing an algorithm for the k-HST. Section 3 presents and analyzes a strategy for this problem that produces a good solution for balanced $\Omega(\log^2 n)$-HST's. Section 4 describes a strategy that can be used for unbalanced $\Omega(\log^2 n)$-HST's whose branching factor is just two. Section 5 combines these two strategies into a strategy that achieves an $O(\log^2 n)$ competitive ratio for any $\Omega(\log^2 n)$-HST, giving this paper's primary result of a $O(\log^6 n/\log\log n)$-competitive algorithm for arbitrary metric spaces. Sections 6 and 7 discuss other implications. Section 6 gives better competitive ratios for special classes of graphs, and Section 7 explains an on-line strategy for combining other on-line strategies so that we do nearly as well as the best of them.

2 Definitions, preliminaries, and intuition

Because the adversary is oblivious, we can view the MTS problem as follows. The on-line algorithm maintains a probability distribution $(p_1, p_2, \ldots, p_n)$ over the n points in the metric space. Given a task, the algorithm may modify this distribution, paying a cost pd to move p units of probability a distance d. If the algorithm's resulting distribution is $(p'_1, p'_2, \ldots, p'_n)$ and the task vector is $(\delta_1, \delta_2, \ldots, \delta_n)$, then the algorithm pays a cost $\sum_i p'_i \delta_i$ to service the task.

One simplification we will make is to assume that each task vector is *elementary*. That is, every task vector has only one non-zero element. It is a folklore result that this can be assumed without loss of generality (a proof is in the Appendix). We will represent the elementary task with cost δ in state i by the pair (i, δ).

A second simplifying assumption we can make is the following.

Assumption 1 *When a task (i, δ) is received that causes the on-line algorithm to remove all probability from point i, we may assume that δ is the* least *value causing the algorithm to do so. Similarly, we can assume the algorithm never receives a task (i, δ) when p_i is already 0.*

This assumption is made without loss of generality since reducing the value of δ to the minimum that results in $p_i = 0$ does not affect the on-line cost (the on-line player pays only the movement cost) and does not increase the optimal off-line cost. The purpose of this assumption is just to simplify the description of the algorithm, allowing its probability distribution to be solely a function of the "work function" (see below). Alternatively, this assumption can be viewed as a preprocessor to the algorithm.

For a point i in the metric space, let w_i denote the optimal off-line cost of ending in state i after servicing all tasks so far. (This is sometimes called the *work function* [CL91].) Given a task (i, δ), the value of w_j for

$j \neq i$ does not change, and w_i becomes $\min\{w_i + \delta, \min_{j \neq i} w_j + d_{ij}\}$, where d_{ij} is the distance between i and j. One can see the latter by noticing that there are two ways the algorithm could end at state i after the task (i, δ). The off-line algorithm could have already been at i, so that the cumulative cost would be $w_i + \delta$. Or it could have moved from some state j to avoid the δ charge, at a cumulative cost of $w_j + d_{ji}$.

The algorithms we consider will have the property that their distribution of probabilities $(p_1, \ldots, p_n)$ is a function of the w values. We call such an algorithm *w-based*. We will say such an algorithm is *reasonable* if it has the property that p_i is zero when there exists a state j such that $w_i = w_j + d_{ij}$. Notice that an unreasonable w-based algorithm has an unbounded competitive ratio.

Assumption 1 immediately implies the following.

Lemma 1 *For a reasonable w-based on-line algorithm, we may assume that for each request (i, δ), w_i increases by δ.*

Proof. Say we get a request (i, δ) for which w_i increases by $\delta' < \delta$. This can only mean that for some j, $w_i = w_j + d_{ij} - \delta'$ before the request and w_i becomes $w_j + d_{ij}$ after the request. But the request (i, δ') would have produced the same result and, since the algorithm is reasonable, would also result in $p_i = 0$. This contradicts Assumption 1. ■

The algorithms we combine recursively will be even more than reasonable: they will be *hierarchically reasonable*. Suppose the metric space M is partitioned into b subspaces $M_1, \ldots, M_b$, and algorithm A partitions its probability mass over sub-algorithms $A_1, \ldots, A_b$ running on each subspace. We say A is hierarchically reasonable if, when there exist two states i and j in different subspaces such that $w_i = w_j + d_{ij}$, A assigns probability zero to the entire subspace containing point i. This property, combined with Assumption 1, ensures that the algorithm will be reasonable even if each sub-algorithm behaves independently of the w values of points in other subspaces.

2.1 Modeling a k-HST's recursive structure

A k-HST metric space can be understood as a collection of metric spaces separated by some large distance Δ, where each metric space is a smaller k-HST space with diameter at most Δ/k. It is natural, then, to attempt to solve the MTS problem on a k-HST with a recursive algorithm that combines sub-algorithms for the subspaces into an algorithm for the entire space. Say each sub-algorithm is r-competitive. In this case, the problem of combining the sub-algorithms is roughly abstracted by the following scenario.

Scenario 0 *We have an MTS problem on a uniform metric space of b states, but with the following change: when the on-line algorithm services a task, it must pay r times the cost specified in the task vector; the off-line algorithm, however, only incurs the cost specified. In other words, the on-line and off-line algorithms are charged equally for movement, but the on-line algorithm is charged r times more for servicing tasks.*

Because this scenario is a generalization of the MTS problem on a uniform metric space, one natural algorithm to apply is the well-known Marking Algorithm. This algorithm will achieve a competitive ratio of $O(r \log b)$ for Scenario 0. One main result in this paper (Section 3) is an algorithm that improves this ratio to $r + O(\log b)$. This is interesting in itself (see Section 7) and also suggests that applying the algorithm recursively should achieve a ratio of $O(\log n)$ on a balanced k-HST.

2.2 Some complications

Unfortunately, Scenario 0 is too simplistic even for modeling balanced trees. The main problem is that, because the sub-algorithms' ratios are amortized, an r-competitive algorithm for a subspace may pay more than r times the off-line cost for servicing any given request.

To get a handle on this and related issues, it will be helpful to make one additional definition. Notice that the optimal off-line cost is $\min_i w_i$. Since, however, the w_i values differ from each other by at most the diameter of the space, it is legitimate for the on-line algorithm to compete against any one, or even a convex combination $\widehat{w}$, of these values. We will say that *algorithm A achieves competitive ratio r with potential function Φ against convex combination $\widehat{w}$* (assume Φ is non-negative and bounded) if for *every* task t,

$$c_A(t) + \Delta\Phi \leq r \cdot \Delta\widehat{w}$$

where $\Delta\Phi$ and $\Delta\widehat{w}$ represent the changes in Φ and $\widehat{w}$ for the task.

Why is this definition useful? Suppose the r-competitive algorithm for subspace i has potential function Φ_i and competes against the convex combination $\widehat{w}_i$. Consider the potential function $\sum_i p_i \Phi_i$ for the global algorithm (p_i is the total probability in subspace i). Say the on-line algorithm receives a task causing $\widehat{w}_i$ to increase by δ, and as a result the global algorithm moves probability p a distance d from subspace i to subspace j before servicing the task. Then the cost plus potential change incurred by the global algorithm is just $pd + p(\Phi_j - \Phi_i) + p'_i r\delta$, where $p'_i = p_i - p$. In other words, we can ignore the internal amortizations at the expense of an additional cost for movement, where this additional cost is at most p times the maximum value of Φ_j.

The concept of paying more than the off-line player for movement motivates adding a distance ratio to the scenario. We add this, and account for differences in subspaces' sizes, in the following, more careful version of Scenario 0.

Scenario 1 *As before, there are b regions. Each pair of regions is separated by a distance d. Associated with the regions are cost ratios $r_1 \geq r_2 \geq \cdots \geq r_b$, and with the distance is associated a distance ratio s.*

Suppose the on-line algorithm has p_i probability on region i when it receives a request (i, δ). In reaction to the request, the algorithm moves some probability from i, leaving p'_i behind. Then the on-line algorithm's cost for the request is $p'_i r_i \delta + (p_i - p'_i)sd$.

The off-line player, on the other hand, pays only δ for servicing (i, δ) in region i and only d when it moves between regions.

This scenario is a generalization of the aptly named "unfair two state problem" of Seiden [Sei96].

While the primary goal in developing algorithms for this scenario is to optimize the competitive ratio, our secondary goal is to limit the maximum value of the potential function used by the algorithm. This is because, as suggested earlier, if a potential's maximum is large, the distance ratio s will also be large at the next higher level of the recursion used in solving for the k-HST.

3 Combining equal-ratio regions

3.1 Strategy

We develop two new strategies for Scenario 1. The first will loosely approximate all the cost ratios by r_1. The second will handle different ratios more carefully but it will only apply when $b = 2$. We will then combine these to construct an algorithm for the k-HST.

The first strategy will take an odd integer parameter t, which we will later choose to be $O(\log n)$.

Strategy 1 *The strategy takes an odd integer t as a parameter. We allocate to region j the probability*

$$p_j = \frac{1}{b} + \frac{1}{b} \sum_{i=1}^{b} \left(\frac{w_i - w_j}{d} \right)^t \tag{1}$$

For two regions with equal cost ratios, Strategy 1 with $t = 1$ is equivalent to that of Blum, Karloff, Rabani, and Saks [BKRS92]. The following lemma tells us that Strategy 1 fulfills the basic properties described in the previous section.

Lemma 2 *Strategy 1 is w-based, legal (that is, $\sum_j p_j = 1$ and each p_j is nonnegative), and reasonable.*

Proof. That the strategy is w-based is obvious. It maintains a legal probability distribution because, since t is odd, $\sum_j \sum_i (\frac{w_i - w_j}{d})^t = 0$. Because p_j is a decreasing function of only w_j among the w values, Assumption 1 implies that each p_j remains non-negative. (Requests to $i \neq j$ will only increase p_j. Say we receive a request (j, δ) that would make p_j negative if w_j increased by δ. Since the distribution (Equation 1) is continuous, there is an $\delta' < \delta$ for which the algorithm sets p_j to be zero. Assumption 1 implies that we can use (j, δ') instead so that p_j becomes exactly zero.)

Why is Strategy 1 reasonable? Say that $w_j = w_k + d$. Consider the following term from Equation 1.

$$\sum_{i=1}^{b} \left(\frac{w_i - w_j}{d} \right)^t$$

The kth term of the summation is $(-1)^t = -1$. And the ith term of the summation is at most zero for $i \neq k$, since $w_i \leq w_k + d = w_j$. So the summation is at most -1, and p_j is at most zero. ∎

In the remainder of this section we will analyze the strategy's performance and find that its amortized competitive ratio is at most $r_1 + 2sb^{1/t}t$. We will then bound the potential used in the analysis. Finally, we will examine how this strategy performs alone on a k-HST and find that it gives polylog(n) performance for metric spaces of poly(n) diameter.

3.2 Performance

To analyze the performance we will require a simple general lemma.

Lemma 3 *Consider n nonnegative reals $x_1, \ldots, x_n$ and two positive integers $s < t$. If $\sum_i x_i^t \leq 1$, then $\sum_i x_i^s \leq n^{(t-s)/t}$.*

This lemma, presented here without proof, is not difficult to understand. The value of $\sum_i x_i^s$ is maximum when all the terms are equal.

Lemma 4 *The competitive ratio of Strategy 1 is at most $r_1 + 2sb^{1/t}t$. For t being roughly $\lg n$, the ratio is roughly $r_1 + 4 \lg n$.*

Proof. We will use two potential functions Φ_ℓ and Φ_m. The potential function Φ_ℓ will amortize the *local* cost within each region.

$$\Phi_\ell = \frac{r_1 d}{2(t+1)b} \sum_{i=1}^{b} \sum_{j=1}^{b} \left(\frac{w_i - w_j}{d} \right)^{t+1}$$

Notice that Φ_ℓ has the property that, for any j,

$$\frac{\partial \Phi_\ell}{\partial w_j} = -\left(p_j - \frac{1}{b}\right) r_1 \tag{2}$$

The other potential, Φ_m, will amortize the *movement* cost between regions.

$$\Phi_m = \frac{sd}{2b} \sum_{i=1}^{b} \sum_{j=1}^{b} \left| \frac{w_i - w_j}{d} \right|^t$$

The potential Φ for the strategy is simply $\Phi_\ell + \Phi_m$.

We will show that the algorithm's local cost is at most r_1 times the off-line cost and that for movement the algorithm pays at most $2sb^{1/t}t$ times the off-line cost. This will yield the desired bound.

Justified by Lemma 1, we assume that, for a request (k, δ), w_k increases from some value y to $y + \delta$. In this analysis the strategy will compete against the average w value, $\sum w_i / b$. So the off-line cost is δ / b.

Let p_k and p'_k represent the probability in region k before and after the task vector, and let Φ_ℓ (Φ_m) and Φ'_ℓ (Φ'_m) represent the local (movement) potential before and after the task vector. Then the on-line strategy's cost will be

$$p'_k r_k \delta + (p_k - p'_k) sd + \Phi'_\ell + \Phi'_m - \Phi_\ell - \Phi_m$$

Because p_k decreases as a function of w_k, we can upper-bound this cost using an integral.

$$\int_y^{y+\delta} \left(p_k r_k + \frac{\partial \Phi_\ell}{\partial w_k} - \frac{\partial p_k}{\partial w_k} sd + \frac{\partial \Phi_m}{\partial w_k} \right) dw_k \tag{3}$$

We will examine the first two terms, representing the local cost, and the last two terms, representing the movement cost, separately. For the local cost, we have (using Equation 2)

$$p_k r_k + \frac{\partial \Phi_\ell}{\partial w_k} = p_k r_1 - (p_k - \frac{1}{b}) r_1 = \frac{r_1}{b} \tag{4}$$

Thus the total local cost,

$$\int_y^{y+\delta} \left(p_k r_k + \frac{\partial \Phi_\ell}{\partial w_k} \right) dw_k \, ,$$

is at most $\delta r_1 / b$, which is r_1 times the off-line cost as desired.

Analyzing the movement cost requires more work.

$$\begin{aligned}
& -\frac{\partial p_k}{\partial w_k} sd + \frac{\partial \Phi_m}{\partial w_k} \\
= \; & \frac{st}{b} \sum_{i \neq k} \left(\frac{w_i - w_k}{d} \right)^{t-1} + \frac{st}{b} \sum_{w_i < w_k} \left(\frac{w_k - w_i}{d} \right)^{t-1} \\
& -\frac{st}{b} \sum_{w_i > w_k} \left(\frac{w_i - w_k}{d} \right)^{t-1} \\
= \; & \frac{2st}{b} \sum_{w_i < w_k} \left(\frac{w_k - w_i}{d} \right)^{t-1}
\end{aligned} \tag{5}$$

We would like to simplify the summation. Say that w_a is currently the maximum w value. Observe using the probability allocation (Equation 1) that, since p_a is not negative, the following holds.

$$\sum_{i \neq a} \left(\frac{w_a - w_i}{d} \right)^t \leq 1 \tag{6}$$

Because w_a is maximum, each term of the summation is positive. Thus it follows from Lemma 3 that

$$\sum_{i \neq a} \left(\frac{w_a - w_i}{d} \right)^{t-1} \leq (b-1)^{1/t} < b^{1/t}$$

Using the definition of a again we can continue from Equation 5 to finish approximating the movement cost.

$$\begin{aligned} \frac{2st}{b} \sum_{w_i < w_k} \left(\frac{w_k - w_i}{d} \right)^{t-1} &\leq \frac{2st}{b} \sum_{i \neq a} \left(\frac{w_a - w_i}{d} \right)^{t-1} \\ &< \frac{2sb^{1/t}t}{b} \end{aligned} \tag{7}$$

The estimates of the local cost (4) and movement cost (7) bound the total cost (3) by

$$\begin{aligned} &\int_{w_k}^{w_k+\delta} \left(p_k r_k + \frac{\partial \Phi_\ell}{\partial w_k} + \frac{\partial p_k}{\partial w_k} sd + \frac{\partial \Phi_m}{\partial w_k} \right) dw_k \\ &\quad \leq \int_{w_k}^{w_k+\delta} \frac{r_1 + 2sb^{1/t}t}{b} \, dw_k \\ &\quad = \frac{\delta}{b} (r_1 + 2sb^{1/t}t) \end{aligned}$$

So the competitive ratio is $r_1 + 2sb^{1/t}t$ as desired. ∎

As Lemma 4 states, if we let t be at least $\lg b$, then $b^{1/t}$ is at most 2, so the ratio is $r_1 + 4st$. Since we approximate $b^{1/t}$ in exactly this way, why would we ever want t to increase beyond $\lg b$? A larger exponent t implies that the maximum potential is smaller.

3.3 Potential

To apply Strategy 1 recursively on a k-HST, we must bound the potential.

Lemma 5 *The potential in Lemma 4 is bounded by*

$$0 \leq \Phi \leq \left(\frac{r_1}{t+1} + s \right) d$$

Proof. The lower bound is trivial. Let us concern ourselves with the upper bound, bounding Φ_ℓ and Φ_m separately. To bound Φ_ℓ, let a be the index of the maximum w value.

$$\begin{aligned} \Phi_\ell &= \frac{r_1 d}{2(t+1)b} \sum_i \sum_j \left(\frac{w_i - w_j}{d} \right)^{t+1} \\ &= \frac{r_1 d}{(t+1)b} \sum_i \sum_{w_j < w_i} \left(\frac{w_i - w_j}{d} \right)^{t+1} \\ &\leq \frac{r_1 d}{(t+1)b} \sum_i \sum_{w_j < w_i} \left(\frac{w_a - w_j}{d} \right)^{t+1} \\ &\leq \frac{r_1 d}{t+1} \sum_j \left(\frac{w_a - w_j}{d} \right)^{t+1} \end{aligned}$$

$$\leq \frac{r_1 d}{t+1} \sum_j \left(\frac{w_a - w_j}{d}\right)^t \tag{8}$$

$$\leq \frac{r_1 d}{t+1} \tag{9}$$

Inequality 8 follows because, since $w_a \leq w_j + d$, each term of the summation is at most one, so reducing the term's exponent increases the term's value. Inequality 9 comes from Inequality 6.

Bounding Φ_m is similar. Again, let a be the index of the maximum w value.

$$\begin{aligned}
\Phi_m &= \frac{sd}{2b} \sum_i \sum_j \left|\frac{w_i - w_j}{d}\right|^t \\
&= \frac{sd}{b} \sum_i \sum_{w_j < w_i} \left(\frac{w_i - w_j}{d}\right)^t \\
&\leq \frac{sd}{b} \sum_i \sum_{w_j < w_i} \left(\frac{w_a - w_j}{d}\right)^t \\
&\leq sd \sum_j \left(\frac{w_a - w_j}{d}\right)^t \\
&\leq sd
\end{aligned}$$

Adding this to the bound for Φ_ℓ (9) gives a bound for the total potential of Strategy 1.

$$\Phi = \Phi_\ell + \Phi_m \leq \left(\frac{r_1}{t+1} + s\right) d$$

This is as promised. ■

3.4 Recursive application

With the strategy's performance and potential bounded we can quickly look at how the strategy performs by itself on k-HST's. While this analysis is not necessary for the final result, seeing a simpler strategy applied to a k-HST will make the later presentation clearer. It will also suggest why this strategy alone is not enough to get a bound polylogarithmic in n.

First, let us formalize the argument that a recursive algorithm can ignore sub-algorithms' potentials by increasing the distance ratio. Say the algorithm receives a task whose non-zero component is in region i. The sub-algorithm for the region has amortized ratio r_i, so the amortized local cost for being in that region is $r_i\delta$. We are interested in its true cost, however. This is $r_i\delta - (\Phi_i' - \Phi_i)$ if Φ_i and Φ_i' respectively represent the sub-algorithm's potential before and after the request.

Lemma 6 *Consider a version of Scenario 1 where cost ratios are amortized with potentials bounded by Φ_{max}. That is, requests to region i cost not $p_i r_i \delta$ but $p_i(r_i\delta - (\Phi_i' - \Phi_i))$. Let A be an r-competitive algorithm for the original Scenario 1 whose distance ratio is $\hat{s} + \Phi_{max}/d$. Then there exists an r-competitive algorithm $\tilde{A}$ for the scenario with amortized ratios and distance ratio $\hat{s}$. The potential of $\tilde{A}$ is at most Φ_{max} plus the potential of A.*

Proof. In $\tilde{A}$ we allocate probability as in A. Let Φ_A be the potential of A. The potential of $\tilde{A}$ will be

$$\Phi_{\tilde{A}} = \Phi_A + \sum_{j=1}^{b} p_j \Phi_j$$

That this is at most $\Phi_A + \Phi_{max}$ is clear.

Say $\tilde{A}$ receives a request (i, δ). It acts as A would with this request, leaving $p'_i < p_i$ probability on region i. The amortized cost to $\tilde{A}$ for processing the request can be split into two pieces. First we will move probability. Movement costs us

$$
\begin{aligned}
& (p_i - p'_i)\hat{s}d + \sum_{j=1}^{b} p'_j \Phi_j - \sum_{j=1}^{b} p_j \Phi_j \\
\leq \quad & (p_i - p'_i)\hat{s}d - (p_i - p'_i)\Phi_i + \sum_{j \neq i}^{b} (p'_j - p_j)\Phi_{max} \\
= \quad & (p_i - p'_i)\hat{s}d - (p_i - p'_i)\Phi_i + (p_i - p'_i)\Phi_{max} \\
\leq \quad & (p_i - p'_i)(\hat{s} + \Phi_{max}/d)d
\end{aligned}
$$

After we move probability, we pay the amortized cost ratio. (Notice that Φ_j remains unchanged for $j \neq i$, since nothing has changed in that region.) This cost is

$$p'_i(r_i\delta - \Phi'_i + \Phi_i) + p'_i(\Phi'_i - \Phi_i) + \Phi'_A - \Phi_A = p'_i r_i \delta + \Phi'_A - \Phi_A$$

So the total cost to $\tilde{A}$ is

$$p'_i r_i \delta + (p_i - p'_i)(\hat{s} + \Phi_{max}/d)d + \Phi'_A - \Phi_A$$

Since this is the amortized cost to A and the off-line cost is the same in both cases, $\tilde{A}$ is r-competitive. ∎

Now we can examine applying Strategy 1 to a k-HST.

Lemma 7 *Consider an n-point metric space induced by a k-HST of depth D with $k \geq 9D$. The competitive ratio of applying Strategy 1 with* $\lg n \leq t < \lg n + 2$ *is at most* $9D \lg n$.

Corollary 8 *There is a randomized algorithm for the MTS problem with competitive ratio*

$$O(\log^3 n \log^2 \Delta / \log^3 \log \Delta)$$

Proof. Choose k to be $18 \lg \Delta / \lg \lg \Delta$. Then the depth of the k-HST is at most $\log_k \Delta < k/9$. This choice satisfies the conditions of the Lemma 7. Applying Theorem 1 gives the result. ∎

Proof of Lemma 7. We will use induction on D to show that Strategy 1 on a D-depth k-HST (with $k \geq 9D$ and diameter Δ) has a competitive ratio of $9D \lg n$ with maximum potential $9D\Delta$. This is clearly true when D is zero.

For the induction step, we see that the diameter of each subspace is at most Δ/k, and the depth of each is at most $D - 1$. So the strategy for each subspace has a competitive ratio of at most $9(D-1)\lg n$ and a potential of at most $9(D-1)\Delta/k$.

If we were to simply apply Strategy 1 with $d = \Delta$, the strategy would not be hierarchically reasonable. Consider a node n_i in subspace R_i whose w value is w_{n_i} and a node n_j in subspace R_j with w value w_{n_j}. We need the probability on n_i to be zero if $w_{n_i} = w_{n_j} + \Delta$. Because the strategy for R_j uses a convex combination of the w values, however, the w_j that Strategy 1 sees may be as much as $w_{n_j} + \Delta/k$. Meanwhile, the internal node may see w_i as small as $w_{n_i} - \Delta/k$. So the difference of w_i and w_j may be only $\Delta - 2\Delta/k$. That they differ by less than Δ implies that the strategy may allocate probability to R_j. Unaware of even the existence of n_i, R_j may allocate some of this probability to n_j, which we must avoid.

One solution to this problem is the following. We will set d to be $\Delta - 2\Delta/k$ while setting the distance ratio $\hat{s}$ to be $k/(k-2)$ to keep $\hat{s}d$ at the true distance between subspaces, Δ. This reduces the off-line player's distance cost, hurting the strategy's ratio slightly. But these parameters guarantee that the strategy is hierarchically reasonable.

The other issue we must consider is internal potentials. We can apply Lemma 6 to take care of this. Since $k \geq 9D$, the maximum potential of each subspace ($9(D-1)\Delta/k$) is at most d. So we can use a distance ratio of $\hat{s}+1 \leq 9/4$ (because $k \geq 10$).

To calculate the competitive ratio for the k-HST, we use Lemma 4.

$$r_1 + 2sb^{1/t}t \leq 9(D-1)\lg n + 9\lg n = 9D\lg n$$

(We bound $b^{1/t}$ by two because $t \geq \lg b$.) Lemmas 5 and 6 bound the potential.

$$\begin{aligned} & \left(\frac{r_1}{t+1} + s\right)\Delta + 9(D-1)\frac{\Delta}{k} \\ \leq\ & \left(\frac{9(D-1)\lg n}{\lg n} + \frac{13}{4}\right)\Delta \\ \leq\ & 9D\Delta \end{aligned}$$

These two bounds satisfy the induction. ■

Notice that this ratio is polylogarithmic in n for poly(n)-diameter graphs. This is already an improvement on the result of [Bar96]. In the remainder of this paper, we show how to achieve a ratio polylogarithmic in n only, without restricting the class of metric spaces in any way.

4 Combining two regions

4.1 Strategy

We wish to remove the appearance of Δ in the ratio. The diameter appeared when we bounded the depth of the tree by $\log_k \Delta$. This occurs; for example, consider a k-HST decomposition of a superincreasing metric space, where the points lie on a line at $1, k, k^2, \ldots k^n$. In such a tree, however, many internal nodes have a subtree much larger than any of its siblings. This motivates the following idea: if one subtree is much larger than the remaining $b-1$ combined, then we will use Strategy 1 on the $b-1$ smaller trees, and then carefully combine the result with the larger one. To do this, we need a method for carefully combining two spaces of unequal ratios.

In this section we consider this problem of carefully combining two regions. For $s = 1$, the problem is one examined by Blum, Karloff, Rabani, and Saks [BKRS92]. (They were concerned with a metric space consisting of two spaces separated by a large distance. That paper was able to ignore the internal potential functions and additive constants by assuming the two spaces were sufficiently far apart. Because we cannot afford to assume the spaces are so separated, we must be more careful and introduce $s > 1$.) By appropriately modifying the technique used in that paper, we get a strategy for Scenario 1 with two regions. (Seiden [Sei96] independently developed the same algorithm. We present it here for completeness.)

Strategy 2 *Let $p_1(y)$ be the following function.*

$$p_1(y) = \frac{e^{\frac{r_1-r_2}{s}} - e^{\frac{r_1-r_2}{s}\left(\frac{1}{2}+\frac{y}{2d}\right)}}{e^{(r_1-r_2)/s} - 1} \tag{10}$$

When $b = 2$ in Scenario 1, the strategy places $p_1(w_1 - w_2)$ probability in the first region and the rest in the second.

While the strategy is hardly intuitive, the analysis will make the reason for the selection clear.

4.2 Performance

Lemma 9 *The competitive ratio of Strategy 2 is*

$$r_1 + \frac{r_1 - r_2}{e^{(r_1 - r_2)/s} - 1}$$

The potential of the strategy never exceeds $(2r_2 + s)d$

Proof. Notice that the strategy is w-based. Because $p_1(d) = 0$ and $p_1(-d) = 1$, it is legal and reasonable.

Our analysis will compete against w_1. This means that the cost must be zero when w_2 increases, so these costs will be absorbed by the potential. Our potential, therefore, is

$$\Phi = (1 - p_1)sd + \int_{-d}^{w_1 - w_2} (1 - p_1(y))r_2 \, dy$$

Because $w_1 - w_2$ is always at least $-d$, this potential is nonnegative. And because the integrand is at most r_2, the second term is at most $2r_2d$, while the first term is at most sd. So the potential is bounded by $\Phi \leq (2r_2 + s)d$.

A request $(2, \delta)$ will be absorbed completely by the potential. Let us consider a request $(1, \delta)$ bringing $w_1 - w_2$ from z to $z + \delta$. Then, the strategy's cost is at most

$$\begin{aligned} & \int_z^{z+\delta} \left(p_1(y)r_1 - sd\frac{dp_1}{dy} + \frac{d\Phi}{dy} \right) dy \\ \leq & \int_z^{z+\delta} \left(p_1(y)r_1 - 2sd\frac{dp_1}{dy} + (1 - p_1(y))r_2 \right) dy \end{aligned}$$

(The integral approximates the cost because p_1 is a decreasing function.) By setting this to a constant we obtain a first-order differential equation in p_1, which can be solved with the boundary conditions $p_1(d) = 0$ and $p_1(-d) = 1$. It is easy to verify that if p_1 is as in Equation 10, the integrand is constant.

$$\begin{aligned} & \int_z^{z+\delta} \left(p_1(y)r_1 - 2sd\frac{dp_1}{dy} + (1 - p_1(y))r_2 \right) dy \\ & \qquad = \int_z^{z+\delta} r_1 + \frac{r_1 - r_2}{e^{(r_1 - r_2)/s} - 1} \, dy \\ & \qquad = \left(r_1 + \frac{r_1 - r_2}{e^{(r_1 - r_2)/s} - 1} \right) \delta \end{aligned}$$

Since the off-line player pays δ, the competitive ratio for the strategy is as advertised. ∎

5 Combining the strategies on a k-HST

5.1 Strategy

Our strategy combines Strategy 2 with Strategy 1 at internal nodes of the k-HST where, roughly, the first subtree contains a disproportionate number of nodes.

Strategy 3 *Consider an internal node of a k-HST space whose b subspaces have ratios $r_1 \geq r_2 \geq \cdots \geq r_b$. The ith subtree contains n_i nodes. Let n represent $\sum_{i=1}^{b} n_i$.*

Our strategy for the node will be the following.

1. *If $r_1 \leq 128 \lg^2 n - 32 \lg n$, we use Strategy 1 with t an odd integer between $2 \lg n$ and $2 \lg n + 2$.*
2. *If $r_1 > 128 \lg^2 n - 32 \lg n$, we combine all but the first subspace using Strategy 1 with $2 \lg n \leq t \leq 2 \lg n + 2$. Then we use Strategy 2 to combine this with the first subspace.*

In the analysis we will determine acceptable values to use for k, s, and d.

5.2 Performance

We will show that the strategy is $O(\log^2 n)$-competitive on an $\Omega(\log^2 n)$-HST using induction. The following lemma allows us to combine subspaces with Strategy 1.

Lemma 10 *Consider a k-HST with diameter Δ. Say that we have a strategy for each subspace with competitive ratio $r_i \leq 128 \lg^2 n_i$ and maximum potential $((k-2)/2)\Delta/k$. To combine the subspaces, we apply Strategy 1 with $2 \lg n \leq t \leq 2 \lg n + 2$, $d = ((k-2)/2k)\Delta$, and $s = 2k/(k-2) + 1$. Then the total competitive ratio is at most $r_1 + 32 \lg n$ and the maximum potential is at most $(64 \lg n + 17/4)((k-2)/2k)\Delta$.*

Proof. Let $\hat{s}$ be $2k/(k-2)$ so that in paying $\hat{s}d$ to move between subspaces the on-line strategy pays Δ. Because the potential for each subspace is at most d, we can avoid the potentials through Lemma 6 if the distance ratio s is $\hat{s} + 1$. This is at most $13/4$ if $k \geq 18$. Because $t \geq \lg b$, $b^{1/t}$ is at most 2, so Lemma 4 gives a ratio of $r_1 + 2sb^{1/t}t \leq r_1 + 32 \lg n$. The maximum potential is

$$\left(\frac{r_1}{t+1} + s\right) d + d \leq (64 \lg n + 17/4)((k-2)/2k)\Delta$$

by Lemmas 5 and 6. ■

This lemma will help in the final proof giving the performance of Strategy 3 on a k-HST.

Lemma 11 *For a k-HST with $k \geq 256 \lg^2 n + 128 \lg n + 11$, applying Strategy 3 achieves a competitive ratio of at most $128 \lg^2 n$.*

Corollary 12 *There is a randomized algorithm for the MTS problem with competitive ratio*

$$O(\log^6 n / \log \log n)$$

Proof. Combine Lemma 11 with Theorem 1. ■

Proof of Lemma 11. Consider a k-HST whose diameter is Δ. Inductively we assume that each r_i is at most $128 \lg^2 n_i$ and that $\Phi_i \leq ((k-2)/2)\Delta/k$. We want to show that the strategy's ratio is at most $128 \lg^2 n$ and the potential is at most $((k-2)/2)\Delta$. Our strategy has two cases, which we analyze separately.

Case 1 Apply Lemma 10. The ratio will be at most $r_1 + 32 \lg n \leq 128 \lg^2 n$. Because $k \geq 64 \lg n + 17/4$, the maximum potential is at most $((k-2)/2)\Delta$.

Case 2 Due to the requirement of hierarchical reasonableness, in applying both Strategy 1 and Strategy 2 we will take d to be $(\Delta - 2\Delta/k)/2$ while setting $\hat{s}$ at $2k/(k-2)$ so that our strategies still pay Δ to move between subspaces. In this way Strategy 1 will not allow w values in different regions to become more than $(\Delta - 2\Delta/k)/2$ apart, nor will Strategy 2 allow w values to grow more than $(\Delta - 2\Delta/k)/2$ apart, so together they will not allow w values to differ by more than $\Delta - 2\Delta/k$. Since each subtree's strategy will never allow any two of its nodes to differ by more than Δ/k, a node whose w value is Δ more than another's will receive no probability.

Let x be so that $n_1 = n(1 - 1/x)$. (Because $r_1 > 128\lg^2 n - 32\lg n$, $4 \leq x \leq n$.) By Lemma 10, the ratio r_2' of the combination of the smaller subspaces is at most

$$\begin{aligned} r_2' &\leq 128\lg^2(n/x) + 32\lg n \\ &\leq 128\lg^2 n - 256\lg x\lg n + 128\lg^2 x + 32\lg n \end{aligned}$$

Because the maximum potential is $(64\lg n + 17/4)d$ and $\hat{s}$ is at most $9/4$ if $k \geq 18$, we will bound s by $64\lg n + 13/2$ in combining r_1 with r_2'.

To calculate the ratio of the entire space we will first bound $r_1 - r_2'$.

$$\begin{aligned} r_1 - r_2' &> 128\lg^2 n - 32\lg n - (128\lg^2 n \\ &\quad -256\lg x\lg n + 128\lg^2 x + 32\lg n) \\ &= 256\lg x\lg n - 128\lg^2 x - 64\lg n \\ &\geq 96\lg x\lg n \end{aligned} \tag{11}$$

The ratio for the combination of r_1 with r_2' is that of Strategy 2,

$$r \leq r_1 + \frac{r_1 - r_2'}{e^{(r_1 - r_2')/s} - 1} \tag{12}$$

The second term of the ratio $((r_1 - r_2')/(e^{(r_1 - r_2')/s} - 1))$ decreases when $r_1 - r_2'$ increases beyond s. So we can use Equation 11 to bound the ratio of Lemma 9.

$$\begin{aligned} r &\leq r_1 + \frac{r_1 - r_2'}{e^{(r_1 - r_2')/s} - 1} \\ &\leq r_1 + \frac{96\lg x\lg n}{e^{\frac{96\lg x\lg n}{64\lg n + 13/2}} - 1} \\ &\leq r_1 + \frac{96\lg x\lg n}{x^2 - 1} \\ &\leq 128\left(\lg n + \lg(1 - \frac{1}{x})\right)^2 + \frac{128\lg x\lg n}{x^2} \\ &\leq 128\lg^2 n - \frac{256\lg n}{x} + \frac{128}{x^2} + \frac{128\lg x\lg n}{x^2} \\ &\leq 128\lg^2 n \end{aligned}$$

From Lemma 9 the maximum potential for combining the two subspaces is $(2r_2' + s)d$, and we add at most $(64\lg n + 17/4)d$ through Lemma 6. So the potential is at most

$$\begin{aligned} &(2r_2' + s)d + (64\lg n + 17/4)d \\ &\quad\leq (256\lg^2 n + 128\lg n + 11)\frac{\Delta - 2\Delta/k}{2} \end{aligned}$$

If the potential is to be at most $((k-2)/2)\Delta$, we should choose k to be at least $256\lg^2 n + 128\lg n + 11$, as specified in the statement of the Lemma. ∎

6 Results for specific metric space

The result for arbitrary metric spaces is based on Theorem 1. More specifically, the algorithm would probabilistically construct an $\Omega(\log^2 n)$-HST space according to [Bar96] and then apply Strategy 3 for the HST. The algorithm for the original metric space uses the same states as the simulated HST algorithm. Strategy 3 has competitive ratio $r = O(\log^2 n)$, so the competitive ratio of the algorithm for the original metric space is $O(rk \log n \log_k n) = O(\log^6 n/\log\log n)$.

This bound improves as the k-HST approximation ratio improves for specific metric spaces. In particular the results of [Bar96] imply that for weighted trees the ratio improves to $O(rk \log_k n) = O(\log^5 n/\log\log n)$.

Note that if the metric space is induced by distances in an unweighted graph then using Corollary 8 we get a ratio of $O(\log^5 n/\log^3 \log n)$.

For d-dimensional meshes we can achieve better bounds by approximating them using balanced k-HST spaces. An HST is called *balanced* if at every internal node the size of the subtrees rooted at this node are equal. Bartal [Bar96] shows that an r-competitive algorithm for balanced k-HST spaces implies a competitive ratio of $O(rk \log_k n)$ for meshes. In the rest of this section we will prove an $O(\log n)$ competitive ratio for balanced $\Omega(\log n)$-HST spaces and hence we get an $O(\log^3 n/\log\log n)$ ratio for meshes.

Consider a k-HST with $k \geq 18 \log n$. We will prove by induction on the size of the tree that there is an algorithm for the MTS problem with competitive ratio at most $9 \log n$ and maximum potential $(k-2)\Delta$. This is clearly true for $n = 1$. Let the degree of the root be b. Then each of its subtree has size n/b. By induction the algorithm for each subtree has competitive ratio at most $9 \log(n/b)$ and maximum potential $((k-2)/k)\Delta$. We apply Strategy 1 to combine all b algorithms setting $t = \log b$, $d = ((k-2)/k)\Delta$ and $\hat{s} = k/(k-2)$. As in Lemma 7 this choice of d and $\hat{s}$ ensures that the strategy will be hierarchically reasonable. Since the maximum potential of a subspace is at most d we can use a distance ratio $s = \hat{s} + 1 \leq 9/4$. The competitive ratio for the entire tree is:

$$r_1 + 2sb^{1/t}t \leq 9\log(n/b) + 9\log b = 9\log n.$$

The potential is bounded by

$$\left(\frac{r_1}{t+1} + s\right)\Delta + \frac{k-2}{k}\Delta \leq \left(\frac{9\log(n/b)}{\log b} + \frac{13}{4}\right)\Delta \leq (k-2)\Delta.$$

7 Side notes

One of the key pieces to our main theorem is the result that Strategy 1 achieves a competitive ratio $r + O(\log b)$ for the setup of Scenario 0: that is, a uniform space of b points where the on-line algorithm is charged a factor r more than the off-line for processing tasks. This result has the following interesting application. Consider the *standard* MTS problem on an n-point uniform metric space and suppose we apply Strategy 1 as if $r = \log n$ and as if each component of the task vector were multiplied by $1/r$. In other words, given a task vector $(\delta_1, \ldots, \delta_n)$, the on-line algorithm believes (which happens to be the truth) that it is paying δ_i to process the task in state i, but it also believes that the off-line algorithm would pay only δ_i/r to process the task in that state.

Given a task sequence σ, consider some off-line solution that (in truth) spends α to process tasks and β to move among states (for a total of $\alpha + \beta$). The on-line algorithm believes, however, that the off-line cost of that solution is only $\alpha/r + \beta$. Therefore, we know that the on-line cost will be at most $(r + 4\lg n)(\alpha/r + \beta) + \mathit{constant} = O(\alpha + \beta \log n)$. In other words, Strategy 1 is simultaneously $O(\log n)$-competitive with respect to the optimal off-line solution (just like the Marking Algorithm), but also constant-competitive with respect to the optimal solution that does not move between states (i.e., $\beta = 0$) and even constant-competitive with respect to the optimal solution that spends at most an $O(1/\log n)$ fraction of its cost for movement (i.e., $\alpha + \beta \log n = O(\alpha)$).

So, this algorithm has the property that its performance (as measured by the competitive ratio) is a function of how "hard" the sequence is (as measured by how much movement is needed to perform well off-line).

If instead we apply Strategy 1 as if $r = \frac{1}{\epsilon} \lg n$, then we get a cost of $\alpha(1+4\epsilon) + \beta(1+4/\epsilon) \lg n$. If we used this to combine on-line schemes on-line, we would pay only $(1 + 4\epsilon)$ times as much as the best of them, plus a cost that depends on $1/\epsilon$, $\lg(\#\,\mathrm{schemes})$, and the maximum cost of switching between two schemes' states. (This follows from the fact that the bound holds for the strategy that remains at one scheme after paying only to move from the initial scheme's state to that scheme.) This is closely related to the kinds of bounds known for algorithms for "predicting from expert advice" in the Machine Learning setting [HW95] and these parallels are discussed further in [BB97].

8 Conclusions

The strategy implied by Corollary 12 is this paper's main result, a randomized on-line MTS algorithm whose competitive ratio is $O(\log^6 n/\log\log n)$ for any metric space. As noted in Section 7, some of the key ingredients to this result have other interesting implications as well.

The MTS problem is related to the k-server problem introduced by Manasse, McGeoch, and Sleator [MMS90]. In particular, a k-server problem on $k + c$ points can be expressed as a $\binom{k+c}{c}$-state MTS problem in which each state represents a configuration of the servers. Thus Corollary 12 implies a competitive ratio of $O(c^6 \log^6 k)$ for the k-server problem on a metric space of $k + c$ points. The best general known result for the k-server problem, due to Koutsoupias and Papadimitriou [KP95], is a competitive ratio of $2k - 1$.

Two interesting open questions that remain are: Can one achieve an $O(\log n)$-competitive ratio for the MTS problem? And, for the k-server problem, can one achieve a polylog(k) competitive ratio, perhaps by extending the ideas of this paper?

We would like to acknowledge helpful discussion with Mike Saks.

References

[Bar96] Y. Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 183–193, October 1996.

[BB97] A. Blum and C. Burch. On-line learning and the metrical task system problem. In *Proceedings of the 10th Annual Conference on Computational Learning Theory*, pages 45–53, 1997.

[BKRS92] A. Blum, H. Karloff, Y. Rabani, and M. Saks. A decomposition theorem and lower bounds for randomized server problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 197–207, October 1992.

[BLS92] A. Borodin, N. Linial, and M. Saks. An optimal online algorithm for metrical task systems. *JACM*, 39(4):745–763, 1992.

[BRS91] A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar geometric terrain. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, New Orleans, May 1991.

[CL91] M. Chrobak and L. Larmore. The server problem and on-line games. In *On-Line Algorithms: Proceedings of a DIMACS Workshop*, pages 11–64, February 1991.

[FK90] D. Foster and H. Karloff. Personal communication, 1990.

[FKL+91] A. Fiat, R. Karp, M. Luby, L. A. McGeoch, D. Sleator, and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12:685–699, 1991.

[HW95] M. Herbster and M. Warmuth. Tracking the best expert. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 286–294. Morgan Kaufmann, 1995.

[IS95] S. Irani and S. Seiden. Randomized algorithms for metrical task systems. In *WADS*, pages 159–170, 1995.

[KP95] E. Koutsoupias and C. Papadimitriou. On the k-server conjecture. *JACM*, 42(5):971–983, 1995.

[KRR91] H. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion planning algorithms. In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, pages 278–288, May 1991.

[MMS90] M. Manasse, L. McGeoch, and D. Sleator. Competitive algorithms for on-line problems. *Journal of Algorithms*, 11:208–230, 1990.

[Sei96] S. Seiden. Unfair problems and randomized algorithms for metrical task systems. Manuscript, April 1996.

Appendix

The following is a folklore theorem. The proof has not appeared in print, however, so we include one here.

Theorem 2 *For any metric space, given an r-competitive algorithm for the metrical task system problem with elementary task vectors, it is possible to construct an algorithm for the general metrical task system problem with competitive ratio $(1+\epsilon)r$, for any $\epsilon > 0$.*

Proof. Let σ be a sequence of arbitrary (not necessarily elementary) task vectors. First, we will show how to construct a subsequence of elementary task vectors for each single task vector of σ, and will concatenate the resulting subsequences into a new sequence of elementary task vectors τ. Next, we will show how the behavior of an r-competitive algorithm A for sequences of elementary task vectors, operating on sequence τ, can be used to induce a new algorithm B for the original sequence σ. Finally, we will show that $B(\sigma) \leq (1+\epsilon)A(\tau)$ and $\mathrm{OPT}(\tau) \leq \mathrm{OPT}(\sigma)$, which gives the result.

An individual task vector v of σ can be converted into a subsequence of elementary task vectors of τ as follows. Let $v = (\delta_1, \delta_2, \ldots, \delta_n)$ be an arbitrary task vector, and let δ be some small value to be determined later.

while some $\delta_i > 0$ do {
    /* begin next stripe */
    for $j \leftarrow 1$ to n
        if ($\delta_j > 0$) {
            output task $(\overbrace{0, \ldots, 0}^{j-1}, min(\delta_j, \delta), 0 \ldots 0)$

```
            δ_j ← max(0, δ_j − δ)
        }
}
```

This construction shows how to create τ from σ. Algorithm B on σ works as follows. It begins by initializing a copy of algorithm A, and maintains the invariant that the state of B after processing some vector v is the state of A after processing the corresponding subsequence of elementary task vectors. Specifically, when B is presented with v, it creates a subsequence of elementary task vectors according to the construction above, and then passes the resulting vectors to algorithm A one at a time. A begins in some state s_0, then passes through some set of states S in the course of processing the elementary task vectors, and ends in some final state s_2. Each of these states will have some cost in the original vector $v = (\delta_1, \delta_2, \ldots, \delta_n)$; let state s_1 be the state of S with lowest cost: $s_1 = argmin_{s \in S}\{\delta_s\}$. Algorithm B begins in state s_0, immediately jumps to state s_1 to process v, and finally jumps to state s_2 to remain in correspondence with A.

Having defined τ and algorithm B, we must show that $B(\sigma) \leq (1+\epsilon)A(\tau)$ and $\mathrm{OPT}(\tau) \leq \mathrm{OPT}(\sigma)$. The second inequality is simpler. *Any* solution for σ can be used as a solution for τ with the same cost. If the solution for σ jumps to state j to process some task v, then jumping to state j results in the same total cost to process the subsequence of τ corresponding to v.

The first inequality states $B(\sigma) \leq (1+\epsilon)A(\tau)$. Consider some task vector $v = (\delta_1, \delta_2, \ldots, \delta_n)$ of σ. Let V be the corresponding subsequence of elementary task vectors in τ. The construction breaks V into a number of logical units called "stripes," each stripe consisting of up to n elementary task vectors, and no two vectors in a stripe having non-zero values for the same state. We denote the stripes STRIPE1, STRIPE2,....

In the course of processing V, say that algorithm A changes state k times, paying total distance d_{total}. Define $n_1 = \lfloor \delta_{s_1}/\delta \rfloor$. During processing of STRIPEj for $j \leq n_1$, A must either move (which it can do at most k times), or else pay cost δ by the definition of s_1. The total cost to algorithm A over V, Cost(A, V), is therefore at least $d_{total} + (n_1 - k)\delta$. Let $d_{\min}$ be the smallest distance in the space, and choose $\delta < \epsilon d_{\min}/2$. Additionally, choose $\delta < \epsilon\delta_j/2$ for all j, so $n_1\delta > \delta_{s_1}(1-\epsilon/2)$. Then

$$\begin{aligned}
\text{Cost}(A,V) &\geq (1-\epsilon/2)d_{total} + \epsilon k d_{\min}/2 + (n_1 - k)\delta \\
&\geq (1-\epsilon/2)d_{total} + n_1\delta \\
&\geq (1-\epsilon/2)d_{total} + (1-\epsilon/2)\delta_{s_1}
\end{aligned}$$

Finally, note that A must travel from s_0 via s_1 to s_2, so we have $d_{total} \geq d_{s_0,s_1} + d_{s_1,s_2}$. So we can write the final lower bound on the cost of algorithm A as follows:

$$\text{Cost}(A,V) \geq (1-\epsilon/2)(d_{s_0,s_1} + d_{s_1,s_2} + \delta_{s_1}).$$

Recall that B begins servicing v in s_0, jumps to s_1 to service the vector, and finally jumps to s_2. So the cost to B for servicing v is given by Cost$(B, v) = d_{s_0,s_1} + \delta_{s_1} + d_{s_1,s_2}$. Thus,

$$(1+\epsilon)\text{Cost}(A,V) \geq \text{Cost}(B,v).$$

■