BETTER ALGORITHMS FOR UNFAIR METRICAL TASK SYSTEMS AND APPLICATIONS*

AMOS FIAT† AND MANOR MENDEL‡

Abstract. Unfair metrical task systems are a generalization of online metrical task systems. In this paper we introduce new techniques to combine algorithms for unfair metrical task systems and apply these techniques to obtain the following results:

1. Better randomized algorithms for unfair metrical task systems on the uniform metric space.
2. A randomized algorithm for metrical task systems on general metric spaces, $O((\log n \log\log n)^2)$ competitive, improving on the best previous result of $O(\log^5 n \log\log n)$.
3. A tight randomized competitive ratio for the k-weighted caching problem on $k+1$ points, $O(\log k)$, improving on the best previous result of $O(\log^2 k)$.
4. An $O(\log^2 n)$ competitive randomized algorithm for metrical task systems on n equally spaced points on the line.

Key words. online algorithms, randomized algorithms

AMS subject classifications. 68W20, 68W25, 68W40

1. Introduction.

Metrical task systems, introduced by Borodin, Linial, and Saks [10], can be described as follows: A server in some internal state receives *tasks* that have a service cost associated with each of the internal states. The server may switch states, paying a cost given by a metric space defined on the state space, and then pays the service cost associated with the new state.

Metrical task systems have been the subject of a great deal of study. A large part of the research into online algorithms can be viewed as a study of some particular metrical task system. In modeling some of these problems as metrical task systems, the set of permissible tasks is constrained to fit the particulars of the problem. In this paper we consider the original definition of metrical task systems where the set of tasks can be arbitrary.

A deterministic algorithm for any n-state metrical task system with a competitive ratio of $2n-1$ is given in the original paper [10], along with a matching lower bound for any metric space.

The *randomized* competitive ratio for the MTS problem is not as well understood. For the uniform metric space, where all distances are equal, the randomized competitive ratio is known to within a constant factor, and is $\Theta(\log n)$ [10, 14]. In fact, it has been conjectured that the randomized competitive ratio for MTS is $\Theta(\log n)$ in any n-point metric space. Previously, the best upper bound on the competitive ratio for arbitrary metric space was $O(\log^5 n \log\log n)$ due Bartal, Blum, Burch and Tomkins [4] and Bartal [3]. The best lower for any metric space is $\Omega(\log n / \log^2 \log n)$ due to Bartal, Bollobás and Mendel [5], improving previous lower bounds of Karloff, Rabani and Ravid [16], and Blum, Karloff, Rabani, and Saks [9].

An observation made in [16, 9, 2] that played a fundamental role in previous results for this problem is that the randomized competitive ratio for MTS is conceptually easier to analyze on "decomposable spaces", spaces that are composed of subspaces with small diameter compared to that of the entire space. Bartal [2] introduced a class

*This work was partly supported by United States Israel Bi-national Science Foundation Grant 96-00247/1. Preliminary version appeared in 32nd Annual ACM Symposium on Theory of Computing, 2000

†School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel (fiat@tau.ac.il).

‡School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel (mendelma@tau.ac.il).

of decomposable spaces called *hierarchically well-separated trees* (HST). A k-HST is a metric space defined on the leaves of a tree such that, for each level of the tree, the diameters of the subtrees decrease by a factor of k between the levels. Consider a particular level of an HST. The distances to all subtrees are approximately the same and thus it is natural to use a recursive solution for the HST where the problem at a particular level is essentially on a uniform space.

If we are interested in the competitive ratio for a specific metric space M, then it is useful to consider how close it is to a simpler metric space N (such as an HST). We say that N α-approximates M if the distances in N are within a factor α from those in M. Clearly, if there is a r-competitive algorithm for N then there is αr-competitive algorithm for M. This notion can be generalized to a probabilistic metric approximation [2] by considering a set of approximating metric spaces that dominate the original metric space and bounding the expectation of the distances. In [2, 3] it is proved that any metric space on n points can be $O(\log n \log\log n)$-*probabilistically approximated* by an HST, thus reducing the problem of devising algorithm for MTS on any metric space to devising an algorithm for HSTs only [4].

Our upper bound for HST spaces follows the general framework originated in [9] and explicitly formulated in [20, 4], where the recursive structure of the HST is modeled by defining an *unfair metrical task system* problem [20, 4] on an uniform metric space. In an unfair MTS problem, to every point v_i of the metric space we associate a cost ratio r_i which multiplies the online costs for processing tasks in that point. Offline costs remain as before. The cost ratio r_i roughly corresponds to the competitive ratio of the online algorithm in a subspace of the HST. For UMTSs on uniform metric spaces, tight upper bounds are only known for two point spaces [9, 20, 4] and for n point spaces with equal cost ratios [4]. A tight lower bound is known for any number of points and any cost ratios [5].

We also consider a variant of the paging problem. The basic paging problem is the online problem of deciding what page to evict upon a page fault. The performance measure is the number of page faults. This problem has tight deterministic and randomized bounds on the competitive ratio of k and $\ln k + O(1)$, where k is the number of page slots in memory [22, 13, 18, 1]. Weighted caching is the paging problem when there are different costs to fetch different pages. Deterministically, a competitive ratio of k is achievable [11, 23], with a matching lower bound following from the k-server bound in [17]. No randomized algorithm is known to have a competitive ratio better than the deterministic ratio of k. However, in some special cases progress has been made. Irani [private communication] has shown an $O(\log k)$ competitive algorithm when page fetch costs are one of two possible values. Blum, Furst, and Tomkins [8] have given an $O(\log^2 k)$ competitive algorithm for arbitrary page costs, when the total number of pages is $k+1$, they also give a lower bound of $\Omega(\log k)$ for any page costs.

Our contributions. In this paper we introduce a general notation and technique for combining algorithms for unfair metrical task systems on hierarchically decomposable metric spaces. This technique is an improvement on the previous methods [9, 20, 4] and we believe that it is of independent interest beyond the applications we have given in this paper. We also believe that using some notation similar to ours is imperative to avoid possible confusion in combining algorithms from different spaces. Using this technique, we obtain randomized algorithms for unfair metrical task systems on the uniform metric space that are better than the algorithm of [4]. Using the algorithm for unfair metrical task systems on uniform metric space and the new method for combining algorithms, we obtain $O(\log n \log\log n)$ competitive algorithms for MTS on HST spaces, which implies $O((\log n \log\log n)^2)$-competitive randomized algorithm for metrical task systems on any metric space. As the weighted caching with cache size k on $k+1$ pages is a special case of MTS on star-like metric spaces, we are able to obtain an $O(\log k)$ competitive algorithm. This is tight up to a constant factor. We also consider the MTS on n equally spaced points on the line, and show an $O(\log^2 n)$ upper bound on the competitive ratio for this problem. This is a slight improvement over the general bound.

Overview of the paper. In Section 2 the MTS problem is formally defined, along with several technical conditions, that allows us later to combine algorithms for subspaces together. In Section 3 we deal with the main technical contribution of our paper. We introduce a novel technique to combine algorithms for subspace into an algorithm for the entire space. The technique is summarized in Theorem 3.1. In Section 4 we consider an unfair metrical task system where the underling metric space is the uniform metric space. We make use of previous algorithms for two points and for equal cost ratios and using Theorem 3.1, we obtain new algorithms that for some sets of costs ratio their competitive ratios are better than previously known, and we make use of these later. In Section 5 we give the applications mentioned above by combining the algorithms of Section 4 using Theorem 3.1.

2. Preliminaries. *Unfair metrical task systems* (UMTSs) [20, 4] are a generalization of metrical task systems [10]. An UMTS $U = (M; (r_u)_{u\in M}; s)$ consists of a metric space M on b states with a distance metric d_M, a sequence of *cost ratios* $r_u \in \mathbb{R}^+$ for $u \in M$, and a *distance ratio* $s \in \mathbb{R}^+$.

Given some UMTS U, the associated online problem is defined as follows. An online algorithm A occupies some state $u \in M$. When a task arrives the algorithm may change state to v in order to minimize costs. A task is a vector $(c_x)_{x\in M}$, and the cost for algorithm A associated with servicing the task is $s \cdot \mathrm{d}_M(u,v) + r_v c_v$. The cost for A associated with servicing a sequence of tasks σ is simply the sum of costs for servicing the individual tasks of the sequence consecutively and is denoted by $\mathrm{cost}_A(\sigma)$. An online algorithm makes its decisions based only upon tasks seen so far.

An off-line player is defined that services the same sequence of tasks over U. The cost of an off-line player, if it were to do exactly as above, would be $\mathrm{d}_M(u,v) + c_v$. Thus, the concept of *unfairness*, the costs for doing the same thing are different.

Given a sequence of tasks σ we define the *work function* [12] at v, $w_{\sigma,U}(v)$, to be the minimal cost, for any off-line player, to start at the initial state in U, deal with all tasks in σ, and end up in state v. We omit the use of the subscript U if the UMTS is clear from the context. We note that for all $u, v \in M$, $w_\sigma(u) - w_\sigma(v) \le \mathrm{d}_M(u,v)$. If $w_\sigma(u) = w_\sigma(v) + \mathrm{d}_M(u,v)$, u is said to be *supported* by v. We say that $u \in M$ is *supported* if there exists some $v \in M$ such that u is supported by v.

We define $\mathrm{cost}_{\mathrm{OPT}}(\sigma)$ to be $\min_v w_\sigma(v)$. This is simply the minimal cost, for any off-line player, to start at the initial state and process σ. As the differences between the work function values on different states is bounded by a constant (the diameter of the metric space) independent of the task sequence, it is possible to use in convex combination of the work function values instead of the minimal one. Let $\alpha = (\alpha(u))_{u\in M}$ be a *weight vector* of real non-negative numbers such that $\sum_{u\in M} \alpha(u) = 1$. We define the α-optimal-cost of a sequence of tasks σ to be $\mathrm{cost}_{\alpha\text{-OPT}}(\sigma) = \langle \alpha, w_\sigma \rangle = \sum_{u\in M} \alpha(u) w_\sigma(u)$. As observed above, $\mathrm{cost}_{\alpha\text{-OPT}}(\sigma) \le \mathrm{cost}_{\mathrm{OPT}}(\sigma) + \Delta(M)$, where $\Delta(M) = \max_{u,v\in M} \mathrm{d}_M(u,v)$ is the diameter of M.

A randomized online algorithm A for an UMTS is an online algorithm that decides upon the next state using a random process. The expected cost of a randomized algorithm A on a sequence σ is denoted by $E[\mathrm{cost}_A(\sigma)]$.

DEFINITION 2.1. *[22, 15, 6] A randomized online algorithm is called r competitive against an oblivious adversary if there exists c such that for all task sequences σ, $E[\mathrm{cost}_A(\sigma)] \leq r\ \mathrm{cost}_{OPT}(\sigma) + c$.*

OBSERVATION 1. *We can limit the discussion to distance ratio equals one since an UMTS $U = (M; (r_u)_{u\in M}; s)$ has a competitive ratio of r if and only if $U' = (M; (s^{-1}r_u)_{u\in M}; 1)$ has competitive ratio of rs^{-1}. Moreover an rs^{-1} competitive algorithm for U' is r competitive algorithm for U, since in both U' and U the offline costs are the same but the online costs in U are multiplied by a factor of s compared to the costs in U'. When $s = 1$, we drop it from the notation.*

Given a randomized online algorithm A for an UMTS U with state space M and a sequence of tasks σ, we define $p_{\sigma,A}$ to be the vector of probabilities $(p_{\sigma,A}(u))_{u\in M}$ where $p_{\sigma,A}(u)$ is the probability that A is in state u after serving the request sequence σ. We drop the subscript A if the algorithm is clear from the context.

Let $x \circ y$ denote the concatenation of sequences x and y. Let U be a UMTS over the metric space M with distance ratio s. Given two successive probability distributions on the states of U, p_σ and $p_{\sigma\circ e}$, where e is the next task, we define the set of transfer matrices from p_σ to $p_{\sigma\circ e}$, denoted $T(p_\sigma, p_{\sigma\circ e})$, as the set of all matrices $T = (t_{uv})_{u,v\in M}$ with non negative real entries, where

$$\sum_{v\in M} t_{uv} = p_\sigma(u),\ u \in M; \qquad \sum_{u\in M} t_{uv} = p_{\sigma\circ e}(v),\ u \in M.$$

We define the *unweighted moving cost* from p_σ to $p_{\sigma\circ e}$:

$$\mathrm{mcost}_M(p_\sigma, p_{\sigma\circ e}) = \min_{\substack{(t_{uv})\in \\ T(p_\sigma, p_{\sigma\circ e})}} \sum_{u,v} t_{uv}\, \mathrm{d}_M(u,v),$$

the *moving cost* is defined as $\mathrm{mcost}_U(p_\sigma, p_{\sigma\circ e}) = s \cdot \mathrm{mcost}_M(p_\sigma,\ p_{\sigma\circ e})$, and the *local cost* on a task $e = (c_u)_{u\in M}$ is defined as $\sum_{u\in M} p_{\sigma\circ e}(u)c_u r_u$. Due to linearity of expectation, $E[\mathrm{cost}_A(\sigma\circ e)] - E[\mathrm{cost}_A(\sigma)]$ is equal to the sum of moving costs from p_σ to $p_{\sigma\circ e}$ plus the local cost on e. Hence we can view A as a deterministic algorithm that maintains the probability mass on the states whose cost on task e given after sequence σ is

$$\mathrm{cost}_A(\sigma\circ e) - \mathrm{cost}_A(\sigma) = \mathrm{mcost}_U(p_\sigma, p_{\sigma\circ e}) + \sum_{u\in M} p_{\sigma\circ e}(u)c_u r_u. \tag{2.1}$$

In the sequel we will use the terminology of changing probabilities, with the understanding that we are referring to a deterministic algorithm charged according to Equation (2.1).

Elementary tasks are task vectors with only one non-zero entry, we use the notation (v,δ), $\delta \geq 0$, for an elementary task of cost δ at state v. Tasks $(v, 0)$ can simply be ignored by the algorithm.

DEFINITION 2.2 ([4]). *A* reasonable *algorithm is an online algorithm that never assigns a positive probability to a supported state.*

DEFINITION 2.3 ([4]). *A* reasonable task sequence *for algorithm A, is a sequence of tasks that obeys the following:*

1. All tasks are elementary.

2. *For all σ, the next task (v,δ) must obey that for all δ', if $\delta > \delta' \geq 0$ then $p_{\sigma\circ(v,\delta')}(v) > 0$.*

It follows that a reasonable task sequence for A never includes tasks (v,δ), $\delta > 0$, if the current probability of A on v is zero.

The following lemma is from [4].

LEMMA 2.4. *Given a randomized online algorithm A that obtains a competitive ratio of r when the task sequences are limited to being reasonable task sequences for A, then, for all $\varepsilon > 0$, there also exists a randomized algorithm A' that obtains a competitive ratio of $r+\varepsilon$ on all possible sequences.*

Proof. For the sake of completeness we sketch a proof here. The proof proceeds in three stages. In the first stage, we convert an algorithm A_0 for reasonable task sequences to a *lazy* algorithm A_1 (an algorithm that dose not move the server when receiving a task with zero cost) for reasonable task sequences. In the second stage, we convert an algorithm A_1 to an algorithm A_2 for elementary task sequences, and then, in the third stage, we convert A_2 to an algorithm A_3 for general task sequences.

The first stage is well known.

The second stage. Given an elementary task sequence, every elementary task $e = (v,x)$ is converted to a task (v,y) such that $y = \sup\{z | z < x$ and the probability induced by A_0 on v is greater than $0\}$. The resulting task sequence is reasonable and is fed to A_1. A_2 imitates the movements of A_1.

The third stage. Let σ be an arbitrary task sequence. First, we convert σ into an elementary task sequence $\hat{\sigma}$, each task $\tau = (\delta_1,\ldots,\delta_n)$ in σ is converted to a sequence of tasks $\hat{\sigma}_\tau$ as follows: Let $\varepsilon' > 0$ be small constant to be determined later, and assume for simplicity that $\delta_i \geq \delta_{i+1}$. Then $\hat{\sigma}_\tau = \varsigma_1 \circ \varsigma_2 \circ \cdots \varsigma_N$, where $N = \lfloor \delta_1/\varepsilon' \rfloor$ and $\varsigma_j = (v_1,\varepsilon') \circ (v_2,\varepsilon') \circ \cdots \circ (v_{k_j},\varepsilon')$, where $k_j = \max\{i | \delta_i \geq j \cdot \varepsilon'\}$. Note that the optimal offline cost $\hat{\sigma}$ is at most the optimal offline cost on σ, since any servicing for σ, when applied to $\hat{\sigma}$ would have a cost no bigger than the original cost. Consider an r-competitive online algorithm A for elementary tasks operating on $\hat{\sigma}$, and construct an online algorithm B for σ. B maintains the invariant that the state of B after processing some task τ is the same state as A after processing the sequence $\hat{\sigma}_\tau$. Consider the behavior of A on $\hat{\sigma}_\tau$. It begins in some state v_{i_0}, passes through some set S of states and ends up in some state v_{i_2}. Consider the original task $\tau = (\delta_1,\ldots,\delta_n)$. Let v_{i_1} be the state in S with the lowest cost in τ. Algorithm B begins in state v_{i_0}, immediately moves to v_{i_1}, serves τ in v_{i_1} and then moves to v_{i_2}.

Informally, on each task A pays either a local cost of ε' or moving cost of at least ε' and therefore these costs are larger than the local cost of B. B also has a moving cost at least as A. By careful combination of these two we can conclude that the cost of B on σ is at most $(1+\varepsilon)$ times the cost of A on $\hat{\sigma}$. □

Hereafter, we assume only reasonable task sequences. This is without lost of generality due to Lemma 2.4.

OBSERVATION 2. *For a reasonable algorithm A competing against applied on reasonable task sequence $\sigma = \tau_1\tau_2\cdots\tau_m$, any elementary task $\tau = (v,\delta)$ causes $w(v)$ to increase by δ. This follows because v would not have been supported following any alternative request (v,δ'), $\delta' < \delta$. See [4, Lemma 1]. This also implies that for any state v, $w_\sigma(v) = \sum_{j=1}^{m} \tau_j(v)$.*

DEFINITION 2.5. *An online algorithm A is said to be* sensible *and r-competitive on UMTS $U = (M; (r_u)_{u\in M}; s)$ if it obeys the following:*

1. *A is reasonable.*
2. *A is a* stable algorithm *[12],* i.e., *the probabilities that A assigns to the dif-*

ferent states are purely a function of the work function.

3. *Associated with A are a weight vector α_A and a potential function Φ_A such that*
   - $\Phi_A : \mathbb{R}^b \mapsto \mathbb{R}^+$, *is purely a function of the work-function, bounded, non-negative, and continuous.*
   - *For all task sequences σ and all tasks e,*

$$\mathrm{cost}_A(\sigma\circ e)-\mathrm{cost}_A(\sigma)+\Phi_A(w_{\sigma\circ e})-\Phi_A(w_\sigma)\le r\cdot\langle\alpha_A, w_{\sigma\circ e}-w_\sigma\rangle. \quad (2.2)$$

OBSERVATION 3. *An online algorithm that is sensible and r-competitive according to Def. 2.5 is also r-competitive according to Def. 2.1. This is so since summing up the two sides in Inequality* (2.2) *over the individual tasks in the task sequence, we get a telescopic sum such that* $\mathrm{cost}_A(\sigma)+\Phi_A(w_\sigma)-\Phi_A(w_\varepsilon)\le r\cdot\langle\alpha_A, w_\sigma - w_\varepsilon\rangle$, *where w_ε is the initial work function. We conclude that* $\mathrm{cost}_A(\sigma)\le r\cdot\mathrm{cost}_{OPT}(\sigma)+r\Delta(M)+\sup_w\Phi(w)$.

When combining sensible algorithms we would like the resulting algorithm to be also sensible. The problematic invariant to maintain is reasonableness. To maintain it there is a need for a "stronger" concept than reasonableness, which we call *constrained algorithms.*

DEFINITION 2.6. *A sensible r-competitive algorithm A for UMTS* $U=(M;(r_u)_{u\in M};s)$ *with associated potential function Φ is called (β,η)-constrained,* $0\le\beta\le 1$, $0\le\eta$, *if the following hold:*

1. *For all $u,v\in M$: if $w(u)-w(v)\ge\beta\,\mathrm{d}_M(u,v)$ then the probability that A assigns to u is zero ($p_{w,A}(u)=0$).*
2. $|\Phi|\le\eta\,\Delta(M)r$, *where $|\Phi|$ denotes* $\sup_w\Phi(w)$.

OBSERVATION 4.

1. *For a (β,η)-constrained algorithm competing against a reasonable task sequence, $\forall u,v\in M$, $|w(u)-w(v)|\le\beta\;\mathrm{d}_M(u,v)$. The argument here is similar to the one given in Observation 2.*
2. *A sensible r-competitive algorithm A for a space of diameter Δ is by definition $(1,|\Phi_A|/(r\Delta))$-constrained.*
3. *If A is (β,η)-constrained then it is trivially (β',η')-constrained for all $\beta\le\beta'\le 1$ and $\eta\le\eta'$.*

3. A Combining Lemma for Unfair Metrical Task Systems. We are given a metric space M, composed of sub-spaces $M_1,\ldots,M_b$, with "large" distances between sub-spaces compared to the diameters of the sub-spaces. A metrical task system on M induces a metrical task system on M_i. Assume that for every i, we have a $\hat{r}_i$-competitive algorithm A_i for the induced MTS on M_i. Our goal is to combine the A_i algorithms so as to obtain an algorithm for the original MTS defined on M. To do so we make use of a "combining algorithm" $\hat{A}$. $\hat{A}$ has the role of determining which of the M_i sub-spaces contains the server. Since the "local cost" of $\hat{A}$ on sub-space M_i is $\hat{r}_i$ times the optimal cost on subspace M_i, it is natural that $\hat{A}$ should be an algorithm for the UMTS $\hat{U}=(\hat{M};(\hat{r}_1,\ldots,\hat{r}_b))$, where $\hat{M}=\{z_1,\ldots,z_b\}$ is a space with points corresponding to the sub-spaces and distances that are roughly the distances between the corresponding sub-spaces. Tasks for the original MTS are translated to tasks for the M_i induced metrical task systems simply by restriction. It remains to define how one translates tasks for the original MTS to tasks for $\hat{U}$.

Previous papers [9, 20, 4] use the average cost of the task on the sub-space M_i as the cost for z_i in the task for $\hat{U}$. This way the cost for the online algorithm is $\hat{r}_i$ times the cost for the optimum (we assume reasonableness of the algorithm and the task sequence), however, *this is true only in the amortized sense.* In order to bound the fluctuation around the amortized cost, they have to assume that the diameters of the sub-space are very small compared with the distances between M_i sub-spaces. We take a different approach: the cost for a point $z_i \in \hat{U}$ is (an upper bound for) *the cost of A_i on the corresponding task, divided by $\hat{r}_i$.* In this way the amortization problem disappears, and we are able to combine sub-spaces with a relatively large diameter. A formal description of the construction is given in Theorem 3.1 and its proof.

THEOREM 3.1. *Let U be an UMTS $U = (M; (r_u)_{u\in M}; s)$, where M is a metric space on n points. Consider an arbitrary partition of the points of M, $P = (M_1, M_2, \ldots, M_b)$. $U_j = (M_j; (r_u)_{u\in M_j}; s)$ is the UMTS induced by U on the subspace M_j. Let $\hat{M}$ be a metric space defined over the set of points $\{z_1, z_2, \ldots, z_b\}$ with a distance metric $\mathrm{d}_{\hat{M}}(z_i, z_j) \geq \max\{\mathrm{d}_M(u, v) : u \in M_i, v \in M_j\}$. Assume that*

- *For all j, there is a (β_j, η_j)-constrained $\hat{r}_j$-competitive algorithm A_j for the UMTS U_j.*
- *There is a $(\hat{\beta}, \hat{\eta})$-constrained r-competitive algorithm $\hat{A}$ for the UMTS $\hat{U} = (\hat{M}; (\hat{r}_1, \ldots, \hat{r}_b); s)$.*

Define

$$\beta = \max\Big\{\max_i \beta_i,\ \max_{i\neq j} \frac{\hat{\beta}\, \mathrm{d}_{\hat{M}}(z_i, z_j) + \beta_j\, \Delta(M_j) + \beta_i\, \Delta(M_i) + \eta_i \Delta(M_i)}{\min_{p\in M_i, q\in M_j} \mathrm{d}_M(p, q)}\Big\}, \tag{3.1}$$

and

$$\eta = \hat{\eta}\frac{\Delta(\hat{M})}{\Delta(M)} + \max_i\ \eta_i \frac{\Delta(M_i)}{\Delta(M)}. \tag{3.2}$$

If $\beta \leq 1$, then there exists a (β, η)-constrained and r-competitive algorithm, A, for the UMTS U, against reasonable task sequences.

3.1. The Construction of the Combined Algorithm. For $1 \leq j \leq b$, denote by Φ_j and α_j the associated potential function and weight vector of algorithm A_j, respectively. Similarly, denote by $\hat{\Phi}$ and $\hat{\alpha}$ the associated potential function and weight vector of algorithm $\hat{A}$, respectively.

Given a sequence of elementary tasks $\sigma = (v_1, \delta_1) \circ (v_2, \delta_2) \circ \cdots \circ (v_{|\sigma|}, \delta_{|\sigma|})$, $v_i \in M$, we define the sequences

$$\sigma|_{M_\ell} = (u_1^\ell, \delta_1^\ell) \circ (u_2^\ell, \delta_2^\ell) \circ \cdots \circ (u_{|\sigma|}^\ell, \delta_{|\sigma|}^\ell), \text{ where}$$

- $u_j^\ell = v_j$ and $\delta_j^\ell = \delta_j$, if $v_j \in M_\ell$.
- u_j^ℓ is an arbitrary point in M_ℓ and $\delta_j^\ell = 0$, if $v_j \notin M_\ell$.

Informally, $\sigma|_{M_\ell}$ is the restriction of σ to subspace M_ℓ.
For $u \in M$, define $s(u) = i$ iff $u \in M_i$. We define the sequence

$$\chi(\sigma) = (z_{s(v_1)}, \hat{\delta}_1) \circ (z_{s(v_2)}, \hat{\delta}_2) \circ \cdots \circ (z_{s(v_{|\sigma|})}, \hat{\delta}_{|\sigma|}),$$

inductively. Let $e = (v, \delta)$, $s(v) = \ell$, then $\chi(\sigma \circ e) = \chi(\sigma) \circ (z_\ell, \hat{\delta})$ where

$$\hat{\delta} = \big(\langle \alpha_\ell, w_{\sigma\circ e|_{M_\ell}, U_\ell}\rangle - \Phi_\ell(w_{\sigma\circ e|_{M_\ell}, U_\ell})/\hat{r}_\ell\big) - \big(\langle \alpha_\ell, w_{\sigma|_{M_\ell}, U_\ell}\rangle - \Phi_\ell(w_{\sigma|_{M_\ell}, U_\ell})/\hat{r}_\ell\big) \tag{3.3}$$

Note that $\hat{\delta}$ is an upper bound on the cost of A_ℓ for the task (v, δ), divided by $\hat{r}_\ell$. This is so since A_ℓ is sensible, and $\chi(\sigma)$ is a reasonable task sequence (see

Proposition 3.2); and it follows from inequality (2.2). This also implies that the task $(z_\ell, \hat{\delta})$ is well defined, *i.e.*, $\hat{\delta} \geq 0$.

ALGORITHM A. The algorithm works as follows:

1. It simulates algorithm A_ℓ on the task sequence $\sigma|_{M_\ell}$, for $1 \leq \ell \leq b$.
2. It also simulates algorithm $\hat{A}$ on the task sequence $\chi(\sigma)$.
3. The probability assigned to a point $v \in M_\ell$ is the product of the probability assigned by A_ℓ to v and the probability assigned by $\hat{A}$ to z_ℓ. (*i.e.*, $p_{\sigma,A}(v) = p_{\sigma|_{M_\ell},A_\ell}(v) \cdot p_{\chi(\sigma),\hat{A}}(z_\ell)$.)

We remark that the simulations above can be performed in an online fashion.

3.2. Proof of Theorem 3.1. To simplify notation and without loss of generality we consider an arbitrary sequence σ and arbitrary task e. With respect to σ we define

$$\begin{aligned} w &= w_{\sigma,U}; & w^e &= w_{\sigma\circ e,U}; \\ w_k &= w_{\sigma|_{M_k},U_k},\ 1 \leq k \leq b; & w^e_k &= w_{\sigma\circ e|_{M_k},U_k},\ 1 \leq k \leq b; \\ \hat{w} &= w_{\chi(\sigma),\hat{U}}; & \hat{w}^e &= w_{\chi(\sigma\circ e),\hat{U}}. \end{aligned}$$

Define p, p_k, and $\hat{p}$ to be the probability distributions on the states of U, U_k and $\hat{U}$ as induced by algorithms A, A_k and $\hat{A}$ on the sequences σ, $\sigma|_{M_k}$, and $\chi(\sigma)$, $1 \leq k \leq b$. Likewise, we define p^e, p^e_k and $\hat{p}^e$ where the sequences are $\sigma \circ e$, $\sigma \circ e|_{M_k}$, and $\chi(\sigma \circ e)$.

PROPOSITION 3.2. *If the task sequence given to algorithm A on U is reasonable, then the simulated task sequences for algorithms A_i on U_i and the simulated task sequence for algorithm $\hat{A}$ on $\hat{U}$ are also reasonable.*

Proof. If the adversary issues a task $e = (v, \delta)$ for A, it implies that had we replaced task e by any other task $e' = (v, \delta')$ where $0 \leq \delta' < \delta$, then $p^{e'}(v)$ would have been strictly greater than zero. This implies that $p^{e'}_\ell(v)$ would also have been strictly greater than zero, since $p^{e'}(v) = p^{e'}_\ell(v)\hat{p}^{e'}(z_\ell)$ (where $v \in M_\ell$).

To see that $\chi(\sigma)$ is a reasonable task sequence for $\hat{A}$, let $\hat{e} = (z_\ell, \hat{\delta})$ the last task in $\chi(\sigma)$, and let $e = (v, \delta)$ be the corresponding task in σ where $v \in M_\ell$. Consider a hypothetical task (v, x) in U, for $0 \leq x \leq \delta$. Denote by $(z_\ell, f(x))$ the corresponding task for $\hat{U}$, where $f(x)$ is determined according to Eq. (3.3). f is continuous (as Φ_ℓ is continuous), $f(0) = 0$, and $f(\delta) = \hat{\delta}$. Therefore for any $0 \leq \hat{\delta}' < \hat{\delta}$ there exists $0 \leq \delta' < \delta$ such that $f(\delta') = \hat{\delta}'$ and since $0 < p^{(v,\delta')} = p^{(v,\delta')}_\ell(v) \cdot \hat{p}^{(v,\delta')}(z_\ell)$ we conclude that $0 < \hat{p}^{(v,\delta')}(z_\ell)$ (the probability induced by $\hat{A}$ on z_ℓ after the task $(z_\ell, \hat{\delta}')$). □

PROPOSITION 3.3. *For all σ and for all ℓ, $\hat{w}(z_\ell) = \langle \alpha_\ell, w_\ell \rangle - \Phi_\ell(w_\ell)/\hat{r}_\ell$.*

Proof. From Proposition 3.2 we know that the task sequence $\chi(\sigma)$ for $\hat{A}$ is reasonable. As $\hat{A}$ is sensible and from Observation 2 we know that $\hat{w}(z_\ell)$ is exactly the sum of costs in $\chi(\sigma)$ for z_ℓ. By the definition of $\chi(\sigma)$ in Equation (3.3) it follows that this sum is $\langle \alpha_\ell, w_\ell \rangle - \Phi_\ell(w_\ell)/\hat{r}_\ell$. □

PROPOSITION 3.4. *If for all $1 \leq \ell \leq b$, $u \in M_\ell$, it holds that $w(u) = w_\ell(u)$ then any state $u \in U$, for which there exists a state v such that $w(u) - w(v) \geq \beta\, \mathrm{d}_M(u, v)$, has $p(u) = 0$.*

Proof. Consider states u and v as above, *i.e.*, $w(u) - w(v) \geq \beta\, \mathrm{d}_M(u, v)$. We now consider two cases:

1. $u \in M_i$ and $v \in M_i$. We want to show that $w_i(u) - w_i(v) \geq \beta_i\, \mathrm{d}_{M_i}(u, v)$, as A_i is (β_i, η_i)-constrained this implies that $p_i(u) = 0$, which implies that $p(u) = 0$. From the conditions above we get

$$w_i(u) - w_i(v) = w(u) - w(v) \geq \beta\, \mathrm{d}_M(u, v) \geq \beta_i\, \mathrm{d}_{M_i}(u, v).$$

2. $u \in M_i$, $v \in M_j$, $i \neq j$. Our goal now will be to show that $\hat{w}(z_i) - \hat{w}(z_j) \geq \hat{\beta}\, \mathrm{d}_{\hat{M}}(z_i, z_j)$, as this implies that $\hat{p}(z_i) = 0$ which implies that $p(u) = 0$.
A lower bound on $\hat{w}(z_i)$ is

$$\begin{aligned} \hat{w}(z_i) &= \langle \alpha_i, w_i \rangle - |\Phi_i|/r_i && (3.4)\\ &\geq w_i(u) - \beta_i \Delta(M_i) - |\Phi_i|/r_i && (3.5)\\ &= w(u) - \beta_i \Delta(M_i) - \eta_i \Delta(M_i). && (3.6) \end{aligned}$$

To justify (3.4) one uses the definitions and Proposition 3.3, (3.5) follows because a convex combination of values is at least an arbitrary value minus the maximal difference. The maximal difference between work function values is bounded by β_i times the distance, see observation 4. The last equation, (3.6), follows from our assumption that the work functions are equal and from the definition of η_i.
Similarly, to obtain an upper bound on $\hat{w}(z_j)$, we derive

$$\hat{w}(z_j) = \langle \alpha_j, w_j \rangle - |\Phi_j|/r_j \leq w(v) + \beta_j \Delta(M_j). \quad (3.7)$$

It now follows from Inequalities (3.6) and (3.7) that,

$$\begin{aligned} \hat{w}(z_i) - \hat{w}(z_j) &\geq (w(u) - w(v)) - \beta_i \Delta(M_i) - \beta_j \Delta(M_j) - \eta_i \Delta M_i \\ &\geq \beta d_M(u,v) - \beta_i \Delta(M_i) - \beta_j \Delta(M_j) - \eta_i \Delta M_i \geq \hat{\beta}\, \mathrm{d}_{\hat{M}}(z_i, z_j). \end{aligned}$$

The last inequality follows from Eq. (3.1).

□

PROPOSITION 3.5. *For any reasonable task sequence σ, subspace M_ℓ, and $v \in M_\ell$: $w_\ell(v) = w(v)$.*

Proof. The proof is by a contradiction. Let σ' be the shortest reasonable task sequence for which there exists v such that $w_{\sigma'|_{M_\ell}}(v) \neq w_{\sigma'}(v)$. It is easy to observe that $\sigma' = \sigma \circ e$ where $e = (v, \delta)$. As the sequence $\sigma \circ e|_{M_\ell}$ is a reasonable task sequence (from Proposition 3.2) and A_ℓ is reasonable, it follows that $w_\ell^e(v) = w_\ell(v) + \delta$. Since $w_\ell(v) = w(v)$ and $w^e(v) \leq w(v) + \delta$ we deduce that $w_\ell^e(v) > w^e(v)$.

Let $e_x = (v, x)$, define $\delta' = \sup\{x : w^{e_x}(v) = w_\ell^{e_x}(v)\}$. Obviously, $0 \leq \delta' \leq \delta$. Define $e' = (v, \delta')$. By continuity of the work function $w^{e'}(v) = w_\ell^{e'}(v)$ and thus $\delta' < \delta$. The conditions above imply that an elementary task in v after $w^{e'}$ will not change the work function, which means that v is supported in $w^{e'}$. Hence, the conditions of Proposition 3.4 are met (here we use the assumption that $\beta \leq 1$). By Proposition 3.4 $p^{e'}(v) = 0$ and as the sequence σ is reasonable for A it follows that $\delta \leq \delta'$. A contradiction. □

PROPOSITION 3.6. *For all σ, and all tasks $e = (v, \delta)$,*

$$\mathrm{cost}_A(\sigma \circ e) - \mathrm{cost}_A(\sigma) \leq \mathrm{cost}_{\hat{A}}(\chi(\sigma \circ e)) - \mathrm{cost}_{\hat{A}}(\chi(\sigma)).$$

Proof. Let us denote the subspace containing v by M_ℓ. We split the cost of A into two main components, the moving cost $\mathrm{mcost}_U(p, p^e)$, and the local cost $r_v p^e(v)\delta = r_v \hat{p}^e(z_\ell) p_\ell(v_i) \delta$ (see Equation (2.1)).

We give an upper bound on the moving cost of A by considering a possibly suboptimal algorithm that works as follows:

1. Move probabilities between the different M_j subspaces. *I.e.*, change the probability $p(u) = \hat{p}(z_j)p_j(u)$ for $u \in M_j$ to an intermediate stage $\hat{p}^e(z_j)p_j(v)$. The moving cost for A to produce this intermediate probability is bounded by $\mathrm{mcost}_{\hat{U}}(\hat{p}, \hat{p}^e)$ as the distances in $\hat{M}$ are an upper bound on the real distances for A ($\mathrm{d}_{\hat{M}}(z_i, z_j) \geq \mathrm{d}_M(u, v)$ for $u \in M_i$, $v \in M_j$). We call this cost the inter-space cost for A.
2. Move probabilities within the M_j subspaces. *I.e.*, move from the intermediate probability $\hat{p}^e(z_j)p_j(u)$, $u \in M_j$ to the probability $p^e(u) = \hat{p}^e(z_j)p^e_j(u)$. As all algorithms A_j, $j \neq \ell$, get a task of zero cost, $p^e_j = p_j$, $j \neq \ell$. The moving cost for A to produce $p^e(u)$, $u \in M_\ell$, from the intermediate stage , is no more than $\hat{p}^e(z_\ell) \cdot \mathrm{mcost}_{U_\ell}(p_\ell, p^e_\ell)$. We call this cost the intra-space cost for A.

Taking the local cost for A and the intra-space cost for A:

$$
\begin{aligned}
&r_u \hat{p}^e(z_\ell)p_\ell(u)\delta + \hat{p}^e(z_\ell) \cdot \mathrm{mcost}_{U_\ell}(p_\ell, p^e_\ell) \\
&\qquad = \hat{p}^e(z_\ell)\left(cost_{A_\ell}(\sigma \circ e) - \mathrm{cost}_{A_\ell}(\sigma)\right) && (3.8)\\
&\qquad \leq \hat{p}^e(z_\ell)\hat{r}_\ell\big((\langle \alpha_\ell, w^e_\ell\rangle - \Phi_\ell(w^e_\ell)/\hat{r}_\ell) - (\langle \alpha_\ell, w_\ell\rangle - \Phi_\ell(w_\ell)/\hat{r}_\ell)\big) && (3.9)
\end{aligned}
$$

To obtain (3.8) we use the definition of online cost (see Eq. (2.1)). To obtain (3.9) we use the fact that A_ℓ is $\hat{r}_\ell$ competitive and sensible (see Eq. (2.2)).

Let $\hat{e}$ be the last task in $\chi(\sigma \circ e)$. Formula (3.9) is simply the local cost for algorithm $\hat{A}$ on task $\hat{e}$. Thus, we have bounded the cost for algorithm A on task e to be no more than the cost for algorithm $\hat{A}$ on task $\hat{e}$. □

Proof of Theorem 3.1. We associate a weight vector α and a bounded potential function Φ with algorithm A, where

$$
\alpha(v) = \hat{\alpha}(z_\ell)\alpha_\ell(v) \quad \text{for } v \in M_\ell\,; \qquad \Phi(w) = \hat{\Phi}(\hat{w}) + r\sum_i \hat{\alpha}(z_i)\Phi_i(w_i)/\hat{r}_i.
$$

We remark that from Proposition 3.3 and Proposition 3.5 it follows that $\hat{w}$ and w_i are determined by w, so $\Phi(w)$ is well defined.

We derive the following upper bound on the cost of A:

$$
\begin{aligned}
&\mathrm{cost}_A(\sigma \circ e) - \mathrm{cost}_A(\sigma) \\
&\qquad \leq \mathrm{cost}_{\hat{A}}(\chi(\sigma \circ e)) - \mathrm{cost}_{\hat{A}}(\chi(\sigma)) && (3.10)\\
&\qquad \leq r\Big(\sum_i \hat{\alpha}(z_i)\hat{w}^e(z_i) - \sum_i \hat{\alpha}(z_i)\hat{w}(z_i)\Big) - \Big(\hat{\Phi}(\hat{w}^e) - \hat{\Phi}(\hat{w})\Big) && (3.11)\\
&\qquad = r\Big(\sum_i \sum_{v \in M_i} \hat{\alpha}(z_i)\alpha_i(v)w^e_i(v) - \sum_i \sum_{v \in M_i} \hat{\alpha}(z_i)\alpha_i(v)w_i(v)\Big) \\
&\qquad\quad - \Big(\big(\hat{\Phi}(\hat{w}^e) + r\sum_i \hat{\alpha}(z_i)\Phi_i(w^e_i)/\hat{r}_i\big) - \big(\hat{\Phi}(\hat{w}) + r\sum_i \hat{\alpha}(z_i)\Phi_i(w_i)/\hat{r}_i\big)\Big) \\
&&& (3.12)\\
&\qquad = r(\langle \alpha, w^e\rangle - \langle \alpha, w\rangle) - (\Phi(w^e) - \Phi(w)). && (3.13)
\end{aligned}
$$

Inequality (3.10) follows from Proposition 3.6, inequality (3.11) is implied as $\hat{A}$ is a sensible r competitive algorithm. We obtain (3.12) by substituting $\hat{w}^e(z_i)$ and $\hat{w}(z_i)$ according to Proposition 3.3 and rearranging the summands. Equation (3.13) follows from the definition of α and Φ above.

We now prove that A is (β, η)-constrained. It follows from Proposition 3.4 and Proposition 3.5 that the condition on β is satisfied (see Definition 2.6). It remains to show the condition on η:

$$\begin{aligned}
|\Phi| &\leq |\hat{\Phi}| + r \sum_i \hat{\alpha}(z_i) |\Phi_i| / \hat{r}_i && (3.14)\\
&\leq \hat{\eta} r \cdot \Delta(\hat{M}) + r \sum_i \hat{\alpha}(z_i) \eta_i \hat{r}_i \cdot \Delta(M_i) / \hat{r}_i && (3.15)\\
&\leq r \cdot \Delta(M) \big(\hat{\eta} \tfrac{\Delta(\hat{M})}{\Delta(M)} + \max_i \{ \eta_i \tfrac{\Delta(M_i)}{\Delta(M)} \} \big)\\
&= r \cdot \Delta(M) \eta.
\end{aligned}$$

Inequality (3.14) follows by the definition of Φ, (3.15) follows because $\hat{A}$ is $(\hat{\beta}, \hat{\eta})$-constrained and A_i is (β_i, η_i)-constrained, $1 \leq i \leq b$.

We have therefore shown that A is a (β, η)-constrained and r-competitive algorithm. □

3.3. More Constrained Algorithms. Theorem 3.1 assumed constrained algorithms. In this section we show how to obtain such algorithms. The proof is motivated by similar ideas from [20, 4].

DEFINITION 3.7. *Fix a metric space M on b states and cost ratios $r_1, \ldots, r_b$. Assume that for all $s > 0$ there is a (β, η) constrained $f(s)$ competitive algorithm A_s for the UMTS $U_s = (M; (r_1, \ldots, r_b); s)$ against reasonable task sequences. For $\rho > 0$ we define the ρ-variant of A_s (if it exists) to be a $(\beta\rho, \eta\rho)$ constrained $f(s/\rho)$ competitive algorithm for U_s.*

PROPOSITION 3.8. *Let $0 < \beta \leq 1$ and $0 < \beta/\rho \leq 1$. Assume we are given a $(\beta/\rho, \eta/\rho)$-constrained, r-competitive online algorithm A', against reasonable task sequences on the UMTS $U' = (\rho M; (r_1, \ldots, r_b); s/\rho)$. This implies that there exists an online algorithm A which is (β, η)-constrained and r competitive, against reasonable task sequences on the UMTS $U = (M; (r_1, \ldots, r_b); s)$.*

Proof. Algorithm A on the UMTS U simulates algorithm A' on the UMTS U' by translating every task (v, δ) to task (v', δ). The probability that A associates with state v is the same as the probability that algorithm A' associates with state v'. If the task sequence for A' is reasonable then the simulated task sequence for A' is also reasonable simply because the probabilities for v and v' are identical.

The costs of A or A' on task (v, δ) or (v', δ) can be partitioned into moving costs and local costs. As the probability distributions are identical, the local costs for A and and A' are the same. The unweighted moving costs for A are $1/\rho$ the unweighted moving costs for A' because all distances are multiplied by $1/\rho$. However, the moving costs for A' are the unweighted moving costs multiplied by a factor of s/ρ whereas the moving costs for A are the unweighted moving costs multiplied by a factor of s. Thus, the moving costs are also identical.

To show that A is (β, η)-constrained (and hence reasonable) we first need to show that if the work functions in U and U' are equal, then this implies that if u and v are two states such that $w(u) \geq w(v) + \beta\, \mathrm{d}_M(u, v)$ then $p(u) = 0$. This is true because A' is $(\beta/\rho, \eta/\rho)$-constrained, and thus $w(u') \geq w(v') + (\beta/\rho) \cdot \mathrm{d}_{\rho M}(u', v')$ implies a probability of zero on u' for A' which implies a probability of zero on u for A. Next, one needs to show that the work functions are the same, this can be done using an argument similar to the proof of Proposition 3.5.

As the work functions and costs are the same for the online algorithms A and A' it follows that we can use the same potential function. To show that $|\Phi| \le \eta \cdot \Delta(M)$ we note that $|\Phi| \le (\eta/\rho)\Delta(\rho M)$. □

OBSERVATION 5. *Assume that there is a (β, η)-constrained, r-competitive algorithm A for a UMTS $U = (M; (r_1, \ldots, r_b); s)$ against a reasonable task sequences. Then, for all $\rho > 0$, a natural modification of A, A', is a (β, η)-constrained, r-competitive algorithm against reasonable task sequences on the UMTS $U' = (\rho M; (r_1, \ldots, r_b); s)$.*

LEMMA 3.9. *Under the assumptions of Definition 3.7, for all $\rho > 0$ such that $\beta\rho \le 1$, and for all $s > 0$, the ρ-variant of A_s exists.*

Proof. For all $\rho > 0$ such that $\beta\rho \le 1$:

1. By assumption, there is a (β, η)-constrained, $f(s/\rho)$-competitive algorithm for the UMTS $(M; r_1, \ldots, r_b; s/\rho)$.
2. It follows from Proposition 3.8 that there exists an online algorithm that is $(\rho\beta, \rho\eta)$-constrained, $f(s/\rho)$-competitive for the UMTS $(\rho^{-1}M; r_1, \ldots, r_b; s)$.
3. It now follows from Observation 5 that there exists a $(\rho\beta, \rho\eta)$-constrained, $f(s/\rho)$-competitive online algorithm for the UMTS $(M; r_1, \ldots, r_b; s)$. This means that the ρ variant of A_s exists.

□

4. The Uniform Metric Space. Let $\mathcal{U}_b^d$ denote the uniform metric space on b points where all pairwise distances are d.

We define algorithm ODDEXPONENT on UMTS $U = (\mathcal{U}_b^d; (r_1, \ldots, r_b); s)$, $b \ge 2$, and without loss of generality assume $r_1 = \max_i r_i$. ODDEXPONENT is from [4] . The following lemma applies our terminology to the results of [4].

LEMMA 4.1. ODDEXPONENT *is $(1, 1)$-constrained, and $(r_1 + 6s \ln b)$-competitive.*

Proof. Algorithm ODDEXPONENT, when servicing a reasonable task sequence, allocates for configuration v the probability $p(v) = \frac{1}{b} + \frac{1}{b}\sum_u \left(\frac{w(u)-w(v)}{d}\right)^t$. t is chosen to be an odd integer in the range $[\ln b, \ln b + 2)$.

Bartal *et. al.* [4] prove that ODDEXPONENT is sensible, $(r_1 + 6s \ln b)$-competitive and that the associated potential function $|\Phi_1| \le (r_1/(t+1) + s)d \le (1/\lceil \ln b \rceil)(r_1 + 6s \ln b)d$. This implies that ODDEXPONENT is $(1, 1/\lceil \ln b \rceil)$-constrained. □

We define algorithm TWOSTABLE on UMTS $U = (\mathcal{U}_2^d; (r_1, r_2); s)$. Algorithm TWOSTABLE is from [20] and [4]; the algorithm also appears implicitly in [9]. The following lemma applies our terminology to the results of [20, 4].

LEMMA 4.2. TWOSTABLE *is $(1, 4)$-constrained, and r competitive where*

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

Proof. TWOSTABLE works as follows: Let $y = w(v_1) - w(v_2)$, and $z = (r_1 - r_2)/s$. The probability on point v_1 is $p(v_1) = \left(e^z - e^{z(\frac{1}{2} + \frac{y}{2d})}\right)/\left(e^z - 1\right)$. TWOSTABLE is shown to be sensible and r competitive in [4, 20] and the potential function associated with TWOSTABLE, Φ_2, obeys $|\Phi_2| \le (2r_2 + s)d$.

It remains to show that $|\Phi_2| \le 4rd$. We use the fact that, in general, if $|z| \le 1/2$ then $1/2 \le z/(e^z - 1)$, and do a simple case analysis. If $\max\{r_1, r_2\} > \frac{1}{2}s$ then $|\Phi_2| \le (2r_2 + s)d \le (2r + 2r)d \le 4rd$. Else $|z| \le 1/2$, so $r = r_2 + \frac{z}{e^z - 1}s \ge r_2 + \frac{s}{2}$. Hence $|\Phi_2| \le 2rd$. □

To gain an insight about the competitive ratio of TWOSTABLE, we have the following proposition.

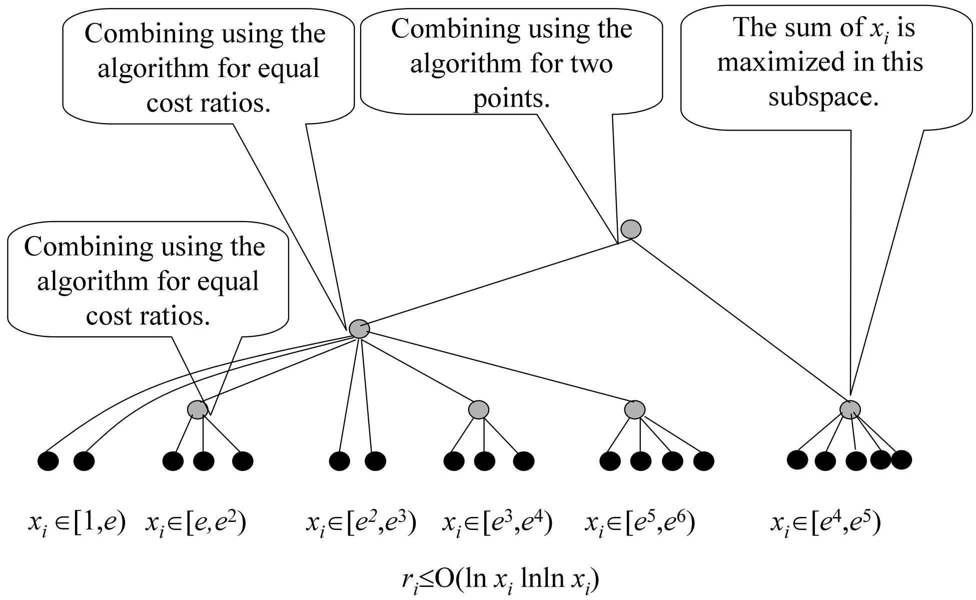

FIG. 4.1. *Schematic description of* COMBINED.

PROPOSITION 4.3. *Let* $f(s, r_1, r_2) = r_1 + (r_1 - r_2)/\left(e^{(r_1-r_2)/s} - 1\right)$. *Let* $x_1, x_2 \in \mathbb{R}^+$ *such that* $r_1 \leq 2s(\ln x_1 + 1)$ *and* $r_2 \leq 2s(\ln x_2 + 1)$. *Then* $f(s, r_1, r_2) \leq 2s(\ln(x_1 + x_2) + 1)$.

Proof. First we show that f is a monotonic non-decreasing function of both r_1 and r_2. Since the formula is symmetric in r_1 and r_2 it is enough to check monotonicity in r_1. Let $x = (r_1 - r_2)/s$, it suffices to show that $g(x) = sx + r_2 + sx/(e^x - 1)$ is monotonic in x. Taking the derivative

$$g'(x) = s \cdot \frac{e^x(e^x - (1 + x))}{(e^x - 1)^2} \geq 0, \text{ since } e^x \geq 1 + x.$$

Therefore we may assume that $r_1 = 2s(\ln x_1 + 1)$ and $r_2 = 2s(\ln x_2 + 1)$. Without loss of generality we can assume that $x_1 \geq x_2$ and let $y \geq 2$ be such that $x_1 = (x_1 + x_2)(1 - 1/y)$. By substitution we get $r_1 - r_2 = 2s \ln(y - 1)$ and

$$\begin{aligned} f(s, r_1, r_2) = r_1 + \frac{r_1 - r_2}{e^{(r_1-r_2)/s} - 1} &= 2s\Big(\ln(x_1 + x_2) + 1 + \ln(y - 1) - \ln y + \frac{\ln(y - 1)}{(y - 1)^2 - 1}\Big) \\ &\leq 2s\Big(\ln(x_1 + x_2) + 1 - \frac{1}{y} + \frac{\ln(y - 1)}{(y - 1)^2 - 1}\Big). \end{aligned}$$

We now prove that for $y \geq 2$, $-\frac{1}{y} + \frac{\ln(y-1)}{(y-1)^2-1} \leq 0$. When y approaches 2, the limit of the expression is zero. For $y > 2$, we multiply the left side by $(y - 1)^2 - 1$, and get $g(y) = -(y - 2) + \ln(y - 1)$. Since $g(2) = 0$ and $g'(y) = -1 + 1/(y - 1) < 0$ for $y > 2$, we are done. ☐

Next, we describe algorithm COMBINED as defined on an UMTS $U = (\mathcal{U}_b^d; (r_1, \ldots, r_b); s)$. This algorithm is inspired by *Strategy 3* [4]. Like *Strategy 3*, COMBINED combine ODDEXPONENT and TWOSTABLE on subspaces of $\mathcal{U}_b^d$, however, it does so in a more sophisticated way that is impossible using the combining technique of [4]. Fig. 4.1 presents a scheme of the combining process.

Algorithm Combined As discussed in Observation 1, we may assume that $s = 1$. Let x_i be the minimal real number such that $r_i \leq 100 \ln x_i \ln \ln x_i$ and $x_i \geq e^{e^6+1}$, and let x denote $\sum_i x_i$. For a set $S \subset M_b^d$ let $U(S)$ denote the UMTS induced by U on S.

Let $\mathcal{U}_b^d = \{v_1, \ldots, v_b\}$, where v_i has cost ratio r_i. We partition the points of $\mathcal{U}_b^d$ as follows: let $Q_\ell = \{v_i : \; e^{\ell-1} \leq x_i < e^\ell\}$. Let $P = \{Q_\ell : |Q_\ell| \geq \ln x\} \cup \{\{v\} : v \in Q_\ell \text{ and } |Q_\ell| < \ln x\}$, P is a partition of $\mathcal{U}_b^d$. For $S \in P$ let $x(S) = \sum_{v_i \in S} x_i$. Without loss of generality we assume $P = \{S_1, S_2, \ldots, S_{b'}\}$ where $b' = |P|$ and $x(S_j) \geq x(S_{j+1})$, $1 \leq j \leq b' - 1$.

We associate with every set S_i an algorithm $A(S_i)$ on the UMTS $U(S_i)$. If $|S_i| \geq \ln x$ we choose $A(S_i)$ to be the $(1/10)$-variant of OddExponent. If $|S_i| < \ln x$ then $|S_i| = 1$ and we choose $A(S_i)$ to be the trivial algorithm on one point, this algorithm has a competitive ratio equals to the cost ratio, and it is $(0,0)$-constrained. Let $r(S_i)$ denote the competitive ratio of $A(S_i)$ on $U(S_i)$.

If $b' = 1$ we choose Combined to be $A(S_1)$ and we are done. If $b' \geq 2$, let $\tilde{M} = \cup_{i=2}^{b'} S_i$. We want to construct an algorithm, $A(\tilde{M})$, for $U(\tilde{M})$. If $b' = 2$, we choose $A(\tilde{M})$ to be $A(S_2)$. Otherwise, we apply Theorem 3.1 on $\tilde{M}$ with the partition $\{S_2, \ldots, S_{b'}\}$. We define $\hat{M}$ from Theorem 3.1 to be $\mathcal{U}_{b'-1}^d$. Likewise, $\hat{A}$ from Theorem 3.1 is the application of the $(1/5)$-variant of OddExponent on $\hat{U} = (\mathcal{U}_{b'-1}^d; r(S_2), \ldots, r(S_{b'}))$. Let $r(\tilde{M})$ denote the competitive ratio of $\hat{A}$.

Now we choose the partition $\{S_1, \tilde{M}\}$ of $\mathcal{U}_b^d$. We combine the two algorithms $A(S_1)$ and $A(\tilde{M})$ using the $(1/10)$ variant of TwoStable (this is the $\hat{A}$ required in Theorem 3.1) on the UMTS $(\mathcal{U}_2^d; r(S_1), r(\tilde{M}))$ (the UMTS $\hat{U}$ of Theorem 3.1). We denote the competitive ratio of $\hat{A}$ by r. The resulting combined algorithm, $A(M)$, is Algorithm Combined.

Lemma 4.4. *Given that $x = \sum_i x_i$, $r_i \leq 100 s \ln x_i \ln \ln x_i$, and $x_i \geq e^{e^6+1}$, algorithm* Combined *for the UMTS $U = (\mathcal{U}_b^d; r_1, \ldots, r_b; s)$ is $(1, 1/2)$-constrained and r-competitive, where $r \leq 100 s \ln x \ln \ln x$.*

Proof. As before, without loss of generality, we assume $s = 1$. First we calculate the constraints of the algorithm.

From Lemma 4.1 and Lemma 3.9, $A(S_i)$ is $(1/10, 1/10)$-constrained, for every $1 \leq i \leq b'$. We would like to show that $A(\tilde{M})$ is $(1/2, 3/10)$-constrained. If $b' = 2$ then it obviously $(1/10, 1/10)$-constrained. Else $(b' > 2)$, the combining algorithm for $\tilde{M}$ is the $(1/5)$-variant of OddExponent which is $(1/5, 1/5)$-constrained. Hence, from Eq. (3.1), $\beta \leq (d/5 + d/10 + d/10 + d/10)/d = 1/2$, and from Eq. (3.2), $\eta \leq (d/5 + d/10)/d = 3/10$. From Theorem 3.1, $A(\tilde{M})$ is $r(\tilde{M})$ competitive.

The (β, η)-constraints of algorithm Combined are calculated as follows: The $(1/10)$-variant of TwoStable is $(1/10, 2/10)$ constrained, therefore $\beta = (d/10 + d/10 + d/2 + 3d/10)/d = 1$ and $\eta = (2d/10 + 3d/10)/d = 1/2$. From Theorem 3.1, $A(M)$ is r-competitive.

To summarize, Combined is $(1, 1/2)$-constrained and r-competitive algorithm for the UMTS U.

It remains to prove the bound on r. First we show that $r(S_j) \leq 100 s \ln x(S_j) \ln \ln x(S_j)$ for all $1 \leq j \leq b'$. If $|S_j| = 1$, we are done. Otherwise, $|S_j| \geq \ln x$, and $S_j = Q_\ell$ for some ℓ.

$$
\begin{aligned}
r(S_j) &\le 100 \ln e^{\ell} \ln\ln e^{\ell} + 6 \cdot 10 \ln |S_j| && (4.1)\\
&\le 100\big(\ln e^{\ell-1} \ln\ln e^{\ell-1} + \ln \ell + \tfrac{1}{\ell-1} \ln e^{\ell-1}\big) + 60 \ln |S_j| \\
&\le 100\big(\ln e^{\ell-1} \ln\ln e^{\ell-1} + \ln\ln x + \tfrac{60}{100} \ln |S_j| + 1\big) && (4.2)\\
&\le 100\big(\ln e^{\ell-1} \ln\ln e^{\ell-1} + 2 \ln |S_j|\big) && (4.3)\\
&\le 100 \ln(|S_j| e^{\ell-1}) \ln\ln(|S_j| e^{\ell-1}) \\
&\le 100 \ln x(S_j) \ln\ln x(S_j). && (4.4)
\end{aligned}
$$

We derive inequality (4.1) by noting that we apply a (1/10) variant of OddExponent to $S_j = Q_\ell$. This implies that $r_i \le 100 s \ln e^{\ell} \ln\ln e^{\ell}$ for all $v_i \in S_j$. By the bound on the competitive ratio of the (1/10) variant of OddExponent (See Lemma 4.1 and Lemma 3.9) we get the inequality. Inequality (4.2) follows since $\ell \le \ln x$. Inequality (4.3) follows because $\ln |S_j| \ge \ln\ln x$, and $\ln\ln x \ge 6$. The last inequality follows because $e^{\ell-1}$ is a lower bound on x_i for $v_i \in S_j$ and thus $|S_j| e^{\ell-1} \le x(S_j)$.

We note that $b' \le \ln^2 x$ as there are at most $\ln x$ sets Q_i, and each such set contributes at most $\ln x$ sets S_i to P. We now derive a bound on $r(\tilde{M})$.

$$
\begin{aligned}
r(\tilde{M}) &\le \max_{2 \le i \le b'} r(S_i) + 6 \cdot 5 \cdot \ln(b' - 1) && (4.5)\\
&\le 100 \cdot \ln x(S_2) \ln\ln x + 30 \cdot (2 \ln\ln x) && (4.6)\\
&= 100(\ln x(S_2) + 0.6) \ln\ln x
\end{aligned}
$$

Inequality (4.5) follows since the algorithm used is a (1/5) variant of OddExponent. Inequality (4.6) follows by using the previously derived bound on $r(S_i)$ and noting that $x(S_2)$ is maximal amongst $x(S_2), \ldots, x(S_{b'})$ and that $x(S_i) \le x$.

From Lemma 3.9 we know that the competitive ratio of the (1/10)-variant of TwoStable is $f(10, r(S_1), r(\tilde{M}))$ where f is the function as given in Proposition 4.3. We give an upper bound on $f(10, r(S_1), r(\tilde{M}))$ using Proposition 4.3. To do this we need to find values y_1 and y_2 such that

$$
\begin{aligned}
r(S_1) &\le 100 \ln x(S_1) \ln\ln x = 2 \cdot 10(\ln y_1 + 1)\\
r(\tilde{M}) &\le 100(\ln x(S_2) + 0.6) \ln\ln x = 2 \cdot 10(\ln y_2 + 1).
\end{aligned}
$$

Indeed, the following values satisfy the conditions above: $y_1 = x(S_1)^{5 \ln\ln x}/e$ and $y_2 = (e^{0.6} x(S_2))^{5 \ln\ln x}/e$. Using Proposition 4.3 we get a bound on r as follows

$$
\begin{aligned}
r &\le 2 \cdot 10(\ln(y_1 + y_2) + 1) && (4.7)\\
&\le 20 \ln\big(x(S_1)^{5 \ln\ln x} + (e^{0.6} x(S_2))^{5 \ln\ln x}\big) \\
&\le 20 \ln\big(x(S_1)^{5 \ln\ln x} + (2^{5 \ln\ln x} - 2) x(S_2)^{5 \ln\ln x}\big) && (4.8)\\
&\le 20 \ln\big((x(S_1) + x(S_2))^{5 \ln\ln x}\big) && (4.9)\\
&\le 100 \ln x \ln\ln x
\end{aligned}
$$

Inequality (4.7) follows from Proposition 4.3. Inequality (4.8) follows because $\ln\ln x \ge 6$. Inequality (4.9) follows since, in general, for $a \ge b > 0$ and $z \ge 1$, $a^z + (2^z - 2) b^z \le (a + b)^z$. This is because

$$
a^z + (2^z - 2) b^z \le (\max\{a, 2b\})^{z - \lfloor z \rfloor} \big(a^{\lfloor z \rfloor} + (2^{\lfloor z \rfloor} - 1) b^{\lfloor z \rfloor}\big) \le (a+b)^{z - \lfloor z \rfloor} (a+b)^{\lfloor z \rfloor} = (a+b)^z
$$

The second inequality follows since $\max\{a, 2b\} \leq a + b$ and from the sum of the binomial coefficients. □

Next, we present a better algorithm when all the cost ratios but one are equal.

LEMMA 4.5. *Given an UMTS $U = (\mathcal{U}_b^d; r_1, r_2, \ldots, r_b)$ with $r_2 = r_3 = \cdots = r_b$, there exists a $(1, 3/5)$-constrained and r-competitive online algorithm,* WCOMBINED, *where*

$$r = 30\Big(\ln\big(e^{\frac{r_1}{30}-\frac{1}{3}} + (b-1)e^{\frac{r_2}{30}-\frac{1}{3}}\big) + \tfrac{1}{3}\Big).$$

Proof. The proof is a simplified version of the proof of Lemma 4.4, and we only sketch it here. We define x_1, x_2, such that

$$r_1 = 30(\ln x_1 + \tfrac{1}{3}) = 2 \cdot 5 \cdot (\ln x_1^3 + 1), \quad r_2 = 30(\ln x_2 + \tfrac{1}{3}) = 2 \cdot 5 \cdot (\ln x_2^3 + 1).$$

Let $\tilde{M} = \{v_2, \ldots v_b\}$. We use a $(1/5)$ variant of ODDEXPONENT on the UMTS $U(\tilde{M})$. The competitive ratio of this algorithm is at most

$$r(\tilde{M}) \leq r_2 + 30\ln(b-1) \leq 30\big(\ln((b-1)x_2) + \tfrac{1}{3}\big) = 10\big(\ln((b-1)x_2)^3 + 1\big)$$

and it is $(1/5, 1/5)$ constrained. We combine it with the trivial algorithm for $U(\{v_1\})$ using a $(1/5)$ variant of algorithm TWOSTABLE, the resulting algorithm is $(1, 3/5)$ constrained, and by Proposition 4.3 we have

$$r \leq 10(\ln(x_1^3 + ((b-1)x_2)^3 + 1) \leq 10(\ln(x_1 + (b-1)x_2)^3 + 1) = 30(\ln(x_1 + (b-1)x_2) + \tfrac{1}{3}).$$

Substituting for x_i gives the required bound. □

5. Applications.

5.1. An $O((\log n \log\log n)^2)$ Competitive algorithm for MTSs.

Bartal [2] defines a class of decomposable spaces called *hierarchically well separated trees* (HST).[1]

DEFINITION 5.1. *For $k \geq 1$, a k*-hierarchically well-separated tree *(k-HST) is a metric space defined on the leaves of a rooted tree T. Associated with each vertex $u \in T$ is a label $\Delta(u) \geq 0$, and $\Delta(u) = 0$ iff u is a leaf of T. The labels obey that any vertex v, a child of u, $\Delta(v) \leq \Delta(u)/k$. The distance between two leaves $x, y \in T$ is defined as $\Delta(\mathrm{lca}(x, y))$, where $\mathrm{lca}(x, y)$ is the least common ancestor of x and y in T. Clearly, this is a metric. We call a vertex with out-degree* 1, degenerate vertex. *For non-degenerate vertex u, $\Delta(u)$ is the diameter of the sub-space induced on the subtree rooted by u. Note that any HST can be transform into an HST without degeneracies with the same metric.*

Bartal [2, 3] shows how to approximate any metric space using an efficiently constructable probability distribution over a set of k-HSTs . His result allows to reduce a MTS problem on an arbitrary metric space to MTS problems on HSTs. Formally, he shows the following theorem.

THEOREM 5.2 ([3]). *Suppose there is a r-competitive algorithm for any n-points k-HST metric space. Then there exists an $O(rk \log n \log\log n)$-competitive randomized algorithm for* any *n-point metric space.*

[1] The definition given here for k-HST differs slightly from the original definition in [2]. We choose the definition given here for simplicity of the presentation. For $k > 1$ the metric spaces given by these two definitions approximate each other to within a factor of $k/(k-1)$.

We seek an online algorithm for a metrical task system where the underlying metric space is a k-HST. Following [4] we use the unfair MTS model to obtain an online algorithm for a MTS over a k-HST metric space.

ALGORITHM RHST. We define the algorithm RHST(T) on the metric space $M(T)$, where T is a k-HST with $k \geq 5$. Algorithm RHST(T) is defined inductively on the depth of the underlying HST, T.

When $|M(T)| = 1$, RHST(T) serves all task sequences optimally. It is $(0,0)$-constrained. Otherwise, let the children of the root of T be $v_1, \ldots, v_b$, and let T_i be the subtree rooted at v_i. Denote $d = \Delta(T)$, and so $\Delta(T_i) \leq d/k$. Every algorithm RHST(T_i) is an algorithm for the UMTS $U_i = (M(T_i); (1, \ldots, 1))$.

We construct a metric space $\hat{M} = \mathcal{U}_b^d$, we define cost ratios $r_1, \ldots, r_b$ where $r_i = r(T_i)$, the competitive ratio of RHST(T_i). We now use Theorem 3.1 to combine algorithms RHST(T_i). The role of $\hat{A}$ is played by the $(1/2)$ variant of COMBINED on the unfair metrical task system $\hat{U} = (\hat{M}; (r_1, \ldots, r_b))$. The combined algorithm is a RHST(T) on the UMTS $(M(T); (1, \ldots, 1))$.

We remark that the application of Theorem 3.1 requires that the algorithms be constrained, we show that this is true in the following lemma.

LEMMA 5.3. *The algorithm* RHST*(T) is* $O(\ln n \ln \ln n)$ *competitive against reasonable task sequences, where* $n = |M(T)|$.

Proof. Let $n' = e^{e^6+1} n$. We prove by induction on the depth of the tree that RHST(T) is $(1,1)$-constrained and $200 \ln n' \ln \ln n'$-competitive.

When $|M(T)| = 1$, it is obvious. Otherwise, let $n_i = |M(T_i)|$, $n'_i = e^{e^6+1} n_i$, and $n' = \sum_i n'_i$. We assume inductively that each of the RHST(T_i) algorithms is $(1,1)$-constrained and $200 \ln n'_i \ln \ln n'_i$ competitive on $M(T_i)$. The combined algorithm, RHST(T), is (β, η)-constrained. From Eq. (3.1), and given that $k \geq 5$, we get that

$$\beta \leq \max\left\{1, \frac{1/2 \cdot d + d/k + d/k + 0.5d/k}{d}\right\} \leq \max\{1, 1\} = 1.$$

From Eq. (3.2) we get that $\eta \leq 1/2 \cdot \frac{d}{d} + \frac{d/k)}{d} \leq 1$, for $k \geq 5$. This proves that the algorithm is well defined and $(1,1)$ constrained.

We bound the competitive ratio using Lemma 4.4, we apply a $(1/2)$ variant of COMBINED, from Lemma 3.9 this means that the competitive ratio obtained by the $(1/2)$ variant of COMBINED on $(\hat{M}; r_1, \ldots, r_b)$ is the same as the competitive ratio attained by COMBINED on $(\hat{M}; r_1, \ldots, r_b; 2)$. Note that x_i computed by COMBINED is at most n'_i, hence by Lemma 4.4 it follows that the competitive ratio is at most $100 \cdot 2 \ln x \ln \ln x \leq 200 \ln n' \ln \ln n'$, since $x = \sum_i x_i$. ☐

Since every HST T can be 5-approximated by a 5-HST T' (see [3]), the bound we have just proved holds for any HST.

Combining Theorem 5.2 with Lemma 5.3, it follows that

THEOREM 5.4. *For any MTS over an* n*-point metric space, the randomized competitive ratio is* $O((\log n \log \log n)^2)$.

5.2. K-Weighted Caching on $K+1$ Points. Weighted caching is a generalized paging problem where there is a different cost to fetch different pages. This problem is equivalent to the K-server problem on a *star metric space* [23, 8]. A star metric space is derived from a depth one tree with distances on the edges, the points of the metric space are the leaves of the tree and the distances between a pair of points is the length of the (2 edge) path between them. This is so, since we can assign any edge (r, u) in the tree a weight of half the fetch cost of u. Together, an entrance of a server into a leaf from the star's middle-point (page in) and leaving the leaf to the star's middle point (page out) have the same cost of fetching the page.

The K-server problem on a metric space on $K+1$ points is a special case of the metrical task system problem on the same metric space, and hence any upper bound for the metrical task system translates to an upper bound for the corresponding K-server problem.

Given a star metric space M, we 12-approximates it with a 6-HST T. T has the special structure that for every internal vertex, all children except perhaps one, are leaves. It is not hard to see that one can find such a tree T such that for any $u, v \in M$, $\mathrm{d}_M(u,v) \le \mathrm{d}_T(u,v) \le 12 \cdot \mathrm{d}_M(u,v)$. Essentially, the vertices furthest away from the root (up to a factor of 6) in the star are children of the root of T and the last child of the root is a recursive construction for the rest of the points.

We now follow the construction of RHST given in the previous section, on an 6-HST T, except that we make use of $(1/2)$-variant of WCOMBINED rather than $(1/2)$-variant of COMBINED. The special structure of T implies that all the children of an inner vertex, except perhaps one, are leaves and therefore have a trivial 1-competitive algorithm on their "subspaces". Hence we can apply WCOMBINED. Using induction on the depth of the tree and Lemma 4.5, it is easy to bound the competitive ratio on $K+1$ leaves tree to be at most $60(\ln(K+1)+1/3)$.

Combining the above with the lower bound of [8] we get:

THEOREM 5.5. *The competitive ratio for the K-weighted caching problem on $K+1$ points is $\Theta(\log K)$.*

5.3. A MTS on Equally Spaced Points on the Line. The metric space of n equally spaced points in the line is considered important because of its simplicity, and the practical significance of the k-server on the line (for which this problem is a special case). The best lower bound currently known on the competitive ratio is $\Omega(\log n/\log\log n)$ [9]. Previously, the best upper bound known was $O(\log^3 n/\log\log n)$ due to [4].

We are able to slightly improves the upper bound on the competitive ratio from Section 5.1 to $O(\log^2 n)$. Bartal [2] proves that n equally spaced points on the line can be $O(\log n)$ probabilistically approximated by a set of *binary* 4-HSTs. We present an $O(\log n)$ competitive randomized algorithm for binary 4-HST, similar to RHST except that we make use of $(1/4)$-variant of TWOSTABLE instead of $(1/2)$-variant of COMBINED. Similar arguments show that this algorithm is $(1,1)$-constrained, and using Proposition 4.3 we conclude that the algorithm is $8\ln n$ competitive. Combining the path-approximation by binary 4-HST with the algorithm for binary 8-HST we get

THEOREM 5.6. *The competitive ratio of the MTS problem on metric space of n equally spaced points on the line is $O(\log^2 n)$.*

6. Concluding Remarks. This paper present algorithms for MTS problem and related problems with significantly improved competitive ratio. An obvious avenue of research is to further improve the the upper bound on the competitive ratio for the MTS problem. A slight improvement to the competitive ratio is reported in [19]. The resulting competitive ratio is $O(\log^2 n \log\log n \log\log\log n)$ and the improvement is achieved by refining the reduction from arbitrary metric spaces to HST spaces (*i.e.*, that improvement is orthogonal to the improvement presented in this paper). However, in order to break the $O(\log^2 n)$ bound, it seems that one needs to deviate from the black box usage of Theorem 5.2. Maybe the easiest special case to start with is the metric space on equally spaced points on the line.

An interesting line of research would be to attempt to apply the techniques of this and previous papers to the randomized k-server problem, or even for a special case such as the randomized weighted caching on k pages problem; see also [7, 21].

The careful reader might have noticed that the combining technique (Theorem 3.1) can also be applied to deterministic algorithms, and also has limited applicability to constrained set of tasks. It is interesting to investigate possible applications of the technique in these, and maybe others, contexts.

Acknowledgments. We would like to thank Yair Bartal, Avrim Blum and Steve Seiden for helpful discussions.

REFERENCES

[1] D. Achlioptas, M. Chrobak, and J. Noga, *Competitive analysis of randomized paging algorithms*, Theoretical Computer Science, 234 (2000), pp. 203–218.

[2] Y. Bartal, *Probabilistic approximation of metric space and its algorithmic application*, in 37th Annual Symposium on Foundations of Computer Science, Oct. 1996, pp. 183–193.

[3] ———, *On approximating arbitrary metrics by tree metrics*, in Proceedings of the 30th Annual ACM Symposium on Theory of Computing, 1998, pp. 183–193.

[4] Y. Bartal, A. Blum, C. Burch, and A. Tomkins, *A polylog(n)-competitive algorithm for metrical task systems*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, May 1997, pp. 711–719.

[5] Y. Bartal, B. Bollobás, and M. Mendel, *A ramsey-type theorem for metric spaces and its application for metrical task systems and related problems*, in Proceedings of the 42nd Annual Symposium on Foundations of Comptuer Science, Las Vegas, Nevada, 2001.

[6] S. Ben-David, A. Borodin, R. Karp, G. Tardos, and A. Wigderson, *On the power of randomization in on-line algorithms*, Algorithmica, 11 (1994), pp. 2–14.

[7] A. Blum, C. Burch, and A. Kalai, *Finely-competitive paging*, in 40th IEEE Symposium on Foundations of Computer Science, 1999.

[8] A. Blum, M. L. Furst, and A. Tomkins, *What to do with your free time: algorithms for infrequent requests and randomized weighted caching.* manuscript, Apr. 1996.

[9] A. Blum, H. Karloff, Y. Rabani, and M. Saks, *A decomposition theorem and lower bounds for randomized server problems*, SIAM J. Comput., 30 (2000), pp. 1624–1661.

[10] A. Borodin, N. Linial, and M. Saks, *An optimal online algorithm for metrical task systems*, J. Assoc. Comput. Mach., 39 (1992), pp. 745–763.

[11] M. Chrobak, H. Karloff, T. Payne, and S. Vishwanathan, *New results on server problems*, in 1st Annual ACM-SIAM Symposium on Discrete Algorithms, 1990, pp. 291–300.

[12] M. Chrobak and L. L. Larmore, *The server problem and on-line games*, in On-line Algorithms, L. A. McGeoch and D. D. Sleator, eds., vol. 7 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Feb. 1991, pp. 11–64.

[13] A. Fiat, R. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young, *Competitive paging algorithms*, Journal of Algorithms, 12 (1991), pp. 685–699.

[14] S. Irani and S. Seiden, *Randomized algorithms for metrical task systems*, Theoretical Computer Science, 194 (1998), pp. 163–182.

[15] A. Karlin, M. Manasse, L. Rudolph, and D. D. Sleator, *Competitive snoopy caching*, Algorithmica, 3 (1988), pp. 79–119.

[16] H. Karloff, Y. Rabani, and Y. Ravid, *Lower bounds for randomized k-server and motion-planning algorithms*, SIAM Journal on Computing, 23 (1994), pp. 293–312.

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

[18] L. McGeoch and D. Sleator, *A strongly competitive randomized paging algorithm*, J. Algorithms, 6 (1991), pp. 816–825.

[19] M. Mendel, *Online Algorithms for Paging and Metrical Task Systems*, PhD thesis, School of Computer Science, Tel-Aviv University, Tel-Aviv, Israel, October 2001.

[20] S. Seiden, *Unfair problems and randomized algorithms for metrical task systems*, Information and Computation, 148 (1999), pp. 219–240.

[21] ———, *A general decomposition theorem for the k-server problem*, manuscript, Louisiana State University, 2000.

[22] D. D. Sleator and R. E. Tarjan, *Amortized efficiency of list update and paging rules*, Communication of the ACM, 28 (1985), pp. 202–208.

[23] N. E. Young, *The k-server dual and loose competitiveness for paging*, Algorithmica, 11 (1994), pp. 525–541.