Improved Algorithms and Lower Bounds for Parametrized Metrical Service Systems

Junhao Gan
School of Computing and Information Systems, The University of Melbourne, Australia

Xiao Sun
School of Computing and Information Systems, The University of Melbourne, Australia

Seeun William Umboh
School of Computing and Information Systems, The University of Melbourne, Australia
ARC Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Melbourne, Australia

Abstract

We consider the parametrized setting of the classical metrical service system (MSS) problem first studied by Bubeck and Rabani (APPROX/RANDOM 2020). In this setting, the adversary is restricted to a set of m distinct request types, known to the algorithm in advance. The goal is to obtain competitive ratio bounds in terms of m. In this work, we make significant progress in understanding the landscape of parametrized MSS and resolve several open problems from Bubeck and Rabani.

Our first main result is a tight bound for parametrized MSS on weighted stars. Previously, Bubeck and Rabani gave a randomized lower bound of $\Omega(m)$ and deterministic upper bound of $O(2^m)$. We show that, surprisingly, a deterministic $O(m)$-competitive algorithm exists, matching the randomized lower bound. Our key insight is an interval covering formulation of MSS on weighted stars which enables an application of the primal-dual method.

Our second main contribution is an improved lower bound construction for parametrized MSS on hierarchically separated trees (HSTs). Bubeck and Rabani's construction gave a $\omega(1)$ lower bound when $m \geq 6$. Our improved lower bounds are tight for 2-level HSTs and also rule out $O(1)$-competitive algorithms on HSTs when the parameter $m \geq 4$. We also complement these results by giving a deterministic $O(1)$-competitive algorithm on general metrics when $m = 2$ while showing that it is impossible when $m \geq 3$.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, competitive analysis, metrical service systems, metrical task systems

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2026.11

Category APPROX

Related Version *Full Version*: https://arxiv.org/abs/2607.07098

Funding *Junhao Gan*: Supported in part by the Australian Government through the Australian Research Council ARC DP230102908.
Xiao Sun: Supported by the Australian Government through the Australian Research Council DP240101353 and by the University of Melbourne through the Melbourne Research Scholarship.
Seeun William Umboh: Supported by the Australian Government through the Australian Research Council DP240101353.

Acknowledgements We thank Christian Coester for insight into the case of $m = 3$ on general metrics.

1 Introduction

The *metrical task system* (MTS) problem introduced by Borodin et al. [2] is a general framework that models a wide range of online computing problems, e.g., paging [15] and k-server [13]. In this problem, one server moves between different points (also called states) in an n-point metric space to process a sequence of requests that arrive one by one; each request incurs different costs when the server is at different states. The total cost is the sum of the transition cost of moving between different states and the cost of serving the requests. An important special case is the *metrical service system* (MSS) problem [9] where every request has cost either 0 or infinity everywhere in the whole metric space. Note that paging and k-server are both MSS problems.

Since its introduction almost four decades ago, MTS has been intensively studied on both general metrics and some special classes of metrics. On general metrics, in a series of recent breakthrough results, Bubeck et al. [4] showed that the randomized competitive ratio is $O(\log^2 n)$ and Bubeck et al. [3] gave a matching *existential lower bound*[1], refuting the previous widely-believed conjecture of the existence of a logarithmic upper bound. Bubeck et al. [3] also showed a *universal lower bound*[2] of $\Omega(\log n)$, which is also tight in the sense that an $O(\log n)$ upper bound can be achieved on some special metrics, e.g. uniform metrics [2] and weighted stars [4]. While MSS is a special case of MTS, the best competitive ratio that can be achieved by both deterministic and randomized algorithms for MSS in all cases is asymptotically the same as that of MTS only up to a constant factor.

While many online problems are metrical task systems, the best bounds on their competitive ratios are much lower than those implied by the general worst-case MTS bounds. For instance, while the k-server problem in an N-point uniform metric space can be reduced to an MSS instance with $n = \binom{N}{k}$ states, the randomized competitive ratio is known to be $O(\log k)$, which is much lower than the lower bound of $\Omega(\log \binom{N}{k})$ implied by the universal lower bound for MSS with $\binom{N}{k}$ states. The reason for this discrepancy is because while there are $2^n - 1$ possible requests in a general n-state MSS, there are only N possible requests for a k-server.

The above example motivates studying the competitive ratio of MTS in terms of a parameter of the set of possible requests, instead of the number of states n. Burley and Irani [8] proposed examining request sequences drawn from a given set. For uniform metrics, they provided a deterministic algorithm with competitive ratio $O(\log n)$ times the (unknown) best possible competitive ratio. Another example that shows the power of restricting the set of possible requests is the power management problem [11] in which there are only two possible requests and the competitive ratio is $O(1)$ [1]. For MSS, one parameter that has been studied is the *width* (e.g. [10, 14, 7]), which is the maximum size of every requested set. Width is well-motivated as MSS can also be viewed as online set chasing, in the sense that the server needs to move to a state in the feasible set where the cost of the current request is 0 rather than ∞, and we only need to consider transition cost while the service cost is zero if the server stays in the feasible set at every step.

Parametrized MTS. The parametrized MTS proposed by Bubeck and Rabani [5] studies the number of possible requests as the parameter. In particular, the adversary can only generate requests from a fixed set of m requests known to the algorithm in advance. On

[1] The existential lower bound means that there exist n-point metrics on which the lower bound holds.
[2] The universal lower bound means that it holds for every n-point metric space.

uniform metrics, they showed that the deterministic competitive ratio is $\Theta(m \log(en/m))$, which is significantly better than the general bound of $\Theta(n)$. Interestingly, the randomized competitive ratio on uniform metrics is still $\Theta(\log n)$ even when $m = 2$. On the other hand, for two-level hierarchically separated tree (HST) metrics, the deterministic competitive ratio is $\Theta(n)$ and the randomized competitive ratio is $\Theta(\log n)$, even when m is constant. In summary, restricting m only improves the deterministic competitive ratio for uniform metrics, and does not help for two-level HSTs.

Parametrized MSS. The more interesting case is parametrized MSS, where parametrization leads to substantial improvements of the competitive ratio from the general case. For MSS, restricting possible requests within a fixed set of size m improves both deterministic and randomized competitive ratios: for uniform metrics, the deterministic and randomized competitive ratios are both $\Theta(m)$ and thus significantly better when $n \gg m$. Another interesting aspect of this result is that deterministic algorithms are essentially as powerful as randomized algorithms, which refutes the folklore that the randomized competitive ratio is polylogarithmic in the deterministic competitive ratio for most problems that can be interpreted as metrical task systems. As for the case of two-level HSTs, there are still gaps between the currently known upper and lower bounds: the deterministic competitive ratio is between $\Omega(\binom{m}{\lfloor m/2 \rfloor}) = \Omega(2^m/\sqrt{m})$ and $O(m2^m)$ and the randomized competitive ratio is at least $\Omega(2^{\lfloor m/2 \rfloor})$.

Bubeck and Rabani [5] proposed the following open problems in the study of parametrized MSS, which involve not only the exact tight bound of the competitive ratio in certain scenarios, but also understanding how restricting the number of possible requests m can help improve the competitive ratio, e.g., can we devise an online algorithm whose competitive ratio is purely in terms of m rather than metric space size n, and can we do this for some special classes of metrics like HSTs or even general metrics?

▶ **Question 1.** *What is the deterministic competitive ratio on weighted star metrics?*

▶ Remark 1.1. It is easy to design an $O(2^m)$-competitive deterministic algorithm and an $O(m)$-competitive randomized algorithm by reducing the problem to MSS on a weighted star where there are only $2^m - 1$ different states and applying the state-of-the-art result for MTS on weighted stars. For the special case of uniform metrics, the deterministic and randomized competitive ratios are both known to be $\Theta(m)$ so it is natural to ask if the deterministic algorithm can be generalized to weighted stars.

▶ **Question 2.** *What is the tight bound of the worst-case competitive ratio on all HSTs of two and higher levels?*

▶ **Question 3.** *For what constant values of m can we get constant-competitive algorithms for general metrics or some special classes of metrics like HSTs?*

1.1 Our Contributions

In this paper, we make substantial progress towards understanding the competitiveness of parametrized MSS.

Weighted stars. Our first main result answers Question 1 affirmatively.

▶ **Theorem 1.2.** *There is a $2m$-competitive deterministic algorithm for parameterized MSS on weighted stars.*

The result brings more insight into the foundation of this problem and the power of randomization as well, because it matches the lower bound of $\Omega(m)$ for randomized algorithms since the uniform metric is a special class of the weighted star. Theorem 1.2 shows that deterministic algorithms are essentially as powerful as randomized algorithms in achieving the best possible competitive ratio of $O(m)$ on weighted stars, the same as the case for the parametrized metrical service system problem on uniform metrics.

2-level HSTs. Next, we address Question 2 by settling the randomized competitive ratio on 2-level HSTs. Let $c_{k,m}$ denote the best possible randomized competitive ratio of the MSS problem that can be achieved on all k-level HSTs with m request types:

▶ **Theorem 1.3.** $c_{2,m} = \Theta(\binom{m}{\lceil m/2 \rceil}) = \Theta(2^m/\sqrt{m})$.

This theorem contains two separate claims: $c_{2,m} = O(\binom{m}{\lceil m/2 \rceil})$ (Theorem 4.17) and $c_{2,m} = \Omega(\binom{m}{\lceil m/2 \rceil})$ (Theorem 4.7). The lower bound is existential, in the sense that there exists a 2-level HST and m request types on it such that no randomized algorithm can achieve a better worst-case competitive ratio on all possible request sequences. Theorem 1.3 improves upon previous upper and lower bounds. Previously, the lower bound of $\Omega(\binom{m}{\lceil m/2 \rceil})$ is only known to hold for deterministic algorithms, and we prove that it also holds for randomized algorithms using a new hard instance.

Higher level HSTs. Finally, we turn to Question 3. Previously, Bubeck and Rabani [5] obtained lower bounds on $c_{k,m}$ for $k \geq 3$ by iterating their lower bound construction for $k = 2$. As a consequence, their lower bound rules out $O(1)$-competitive algorithms for $m \geq 6$. By iterating our construction in the proof of Theorem 4.7 to higher level HSTs, we also obtain improved lower bounds on $c_{k,m}$ for $k \geq 3$. The formal theorem statement (Theorem 4.8) is somewhat technical so we defer it to Section 4.1.4 and instead state the following corollary.

▶ **Corollary 1.4.** *For $m \geq 4$, there is no algorithm that achieves constant competitive ratio on HSTs.*

Combined with Observation 1.6, we conclude that the only open case for Question 3 on HSTs is $m = 3$.

General metrics. We resolve Question 3 on general metrics by the following two complement results.

We first show that a constant competitive ratio cannot be achieved on general metrics when m is at least 3 by the following existential lower bound of $\Omega(\log n)$:

▶ **Theorem 1.5.** *For $m \geq 3$, there exist $3n$-point metrics on which every randomized algorithm has competitive ratio of $\Omega(\log n)$.*

We refer the reader to the full version to see the proofs of Theorem 1.5 as well as the following Observation 1.6.

On the other hand, a constant competitive ratio can be achieved on general metrics when $m = 2$.

▶ **Observation 1.6.** *When $m = 2$, there is a 6-competitive deterministic algorithm on general metrics.*

This result follows from the observation that when $m = 2$, we can assume that the input sequence is strictly alternating between two request types, because MSS can be regarded as chasing feasible sets and hence repeating the same request does not incur additional cost. Therefore, the only uncertainty of the input is its total length, and the optimal cost can be expressed as a non-decreasing function of the input length. This observation allows us to apply the Guess-and-Double Method to design a constant-competitive algorithm for general metrics when $m = 2$.

Therefore, the open problems for Question 3 now lie on some special classes of metrics, including HSTs that we study in this paper.

1.2 Our Techniques

Weighted stars. In MSS, we can think of a request as the set of states that the server must move to. Bubeck and Rabani [5] use an algorithm that works in phases similar to the classical marking algorithm to provide a $O(m)$-deterministic algorithm for parametrized MSS on uniform metrics: in each phase, it moves to an arbitrary state in the intersection of the requests in the phase; if the intersection is empty, the algorithm starts a new phase and moves to an arbitrary requested state. It is clear that any solution needs to move at least once during each phase, and they show that the algorithm moves at most m times during each phase to prove the upper bound. This method is heavily based on the fact that the metric space is uniform and hence we can simply compare the number of times the algorithm's server moves with that of the optimal server. It is unclear whether their method can be adapted to solve the problem on weighted stars.

We use a significantly different approach based on an interval covering formulation of the problem and apply the primal-dual method. Each interval corresponds to a consecutive subsequence of requests, and the cost to buy this interval is the minimum possible cost of moving to a state that can satisfy all request types in this subsequence. Our analysis exploits the fact that there are only m different request types to show that the algorithmic cost is at most m times the feasible dual solution that we maintain and thus obtain the desired competitive ratio. Our formulation is similar to that in [6] but we exploit the property of MSS requests that allows us to only consider the unique state with minimum cost to satisfy every subsequence, while for MTS we have to consider various costs of serving the same subsequence of requests in n different states, thus leading to the appearance of n in the competitive ratio instead of purely in terms of m. We hope that similar methods under the linear programming and primal-dual framework can be adapted to solve the more general parametrized MTS problem.

Lower bounds on HSTs. We use the same high-level approach used by Bubeck and Rabani [5] in which a construction for the uniform metric[3] is lifted to HSTs of higher levels. The main conceptual difference is in our construction for the uniform metric. In Bubeck and Rabani's construction, there are $2^{m/2}$ points, each labeled by a node of the Boolean hypercube $\{0,1\}^{m/2}$ (we assume m is even here for simplicity). Each of the m requests correspond to a an axis-aligned halfspace of the Boolean hypercube. Thus, if we construct a sequence of $m/2$ requests by picking a random halfspace for each dimension with equal probability, then for each request, the algorithm has to move with probability $1/2$. On the other hand, the adversary can move to the point in the intersection of the halfspaces. This construction can be lifted to HSTs of higher levels by partitioning the set of all possible labels of the lower level case into pairs of antipodes and treating them as complementary requests.

[3] A 1-level HST is equivalent to a uniform metric.

Our key idea is to use a construction for the uniform metric that has more points. In particular, we use $\binom{m}{m/2}$ points, each corresponding to an $\frac{m}{2}$-sized subset of $[m]$. As a result, when we apply the above lifting process, for every fixed level k of the HST, there are more subtrees in our construction when m is sufficiently large. We need to be more careful in our analysis as it is no longer true that each request causes the algorithm to move with constant probability. Instead, we prove that its average is still lower bounded by a positive constant.

2 Preliminaries

Given a metric space $\mathcal{M} = (X, d)$ where X is the set of points and $d : X \times X \to \mathbb{R}_{\geq 0}$ is the distance function, a parametrized metrical service system instance consists of an initial location $s_0 \in X$ of the server and a sequence of requests $\{\rho_t\}_{t\geq 1}$, where each ρ_t is from a fixed collection $\{K_1, \ldots, K_m\}$ of m non-empty subsets of X. Upon the arrival of a request $\rho_t = K_i$ for some $1 \leq i \leq m$ at every step $t \geq 1$, one must move the server to some state $s_t \in K_i$ to maintain a feasible solution if its previous state s_{t-1} does not satisfy this, and the decision must be made without knowing future requests after ρ_t. As a result, MSS can be regarded as the online set chasing problem, illustrating that we always need to move the server to a state in the current requested set, and the parametrized setting restricts that there are only m different request types. The cost of a solution $\{s_t\}_{t\geq 1}$is defined to be

$$\sum_{t\geq 1} d(s_{t-1}, s_t),$$

where $d(s_{t-1}, s_t)$ denotes the transition cost to move from s_{t-1} to s_t upon the arrival of ρ_t. This can be interpreted as the total length of the trajectory of the server.

A randomized algorithm $\mathcal{A}$ has competitive ratio c if there exists some constant β that is independent of the input sequence (it might depend on the metric space $\mathcal{M}$) such that for every input sequence $\boldsymbol{\rho}$:

$$\mathbb{E}[\text{cost}_{\mathcal{A}}(\boldsymbol{\rho})] \leq c \cdot \text{cost}^*(\boldsymbol{\rho}) + \beta,$$

where $\mathbb{E}[\text{cost}_{\mathcal{A}}(\boldsymbol{\rho})]$ is the expected cost of $\mathcal{A}$ on input $\boldsymbol{\rho}$ and $\text{cost}^*(\boldsymbol{\rho})$ is the minimum cost among all feasible solutions of $\boldsymbol{\rho}$. If $\beta = 0$, then we say that c is a strict competitive ratio of $\mathcal{A}$. When we talk about the competitive ratio of a problem class (e.g. parametrized MSS with m requests on a class of metrics), we are referring to the best possible competitive ratio that can be achieved by some algorithm $\mathcal{A}$ on all possible inputs from this problem class.

3 Weighted Stars

On every weighted star metric, there is one root node to which every leaf node (state) is connected with a single edge of non-negative weight. Let d_s denote the weight of the edge between state s and the root, then the distance between two different states s_1 and s_2 is $d(s_1, s_2) = d_{s_1} + d_{s_2}$, where d_{s_1} can be charged as the cost of leaving state s_1 and d_{s_2} can be charged as that of entering state s_2 if the server moves from s_1 to s_2.

A common practice on weighted stars is that instead of charging the cost of both leaving s_1 and entering s_2, we only need to consider the cost of entering the new state at every step, i.e. we can regard the cost of moving from s_1 to a different state s_2 as only d_{s_2}, for both the algorithmic cost and the optimal solution. Over the entire process, the number of times that the server leaves any state is the same as the number of times that the server enters the same state, except for the initial and the last state. Now we formally show that this setting will only lose the competitive ratio of the original problem setting by a factor of at most 2.

▶ **Lemma 3.1.** *Every strictly c-competitive algorithm for the problem setting under the new transition costs has strict competitive ratio at most $2c$ in the original setting.*

Proof. Suppose that the initial state of the server is s_0, the algorithm moves to states $s_1, \ldots, s_n$ by the chronological order during its execution, and some fixed solution moves to states $s'_1, \ldots, s'_m$ one by one. By the fact that the algorithm achieves strict competitive ratio c under the new setting, it holds that $\sum_{j=1}^{n} d_{s_j} \leq c \sum_{j=1}^{m} d_{s'_j}$. Then we have $d_{s_0} + 2\sum_{j=1}^{n-1} d_{s_j} + d_{s_n} \leq d_{s_0} + 2\sum_{j=1}^{n} d_{s_j} \leq d_{s_0} + 2c\sum_{j=1}^{m} d_{s'_j} \leq 2c(d_{s_0} + 2\sum_{j=1}^{m-1} d_{s'_j} + d_{s'_m})$, which suggests that the same algorithm achieves strict competitive ratio $2c$ in the original setting by the arbitrariness of the solution that we are comparing the algorithmic cost with. ◀

State labels. In this setting, we do not need to consider the initial state. Moreover, each state can be labeled with a subset of $[m] = \{1, 2, \ldots, m\}$, which implies the set of request types that are feasible at this position. To formalize, given the label $S \subseteq [m]$ of some state s and $\{K_1, \ldots, K_m\}$ of the parametrized MSS setting of m possible request types as subsets of the metric space, $S = \{i \mid s \in K_i\}$. We may use the notations of a state s and its label S interchangeably, as well as a set of natural numbers and the set of request types K_i's whose indices are in this set.

Assumptions. It suffices to consider $2^m - 1$ states that correspond to $2^m - 1$ different non-empty subsets of $[m]$, since if there are at least two states with exactly the same label, then any reasonable algorithm should only visit the one whose edge to the node has the minimum weight. For every non-empty $S \subseteq [m] = \{1, 2, \ldots, m\}$, we use d_S to denote the minimum edge weight between the root and any node whose label is exactly equal to S. If no such node exists, just set $d_S = \infty$.

We now make another assumption, which is without loss of generality, yet important for the validity of the algorithm that we are going to propose.

▶ **Assumption 3.2.** *$d_S \leq d_{S'}$ for every $S \subseteq S'$.*

Otherwise, if there exist two different non-empty subsets S and S' of $[m]$ such that $S \subseteq S'$ yet $d_S > d_{S'}$, then every visit to state S can be replaced by a visit to state S', maintaining the feasibility because $S' \supseteq S$ while only reducing the cost. In this case, we can set d_S to $d_{S'}$ instead, which does not increase the optimal cost since we are only reducing edge lengths, while the algorithmic solution can replace each visit to S with one visit to S' in the original setting, which means that every algorithm can be transformed into one whose cost in the original setting is no more than that under this assumption.

We make one final assumption. Against any deterministic algorithm, one can assume that the adversary always forces the algorithm to move to a different state at every step as long as there is some K_i such that the current state of the algorithm is not in K_i, and the input ends when the current state is in the intersection of all K_i's for $1 \leq i \leq m$, if such a scenario exists. A straightforward consequence of this assumption on our problem setting is that any two consecutive requests from the input sequence are of different types, but we will eventually use a stronger argument that also exploits the algorithmic decisions seen by the adversary in the deterministic model to bound the desired competitive ratio.

Integer programming formulation. Under the input sequence $\{\rho_t\}_{t \geq 1}$, we use $S_{i,j}$ to denote the set of request types that appear in the subsequence $\rho_i, \ldots, \rho_j$, i.e. $S_{i,j} = \{\ell \mid K_\ell = \rho_t \text{ for some } i \leq t \leq j\}$, which also corresponds to a unique node of the weighted star in our

notation. Then we can write down an integer programming formulation of the problem whose minimum objective function value is a lower bound of the optimal solution of the original problem. We use T to express the number of steps of the whole input sequence.

$$\begin{aligned} \text{minimize} \quad & \sum_{1\le i\le j\le T} d_{S_{i,j}} x_{i,j} \\ \text{subject to} \quad & \sum_{i\le t\le j} x_{i,j} \ge 1, \quad \forall 1 \le t \le T \\ & x_{i,j} \in \{0,1\}, \quad \forall 1 \le i \le j \le T \end{aligned} \tag{1}$$

▶ **Lemma 3.3.** *The minimum value of (1) is no more than the optimal value of the parametrized MSS problem on the weighted star in our setting.*

Proof. Given any solution of the parametrized MSS problem on the weighted star, for every state that the server stays at from step i to j, we can just assume that it stays at state $S_{i,j}$ due to Assumption 3.2, otherwise the cost can only be higher. We set $x_{i,j} = 1$ for every such state that the server visits, then this is a feasible solution of (1), and its value is the same as the given solution of the parametrized MSS on the weighted star instance. ◀

The next step is to write down the dual program of the linear relaxation of (1), whose maximum value is no more than the minimum value of its primal linear program and thus the minimum value of the original program (1) as well:

$$\begin{aligned} \text{maximize} \quad & \sum_{1\le t\le T} y_t \\ \text{subject to} \quad & \sum_{t:i\le t\le j} y_t \le d_{S_{i,j}}, \forall 1 \le i \le j \le T \\ & y_t \ge 0, \quad \forall 1 \le t \le T \end{aligned} \tag{2}$$

Primal-dual algorithm. We are now ready to state our primal-dual algorithm. From a high level, our algorithm maintains a feasible dual solution and at every step t: increases the dual variable y_t until some dual constraint becomes tight, and then decides which state to move to depending on the tight dual constraint.

Now we come to its formal description. Upon the arrival of request ρ_t, increase the value of the dual variable y_t from the initial value 0 until some dual constraint is tight, i.e.

$$\sum_{\ell=i}^{t} y_\ell = d_{S_{i,t}}$$

for some $i \le \ell \le t$; or equivalently, we can write the value as

$$y_t = \min_{1\le i\le t} \left\{ d_{S_{i,t}} - \sum_{\ell=i}^{t-1} y_\ell \right\}, \tag{3}$$

where the value of $\sum_{\ell=i}^{t-1} y_\ell$ is defined to be zero for $i = t$.

If the tight dual constraint is $\sum_{\ell=i}^{t} y_\ell = d_{S_{i,t}}$, then the algorithm moves the server to the state labeled with $S_{i,t}$ (the one with the minimum edge weight to the root based on our previous assumption), to process request ρ_t. Ties are broken in an arbitrary way.

Algorithm analysis. We first prove that the algorithm indeed maintains a feasible dual solution.

▶ **Lemma 3.4.** *The algorithm maintains a feasible dual solution.*

Proof. At every step t, a new dual variable y_t whose initial value is zero appears with a set of new dual constraints, and we need to show that these constraints are not violated when $y_t = 0$, so that it is feasible to increase the value of y_t from zero until some dual constraint becomes tight.

It suffices to prove that $\sum_{\ell=i}^{t-1} y_\ell \le d_{S_{i,t}}$ for every $i \le t$, given the values y_1 to y_{t-1} well-defined from previous steps. The case that $i = t$ is trivial, since $d_{S_{i,t}}$ is non-negative based on the problem definition. For every $1 \le i \le t-1$, it holds that $\sum_{\ell=i}^{t-1} y_\ell \le d_{S_{i,t-1}}$ given that we maintain a feasible solution for dual variables and constraints that have appeared until step $t-1$, and the result follows by Assumption 3.2 and the fact that $S_{i,t-1} \subseteq S_{i,t}$. ◀

Now we are ready to use values of y_t's that the algorithm computes under its execution to provide an upper bound of its cost.

▶ **Lemma 3.5.** *The cost of the algorithm is at most* $m \sum_{\ell=1}^{T} y_\ell$.

Proof. When the algorithm moves to state $S_{i,t}$ at step t, $S_{i,t}$ is the set of request types in some subsequence $\rho_i, \ldots, \rho_t$ and the values of dual variables assigned by the algorithm so far satisfy that $\sum_{\ell=i}^{t} y_\ell = d_{S_{i,t}}$, since the algorithm chooses state $S_{i,t}$ to move to according to the tight dual constraint at every step. As a result, the cost of moving to state $S_{i,t}$ at every step t can be charged to some y_ℓ's. Now we show that every y_ℓ is only charged to at no more than m different steps t and their corresponding states $S_{i,t}$'s, which means that we need to show there are at most m pairs of (i,t)'s such that $i \le \ell \le t$ and the algorithm moves to state $S_{i,t}$ at step t.

Suppose there are two different steps $t_1 < t_2$ such that the algorithm moves to state S_{i_1,t_1} at step t_1 and state S_{i_2,t_2} at step t_2, and $i_1 \le \ell \le t_1$ and $i_2 \le \ell \le t_2$. The subsequence $\rho_{t_1+1}, \ldots, \rho_{t_2}$ contains at least one request that is not in set S_{i_1,t_1} – namely, ρ_{t_1+1} – as otherwise the algorithm has no incentive to move to a different state. As a result, since $S_{t_1+1,t_2} \subseteq S_{\ell,t_2}$ and $S_{\ell,t_1} \subseteq S_{i_1,t_1}$, we get $S_{\ell,t_2} \setminus S_{\ell,t_1} \supseteq S_{t_1+1,t_2} \setminus S_{i_1,t_1} \neq \varnothing$, and hence $|S_{\ell,t_2}| \ge |S_{\ell,t_1}| + 1$ because $S_{\ell,t_1} \subseteq S_{\ell,t_2}$. Now if we have in total k such different steps $\ell \le t_1 < \ldots < t_k$, then it holds that $|S_{\ell,t_k}| \ge k$, which directly implies that $k \le m$. ◀

Lemmas 3.1 and 3.5 and the fact that the maximum value of the linear dual program (2) is no more than the minimum value of the original program (1) imply Theorem 1.2.

4 Randomized Competitive Ratio on HSTs

In this section, we consider hierarchically separated trees (HSTs), a highly symmetric rooted tree structure where the edge weights between a node and all its children are equal. The path between one internal node to all leaves in the subtree rooted at this node has the same number of edges, meaning that the level of a node as the number of edges from it to any of its descendant leaf is well-defined (the level of every leaf node is 0). Moreover, the edge weight between an internal node and all its children is a function of the node level. Since we regard all leaf nodes as the points in the metric space induced by a rooted tree, the distance between two points is a function of the level of their lowest common ancestor. We say that an HST is a k-level HST if its root is at level k.

The bulk of this section is devoted to proving our lower bounds in Section 4.1. We finish with the tight upper bound for 2-level HSTs in Section 4.2.

4.1 Iterative Lower Bounds

The key ingredient for our improved lower bounds is a new construction for uniform metrics (i.e. 1-level HSTs). Thus, we begin in Section 4.1.1 by giving the construction and analysis for uniform metrics. Then, we show how to lift our ideas to construct higher level HSTs in Section 4.1.2. In Section 4.1.3, we prove our improved lower bound $c_{2,m} = \Omega(\binom{m}{\lceil m/2 \rceil})$ on 2-level HSTs. Finally, we prove our lower bounds for higher level HSTs.

Iterative sequence. We first introduce the following iterative sequence which will be used to express the constructions of hard instances and the lower bounds:

$$A_{1,m} = \binom{m}{\lceil m/2 \rceil}, \qquad A_{k,m} = \binom{A_{k-1,m}}{\lceil A_{k-1,m}/2 \rceil} \quad \text{for } k \geq 2. \tag{4}$$

▶ **Proposition 4.1.** *For every fixed $m \geq 4$, we have $A_{1,m} > m$ and so the sequence $\{A_{k,m}\}_{k \geq 1}$ is unbounded.*

Labels and requests. In this section, the requests are defined by assigning a label $S \subseteq [m]$ to each leaf node. The request type K_i (as a feasible set) is then defined to be the leaf nodes whose labels contain i, i.e. request type K_i is served by moving to a node v whose label S contains i. We may use number i to refer to request type K_i when the context is clear, especially when we use the corresponding subset of $[m]$ to refer to a set of request types.

Randomized lower bounds. To prove a lower bound of α on the competitive ratio of randomized online algorithms, we use Yao's minimax principle [17], and give a distribution over input instances such that every deterministic algorithm has expected competitive ratio at least α.

In the following, we will only describe and analyze a randomized input sequence whose length is a function of k and m. This process can be repeated to generate arbitrarily many rounds and hence the request sequence can be arbitrarily long and have arbitrarily large cost.

4.1.1 Warm-up: Uniform Metrics

We begin with a warm-up on a 1-level HST (uniform metric) which we will lift to higher level HSTs.

Lower bound instance. The key property of our 1-level HST is that for every request sequence that is a permutation of $[m]$, there is a leaf that can serve the first $\lceil \frac{m}{2} \rceil$ requests and another leaf that can serve the remaining requests, and thus there is always a solution that moves at most twice. Define the 1-level HST structure $\mathcal{T}_{1,m}$ as follows: it has $\binom{m}{\lceil m/2 \rceil} = A_{1,m}$ leaves, each labeled with a $\lceil \frac{m}{2} \rceil$-size subset of $[m]$ and the distance between every two leaves is 1. The adversarial request sequence is a uniform random permutation of $[m]$.

Before we start our analysis, we prove the following proposition which will be useful in bounding the expected cost of the algorithm.

▶ **Proposition 4.2.** *There exists a universal constant $\kappa > 0$ such that for every $M \geq 2$, we have*

$$\sum_{t=1}^{\lfloor M/2 \rfloor} \frac{\lfloor M/2 \rfloor - t + 1}{M - t + 1} \geq \kappa \cdot M.$$

Proof. We have

$$\begin{aligned}\sum_{t=1}^{\lfloor M/2\rfloor}\frac{\lfloor M/2\rfloor-t+1}{M-t+1} &= \sum_{t=1}^{\lfloor M/2\rfloor}\frac{(M-t+1)-\lceil M/2\rceil}{M-t+1}\\ &= \lfloor M/2\rfloor - \lfloor M/2\rfloor\sum_{i=\lceil M/2\rceil+1}^{M}\frac{1}{i}\\ &= \lfloor M/2\rfloor - \lceil M/2\rceil(H_M - H_{\lceil M/2\rceil})\\ &= \lfloor M/2\rfloor - \lceil M/2\rceil(\ln 2 + o(1)) = \frac{1-\ln 2}{2}M(1+o(1)),\end{aligned}$$

where H_i is the ith harmonic number. ◀

▶ **Theorem 4.3.** *The expected competitive ratio of every deterministic algorithm against the uniform random permutation of m request types on $\mathcal{T}_{1,m}$ is $\Omega(m)$.*

Proof. First, we show that there is a solution with cost at most 2. The whole sequence $\rho_1,\ldots,\rho_m$ has each of the possible request types $K_1,\ldots,K_m$ exactly once. The solution processes requests $\rho_1,\ldots,\rho_{\lceil m/2\rceil}$ at the state labeled by the set of all indices of request types in $\rho_1,\ldots,\rho_{\lceil m/2\rceil}$, i.e. at $S=\{i \mid K_i \text{ is in subsequence } \rho_1,\ldots,\rho_{\lceil m/2\rceil}\}$, and then moves to the state labeled by the set of all indices of request types in the subsequence $\rho_{\lfloor m/2\rfloor+1},\ldots,\rho_m$. The total cost of this solution is at most 2, no matter which initial state the server is in.

We now lower bound the expected cost of every deterministic algorithm. Fix a deterministic algorithm. For each step t, let R_t be the set of request types that have not appeared prior to this step and S_{t-1} denote the label of the state of the algorithm right before the arrival of ρ_t. It holds that $|R_t| = m-t+1$ and $|S_{t-1}| = \lceil m/2\rceil$, and hence $|R_t \setminus S_{t-1}| \geq \lfloor m/2\rfloor - t + 1$. Let Y_t denote the event that the algorithm has to move at step t (i.e. because $\rho_t \notin S_{t-1}$). Since ρ_t is chosen u.a.r. from R_t, we have $\mathbb{E}[Y_t] \geq (\lfloor m/2\rfloor - t + 1)/(m-t+1)$ for every $1 \leq t \leq \lfloor m/2\rfloor$. For $\lfloor m/2\rfloor + 1 \leq t \leq m$, we use the trivial bound $\mathbb{E}[Y_t] \geq 0$. Thus, the total expected cost of the deterministic algorithm is

$$\sum_{t=1}^{m}\mathbb{E}[Y_t] \geq \sum_{t=1}^{\lfloor m/2\rfloor}\frac{\lfloor m/2\rfloor-t+1}{m-t+1} = \Omega(m),$$

by Proposition 4.2. ◀

4.1.2 Lifting to Higher Levels

For every k and m, we construct an HST $\mathcal{T}_{k,m}$ to lower bound $c_{k,m}$.

Key property of $\mathcal{T}_{k,m}$. The key property of $\mathcal{T}_{1,m}$ and the distribution of request sequences is that for every request sequence in the support of the distribution, there is a leaf that can serve the first half of the requests, and another leaf that can serve the second half. Thus, there is a solution that only moves twice. We generalize this to $\mathcal{T}_{k,m}$ as follows. The distribution of request sequences and $\mathcal{T}_{k,m}$ have the property that for every sequence in the support of the distribution, there are two "safe" $(k-1)$-level subtrees such that there is a low-cost solution that can serve the first half of the request sequence in one of them and the second half in the other. In other words, the low-cost solution only moves twice between $(k-1)$-level subtrees.

Higher-level labels. In our construction of $\mathcal{T}_{k,m}$, we will assign labels to all nodes except the root. The labels for leaves come from the set $\mathcal{P}_0 := \binom{[m]}{\lceil m/2 \rceil}$[4]. The labels for ℓ-level nodes are collections of those for $(\ell-1)$-level nodes as follows: Let $\mathcal{P}_\ell$ be the set of ℓ-level labels, then $\mathcal{P}_\ell := \binom{\mathcal{P}_{\ell-1}}{\lceil |\mathcal{P}_{\ell-1}|/2 \rceil}$ for every $1 \le \ell \le k-1$. It is easy to see that $|\mathcal{P}_\ell| = A_{\ell+1,m}$. We also define the label of a subtree to be the label of its root node.

Construction of $\mathcal{T}_{k,m}$. For $k \ge 2$, the HST $\mathcal{T}_{k,m}$ is constructed iteratively in a top-down fashion as follows. For every label $P \in \mathcal{P}_{k-1}$, the root of the HST has a child labeled P. For every level $k-1 \ge \ell \ge 1$, every level-ℓ node v has $|P(v)|$ children where $P(v)$ is its label, each given a unique label $p \in P(v)$.[5] We remark that there is exactly one level-$(k-1)$ node per label in $\mathcal{P}_{k-1}$ but for $\ell < k-1$, there are several level-ℓ nodes from different $(\ell+1)$-level subtrees that share the same label. For every $1 \le \ell \le k$, the distance between every two leaves whose lowest common ancestor is on level ℓ is $C_{\ell-1}$[6] ($C_0 = 1$), where C_ℓ's depend on k and m and will be defined later in Section 4.1.4.

▶ **Observation 4.4.** *In $\mathcal{T}_{k,m}$, every ℓ-level node has exactly $\lceil A_{\ell,m}/2 \rceil$ level-$(\ell-1)$ children with distinct labels, for every $1 \le \ell \le k-1$.*

4.1.3 Lower Bound Sequence for $\mathcal{T}_{2,m}$

We now prove our lower bound of $c_{2,m}$. Recall that C_1 is the distance between two leaves in different 1-level subtrees, and $\mathcal{P}_0$ is the set of all $\lceil m/2 \rceil$-size subsets of $[m]$. For each label $S \in \mathcal{P}_0$, define $\boldsymbol{\rho}(S)$ to be an arbitrary permutation of S.

Now we describe the adversarial randomized input sequence. The adversary first samples a uniformly random permutation $\{S_t\}_{t=1}^{|\mathcal{P}_0|}$ of $\mathcal{P}_0$ and then proceeds in $|\mathcal{P}_0| = A_{1,m}$ phases as follows: in phase t, it repeats the sequence $\boldsymbol{\rho}(S_t)$ for C_1 times. Thus, the entire input is a concatenation of all those phases:

$$\underbrace{\boldsymbol{\rho}(S_1), \ldots, \boldsymbol{\rho}(S_1)}_{C_1 \text{ times}}, \underbrace{\boldsymbol{\rho}(S_2), \ldots, \boldsymbol{\rho}(S_2)}_{C_1 \text{ times}}, \ldots, \underbrace{\boldsymbol{\rho}(S_{|\mathcal{P}_0|}), \ldots, \boldsymbol{\rho}(S_{|\mathcal{P}_0|})}_{C_1 \text{ times}}.$$

For every $1 \le t \le A_{1,m}$, we use T^1_{t-1} to denote the label of the 1-level subtree in which the state of the algorithm is located right before the beginning of the phase t.

▶ **Lemma 4.5.** *For every C_1, every deterministic algorithm has expected cost of at least $\Omega(A_{1,m}C_1)$.*

Proof. We first show that: the cost incurred during the whole phase t is at least C_1 if T^1_{t-1} does not contain S_t. If $S_t \notin T^1_{t-1}$, either the algorithm moves to another 1-level subtree that contains S_t as a leaf node, or the algorithm needs to move at least once during every loop of $\boldsymbol{\rho}(S_t)$ since there is no state in T^1_{t-1} that can satisfy all request types in S_t. In either way, the cost is at least C_1.

Now we consider the probability of the event that T^1_{t-1} does not contain S_t. Similar to the proof of Theorem 4.3, at every phase $1 \le t \le \lfloor A_{1,m}/2 \rfloor$, the probability that T^1_{t-1} does not contain a node labeled S_t is at least $(\lfloor A_{1,m}/2 \rfloor - t + 1)/(A_{1,m} - t + 1)$ because T^1_{t-1} contains exactly $\lceil A_{1,m}/2 \rceil$ different leaf labels. So, Proposition 4.2 implies that the total expected cost incurred during the first $\lfloor A_{1,m}/2 \rfloor$ phases is $\Omega(A_{1,m}C_1)$. ◀

---

[4] The notation $\binom{S}{k}$ means the collection of all k-size subsets of S.
[5] Recall that $P(v)$, a label of a level-ℓ node, is a collection of level-$(\ell-1)$ labels.
[6] We always assume that $C_\ell/C_{\ell-1}$ is an integer for every $\ell \ge 1$.

▶ **Lemma 4.6.** *For every C_1, the optimal offline cost is at most $2C_1 + A_{1,m}$ no matter where the initial state is.*

Proof. Consider the following solution. It moves to the subtree labeled $\{S_1, \ldots, S_{\lceil A_{1,m}/2 \rceil}\}$ and for phases $1 \le t \le \lfloor A_{1,m}/2 \rfloor$, it serves the repetitions of the subsequence $\boldsymbol{\rho}(S_t)$ at the node labeled S_t in the subtree. Then, it moves to the subtree labeled $\{S_{\lfloor A_{1,m}/2 \rfloor + 1}, \ldots, S_{A_{1,m}}\}$ and for $\lfloor A_{1,m}/2 \rfloor < t \le A_{1,m}$, serves the repetitions of the subsequence $\boldsymbol{\rho}(S_t)$ at the node labeled S_t in the subtree. It incurs a cost of at most $2C_1$ for movement between different 1-level subtrees and at most $A_{1,m}$ for movement within each of the two 1-level subtrees. Thus, the total cost is at most $2C_1 + A_{1,m}$. ◀

Since the two lemmata that we have just proven hold for every value C_1, we can set it to be some value such that $C_1 = \omega(A_{1,m})$ with both values regarded as functions of m, and the desired lower bound immediately follows.

▶ **Theorem 4.7.** *Every randomized algorithm has a worst case competitive ratio $\Omega(A_{1,m})$ on some instances of 2-level HSTs.*

4.1.4 General Iterative Lower Bounds on Higher Levels

We now prove the lower bound on $c_{k,m}$ for $k \ge 3$ expressed with the iterative sequence $\{A_{k,m}\}_{k \ge 1}$ defined in Equation (4) at the beginning of Section 4.1.

▶ **Theorem 4.8.** *$c_{k,m} = \Omega(A_{k-1,m})$ for every $k \ge 3$.*

Corollary 1.4 now follows from Theorem 4.8 and Proposition 4.1.

Sequences of labels. We have provided the construction of $\mathcal{T}_{k,m}$ for every k, and now we need to describe the adversary's strategy to generate a randomized input sequence on which no deterministic algorithm achieves competitive ratio better than $c_{k,m}$. Recall that every sequence in the support of the randomized input on $\mathcal{T}_{2,m}$ is a concatenation of sequences $\boldsymbol{\rho}(S)$ for labels $S \in \mathcal{P}_0$. Now we want to generalize this concept, the sequence of a label, to every level-ℓ label (i.e. every element of $\mathcal{P}_l$).

We define inductively on ℓ with $\ell = 0$ as the base case. For every $\ell \ge 1$ and every label $T^\ell \in \mathcal{P}_\ell$, the sequence $\boldsymbol{\rho}(T^\ell)$ is constructed in $|T^\ell|$ phases as follows. Let $T_1^{\ell-1}, \ldots, T_{|T^\ell|}^{\ell-1}$ be an arbitrary permutation of T^ℓ. During each phase $1 \le t \le |T^\ell|$, the sequence $\boldsymbol{\rho}(T^{\ell-1})$ is repeated consecutively for $\frac{C_\ell}{C_{\ell-1}}$ (we can choose values of C_ℓ's to make this amount always an integer at the end) times, and the entire sequence $\boldsymbol{\rho}(T^\ell)$ is a concatenation of those phases:

$$\underbrace{\boldsymbol{\rho}(T_1^{\ell-1}), \ldots, \boldsymbol{\rho}(T_1^{\ell-1})}_{\frac{C_\ell}{C_{\ell-1}} \text{ times}}, \underbrace{\boldsymbol{\rho}(T_2^{\ell-1}), \ldots, \boldsymbol{\rho}(T_2^{\ell-1})}_{\frac{C_\ell}{C_{\ell-1}} \text{ times}}, \ldots, \underbrace{\boldsymbol{\rho}(T_{|T^\ell|}^{\ell-1}), \ldots, \boldsymbol{\rho}(T_{|T^\ell|}^{\ell-1})}_{\frac{C_\ell}{C_{\ell-1}} \text{ times}}.$$

▶ **Lemma 4.9.** *Let $1 \le \ell \le k-2$ and $P \in \mathcal{P}_\ell$. Consider a solution for $\boldsymbol{\rho}(P)$ whose initial state is not in an ℓ-level subtree with label P. Then, the solution incurs a cost of at least C_ℓ on $\boldsymbol{\rho}(P)$.*

Proof. We prove by induction on ℓ. The base case is when $\ell = 1$. Fix some $P \in \mathcal{P}_1$. The sequence $\boldsymbol{\rho}(P)$ consists of C_1 consecutive repetitions of the sequence $\boldsymbol{\rho}(p)$ for every child $p \in P$. If the initial state is in a 1-level subtree labeled $T^1 \ne P$, then there exists some $p \in P$ such that $p \notin T^1$ and thus there is no leaf in the subtree labeled p. As a result, the algorithm either visits a different 1-level subtree, or incurs cost 1 during every repetition of $\boldsymbol{\rho}(p)$, thus costing at least C_1.

Suppose the claim holds for every sequence $\boldsymbol{\rho}(p)$ such that $p \in \mathcal{P}_\ell$ $(1 \le \ell \le k-3)$. Now consider a sequence $\boldsymbol{\rho}(P)$ for some $P \in \mathcal{P}_{\ell+1}$. For every $p \in P$, it contains $\frac{C_{\ell+1}}{C_\ell}$ consecutive repetitions of $\boldsymbol{\rho}(p)$. If the initial state is in an $(\ell+1)$-level subtree labeled $T^{\ell+1} \neq P$, then there exists $p \in P \setminus T^{\ell+1}$ because $|T^{\ell+1}| = |P|$. If the algorithm moves to a different $(\ell+1)$-level subtree, it incurs a cost of $C_{\ell+1}$. Now suppose the algorithm does not move to a different $(\ell+1)$-level subtree. Since $p \notin T_{\ell+1}$, the current subtree does not have an ℓ-level subtree labeled p. Thus at the start of each of the $C_{\ell+1}/C_\ell$ repetitions of $\boldsymbol{\rho}(p)$, by the inductive hypothesis, it incurs a cost of at least C_ℓ. The total cost is thus at least $C_{\ell+1}$. ◀

We now define another iterative sequence $\{B_\ell\}_{1 \le \ell \le k-1}$ and show that it can be used to express the upper bound of the optimal solution of the request sequences:

$$B_1 = \left\lceil \frac{A_{1,m}}{2} \right\rceil, \qquad B_\ell = \left\lceil \frac{A_{\ell,m}}{2} \right\rceil \left[(B_{\ell-1} + C_{\ell-2}) \cdot \frac{C_\ell}{C_{\ell-1}} + C_{\ell-1} \right] \quad \text{for } 2 \le \ell \le k-1.$$

▶ **Lemma 4.10.** *For every label T^ℓ in $\mathcal{P}_\ell$ and subtree labeled T^ℓ in $\mathcal{T}_{k,m}$, there exists a trajectory contained within the subtree that incurs cost of no more than B_ℓ to serve the sequence $\boldsymbol{\rho}(T^\ell)$, for every $1 \le \ell \le k-1$.*

Proof. We prove by induction on ℓ. Consider the base case $\ell = 1$. Fix some $T^1 \in \mathcal{P}_1$. Order the elements $S_1, \ldots, S_{\lceil A_{1,m}/2 \rceil}$ of T in the same order as the phases in $\boldsymbol{\rho}(T^1)$. Consider the trajectory that first moves to the leaf node labeled S_1 in the 1-level subtree labeled T^1 and then moves to leaves $S_2, \ldots, S_{\lceil A_{1,m}/2 \rceil}$ one by one with cost 1 each time since they are all in the same 1-level subtree. The total cost is at most $\left\lceil \frac{A_{1,m}}{2} \right\rceil = B_1$.

Now we consider $\boldsymbol{\rho}(T^\ell)$ for $2 \le \ell \le k-1$, given that the property for $\ell-1$ has been proven. Suppose that $\boldsymbol{\rho}(T^\ell)$ is the concatenation of $\boldsymbol{\rho}(T_1^{\ell-1}), \ldots, \boldsymbol{\rho}(T^{\ell-1}_{\lceil A_{\ell,m}/2 \rceil})$ by the chronological order where each $\boldsymbol{\rho}(T_j^{\ell-1})$ is repeated for $\frac{C_\ell}{C_{\ell-1}}$ times before proceeding to the next one. Consider the trajectory that for each phase $1 \le j \le \lceil A_{\ell,m}/2 \rceil$, consists of $\frac{C_\ell}{C_{\ell-1}}$ copies of the trajectory obtained by applying the inductive hypothesis to $\boldsymbol{\rho}(T_j^{\ell-1})$. In each phase j, for each copy of $\boldsymbol{\rho}(T_j^{\ell-1})$, the trajectory incurs a cost of at most $C_{\ell-2}$ to move to its initial state within the subtree with label $T_j^{\ell-1}$ and then a cost of $B_{\ell-1}$ to serve $\boldsymbol{\rho}(T_j^{\ell-1})$ within the same subtree. Thus, for each phase j, the trajectory's cost is at most $\frac{C_\ell}{C_{\ell-1}}(B_{\ell-1} + C_{\ell-2})$ plus at most $C_{\ell-1}$ to move to an $(\ell-1)$-level subtree labeled $T_j^{\ell-1}$ at the start of the phase. As there are $\lceil A_{\ell,m}/2 \rceil$ phases, the total cost of the trajectory is bounded by B_ℓ, as desired. ◀

Adversary's strategy and analysis. We are finally ready to describe the adversary's strategy on $\mathcal{T}_{k,m}$. It first samples a uniformly random permutation $\{T_t^{k-2}\}_{t=1}^{A_{k-1,m}}$ of $\mathcal{P}_{k-2}$ (note that $|\mathcal{P}_{k-2}| = A_{k-1,m}$), and then proceeds in $A_{k-1,m}$ phases: in each phase t, $\boldsymbol{\rho}(T_t^{k-2})$ is repeated consecutively for $\frac{C_{k-1}}{C_{k-2}}$ times, and the whole input is the concatenation of all $A_{k-1,m}$ phases.

▶ **Lemma 4.11.** *The optimal cost is no more than $2(C_{k-1} + B_{k-1})$ regardless of the initial state.*

Proof. The whole sequence can be regarded as the concatenation of the sequences $\boldsymbol{\rho}(T^{k-1})$'s of two $(k-1)$-level subtrees:

$$\{T_1^{k-2}, \ldots, T^{k-2}_{\lceil A_{k-1,m}/2 \rceil}\} \text{ and } \{T^{k-2}_{\lfloor A_{k-1,m}/2 \rfloor + 1}, \ldots, T^{k-2}_{A_{k-1,m}}\}.$$

The repetitions of $\boldsymbol{\rho}(T^{k-2}_{\lfloor A_{k-1,m}/2 \rfloor + 1})$ appear in both of the $(k-1)$-level subtrees' sequence when m is odd, but it does not affect our conclusion since we are proving an upper bound. As a result, the optimal solution just pays the cost at most B_{k-1} in each T^{k-1} (Lemma 4.10) plus the cost at most $2C_{k-1}$ to visit two $(k-1)$-level subtrees. ◀

▶ **Lemma 4.12.** *The expected cost of every deterministic algorithm to serve this randomized input on $\mathcal{T}_{k,m}$ is $\Omega(A_{k-1,m}C_{k-1})$.*

Proof. The proof is similar those of Theorem 4.3 and Lemma 4.5. Fix a deterministic algorithm. During every phase $1 \le t \le \lfloor A_{k-1,m}/2 \rfloor$ when some $\boldsymbol{\rho}(T_t^{k-2})$ is repeated for $\frac{C_{k-1}}{C_{k-2}}$ times, the probability that $\boldsymbol{\rho}(T_t^{k-2})$ is not in the current $(k-1)$-level subtree containing the algorithm's server has probability is at least $\frac{\lfloor A_{k-1,m}/2 \rfloor - t+1}{A_{k-1,m}-t+1}$. When this happens, the algorithm only has two choices: either move to a new $(k-1)$-level subtree with cost C_{k-1}, or incur cost of at least C_{k-2} during each repetition of $\boldsymbol{\rho}(T_t^{k-2})$ according to Lemma 4.9 since the initial state cannot be in a $(k-2)$-level subtree with label T_t^{k-2}. ◀

It remains to show that there exist possible values of C_ℓ's and B_ℓ's for $1 \le \ell \le k-1$ such that $C_{k-1} = \omega(B_{k-1})$ as m tends to infinity[7]. This would show that the randomized competitive ratio on $\mathcal{T}_{k,m}$ is at least $\Omega(\frac{A_{k-1,m}C_{k-1}}{C_{k-1}+B_{k-1}}) = \Omega(A_{k-1,m})$ as a function of m.

Let $C_1 = \omega(\prod_{j=1}^{k-1} A_{j,m})$ and $C_\ell = \omega((\prod_{j=\ell}^{k-1} A_{j,m})C_{\ell-1})$ for $2 \le \ell \le k-1$. We now prove that such a sequence satisfies the desired property.

▶ **Lemma 4.13.** $C_{k-1} = \omega(B_{k-1})$.

Proof. We prove that $C_\ell = \omega((\prod_{j=\ell+1}^{k-1} A_{j,m})B_\ell)$ as m tends to infinity for every $1 \le \ell \le k-1$ inductively, where $\prod_{j=k}^{k-1} A_{j,m}$ is defined to be 1.

Clearly the result holds for $\ell = 1$ because $B_1 = \Theta(A_{1,m})$. With the inductive hypothesis that $C_\ell = \omega((\prod_{j=\ell+1}^{k-1} A_{j,m})B_\ell)$ for some $1 \le \ell \le k-2$, then in order to prove that $C_{\ell+1} = \omega((\prod_{j=\ell+2}^{k-1} A_{j,m})B_{\ell+1})$, according to the definition that $B_{\ell+1} = \left\lceil \frac{A_{\ell+1,m}}{2} \right\rceil [(B_\ell + C_{\ell-1}) \cdot \frac{C_{\ell+1}}{C_\ell} + C_\ell]$, it suffices to show that the following two statements both hold:

$$C_{\ell+1} = \omega(A_{\ell+1,m} \cdot (\prod_{j=\ell+2}^{k-1} A_{j,m})C_\ell) = \omega(\prod_{j=\ell+1}^{k-1} A_{j,m}C_\ell), \tag{5}$$

which is exactly our definition of $C_{\ell+1}$, and

$$C_\ell = \omega((\prod_{j=\ell+1}^{k-1} A_{j,m})(B_\ell + C_{\ell-1})), \tag{6}$$

while (6) can be further reduced to that the following two formulae $C_\ell = \omega((\prod_{j=\ell+1}^{k-1} A_{j,m})B_\ell)$ and $C_\ell = \omega((\prod_{j=\ell+1}^{k-1} A_{j,m})C_{\ell-1})$ both hold, where the first one holds due to the inductive hypothesis and the second holds due to the definition of C_ℓ. ◀

This completes the proof of Theorem 4.8.

▶ Remark 4.14. Our iterative sequence of lower bounds on k-level HSTs marks an improvement from the one provided in Section 4.2 of [5] in the sense that: given every fixed $k \ge 2$, the lower bound provided by our sequence is asymptotically tighter than theirs when the parameter m tends to infinity. To formalize, consider their iterative sequence $\{\hat{c}_{k,m}\}_{k \ge 1}$ defined as the following: $\hat{c}_{1,m} = \lfloor m/2 \rfloor$ and $\hat{c}_{k,m} = 2^{\hat{c}_{k-1,m}-1}$. We can use inductive arguments on k to show that, $A_{k,m} = \omega(\hat{c}_{k,m})$ for every $k \ge 2$: it is clear that the statement

[7] Here both C_{k-1} and B_{k-1} are functions of m. When we consider both m and k as constants, the lower bound can be achieved by setting the ratio $C_\ell/C_{\ell-1}$ sufficiently large for every $1 \le \ell \le k-1$ and the proof idea is similar.

holds for $k = 2$, since $A_{2,m} = \binom{m}{\lceil m/2 \rceil} = \Theta(2^m/\sqrt{m})$ and $\hat{c}_{2,m} = \Theta(2^{m/2})$; suppose that $\lim_{m\to\infty} A_{k-1,m}/\hat{c}_{k-1,m} = \infty$, then we show that $\lim_{m\to\infty} \log(A_{k,m}/\hat{c}_{k,m}) = \infty$ and hence $\lim_{m\to\infty} A_{k,m}/\hat{c}_{k,m} = \infty$. To see this, $\log(A_{k,m}/\hat{c}_{k,m}) = \log A_{k,m} - \log \hat{c}_{k,m} = A_{k-1,m} - \log(A_{k-1,m})/2 - \hat{c}_{k-1,m} + O(1)$. Due to the inductive hypothesis, $\hat{c}_{k-1,m} = o(A_{k-1,m})$. Therefore, $A_{k-1,m} - \log(A_{k-1,m})/2 - \hat{c}_{k-1,m} \sim A_{k-1,m}$ and $\lim_{m\to\infty} \log(A_{k,m}/\hat{c}_{k,m}) = \infty$.

4.2 Tight Upper Bound for 2-Level HSTs

The idea to show a general upper bound for 2-level HSTs is as follows: even if there is no restriction on the size of the whole tree itself, some leaves and subtrees are redundant in the sense that any reasonable solution should not move to them, or does not need to move to them while maintaining feasibility and not increasing the cost. As a result, we can trim off those redundant nodes and hence reduce the graph size, just as how we show that we only need to consider $2^m - 1$ states for the weighted star case. This idea has also been applied by Bubeck and Rabani to give iterative upper bounds for 2 and higher levels of HSTs (see Section 4.2 Deeper Trees in [5]), and we present a refined argument on 2-level HSTs that closes the gap between upper and lower bounds. We show that the number of leaves remained after such preprocessing is bounded in terms of m and apply the optimal MTS algorithm for HSTs from [4].

Since the initial state can be arbitrarily given, in this part we only consider those leaves different from the initial state, to evaluate whether some states can be totally replaced by others and every algorithm can still maintain a feasible solution with equal or less cost. This is without loss of generality up to a constant factor, since there is only one initial state and thus it is sufficient to give an upper bound of the amount of states that are not the initial state.

In this part, we express leaf nodes lexicographically by a tuple (S, T) where S is the label of the leaf node itself and T is the label of the 1-level subtree that it is in. The label of every node is defined in the same way as the previous subsection[8] . A preliminary observation is that, if two leaf nodes share exactly the same tuple, then they are isomorphic and it suffices to only keep one in the tree. The feasibility of removing redundant nodes is that, when m possible requests and their corresponding feasible sets in the whole metric are fixed, running an algorithm over a subset of states produces a feasible solution for the whole metric as well, therefore what we need to show is that the optimal cost does not increase while we trim off some states.

▶ **Proposition 4.15.** *If there are two leaves (S_1, T) and (S_2, T) in the same 1-level subtree such that $S_1 \subseteq S_2$, then any move to (S_1, T) can be replaced with one to (S_2, T) without increasing the cost.*

Proof. Any request that is feasible at (S_1, T) is also feasible at (S_2, T) since $S_1 \subseteq S_2$. Moreover, for any state different from (S_1, T) and (S_2, T), it has the same distance from those two nodes that are within the same 1-level subtree T according to the 2-level HST structure. ◀

As a result, we can assume that every 1-level subtree is an antichain over subsets of $[m]$, which contains at most $\binom{m}{\lceil m/2 \rceil} = A_{1,m}$ elements according to Sperner's theorem [16]. The next thing is to consider the maximum amount of such 1-level subtrees, which is equivalent to the amount of antichains over subsets of $[m]$. This amount is called Dedekind number $D(m)$ and its following asymptotic approximation property has been proven in [12]:

[8] Recall that the label of a leaf node is the set of request types that this state can satisfy, and the label of a 1-level subtree is the set of labels of leaf nodes contained in it.

▶ **Lemma 4.16** ([12]). $\log D(m) = O(A_{1,m})$.

▶ **Theorem 4.17.** $c_{2,m} = O(A_{1,m})$.

Proof. It suffices to consider at most $O(D(m)A_{1,m})$ leaf nodes (including the initial state and at most $A_{1,m}$ other leaves in the same 1-level subtree as they cannot be pruned by the above process) and apply the optimal online MTS algorithm on an HST that has competitive ratio $O(\log n)$, where n is the number of points in the metric. According to the asymptotic property of $D(m)$ as shown in Lemma 4.16, $\log(D(m)A_{1,m}) = O(A_{1,m} + \log A_{1,m}) = O(A_{1,m})$. ◀

5 Open Problems

Following the initial study of parametrized metrical task systems (MTS) and metrical service systems (MSS) by Bubeck and Rabani [5], our research answers several open problems posed by Bubeck and Rabani. We provide a deterministic $O(m)$-competitive algorithm on weighted stars, tighten the gap between the upper and lower bounds of randomized competitive ratios on HSTs of different levels, and show that it is possible to design a strictly constant-competitive algorithm on general metrics when $m = 2$. However, multiple problems remain open: as Question 3 has been resolved on general metrics by showing the contrast between $m = 2$ and $m = 3$ cases, the focus is now on special classes of metrics, e.g., is it possible to obtain a finite competitive ratio for chasing 3 sets on HSTs or line metrics? Another interesting open problem is whether the $O(\binom{m}{\lceil m/2 \rceil})$ upper bound of randomized algorithms on 2-level HSTs can also be achieved by a deterministic algorithm. The progress on this problem will not only close the gap between the upper and lower bounds known for deterministic algorithms for this particular class of metrics, but also deepen our fundamental understanding of the parametrized MSS problem, as it concerns whether deterministic algorithms are essentially as powerful as randomized algorithms on 2-level HSTs and other classes of metrics. Moreover, the technique that we use in Section 3 is a linear programming primal-dual method that exploits the properties of parametrized MSS with m request types, and we wonder whether this framework can be generalized to prove the results of the more general parametrized MTS on uniform metrics (the tight deterministic bound of $\Theta(m \log(en/m))$ was proven by Bubeck and Rabani) or weighted stars (no result yet).

References

1 John Augustine, Sandy Irani, and Chaitanya Swamy. Optimal power-down strategies. *SIAM J. Comput.*, 37(5):1499–1516, 2008. doi:10.1137/05063787X.

2 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal online algorithm for metrical task systems. In Alfred V. Aho, editor, *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 373–382. ACM, 1987. doi:10.1145/28395.28435.

3 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 581–594. ACM, 2023. doi:10.1145/3564246.3585132.

4 Sébastien Bubeck, Michael B. Cohen, James R. Lee, and Yin Tat Lee. Metrical task systems on trees via mirror descent and unfair gluing. *SIAM J. Comput.*, 50(3):909–923, 2021. doi:10.1137/19M1237879.

5 Sébastien Bubeck and Yuval Rabani. Parametrized metrical task systems. In Jaroslaw Byrka and Raghu Meka, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2020, August 17-19, 2020, Virtual Conference*, volume 176 of *LIPIcs*, pages 54:1–54:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.54.

6 Niv Buchbinder and Joseph Naor. Online primal-dual algorithms for covering and packing. *Math. Oper. Res.*, 34(2):270–286, 2009. doi:10.1287/MOOR.1080.0363.

7 William R. Burley. Traversing layered graphs using the work function algorithm. *J. Algorithms*, 20(3):479–511, 1996. doi:10.1006/JAGM.1996.0024.

8 William R. Burley and Sandy Irani. On algorithm design for metrical task systems. *Algorithmica*, 18(4):461–485, 1997. doi:10.1007/PL00009166.

9 Marek Chrobak and Lawrence L. Larmore. The server problem and on-line games. In Lyle A. McGeoch and Daniel Dominic Sleator, editors, *On-Line Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, February 11-13, 1991*, volume 7 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 11–64. DIMACS/AMS, 1991. doi:10.1090/DIMACS/007/02.

10 Amos Fiat, Dean P. Foster, Howard J. Karloff, Yuval Rabani, Yiftach Ravid, and Sundar Vishwanathan. Competitive algorithms for layered graph traversal. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, October 1-4, 1991*, pages 288–297. IEEE Computer Society, 1991. doi:10.1109/SFCS.1991.185381.

11 Sandy Irani, Sandeep K. Shukla, and Rajesh K. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Trans. Embed. Comput. Syst.*, 2(3):325–346, 2003. doi:10.1145/860176.860180.

12 Daniel Kleitman and George Markowsky. On dedekind's problem: the number of isotone boolean functions. ii. *Transactions of the American Mathematical Society*, 213:373–390, 1975.

13 Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990. doi:10.1016/0196-6774(90)90003-W.

14 H. Ramesh. On traversing layered graphs on-line. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 412–421. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313843.

15 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985. doi:10.1145/2786.2793.

16 Emanuel Sperner. Ein satz über untermengen einer endlichen menge. *Mathematische Zeitschrift*, 27(1):544–548, 1928.

17 Andrew Chi-Chih Yao. Probabilistic computations: Toward a unified measure of complexity (extended abstract). In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 222–227. IEEE Computer Society, 1977. doi:10.1109/SFCS.1977.24.