The Randomized k-Server Conjecture is False![*]

Sébastien Bubeck
Microsoft Research
Redmond, WA, United States

Christian Coester
University of Oxford
Oxford, United Kingdom

Yuval Rabani[†]
The Hebrew University of Jerusalem
Jerusalem, Israel

ABSTRACT

We prove a few new lower bounds on the randomized competitive ratio for the k-server problem and other related problems, resolving some long-standing conjectures. In particular, for metrical task systems (MTS) we asympotically settle the competitive ratio and obtain the first improvement to an existential lower bound since the introduction of the model 35 years ago (in 1987).

More concretely, we show:

(1) There exist $(k+1)$-point metric spaces in which the randomized competitive ratio for the k-server problem is $\Omega(\log^2 k)$. This refutes the folklore conjecture (which is known to hold in some families of metrics) that in all metric spaces with at least $k+1$ points, the competitive ratio is $\Theta(\log k)$.
(2) Consequently, there exist n-point metric spaces in which the randomized competitive ratio for MTS is $\Omega(\log^2 n)$. This matches the upper bound that holds for all metrics. The previously best existential lower bound was $\Omega(\log n)$ (which was known to be tight for some families of metrics).
(3) For all $k < n \in \mathbb{N}$, for *all* n-point metric spaces the randomized k-server competitive ratio is at least $\Omega(\log k)$, and consequently the randomized MTS competitive ratio is at least $\Omega(\log n)$. These universal lower bounds are asymptotically tight. The previous bounds were $\Omega(\log k/\log\log k)$ and $\Omega(\log n/\log\log n)$, respectively.
(4) The randomized competitive ratio for the w-set metrical service systems problem, and its equivalent width-w layered graph traversal problem, is $\Omega(w^2)$. This slightly improves the previous lower bound and matches the recently discovered upper bound.
(5) Our results imply improved lower bounds for other problems like k-taxi, distributed paging, and metric allocation.

These lower bounds share a common thread, and other than the third bound, also a common construction.

CCS CONCEPTS

• **Theory of computation → K-server algorithms**;

KEYWORDS

online computing, competitive analysis, randomized algorithms, lower bounds, k-server, metrical task systems

ACM Reference Format:

Sébastien Bubeck, Christian Coester, and Yuval Rabani. 2023. The Randomized k-Server Conjecture is False!. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC '23), June 20–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3564246.3585132

Two roads diverged in a wood, and I—
I took the one less traveled by,
And that has made all the difference.
– Robert Frost

1 INTRODUCTION

Since its inception in [42], the k-server problem has been the driving challenge shaping research on competitive analysis of online algorithms, a computing paradigm pioneered in [49]. The problem is simple to state. (See Section 2 for the descriptions of all the problems discussed in the introduction.) There are compellingly beautiful conjectures regarding the competitive ratio for deterministic and randomized algorithms. The attempts, partly successful, to prove these conjectures have led to the development of powerful tools whose impact reaches far beyond the k-server problem, or online computing for that matter. These include the application of and the contribution to concepts such as the work function, quasi-convexity, online choice of online algorithms, hierarchically separated trees (HSTs), non-contracting Lipschitz maps of metric spaces, metric Ramsey theory, the online primal-dual schema, entropic regularization, and the online mirror descent schema.

In this paper, we refute the randomized k-server conjecture, which states that the randomized competitive ratio for the problem in *all* metric spaces (with at least $k+1$ points) is $\Theta(\log k)$. We construct, for all $k \in \mathbb{N}$, metric spaces on $k+1$ points in which the randomized competitive ratio for this problem is $\Omega(\log^2 k)$. This also implies stronger lower bounds for a number of related problems. In particular: (*i*) Metrical task systems were introduced in [14]. The k-server problem in a $(k+1)$-point metric space is a special case of the metrical task systems problem in the same metric space. Therefore, we get that there are, for all $n \in \mathbb{N}$, n-point metric spaces in which the randomized competitive ratio for the metrical task systems problem is $\Omega(\log^2 n)$. (*ii*) The k-taxi

[*] A complete version of this paper can be found at https://arxiv.org/abs/2211.05753.
[†] Research supported in part by ISF grants 3565-21 and 389-22, and by BSF grant 2018687.

STOC '23, June 20–23, 2023, Orlando, FL, USA

ACM ISBN 978-1-4503-9913-5/23/06…$15.00
https://doi.org/10.1145/3564246.3585132

problem was introduced in [29]. It is trivially at least as hard as the k-server problem in the same metric space. Hence, our k-server lower bounds carry over to this problem. (*iii*) The distributed paging (a.k.a. constrained file allocation) problem was introduced in [9]. In [2, Theorem 3.1] it was shown that the k-server problem in a $(k+1)$-point metric space is a special case of distributed paging in a network inducing the same metric on $k+1$ processors, with total capacity of the caches at the processors of m pages, and $m-k+1$ distinct pages in the system. Thus, we get that for all $n \in \mathbb{N}$, there exist n-processor networks in which the randomized competitive ratio for the distributed paging problem is $\Omega(\log^2 n)$. (*iv*) The metric allocation problem was introduced in [6]. The same paper shows that the randomized metrical task systems problem is a special case of this problem in the same metric space. Hence, our results imply an $\Omega(\log^2 n)$ lower bound for metric allocation, where n is the cardinality of the underlying metric space.

Moreover, the ideas behind the construction of the lower bound examples are inspired by a construction of lower bound examples for a different problem. Layered graph traversal (a.k.a. metrical service systems) was introduced in [45]. Here we show a lower bound of $\Omega(w^2)$ on the randomized competitive ratio for traversing width-w layered graphs. This improves upon the previous lower bound of $\Omega(w^2/\log^{1+\epsilon} w)$ (for all $\epsilon > 0$) of [46], and matches asymptotically the recent upper bound of [16]. It also implies the same asymptotically tight lower bound on the randomized competitive ratio for the depth-w evolving tree game, on account of the reduction used in [16] and the matching upper bound in that paper.

The results for the k-server and related problems are quite surprising. The new k-server lower bounds are asymptotically tight for $k+1$ point metric spaces, and more generally this is true of our metrical task systems lower bounds. Matching upper bounds appear in [17, 24]. Previously, it has been widely conjectured (see [3, 5, 12, 13, 27, 30, 34, 35, 37, 40, 48]) that in all metric spaces on more than k points, the randomized competitive ratio for the k-server problem is $\Theta(\log k)$. This was also established in some special cases [4, 27, 28]. The best known upper bounds for general metrics are $O(\log^2 k \log n)$ and $O(\log^3 k \log \Delta)$ [18], where n is the number of points in the metric space and Δ is the ratio between the largest and smallest non-zero distance, and $O(k)$ by using a deterministic algorithm [38] when both n and Δ are very large. Moreover, the deterministic k-server conjecture, which states that in every metric space on at least $k+1$ points the deterministic competitive ratio is exactly k, is nearly resolved. A lower bound of k is known to hold in all metrics [42]. It is tight in many special cases [11, 19, 20, 23, 30, 39, 42, 49], and it is within a factor of $2-\frac{1}{k}$ of the truth in all cases [38]. Thus, it was unexpected that in the randomized case the competitive ratio would vary widely among different metric spaces. Similarly, for metrical task systems, in some metric spaces an asymptotically tight $\Theta(\log n)$ bound is known. In fact, the upper bound holds in all HST metrics [17]; see also previous work in [7, 14, 28, 33, 47]. It has been conjectured that this is the truth in all metric spaces (similarly to the deterministic case, where all metric spaces are known to be equally hard with a competitive ratio of exactly $2n-1$ [14]). Obviously, our results refute this conjecture. The lower bounds for k-taxi, for distributed paging, and for metric allocation are not matched by any known universal upper bound. However, in uniform metric spaces tight bounds of $\Theta(\log n)$ for the latter two problems [2, 6] and $\Theta(\log k)$ for k-taxi [21] are known (for metric allocation and k-taxi this holds even in a somewhat more general setting of weighted star metrics).

Existential and universal lower bounds. The previous best existential lower bound for metrical task systems, $\Omega(\log n)$, was already proved in 1987 when the model was introduced [14]. It uses a coupon collector argument, and the analogous proof had also remained the previously best existential lower bound for the k-server problem. In contrast, the best known *universal* lower bounds (i.e., lower bounds that hold for *all* metrics) had developed over time: Initially it seemed plausible that there might even exist metric spaces with constant competitive ratio, until a first super-constant universal lower bound was shown in [35]. This was based on the idea of showing that every metric space contains a large subspace belonging to a family for which a super-constant lower bound is known. This idea was developed further in [8, 10, 13], eventually resulting in lower bounds of $\Omega(\log n/\log\log n)$ and $\Omega(\log k/\log\log k)$, respectively, for metrical task systems and the k-server problem for all metric spaces of $n > k$ points. These lower bounds are implied by an $\Omega(\log n)$ lower bound on metrical task systems in $\Omega(\log^2 n)$-HST metrics from [8], in combination with lower bounds on the size of a subspace close to such a structure in any metric space from [10]. Here, we close the remaining gap to the upper bounds of $O(\log n)$ and $O(\log k)$ that hold in some metrics by proving universal lower bounds of $\Omega(\log n)$ and $\Omega(\log k)$, respectively. We show that such lower bounds hold for all 1-HST metrics. The universal lower bounds are then implied from [10].

Overview of the existential lower bound proof. Our existential lower bounds use a construction of a metric space akin to the diamond graph used in [1, 15, 41, 43, 44] to prove lower bounds on bi-Lipschitz distortion of metric embeddings and dimension reduction in ℓ_1, and in [32] to prove lower bounds on the online Steiner tree problem. The main idea is demonstrated by the following basic construction that is used in Section 3 to illustrate a somewhat weaker, yet simpler, lower bound than our main result. We use the shortest path metric on the nodes of an inductively constructed graph. Let $w \in \mathbb{N}$. The w graph is constructed as follows. Take a cycle whose length is a multiple of 6. Let s and t be two antipodal points in this cycle. They partition the cycle into two s-t paths whose length is a multiple of 3. We'll consider the first, second, and last third of both paths. But first, we replace every edge in this graph with a copy of the $w-1$ graph, identifying the endpoints of the edge with the two chosen terminals of the recursive graph. Note that in an n-point metric space, if $k = n-1$ then the movement of the servers can be defined alternatively as the movement of the "hole," a.k.a. anti-server—the unique position that no server occupies. The anti-server can be forced to choose a location from a subset of the points by requesting repeatedly the other points.

Now, we describe the bad sequence of requests, informally. Our argument uses Yao's minimax principle—we design a random request sequence that drives the cost of every deterministic algorithm to reach or exceed the lower bound, which we'll denote here

by C_w. This is done in three stages. The first stage makes sure the anti-server traverses the first third of one of the two s-t paths (which now consist of chains of $w-1$ graphs), while paying a factor of C_{w-1} over the shortest path to this goal. The main purpose of this stage is to create two targets to choose from which are far apart. The last third is a mirror image of the first third and is intended to ensure that the anti-server reaches t, while paying the same factor C_{w-1} over the shortest path to that goal. The middle third is where the increase in the competitive ratio happens. There we repeatedly, independently, and with equal probability choose one of the two paths, and force the anti-server either to move forward on the chosen path, or to stay put on the other path. This is repeated for the length of the middle third, so in expectation, wherever the anti-server lies, it has to advance half the length of this interval. The initial distance gained from s guarantees that switching paths does not save any cost. Moreover, in expectation one of the two options "suffered" fewer "hits," roughly square root of the number of repetitions fewer. We now take advantage of this expected gap, and generate more requests that force the anti-server to use "the road less traveled by" and move there all the way to t. The excess steps force C_w to be at least $C_{w-1} + \Theta(\sqrt{C_{w-1}})$, and this shows an $\Omega(w^2)$ lower bound. Finally, the size of this graph is $\exp(w \log w)$, so this argument gives $\Omega((\log k/\log\log k)^2)$. To improve the latter bound, we need to "compress" the graph to use cycles of length 6, recursively, so its size is $\exp(w)$. This complicates the argument considerably. As the graph is symmetric with respect to the roles of s and t, we can then reverse the process (now going from t to s) and thus repeat it as many times as desired.

We note that our constructed metric spaces require distortion $\Omega(\log n)$ to represent as a convex combination of HSTs (where n is the number of points in the metric space). This is obvious from the fact that these metrics contain a path of n^ϵ equally spaced points, for some constant $\epsilon > 0$. In fact, our lower bounds illustrate the somewhat surprising conclusion that the approximation of metric spaces as convex combinations of HSTs (from [25]), which underlies the tight universal upper bounds in [17], cannot be circumvented.

Organization. The rest of the paper is organized as follows. In Section 2 we define the problems we show lower bounds for, and discuss the reductions among them. In Section 3 we prove the basic construction outlined above, and also prove the lower bound of $\Omega(w^2)$ for layered graph traversal. In Section 4 we prove the stronger bound for MTS and k-server, our main result. Finally, in Section 5 we prove our improved universal lower bound.

2 PRELIMINARIES

An online (minimization) problem is a two-player game between an adversary and an online algorithm. Keeping the discussion somewhat informal, the adversary chooses a finite sequence of requests $\rho = \rho[1]\rho[2]\rho[3]\ldots$. The algorithm chooses a response function alg that maps every request sequence to an action. When the game is played, the algorithm responds to each request $\rho[i]$ with $\text{alg}[i] = \text{alg}(\rho[1]\rho[2]\ldots\rho[i])$, thus generating a response sequence $\text{alg} = \text{alg}[1]\,\text{alg}[2]\,\text{alg}[3]\ldots$ (abusing notation slightly). A randomized algorithm $\widetilde{\text{alg}}$ chooses alg from a probability distribution on such functions. Thus, a play of the game is marked by a pair of equal-length sequences (ρ, alg). For each pair of an algorithm and request sequence there is an associated cost. We denote by $c_{\text{alg}}(\rho)$ the cost of algorithm alg on request sequence ρ, and by $c_{\text{opt}}(\rho) = \min_{\text{alg}}\{c_{\text{alg}}(\rho)\}$ the cost of the optimal (offline) algorithm.

Definition 1. *The randomized competitive ratio of an online problem* ONL*, denoted* $C^{\text{ONL}}_{\text{rand}}$*, is the infimum over all* C *that satisfy the following condition. There exists a randomized algorithm* $\widetilde{\text{alg}}$ *and some constant* κ *such that for every adversary strategy* ρ*,*

$$\mathbb{E}[c_{\text{alg}}(\rho) : \text{alg} \sim \widetilde{\text{alg}}] \le C \cdot c_{\text{opt}}(\rho) + \kappa.$$

Unless stated otherwise, the constant κ can be arbitrary. However, for some problems it is more typical to require $\kappa = 0$.

Definition 2. *The distributional lower bound on the randomized competitive ratio of an online problem* ONL*, denoted* $C^{\text{ONL}}_{\text{distr}}$*, is the supremum over all* C *that satisfy the following condition. For all* κ *(which are allowed in Definition 1), there exists a probability distribution* $\tilde{\rho}$ *over adversary strategies such that for every (deterministic) algorithm* alg*,*

$$\mathbb{E}[c_{\text{alg}}(\rho) : \rho \sim \tilde{\rho}] \ge C \cdot \mathbb{E}[c_{\text{opt}}(\rho) : \rho \sim \tilde{\rho}] + \kappa.$$

The following theorem is known as Yao's minimax principle. Notice that it cannot be deduced trivially from von Neumann's minimax principle by setting $\frac{c_{\text{alg}}(\rho)-\kappa}{c_{\text{opt}}(\rho)}$ to be the zero-sum value of a play (ρ, alg), because the distributional lower bound is about the ratio of expectations, not the expected ratio. However, a similar LP duality argument yields it.[1]

Theorem 3 (Yao's minimax). $C^{\text{ONL}}_{\text{rand}} = C^{\text{ONL}}_{\text{distr}}$.

Some notation. Let ρ be a request sequence. We will use subscripts ρ_i to indicate subsequences in an underlying partition $\rho = \rho_1\rho_2\ldots\rho_m$. Note that each ρ_i is a sequence, not necessarily a single request. We denote by $\rho_{\le i}$ the prefix $\rho_1\rho_2\ldots\rho_i$. Also, for notational convenience, $\rho_{\le 0}$ stands for the empty sequence. For two indices $i \le j$, we denote by $\rho_{i\le\cdot\le j}$ the subsequence $\rho_i\rho_{i+1}\ldots\rho_j$.

Escape prices. Given an online problem ONL, we will sometimes consider an *escape price relaxation* of ONL, defined as follows. There is an underlying escape price parameter $p \ge 0$. At certain points in the request sequence (potentially all of them), the online algorithm is allowed as an additional option to respond to a request by bailing out of the game. If the online algorithm responds to a request $\rho[t]$ by bailing out of the game, then the escape price p is added to its cost, but the online algorithm pays no additional cost for any requests thereafter (including the request $\rho[t]$ itself). If we allow this additional option of bailing out only in response to requests $\rho[t]$ belonging to some specific subsequence $\rho_{i\le\cdot\le j}$, then we say that the escape price is available on $\rho_{i\le\cdot\le j}$.

In the escape cost relaxation of a problem, we allow invoking the escape price only to the online algorithm, but not to the (offline) algorithms defining the base cost. Thus, the escape price option only

[1] One may prove it for a fixed κ first, and then take the infimum over all κ.

helps the online player, and the competitive ratio for the escape price relaxation of ONL is at most $C^{\mathrm{ONL}}_{\mathrm{rand}}$, regardless of the value of p.[2]

More notation. In some online problems, there is an adjustable starting configuration s which affects the cost. Given a play (ρ, alg) with starting configuration s, we often write $c_{\mathrm{alg},s}(\rho)$ to denote the cost of the algorithm in this case, and $c_{\mathrm{opt},s}(\rho)$ for the corresponding optimal (offline) cost. Moreover, let $\rho = \rho_1\rho_2$ be the concatenation of two request sequences ρ_1 and ρ_2. For a play (ρ, alg) with starting configuration s, we write

$$c_{\mathrm{alg},s}(\rho_2 \mid \rho_1) := c_{\mathrm{alg},s}(\rho) - c_{\mathrm{alg},s}(\rho_1),$$

denoting the partial cost of the online algorithm on the subsequence ρ_2 when serving the request sequence $\rho = \rho_1\rho_2$. When the initial location s is clear from the context, we will often drop it from the notation.

Now let us discuss the online problems considered in this paper. All the problems considered in this paper are defined in the context of a metric space. See the appendix of the full version of our paper for some basic definitions and notation.

k-server. There is an underlying metric space $\mathcal{M} = (M, d)$. The adversary's strategy ρ is a sequence of points $\rho[1], \rho[2], \cdots \in M$. The algorithm controls k identical servers, initially located at k distinct points in M. We may assume w.l.o.g. that $|M| > k$, otherwise the problem is trivial. The response $\mathrm{alg}[i]$ to a request $\rho[i]$ moves one of the servers from its current location to $\rho[i]$. This adds to the cost of the algorithm the distance travelled by the server. We denote this problem as kSRV.

Metrical task systems. Here, too, there is an underlying metric space $\mathcal{M} = (M, d)$, which must be finite. The elements of M are called *states*. An adversary's request is a vector in $(\mathbb{R}_+ \cup \{\infty\})^M$, where $\mathbb{R}_+$ denotes the set of non-negative real numbers. The algorithm begins at an arbitrary $\mathrm{alg}[0] \in M$. In response to $\rho[i] \in (\mathbb{R}_+ \cup \{\infty\})^M$, the algorithm must choose a state $\mathrm{alg}[i] \in M$. This adds to its cost $d(\mathrm{alg}[i-1], \mathrm{alg}[i]) + \rho[i](\mathrm{alg}[i])$. We denote this problem as MTS.

k-taxi. The setting is identical to that of the k-server problem. The adversary's strategy consists of a sequence of pairs of points in M. In response to a request, the algorithm must move a server to the first point in the pair, and then move the same server from the first point to the second point. There are two flavors to this problem, differing in the definition of the cost. In *easy* k-taxi, the algorithm pays for the entire move. We denote this flavor by ekTX. In *hard* k-taxi, the algorithm pays only for the move to the first point in the request and not for the move to the second point (which of course changes the location of the server towards the following requests). We denote this flavor by hkTX.

[2]The escape price relaxation is reminiscent of a combination of the original problem and the ski rental problem, where the classical cost corresponds to rental price and p is the buying price.

Proposition 4 (folklore). *For all metric spaces $\mathcal{M} = (M, d)$ and for all finite $k < |M|$, $C^{\mathrm{hkTX}}_{\mathrm{rand}}(\mathcal{M}) \geq C^{\mathrm{ekTX}}_{\mathrm{rand}}(\mathcal{M}) \geq C^{\mathrm{kSRV}}_{\mathrm{rand}}(\mathcal{M})$.*[3]

We note that there are metric spaces for which $C^{\mathrm{hkTX}}_{\mathrm{detr}}(\mathcal{M}) \gg C^{\mathrm{kSRV}}_{\mathrm{detr}}(\mathcal{M})$, see [22]. On the other hand, in all settings, $C^{\mathrm{ekTX}}_{\mathrm{detr}}(\mathcal{M}) \leq C^{\mathrm{kSRV}}_{\mathrm{detr}}(\mathcal{M}) + 2$, see [36].

Distributed paging. There is a network of processors. The communication links and delays induce a metric on the set of processors, so we regard the setting as a metric space $\mathcal{M} = (M, d)$, where M is the finite set of processors. Each processor $x \in M$ is endowed with a cache that can hold $m_x \in \mathbb{N}$ memory pages. Let $m = \sum_{x \in M} m_x$. The network in its entirety holds $f \leq m$ distinct pages. They can be replicated in different processors, but for each distinct page at least one copy must be held somewhere in the network at all times. The adversary requests pairs (p, x), where p is one of the f pages, and $x \in M$ is a processor. In response, the algorithm must bring a copy of p to x. If the page is not already there, the algorithm can replicate a copy at another processor, or move that copy, to x. Either way, this costs the distance to the other processor. If x's cache is full, the algorithm must evict a page to make room for p, and the evicted page can be discarded, or (this is necessary if it is the last copy) moved to a vacant slot in another processor, incurring a cost equal to the distance between the processors. We denote this problem as DPG. The following proposition is a special case of [2, Theorem 3.1].

Proposition 5 (Awerbuch, Bartal and Fiat [2]). *Consider any n-node network $\mathcal{M}$. Let $k = n - 1$, and let $f = m - n + 2$. Then, $C^{\mathrm{DPG}}_{\mathrm{rand}}(\mathcal{M}, m, f) = \Omega\left(C^{\mathrm{kSRV}}_{\mathrm{rand}}(\mathcal{M})\right)$.*

Metric allocation. As usual, there is an underlying finite metric space $\mathcal{M} = (M, d)$. The algorithm maintains a fractional allocation of a resource to the points on M, denoted by a vector $p \in \mathbb{R}_+^M$ with $\sum_{x \in M} p_x = 1$. Moving from one allocation to another adds to the algorithm's cost the transportation (a.k.a. earthmover) distance between the two, under d. Each request of the adversary is defined by assigning to every $x \in M$ a non-increasing convex function $\phi_x : [0,1] \to \mathbb{R}_+$. If the algorithm serves the request using the allocation p, then this adds to its cost $\sum_{x \in M} \phi_x(p_x)$. We denote this problem by MA.

Proposition 6 (Bansal and Coester [6]). *For all finite metric spaces $\mathcal{M}$, $C^{\mathrm{MA}}_{\mathrm{rand}}(\mathcal{M}) = C^{\mathrm{MA}}_{\mathrm{detr}}(\mathcal{M}) \geq C^{\mathrm{MTS}}_{\mathrm{rand}}(\mathcal{M})$.*

Layered graph traversal. Here the adversary selects a layered graph $G = (V, E)$, with edge lengths $L : E \to \mathbb{R}_+$, first layer $\{s\}$, last layer $\{t\}$, and edges only between vertices of adjacent layers. The graph is presented to the algorithm layer by layer. When a layer is presented, all the edges to the previous layers and their lengths are revealed. Starting at s, in order to reveal a new layer, the algorithm must reach the previous layer (so at the start the second layer is revealed). The game ends when the algorithm reaches t. The number of layers is not known until t is revealed. The cost of the algorithm is the total length of its traversed path in G. We

[3]We will use the notation $C^{\mathrm{ONL}}_{\mathrm{rand}}(P)$ (also $C^{\mathrm{ONL}}_{\mathrm{detr}}(P)$ for the deterministic competitive ratio) to specify the setting P in which the problem ONL is considered.

denote this problem by LGT. Let w denote the maximum number of nodes within a layer of G.

Proposition 7 (folklore). *Let $\mathcal{M}$ be a finite metric space. Then, $C^{\mathrm{LGT}}_{\mathrm{rand}}(w) \geq C^{\mathrm{MTS}}_{\mathrm{rand}}(\mathcal{M})$, where w is the number of points in $\mathcal{M}$.*

Proof idea. When cost vector $\rho[i]$ is revealed in MTS, construct a new layer L_i with a vertex for each point in $\mathcal{M}$, and the edge between any $x \in L_{i-1}$ and $y \in L_i$ has length $d(x,y) + \rho[i](y)$. □

We note that the inequality here is known to be far from tight.

Small set chasing. There is an underlying metric space $\mathcal{M} = (M, d)$, not necessarily finite, and a starting point $x_0 \in M$. The adversary presents requests which are subsets of M, each of cardinality at most w. To serve a request, the algorithm must move to one of the points in the finite subset, incurring a cost equal to the distance traversed. We denote this problem by MSS.[4] The following propositions are known.

Proposition 8 (folklore). *Let $\mathcal{M}$ be an n-point metric space and let $k = n - 1$. Then for every $w \leq k$, $C^{\mathrm{MTS}}_{\mathrm{rand}}(\mathcal{M}) \geq C^{\mathrm{kSRV}}_{\mathrm{rand}}(\mathcal{M}) \geq C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}, w)$. The last inequality is an equality if $w = k$.*

Proof idea. In $(n-1)$-server there is exactly one point not covered by a server, and this point corresponds to the server location in MSS and MTS. Then a request to a set $S \subset M$ in MSS corresponds to repeated k-server requests to all the points in $M \setminus S$. A k-server request at a point $p \in M$ corresponds to the MTS cost vector that assigns cost ∞ to p and cost 0 to other points. □

Proposition 9 (Fiat, Foster, Karloff, Rabani, Ravid, Vishwanathan [26]). *Let $\mathcal{M}$ be any metric space, and let $\mathcal{U}$ be the Urysohn universal metric space. Then for every $w \in \mathbb{N}$, $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{U}, w) \geq C^{\mathrm{LGT}}_{\mathrm{rand}}(w) \geq C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}, w)$.*

Evolving tree game. The adversary maintains a rooted tree T of maximum depth w and non-negative edge lengths. The root r is fixed throughout the game, and always has one child. Initially, T has a single edge of length 0. At each round of the game, the adversary can choose one of three types of moves: (a) Pick a non-root leaf and increase the length of the edge incident on it. (b) Pick a leaf other than the root and its child and delete it, and if the parent's degree drops to 2 merge the two edges. (c) Create two or more new nodes and attach them with edges of length 0 to an existing leaf. The algorithm must occupy a leaf at all times. Moving between nodes adds to the algorithm's cost the length of the path connecting them. Staying at a leaf while the adversary increases its incident edge length adds to the cost of the algorithm the increase in edge length. We denote this problem by ETG.

Proposition 10 (Bubeck, Coester, Rabani [16]). $C^{\mathrm{ETG}}_{\mathrm{rand}}(w) \geq C^{\mathrm{LGT}}_{\mathrm{rand}}(w)$.

Our lower bound for LGT in this paper, along with the upper bounds in [16], establish that the competitive ratio of LGT is actually the same (up to constant factors) as that of the version of ETG where the tree is binary (or of constantly bounded degree) at all times.

[4] The abbreviation stands for *metrical service systems*, which is the historical name of the problem.

3 BASIC CONSTRUCTION AND ANALYSIS

The main goal of this section is to describe, in a somewhat informal style, a simple construction and analysis which yields a lower bound of $\Omega\left(\left(\frac{\log n}{\log\log n}\right)^2\right)$ for MSS in n-point metric spaces with request sets of arbitrary size. Notice that this lower bound implies immediately the same lower bound for MTS and for kSRV, putting $k = n - 1$, on account of Proposition 8. In Section 3.6 we show that this lower bound can also be achieved with request sets of size at most $w = O\left(\frac{\log n}{\log\log n}\right)$, which implies the lower bound of $\Omega(w^2)$ for LGT on account of Proposition 9.

3.1 The family of metric spaces

First we describe our family of metric spaces by induction. Let $m_0 \leq m_1 \leq m_2 \leq \cdots$ be a sequence of natural numbers, to be determined later. We construct a sequence of finite metric spaces $\mathcal{M}_0 = (M_0, d_0)$, $\mathcal{M}_1 = (M_1, d_1)$, $\mathcal{M}_2 = (M_2, d_2)$, ... of growing size as follows. Each metric in this sequence is the shortest paths metric of an underlying graph. The base case $\mathcal{M}_0$ is a single edge of weight 1. Next, $\mathcal{M}_1$ is a cycle with $6m_0$ edges. In other words, we form a cycle of $6m_0$ copies of $\mathcal{M}_0$. We also choose two special antipodal vertices/points s and t, so $\mathrm{diam}(\mathcal{M}_1) = d_1(s,t)$.

More generally, $\mathcal{M}_{w+1}$ is built from $\mathcal{M}_w$ in the same way as $\mathcal{M}_1$ was built from $\mathcal{M}_0$ (see Figure 1). Namely, M_{w+1} is a "cycle" made of $6m_w$ copies of $\mathcal{M}_w$. Slightly more precisely, consider a cycle with $6m_w$ edges, where each edge $\{u, v\}$ is replaced by putting a copy of $\mathcal{M}_w$ with the two special vertices at u and v. In particular, in $\mathcal{M}_{w+1}$, each special vertex of one copy of $\mathcal{M}_w$ is identified with another special vertex in another copy of $\mathcal{M}_w$. In $\mathcal{M}_{w+1}$, the special vertices s and t are special vertices in two distinct copies of $\mathcal{M}_w$ such that $\mathrm{diam}(\mathcal{M}_{w+1}) = d_{w+1}(s,t)$. Notice that $\mathcal{M}_{w+1}$ can be viewed as consisting of a *left path* of $3m_w$ copies of $\mathcal{M}_w$ and a *right path* of another $3m_w$ copies of $\mathcal{M}_w$. For two subsets of points $S, S' \subset M_w$, we denote by (S, S', i, j) the subset of M_{w+1} that is made of the union of the set S in the i^{th} copy of $\mathcal{M}_w$ on the left path, and the set S' in the j^{th} copy of $\mathcal{M}_w$ on the right path.

3.2 The hard sequence

Now let us describe by induction a hard random sequence ρ^w of subsets of M_w. Recall that $\rho^w[j]$ denotes the j^{th} request in the sequence ρ^w. This is a random subset of M_w. Also let T_w denote the length of ρ^w. The construction gives T_w as a random variable, but for simplicity we ignore this aspect as we can always pad a sequence with some dummy sets to attain a fixed length. We will always have $\rho^w[1]$ be the special vertex s (the source) and $\rho^w[T_w]$ be the special vertex t (the target). Moreover, our construction will guarantee that the request sequence can be satisfied by following a shortest path connecting s and t (so $c_{\mathrm{opt}}(\rho^w)$ is simply the length of this path). We denote by $\rho^{w,1}, \rho^{w,2}, \ldots$ a sequence of i.i.d. copies

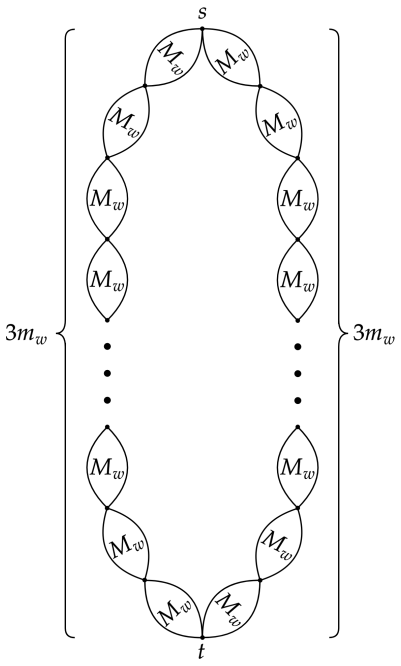

Figure 1: Construction of $\mathcal{M}_{w+1}$ in the $\Omega\left(\left(\frac{\log n}{\log\log n}\right)^2\right)$ lower bound.

of ρ^w. The inductive construction of ρ^{w+1} uses ρ^w and proceeds in three stages.

Stage 1: For every $i = 1, \ldots, m_w$ and $j \in \{1, 2, \ldots, T_w\}$, let

$$\rho^{w+1}[(i-1)T_w + j] = (\rho^{w,i}[j], \rho^{w,i}[j], i, i)\,.$$

In words, we traverse simultaneously the first third of the left and right path of $\mathcal{M}_{w+1}$, one copy of $\mathcal{M}_w$ at a time, using the hard sequence for $\mathcal{M}_w$.

Stage 2: For every $i = 1, \ldots, m_w$, let $\epsilon_i \in \{0, 1\}$ be a Bernoulli random variable independent of everything else. Let $\mathrm{left}(i) = m_w + \epsilon_1 + \ldots + \epsilon_i$, and let $\mathrm{right}(i) = m_w + (1-\epsilon_1) + \ldots + (1-\epsilon_i)$. Also let $\rho^{\mathrm{left}} = \rho^{w,m_w+i}[\epsilon_i j + (1-\epsilon_i)T_w]$, and let $\rho^{\mathrm{right}} = \rho^{w,m_w+i}[(1-\epsilon_i)j + \epsilon_i T_w]$. Now, for every $j \in \{1, 2, \ldots, T_w\}$, let

$$\rho^{w+1}[(m_w + i - 1)T_w + j] = (\rho^{\mathrm{left}}, \rho^{\mathrm{right}}, \mathrm{left}(i), \mathrm{right}(i))\,.$$

In words, if $\epsilon_i = 1$, we present the hard sequence on the next copy of $\mathcal{M}_w$ on the left path, and on the right path we "stay put" (i.e., we keep requesting the target of the last copy of $\mathcal{M}_w$ that was previously traversed). Conversely, if $\epsilon_i = 0$ we "stay put" on the left path, and on the right path we traverse the next copy of $\mathcal{M}_w$ using the hard sequence. (Recall that always $\rho^w[T_w] = \{t\}$.)

Stage 3: Assume that $\mathrm{left}(m_w) \geq \mathrm{right}(m_w)$ (the other case is dealt with by symmetry). Then we set for every $i = 1, \ldots, 3m_w - \mathrm{right}(m_w)$, and for every $j \in \{1, 2, \ldots, T_w\}$,

$$\rho^{w+1}[(2m_w + i - 1)T_w + j] = (\emptyset, \rho^{w,2m_w+i}[j], 1, \mathrm{right}(m_w) + i)\,.$$

In words, the path along which we advanced more at the end of stage 2 is "killed." We continue advancing along the other path, until we reach the final target (which is the target special vertex in $\mathcal{M}_{w+1}$).

3.3 The cost analysis

Let's scale uniformly the edge weights in $\mathcal{M}_{w+1}$, so that the diameter of any copy of $\mathcal{M}_w$ becomes equal to 1 (this is just for notational simplicity). In particular the optimal cost is $3m_w$. Let us assume by induction that we have proved for $\mathcal{M}_w$ a lower bound of C_w on the competitive ratio of any deterministic algorithm against the random sequence ρ^w. We now analyze the cost of any deterministic algorithm in the three stages of the random sequence ρ^{w+1}.

Stage 1: Here, by induction, the expected cost is simply lower bounded by $m_w C_w$.

Stage 2: Assume that $m_w \geq C_w$. (Note that this is a somewhat stricter assumption than seems to be needed here, but we will need this stricter condition later.) Then we claim that the expected cost in this stage is lower bounded by $\frac{m_w}{2} C_w$. Indeed, in each new "phase" (where the hard sequence ρ^w is presented either on the left path or on the right path), the algorithm, with probability 1/2, has to choose between traversing a copy of $\mathcal{M}_w$ against ρ^w, at expected cost C_w, or alternatively backtracking (i.e., switching) to the other path (either right away, or after a while) which costs at least $2m_w \geq C_w$. This tentatively concludes the proof of the claim, up to the minor issue that a priori we do not control the expected cost of traversal of $\mathcal{M}_w$ *conditioned on the fact that the algorithm does not switch*. To see why this might be a problem, consider the fictitious situation (indeed, our request sequence will *not* be like this) where with probability $1-\epsilon$ the traversal is easy (say cost 0) and with probability ϵ the traversal is hard (say cost $1/\epsilon$ times the expectation). In that case switching when the hardness of the traversal is revealed might lower the expected cost by a multiplicative factor ϵ. We explain below in Section 3.5 how to deal with this issue and complete the proof of this paragraph's claim.

Stage 3: This is the key part of the argument. We have

$$\min\{\mathrm{left}(m_w), \mathrm{right}(m_w)\} \leq \frac{3}{2} m_w - \Omega(\sqrt{m_w})$$

with high probability (e.g., on account of the Berry-Esseen Inequality). Therefore, with high probability we face in this stage at least $\frac{3m_w}{2} + \Omega(\sqrt{m_w})$ copies of $\mathcal{M}_w$ to traverse. Thus, the expected cost of this stage is $\left(\frac{3m_w}{2} + \Omega(\sqrt{m_w})\right) C_w$.

3.4 Selecting the parameters and conclusion

Overall the analysis shows that the algorithm pays in expectation $(3m_w + \Omega(\sqrt{m_w}))C_w$, whereas OPT pays just $3m_w$. Thus, we get the recurrence relation $C_{w+1} \geq \left(1 + \Omega\left(\frac{1}{\sqrt{m_w}}\right)\right) C_w$. Clearly, we want to choose m_w to be as small as possible. With the constraint $m_w \geq C_w$ from the stage 2 analysis we obtain $C_{w+1} \geq C_w + \Omega(\sqrt{C_w})$. In particular we easily get by induction $C_w \geq \Omega(w^2)$. Finally, let us calculate $|M_w|$. We have $|M_{w+1}| \leq 6m_w |M_w| \leq 6^w \prod_{w'=1}^{w} m_{w'}$. As

C_w is of order w^2, and m_w is of order of C_w, we get that $|M_w|$ is of order $\exp(Cw \log w)$ for some constant $C > 0$. In particular, if we denote that number by n, we have $\log n = Cw \log w$, so that $w = \Theta\left(\frac{\log n}{\log \log n}\right)$. To extend the bound to any number of points n', simply choose the largest w for which $|M_w| \le n'$, and then extend $\mathcal{M}_w$ arbitrarily to contain exactly n' points (the extra points will be ignored in the request sequence). This concludes the proof of the lower bound up to the conditioning issue in the stage 2 analysis, which we deal with next.

3.5 Escape price

Here we resolve the issue of controlling the expected cost obtained by induction for a traversal of $\mathcal{M}_w$, when the algorithm is allowed to abort this traversal and switch to the other branch in $\mathcal{M}_{w+1}$. Switching to the other branch in stage 2, which is the only stage where it is not obvious that aborting cannot help, costs at least $2m_w \ge 2C_w = 2\,\mathrm{diam}(\mathcal{M}_w)C_w$. (Recall that when considering $\mathcal{M}_{w+1}$, we scale the distances so that the diameter of each copy of $\mathcal{M}_w$ is 1.)

To control this cost, we actually show inductively a lower bound on the competitive ratio of the escape price relaxation of MSS in $\mathcal{M}_w$, rather than on $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}_w, |M_w|)$. Let p_w denote the escape price for the game in $\mathcal{M}_w$. We now show by induction that, for $p_w = 2\,\mathrm{diam}(\mathcal{M}_w)C_w$, the escape price option does not enable an online algorithm facing the hard random sequence ρ^w to overcome the stated lower bound. This in turn will conclude the proof.

The base case is trivial, so let us assume it is true for some w. In $\mathcal{M}_{w+1}$ the escape price is $2\,\mathrm{diam}(\mathcal{M}_{w+1})C_{w+1} = 6m_wC_{w+1}$. (Recall again the uniform scaling of distances when considering $\mathcal{M}_{w+1}$.) Now, assume that the algorithm decides to escape (at the induction level $w+1$), and that this escape happens in some copy of $\mathcal{M}_w$. Using the induction hypothesis, the cost of instead escaping only at the lower induction level w in the aborted copy, but then proceeding to serve the rest of the hard request sequence by going through all the remaining copies of $\mathcal{M}_w$ and paying the expected cost there, is at most the following: $2\,\mathrm{diam}(\mathcal{M}_w)C_w = 2C_w$ for the escape price, plus m_wC_w for stage 1, plus $m_wC_w/2$ for stage 2, plus $2m_wC_w$ for stage 3. Overall, this cost is at most $(2 + 3.5m_w)C_w \le 6m_wC_{w+1}$, as $m_w \ge 1$. In other words, the escape price option on $\mathcal{M}_{w+1}$ does not add any benefit to the algorithm compared to the escape price option on $\mathcal{M}_w$, and thus by induction it does not enable the algorithm to overcome the lower bound.

3.6 Lower bound for LGT

We briefly describe how this construction (slightly modified) also gives an $\Omega(w^2)$ lower bound for MSS with sets of size at most w (and hence for layered graph traversal). Notice that in stages 2 and 3 the sets being requested in ρ^{w+1} have the size of the sets in ρ^w plus at most 1. If this property was also true for stage 1 we would be done. To achieve this, we modify stage 1 as follows: redefine $\mathcal{M}_{w+1}$ by replacing the first m_w copies of $\mathcal{M}_w$ on the left path (and similarly on the right path) by a single edge of length $\frac{C_w}{m_w}$ followed by m_w^2 copies of $\frac{1}{m_w}\mathcal{M}_w$, where $\frac{1}{m_w}\mathcal{M}_w$ denotes the metric space $\mathcal{M}_w$ with distances scaled by $\frac{1}{m_w}$. The request sequence of stage 1 is similar to that of stage 2, in each step presenting the hard sequence on the next copy of $\frac{1}{m_w}\mathcal{M}_w$ on one path while staying put on the other path, but rather than choosing the advancing path at random, we alternate between the two sides. In each pair of steps, the expected online cost is at least $\frac{C_w}{m_w}$ (either due to movement through the advancing copy on the path where the online algorithm is located, or to switch to the other path). Since stage 1 consists of m_w^2 such pairs of steps, the expected cost of stage 1 is still at least m_wC_w, so we get the same lower bound on the online cost as before. The cost of OPT is slightly higher by an additive $\frac{C_w}{m_w} \le 1$ due to the length of the initial extra edge, but since this is much smaller than m_w we still get a recurrence of the same form $C_{w+1} \ge C_w + \Omega(\sqrt{C_w})$, yielding $C_w = \Omega(w^2)$.

4 EXISTENTIAL $\Omega(\log^2 n)$ LOWER BOUND

In this section, we refine the bound from the previous section and show the following theorem.

Theorem 11. *For each $n \in \mathbb{N}$, there exists an n-point metric space $\mathcal{M}$ such that $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}, n-1) = \Omega(\log^2 n)$.*[5]

By Proposition 8, this implies that there exists no $o(\log^2 n)$-competitive algorithm for MTS on general n-point metrics, and no $o(\log^2 k)$-competitive algorithm for kSRV on general $(k+1)$-point metrics. These lower bounds are tight for MTS, and for kSRV at least in the case of $(k+1)$-point metrics, on account of the known $O(\log^2 n)$ upper bound for MTS on arbitrary n-point metrics [17].

To avoid the $\log^2 \log n$ divisor in the bound proved in Section 3, we will use only 6 instead of $6m_w$ copies of $\mathcal{M}_w$ when constructing $\mathcal{M}_{w+1}$, so that the metric space is of smaller size. Note that the reason why m_w needed to be chosen large in Section 3 is to ensure that the cost of switching between the left and right paths is large. The key idea that will enable us to allow smaller switching cost here is that rather than issuing the inductive request sequence in $\mathcal{M}_w$ in its entirety each time, we can break it into smaller subsequences ("chunks") and only issue a single chunk at a time. Since the cost of a chunk is smaller than the cost of the entire inductive sequence, we do not need as large of a switching cost to discourage the algorithm from switching between the left and right paths. A similar idea of decomposing inductive request sequences into chunks is also the key idea that will allow us to tighten the universal lower bounds later in Section 5.

Let $0 < \alpha < 1$ and $\beta \in \mathbb{N}$ be constants that we determine later. We will show that for every $w \in \mathbb{N}$ there exists a metric space $\mathcal{M}_w = (M_w, d_w)$ with $|M_w| \le \beta \cdot 6^w$ where any randomized algorithm for MSS (with request sets of size at most 2^w) has competitive ratio at least αw^2. The metric space $\mathcal{M}_w$ has two special points $s_w, t_w \in M_w$ that we use as the initial and final location of the lower bound instance (i.e., s_w is the initial location of the server, and the request sequence will force the server to terminate at t_w). The request sequence will be such that an optimal offline algorithm

[5] In fact, our construction of the bad sequence uses only sets of size $O(n^{\log_6 2})$ instead of $n-1$.

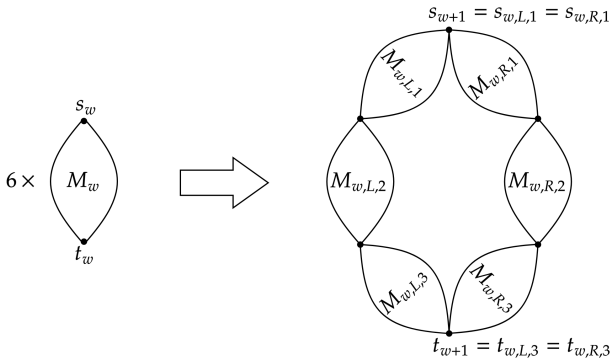

Figure 2: Construction of $\mathcal{M}_{w+1}$ and choice of s_{w+1} and t_{w+1} for the $\Omega(\log^2 n)$ lower bound.

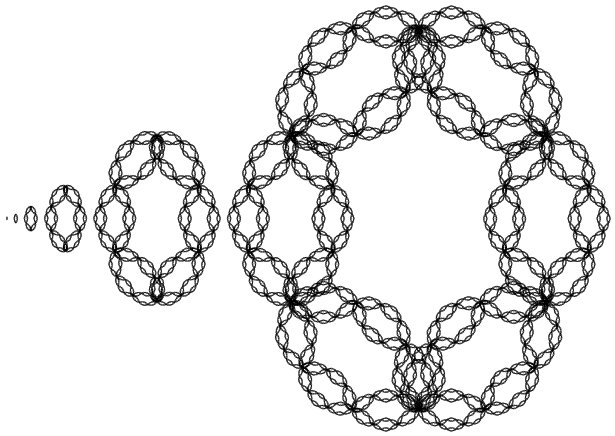

Figure 3: Metric spaces where the competitive ratio of MTS and $(n-1)$-server is $\Theta(\log^2 n)$.

can serve it for cost $d_w(s_w, t_w)$ by moving along a shortest path from s_w to t_w.

If $\alpha w^2 \le 1$, the lower bound is trivial and we choose $\mathcal{M}_w$ to be $\beta + 1$ equally spaced points on a line, with s_w and t_w being the two outermost points.

For larger w, we proceed by induction. Specifically, for $w \in \mathbb{N}$ with $\alpha(w+1)^2 > 1$, we construct $\mathcal{M}_{w+1}$ by taking six copies of $\mathcal{M}_w$ and gluing them together in a circle as shown in Figure 2. We view $\mathcal{M}_{w+1}$ as consisting of a left path and right path of 3 copies of $\mathcal{M}_w$ each. For $j \in \{1,2,3\}$, we denote by $\mathcal{M}_{w,L,j} = (M_{w,L,j}, d_w)$ and $\mathcal{M}_{w,R,j} = (M_{w,R,j}, d_w)$ the jth copy of $\mathcal{M}_w$ on the left and right, respectively. If p is a (set of) point(s) in M_w, we denote by $p_{L,j}$ and $p_{R,j}$ the corresponding (sets of) points in the respective copy. Some members of the family of metric spaces constructed in this way are depicted in Figure 3.

To prove the lower bound, we construct by induction on w a random request sequence ρ for $\mathcal{M}_w$ on which *every* deterministic online algorithm has expected cost at least $\alpha w^2 d_w(s_w, t_w)$. However, to construct these sequences, we will require an induction hypothesis that yields stronger and more delicate properties. Roughly, we will require that the random request sequence ρ can be decomposed into $m \ge \alpha\beta w^2$ (random) subsequences $\rho_1, \dots, \rho_m$ such that any online algorithm's expected cost on each ρ_i is at least some $c_i \approx d_w(s_w, t_w)/\beta$, and the expected sum of these c_i is at least $\alpha w^2 d_w(s_w, t_w)$. This will be captured more precisely by Lemma 12 below.

Terminology and notation. A *chunk* is a sequence of requests. We use this term especially for cases where a sequence is meant to be used as a subsequence of a longer sequence. Sometimes we also use the term *subchunk* instead of chunk, to emphasize that a sequence will be used as a subsequence of a chunk, which in turn will be a subsequence of a longer sequence. Recall that in the context of a sequence of chunks $\rho_1\rho_2\dots\rho_m$, we denote by $\rho_{\le i} = \rho_1\dots\rho_i$ the concatenation of the first i chunks, and by $\rho_{\le 0}$ an empty sequence. If the ρ_i are random, we denote by $\sigma(\rho_{\le i})$ the σ-algebra generated by $\rho_{\le i}$. The property that c_i is $\sigma(\rho_{\le i-1})$-measurable in the following lemma simply means that c_i is a function of $\rho_{\le i-1}$.

Lemma 12. *For every $w \in \mathbb{N}$, there exists a random sequence of chunks $\rho_1\rho_2\dots\rho_m$ for* MSS *in $\mathcal{M}_w$, and a choice of random variables $c_1, \dots, c_m$, such that for all $i = 1, \dots, m$ the following conditions hold.*

1. *The last request of ρ_m is $\{t_w\}$.*
2. *$c_{\mathrm{opt}}(\rho_{\le m}) = d_w(s_w, t_w)$.*
3. *c_i is $\sigma(\rho_{\le i-1})$-measurable.*
4. *For every choice of $\rho_{\le i-1}$ in the sample space,*
$$\mathbb{E}\left[c_{\mathrm{alg}}(\rho_i \mid \rho_{\le i-1}) \,\middle|\, \rho_{\le i-1}\right] \ge c_i$$
for any deterministic online algorithm alg*, even if an escape price of $2d_w(s_w, t_w)$ is available on the suffix ρ_i.*
5. $c_i \in \left[\frac{d_w(s_w,t_w)}{2\beta}, \frac{3d_w(s_w,t_w)}{2\beta}\right]$.
6. $\mathbb{E}\left[\sum_{i=1}^m c_i\right] \ge \alpha w^2 d_w(s_w, t_w)$.
7. $m \ge \alpha\beta w^2$.

In the statement of Lemma 12, we call the number c_i the *size* of the chunk ρ_i. Intuitively, the size of a chunk captures (a lower bound on) the online cost for serving the chunk as part of the longer request sequence $\rho_{\le m}$.

Before proving Lemma 12, we briefly argue that it implies Theorem 11.

Proof of Theorem 11. In the setting of Lemma 12, we have

$$\begin{aligned}\mathbb{E}[c_{\mathrm{alg}}(\rho_{\le m})] &= \mathbb{E}\left[\sum_{i=1}^m \mathbb{E}\left[c_{\mathrm{alg}}(\rho_i \mid \rho_{\le i-1}) \,\middle|\, \rho_{\le i-1}\right]\right] \\ &\ge \mathbb{E}\left[\sum_{i=1}^m c_i\right] \ge \alpha w^2 d(s_w, t_w) = \alpha w^2 c_{\mathrm{opt}}(\rho_{\le m}).\end{aligned}$$

Due to the symmetry in the metric space $\mathcal{M}_w$, we can achieve cost ratio αw^2 also in the opposite direction from t_w to s_w, so

the instance can be repeated indefinitely[6]. Since a randomized algorithm is a random distribution over deterministic algorithms, we conclude that for each randomized algorithm there exists a request sequence of arbitrarily high cost where the ratio between expected online cost and optimal cost is at least αw^2. Since α and β are constants and $n = |M_w| \le \beta \cdot 6^w$, we have $\alpha w^2 \in \Omega(w^2) \subseteq \Omega(\log^2 n)$. □

The proof of Lemma 12 is rather long, so let us first provide a roadmap to its structure. We prove Lemma 12 by induction on w. The base case of the induction are all w with $\alpha w^2 \le 1$: Here, the metric space is $M_w = \{0, 1, \dots, \beta\}$ with the line metric, and $s_w = 0$ and $t_w = \beta$ so that $d_w(s_w, t_w) = \beta$. Each chunk ρ_i consists only of one singleton request. The chunks are sorted from $\{1\}$ to $\{\beta\}$. I.e., $m = \beta$ and $\rho_i = \{i\}$ for $i = 1, \dots, m$. For $c_i = 1$, it is easy to check that all properties are satisfied.

For the induction step, consider some $w+1$ with $\alpha \cdot (w+1)^2 > 1$. Assuming by the induction hypothesis that Lemma 12 holds for w, we will show that it also holds for $w + 1$. This is accomplished by first creating "subchunks" that are smaller than required, but satisfy similar properties to those in Lemma 12, and then combining them to chunks of appropriate size that satisfy all the properties.

To simplify notation, assume from now on (without loss of generality) that distances are scaled such that $d_w(s_w, t_w) = \beta$. Thus, $d_{w+1}(s_{w+1}, t_{w+1}) = 3\beta$. We prove the induction step as follows. Rather than constructing the chunks $\rho_1, \dots, \rho_m$ and sizes $c_1, \dots, c_m$ for $\mathcal{M}_{w+1}$ directly, we will first construct subchunks $\tilde\rho_1, \dots, \tilde\rho_{\tilde m}$ of smaller sizes $\tilde c_1, \dots, \tilde c_{\tilde m}$ that satisfy similar properties. The statement we prove is formalized in Claim 13. Note that the first three properties in the claim are identical to the ones required to complete the induction step of Lemma 12, but the remaining properties are slightly different: Property 4 grants an escape price of 2β instead of 6β, the interval in property 5 is $\left[0, \frac{3}{2}\right]$ instead of $\left[\frac{3}{2}, \frac{9}{2}\right]$, property 6 is slightly strengthened by a "+3" term, and no lower bound on $\tilde m$ is claimed.

Claim 13. *Let $w \in \mathbb{N}$ be a natural number for which the properties listed in Lemma 12 hold, and assume that $\alpha \cdot (w+1)^2 > 1$. There exists a random sequence of subchunks $\tilde\rho_1, \tilde\rho_2, \dots, \tilde\rho_{\tilde m}$ in $\mathcal{M}_{w+1}$ and a choice of random variables $\tilde c_1, \dots, \tilde c_{\tilde m}$ such that for all $i = 1, \dots, \tilde m$ the following conditions hold.*

1. *The last request of $\tilde\rho_{\tilde m}$ is $\{t_{w+1}\}$.*
2. *$c_{\mathrm{opt}}(\tilde\rho_{\le \tilde m}) = 3\beta$.*
3. *$\tilde c_i$ is $\sigma(\tilde\rho_{\le i-1})$-measurable.*
4. *For every choice of $\tilde\rho_{\le i-1}$ in the sample space,*
$$\mathbb{E}\left[c_{\mathrm{alg}}(\tilde\rho_i \mid \tilde\rho_{\le i-1}) \,\middle|\, \tilde\rho_{\le i-1}\right] \ge \tilde c_i$$
for any deterministic online algorithm alg*, even if an escape price of 2β is available on the suffix $\tilde\rho_i$.*
5. *$\tilde c_i \in \left[0, \frac{3}{2}\right]$.*
6. *$\mathbb{E}\left[\sum_{i=1}^{\tilde m} \tilde c_i\right] \ge \alpha(w+1)^2 \cdot 3\beta \;+\; 3$.*

[6] allowing for an arbitrarily large additive constant κ in the definition of competitive ratio

Thus, the proof of Lemma 12 by induction proceeds in two steps. We first prove Claim 13. This asserts that if the properties listed in Lemma 12 hold for w, then some weaker properties hold for $w + 1$. Then we prove that if Claim 13 holds, then also the properties listed in Lemma 12 hold for $w + 1$. This is done in the following sections. In Section 4.1 we construct the subchunks and sizes stipulated in Claim 13. In Section 4.2 we prove the weak inductive step stated in Claim 13. Finally, in Section 4.3 we deduce the properties in Lemma 12 for $w + 1$ from Claim 13.

4.1 Constructing subchunks

Suppose Lemma 12 holds for some fixed w with $\alpha \cdot (w+1)^2 > 1$. To deduce the properties in Claim 13 from the induction hypothesis, we first describe in this subsection the construction of the subchunks $\tilde\rho_1, \tilde\rho_2, \dots, \tilde\rho_{\tilde m}$ and sizes $\tilde c_1, \dots, \tilde c_{\tilde m}$. Recall that distances are scaled such that $d_w(s_w, t_w) = \beta$.

The request sequence $\tilde\rho_{\le \tilde m}$ consists of three stages that are executed one after another. In stage j, requests will only involve points in $M_{w,L,j} \cup M_{w,R,j}$.

Stage 1: Sample a sequence of chunks in $\mathcal{M}_w$ by the induction hypothesis, and replace each request set $u \subseteq \mathcal{M}_w$ in it by $u_{L,1} \cup u_{R,1} \subseteq M_{w,L,1} \cup M_{w,R,1}$. The sequence obtained this way constitutes the start of our sequence of subchunks $\tilde\rho_1, \tilde\rho_2, \dots$, and we define the size $\tilde c_i$ of each subchunks of stage 1 to be equal to the size of the corresponding chunk from which it is created.

Stage 2: Invoking the induction hypothesis two more times,[7] sample independently two sequences of chunks $\rho_{L,1}, \dots, \rho_{L,m_L}$ and $\rho_{R,1}, \dots, \rho_{R,m_R}$ in $M_{w,L,2}$ and $M_{w,R,2}$, respectively, and let $c_{L,i}, c_{R,i} \in \left[\frac{1}{2}, \frac{3}{2}\right]$ be the corresponding sizes.

To construct a subchunk in stage 2, we repeatedly do the following: Select a chunk from one of the two inductively constructed sequences of chunks, say $\rho_{L,i}$, and replace in it each request set by the union of itself and the set u of points on the *other* side (i.e., on the right when the selected chunk is $\rho_{L,i}$) that were contained in the last request set of the previous subchunk. We then say that the construction of this subchunk *uses* chunk $\rho_{L,i}$. The chunk used to construct a subchunk in this way is always the first unused chunk from either the left or the right (with the side determined randomly in a way described shortly). Thus, each subchunk advances the request sequence on one of the two sides by one chunk, incurring cost for any algorithm that has its server on that side, whereas an algorithm would pay cost 0 for the subchunk if it has its server on the other side (since it would already be at a point of u).

Let $\mathrm{left}(j)$ and $\mathrm{right}(j)$ be the number of chunks used on the left and right for the construction of the first j chunks of stage 2 (so $\mathrm{left}(j) + \mathrm{right}(j) = j$). Let

$$n_{L,j} := c_{L,\mathrm{left}(j-1)+1} \qquad \text{and} \qquad n_{R,j} := c_{R,\mathrm{right}(j-1)+1}$$

be the sizes of the *next unused* chunks on the two sides immediately *before* the construction of subchunk j of stage 2. These are the two candidate chunks for use in the construction of subchunk j. Then the construction of the jth subchunk of stage 2 uses the next unused

[7] Recall that we are assuming inductively the stronger statement of Lemma 12

chunk on the left with probability $p_{L,j}$, and it uses the next unused chunk on the right otherwise (with probability $p_{R,j} = 1 - p_{L,j}$), where

$$p_{L,j} := \frac{n_{R,j}}{n_{L,j} + n_{R,j}} \qquad \text{and} \qquad p_{R,j} := \frac{n_{L,j}}{n_{L,j} + n_{R,j}}.$$

These probabilities are chosen so that if we let $L_j := \sum_{i=1}^{\text{left}(j)} c_{L,i}$ and $R_j := \sum_{i=1}^{\text{right}(j)} c_{R,i}$ be the total size of chunks used on the left and right, respectively, for the first j subchunks of stage 2, then for $S_j := L_j - R_j$, the sequence $(S_j)_{j=1,2,\dots}$ is a martingale. We set the size of the jth subchunk to be $n_{L,j}n_{R,j}/(n_{L,j} + n_{R,j})$.

These steps continue until the number of subchunks created in stage 2 is

$$\kappa = \min\left\{k\colon \sum_{j=1}^{k+1} n_{L,j}n_{R,j} \geq \frac{\alpha\beta w^2}{4}\right\}. \tag{1}$$

Note that it can be checked after the construction of each subchunk whether this stopping condition is satisfied, as it only depends on random choices made beforehand. Since each summand $n_{L,j}n_{R,j} \geq \frac{1}{4}$ (as the inductive hypothesis assumes Property 5 of Lemma 12), we have $\kappa < \alpha\beta w^2 \leq \min\{m_L, m_R\}$. Thus, the stopping condition is reached before running out of chunks.

We refer to these first κ subchunks of stage 2 as *stage 2a*, and the following part as *stage 2b*: The subchunks of stage 2b are simply the unused chunks on the side whose total size of used chunks in stage 2a was smaller, and the other side gets "killed". More precisely, if $L_\kappa \leq R_\kappa$ then the subchunks of stage 2b are $\rho_{L,\text{left}(\kappa)+1}$, $\rho_{L,\text{left}(\kappa)+2}$, $\dots$, ρ_{L,m_L}, and otherwise they are $\rho_{R,\text{right}(\kappa)+1}$, $\dots$, ρ_{R,m_R}. The corresponding sizes are the ones given by the induction hypothesis.

Stage 3: The last request of stage 2b was either $\{t_{w,L,2}\} = \{s_{w,L,3}\}$ or $\{t_{w,R,2}\} = \{s_{w,R,3}\}$. In the former case, the subchunks and sizes of stage 3 are obtained by invoking the induction hypothesis in $\mathcal{M}_{w,L,3}$, and otherwise in $\mathcal{M}_{w,R,3}$.

4.2 Analysis

Let $\tilde{\rho}_1, \dots, \tilde{\rho}_{\tilde{m}}$ and $\tilde{c}_1, \dots, \tilde{c}_{\tilde{m}}$ be the entire sequences of subchunks and sizes constructed in stages 1, 2 and 3. The first three properties in Claim 13 follow immediately from the construction and the induction hypothesis of Lemma 12. In particular, an optimal offline algorithm can serve the request sequence for cost 3β by moving through the three copies of $\mathcal{M}_w$ that are on the side (left or right) where stages 2b and 3 are played.

Property 4 follows easily from the induction hypothesis if $\tilde{\rho}_i$ belongs to stage 1, 2b or 3. If $\tilde{\rho}_i$ belongs to stage 2a, say it is the jth subchunk of stage 2a, then suppose the algorithm is on the left side right before this subchunk is issued. (The case that the algorithm is on the right is symmetric.) With probability $p_{L,j}$ the subchunk is constructed using the next unused inductive chunk on the left. Conditioned on this being the case, the algorithm pays expected cost at least $n_{L,j}$ for this subchunk by the induction hypothesis. Indeed, even if the algorithm switches to a point in u (the set on the right included in each request) in order to avoid paying more cost for the subchunk, switching to u costs at least 2β, which is the escape price granted by the induction hypothesis. As the event we conditioned on has probability $p_{L,j}$, the (unconditioned) expected cost of the algorithm for the subchunk is at least $p_{L,j}n_{L,j} = n_{L,j}n_{R,j}/(n_{L,j} + n_{R,j})$. This is precisely the size of this subchunk by definition, proving property 4.

The property $\tilde{c}_i \in \left[0, \frac{3}{2}\right]$ is immediate from the induction hypothesis for stages 1, 2b and 3, recalling $\beta = d_w(s_w, t_w)$. For stage 2a it also holds because the size of the jth subchunk is $n_{L,j}n_{R,j}/(n_{L,j} + n_{R,j}) \leq \max\{n_{L,j}, n_{R,j}\} \leq \frac{3}{2}$.

Finally and most crucially, we wish to show that

$$\mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] \geq 3\alpha\beta(w+1)^2 + 3. \tag{2}$$

By construction and the induction hypothesis, the expected subchunk sizes sum to at least $\alpha\beta w^2$ in stage 1, $\alpha\beta w^2 - \mathbb{E}\left[\min\{L_\kappa, R_\kappa\}\right]$ in stage 2b, and $\alpha\beta w^2$ in stage 3. To analyze stage 2a, recall the definitions $L_j := \sum_{i=1}^{\text{left}(j)} c_{L,i}$ and $R_j := \sum_{i=1}^{\text{right}(j)} c_{R,i}$ and $S_j := L_j - R_j$. Notice that the size of the jth subchunk of stage 2a, which is $n_{L,j}n_{R,j}/(n_{L,j} + n_{R,j}) = p_{L,j}n_{L,j}$, is precisely equal to the expectation of $L_j - L_{j-1}$ (conditioned on prior randomness). Thus, the expected sum of chunk sizes in stage 2a is $\mathbb{E}[L_\kappa]$, which by symmetry is equal to $\mathbb{E}[R_\kappa]$ and thus equal to $\frac{1}{2}\mathbb{E}[L_\kappa + R_\kappa]$. Combining these lower bounds for all stages, we obtain

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] &\geq \underbrace{\alpha\beta w^2}_{\text{stage 1}} + \underbrace{\alpha\beta w^2 - \mathbb{E}\left[\min\{L_\kappa, R_\kappa\}\right]}_{\text{stage 2b}} + \underbrace{\alpha\beta w^2}_{\text{stage 3}} + \underbrace{\frac{\mathbb{E}[L_\kappa + R_\kappa]}{2}}_{\text{stage 2a}} \\ &= 3\alpha\beta w^2 + \frac{1}{2}\,\mathbb{E}[|S_\kappa|].\end{aligned} \tag{3}$$

where the equation uses that $\min\{x, y\} = \frac{x+y-|x-y|}{2}$ for $x, y \in \mathbb{R}$. It remains to bound $\mathbb{E}[|S_\kappa|]$. To do so, we will use the following result about the convergence rate in the martingale central limit theorem.

Lemma 14 (Ibragimov [31]). *Let $\gamma, \eta > 0$ be constants. Let $S_0 = 0, S_1, S_2, \dots$ be a martingale and let $X_j = S_j - S_{j-1}$. If $|X_j| \leq \gamma$ and $\sum_{j=1}^{\infty} \mathbb{E}[X_j^2 | X_1, \dots, X_{j-1}] = \infty$ a.s., then for all $z \in \mathbb{R}$ and $\kappa = \min\left\{k\colon \sum_{j=1}^{k+1} \mathbb{E}[X_j^2 | X_1, \dots, X_{j-1}] \geq \eta^2\right\}$ it holds that*

$$\left|\Pr\left[S_\kappa < z\eta\right] - \Phi(z)\right| \leq 2\sqrt{\frac{\gamma}{\eta}}\left(1 + \frac{3}{2}\frac{\gamma}{\eta} + \frac{\gamma^2}{3\eta^2}\right),$$

where Φ is the standard normal distribution function.

Let us first calculate the conditional variances of the martingale difference terms $X_j = S_j - S_{j-1}$ in our setting:[8]

$$\mathbb{E}[X_j^2 \mid X_1, \ldots, X_{j-1}] = \frac{n_{R,j} n_{L,j}^2}{n_{L,j}+n_{R,j}} + \frac{n_{L,j} n_{R,j}^2}{n_{L,j}+n_{R,j}} = n_{L,j} n_{R,j} \in \left[\frac{1}{4}, \frac{9}{4}\right] \tag{4}$$

The premise $\sum_{j=1}^{\infty} \mathbb{E}[X_j^2 | X_1, \ldots, X_{j-1}] = \infty$ in Lemma 14 can easily be satisfied in our case by extending the martingale with additional terms after step κ.[9] We invoke Lemma 14 with $\gamma = \frac{3}{2}$ and $\eta = \frac{\sqrt{\alpha\beta}\cdot w}{2}$, so that $|X_j| \le \max\{n_{L,j}, n_{R,j}\} \le \gamma$ and the definition of κ in the lemma coincides with the one given in (1). Since $\sqrt{\alpha}\cdot(w+1) > 1$ by the assumption of the induction step, we then have $\frac{\gamma}{\eta} < \frac{3(w+1)}{w\sqrt{\beta}} = O\left(\frac{1}{\sqrt{\beta}}\right)$. Thus,

$$\begin{aligned}\mathbb{E}[|S_\kappa|] &\ge \eta \cdot (\Pr[S_\kappa < -\eta] + \Pr[S_\kappa \ge \eta]) \\ &\ge \eta \cdot \left(2\Phi(-1) - O\left(\beta^{-1/4}\right)\right) \ge \frac{\sqrt{\alpha\beta}\cdot w}{2} \cdot \Phi(-1) \end{aligned} \tag{5}$$

for a sufficiently large constant $\beta \in \mathbb{N}$.

Combining (3) and (5) and choosing α such that $\frac{\Phi(-1)}{4} = 9\sqrt{\alpha\beta}$, we obtain

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] &\ge 3\alpha\beta w^2 + 9\alpha\beta w = 3\alpha\beta(w+1)^2 + 3\alpha\beta(w-1) \\ &> 3\alpha\beta(w+1)^2 + \sqrt{\alpha}\beta = 3\alpha\beta(w+1)^2 + \frac{\Phi(-1)\sqrt{\beta}}{36} \\ &\ge 3\alpha\beta(w+1)^2 + 3\end{aligned}$$

where the strict inequality uses $\sqrt{\alpha}\cdot(w+1) > 1$ and $3(w-1) \ge w+1$, and the last inequality holds for a sufficiently large constant β. This completes the proof of Claim 13.

4.3 Combining subchunks

The next lemma shows that random subchunks $\tilde{\rho}_1, \ldots, \tilde{\rho}_{\tilde{m}}$ of sizes $\tilde{c}_i \le \tilde{c}_{\max}$ can be combined into chunks $\rho_1, \ldots, \rho_m$ of greater sizes $c_i \approx c_{\text{avg}}$, and so that $m = \left\lfloor \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] / c_{\text{avg}} \right\rfloor$. Note that the lemma is not specific to MSS, so it may also be used for constructing similar chunk decompositions for other problems with recursively defined lower bounds that might benefit from this technique.

Lemma 15. *Let $0 < \tilde{c}_{\max} \le c_{\text{avg}}$ be constants. Let $\tilde{\rho}_1, \tilde{\rho}_2, \ldots, \tilde{\rho}_{\tilde{m}}$ be a random sequence of (sub)chunks and let $\tilde{c}_1, \ldots, \tilde{c}_{\tilde{m}} \in [0, \tilde{c}_{\max}]$ be random variables such that $\tilde{c}_j$ is $\sigma(\tilde{\rho}_{\le j-1})$-measurable and*

$$\mathbb{E}\left[c_{\text{alg}}(\tilde{\rho}_j \mid \tilde{\rho}_{\le j-1}) \,\middle|\, \tilde{\rho}_{\le j-1}\right] \ge \tilde{c}_j \tag{6}$$

for any deterministic online algorithm alg. *Then there exists a random sequence $\rho_1, \ldots, \rho_m$ of chunks and random variables $c_1, \ldots, c_m$ such that:*

1. $\rho_{\le m} = \tilde{\rho}_{\le \tilde{m}}$,
2. *c_i is $\sigma(\rho_{\le i-1})$-measurable,*
3. $\mathbb{E}\left[c_{\text{alg}}(\rho_i \mid \rho_{\le i-1}) \,\middle|\, \rho_{\le i-1}\right] \ge c_i$ *for any deterministic online algorithm* alg,
4. $c_i \in [c_{\text{avg}} - \tilde{c}_{\max}, c_{\text{avg}} + \tilde{c}_{\max}]$,
5. $\mathbb{E}\left[\sum_{i=1}^{m} c_i\right] \ge \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] - c_{\text{avg}}$,
6. $m = \left\lfloor \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] / c_{\text{avg}} \right\rfloor$.

If (6) holds also when an escape price p_{esc} is available on the suffix $\tilde{\rho}_j$, then property 3 holds also when an escape price of at least $p_{\text{esc}} + c_{\text{avg}} + \tilde{c}_{\max}$ is available on the suffix ρ_i.

The proof of the induction step of Lemma 12 is completed by applying Lemma 15 with $\tilde{c}_{\max} = 3/2$, $c_{\text{avg}} = 3$ and $p_{\text{esc}} = 2\beta$ to the subchunks guaranteed by Claim 13. For $\beta \ge 2$, the escape price 6β in $\mathcal{M}_{w+1}$ is indeed at least $p_{\text{esc}} + c_{\text{avg}} + \tilde{c}_{\max}$.

Proof of Lemma 15. Let $C := \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right]$ and $m := \lfloor C / c_{\text{avg}} \rfloor$, satisfying property 6. We define the chunks ρ_i and sizes c_i based on random indices h_i as follows. Set for every $i = 0, 1, \ldots, m$,

$$h_i := \min\left\{h\colon \mathbb{E}\left[\sum_{j=h+1}^{\tilde{m}} \tilde{c}_j \,\middle|\, \rho_{\le h}\right] \le C - i \cdot c_{\text{avg}}\right\}.$$

Next, put

$$\rho_i := \begin{cases} \tilde{\rho}_{h_{i-1}+1 \le \cdot \le h_i} & \text{if } i \in \{1, \ldots, m-1\}, \\ \tilde{\rho}_{h_{m-1}+1 \le \cdot \le \tilde{m}} & \text{if } i = m. \end{cases}$$

Finally, put for all $i = 1, 2, \ldots, m$,

$$c_i := \mathbb{E}\left[\sum_{j=h_{i-1}+1}^{h_i} \tilde{c}_j \,\middle|\, \tilde{\rho}_{\le h_{i-1}}\right].$$

Note that $h_0 = 0 \le h_1 \le \cdots \le h_m \le \tilde{m}$, and therefore $\rho_{\le i} = \tilde{\rho}_{\le h_i}$ for $i \le m-1$, and $\rho_{\le m} = \tilde{\rho}_{\le \tilde{m}}$. In particular, c_i is $\sigma(\rho_{\le i-1})$-measurable by design. For any online algorithm alg,

$$\begin{aligned}&\mathbb{E}\left[c_{\text{alg}}(\rho_i \mid \rho_{\le i-1}) \,\middle|\, \rho_{\le i-1}\right] \\ &\ge \mathbb{E}\left[\sum_{j=h_{i-1}+1}^{h_i} \mathbb{E}\left[c_{\text{alg}}(\tilde{\rho}_j \mid \tilde{\rho}_{\le j-1}) \,\middle|\, \tilde{\rho}_{\le j-1}\right] \,\middle|\, \tilde{\rho}_{\le h_{i-1}}\right] \\ &\ge \mathbb{E}\left[\sum_{j=h_{i-1}+1}^{h_i} \tilde{c}_j \,\middle|\, \tilde{\rho}_{\le h_{i-1}}\right] = c_i. \end{aligned} \tag{7}$$

In the setting with escape prices this is still true, but inequality (7) requires a charging argument: Recall that we assume here that inequality (6) holds for algorithms that are allowed to escape during $\tilde{\rho}_j$ (but fails for algorithms that escape earlier). If A escapes for cost $p_{\text{esc}} + c_{\text{avg}} + \tilde{c}_{\max}$ during ρ_i, let $j^* \in \{h_{i-1}+1, \ldots, h_i\}$ be the index of the subchunk $\tilde{\rho}_{j^*}$ of ρ_i during which it escapes. We charge only p_{esc} of this cost $p_{\text{esc}} + c_{\text{avg}} + \tilde{c}_{\max}$ to the subchunk $\tilde{\rho}_{j^*}$, and we

[8] The first equation in (4) only makes sense if we consider the two sequences of chunks sampled in stage 2 as fixed (rather than random), so that the sizes $c_{L,i}$ and $c_{R,i}$ may be viewed as constants rather than random variables. Otherwise, $n_{L,j}$ and $n_{R,j}$ would not be measurable with respect to the σ-algebra generated by $X_1, \ldots, X_{j-1}$, which the expectation is conditioned on. Thus, the bound on $\mathbb{E}[|S_k|]$ that we will obtain in (5) holds for any fixed choice of the two chunk sequences sampled in stage 2. Therefore, it also holds when the expectation is taken over the randomness of these chunks as well.

[9] In the lemma, this condition merely serves to guarantee the existence of κ for any η. For the specific η that we will use, we justified the existence of κ already when we constructed stage 2a.

charge a "fake cost" of $\mathbb{E}[\tilde{c}_j \mid \tilde{\rho}_{\le h_{i-1}}]$ to each subsequent subchunk $\tilde{\rho}_j$ for $j = j^*+1, \dots, h_i$ (where the true cost of A would be 0 because the algorithm already escaped). So the total amount charged in this way is

$$p_{\mathrm{esc}} + \sum_{j=j^*+1}^{h_i} \mathbb{E}[\tilde{c}_j \mid \tilde{\rho}_{\le h_{i-1}}] \le p_{\mathrm{esc}} + c_i,$$

which does not overcharge the escape price of $p_{\mathrm{esc}} + c_{\mathrm{avg}} + \tilde{c}_{\max}$ provided that the bound $c_i \le c_{\mathrm{avg}} + \tilde{c}_{\max}$ of property 4 holds. For this charging scheme, inequality (7) follows by invoking (6) for $j \le j^*$ and substituting the fake charge $\mathbb{E}[\tilde{c}_j \mid \tilde{\rho}_{\le h_{i-1}}]$ for $j > j^*$.

To see that property 4 holds, note that

$$\begin{aligned} c_i &= \mathbb{E}\left[\sum_{j=h_{i-1}+1}^{h_i} \tilde{c}_j \,\middle|\, \tilde{\rho}_{\le h_{i-1}}\right] \\ &= \mathbb{E}\left[\sum_{j=h_{i-1}+1}^{\tilde{m}} \tilde{c}_j \,\middle|\, \tilde{\rho}_{\le h_{i-1}}\right] - \mathbb{E}\left[\mathbb{E}\left[\sum_{j=h_i+1}^{\tilde{m}} \tilde{c}_j \,\middle|\, \tilde{\rho}_{\le h_i}\right] \middle| \tilde{\rho}_{\le h_{i-1}}\right]. \end{aligned}$$

By definition of h_i and the assumption that $\tilde{c}_j \in [0, \tilde{c}_{\max}]$, the first term lies in $[C - (i-1)\cdot c_{\mathrm{avg}} - \tilde{c}_{\max}, C - (i-1)\cdot c_{\mathrm{avg}}]$ and the second in $[C - i\cdot c_{\mathrm{avg}} - \tilde{c}_{\max}, C - i\cdot c_{\mathrm{avg}}]$. Thus, their difference must lie in $[c_{\mathrm{avg}} - \tilde{c}_{\max}, c_{\mathrm{avg}} + \tilde{c}_{\max}]$.

Finally, property 5 holds because

$$\begin{aligned} \mathbb{E}\left[\sum_{i=1}^{m} c_i\right] &= \mathbb{E}\left[\sum_{i=1}^{h_m} \tilde{c}_i\right] \ge \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] - (C - m\cdot c_{\mathrm{avg}}) \\ &\ge \mathbb{E}\left[\sum_{i=1}^{\tilde{m}} \tilde{c}_i\right] - c_{\mathrm{avg}}, \end{aligned}$$

completing the proof. □

5 UNIVERSAL $\Omega(\log n)$ LOWER BOUND

We now prove the tight universal lower bounds for kSRV and MTS. The main technical contribution of this section is a lower bound on MSS in hierarchically separated tree metrics (HSTs). For definitions and further discussion, see the appendix of the full version of our paper. The role that HSTs play in proving universal lower bounds is a consequence of the following theorem.

Theorem 16 (Bartal, Linial, Mendel, Naor [10, Theorem 3.26]). *For every $\epsilon \in (0,1)$ there exists $\delta > 0$ such that for every $q \ge 1$, for every $n \in \mathbb{N}$, and for every n-point metric space, there exists an $n^{\delta/\log 2q}$-point subspace that is a q-HST up to bi-Lipschitz distortion $\le 2+\epsilon$.*

Corollary 17. *Let $q : \mathbb{N} \to [1,\infty)$. Suppose that for all $n \in \mathbb{N}$, that for all n-point $q(n)$-HSTs $\mathcal{H}$ we have that $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{H}, n) = \Omega(\log n)$. Then, for all $n \in \mathbb{N}$, for all n-point metric spaces $\mathcal{M}$, $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}, n-1) = \Omega(\log n / \log 2q(n))$.*

The previously best universal lower bound is implied by the $\Omega(\log n)$ lower bound of [8, Theorem 3] for all n-point $\Omega(\log^2 n)$-HSTs. Here we improve this lower bound by giving a lower bound of $\Omega(\log n)$ for all n-point 1-HSTs. The reason why the lower bound construction in [8, Theorem 3] works for $\Omega(\log^2 n)$-HSTs but not for 1-HSTs is that a stretch of $\Omega(\log^2 n)$ is needed to ensure that the cost of switching between subtrees is large relative to the cost of an inductive request sequence within a subtree. To overcome this, we use a similar idea to the one in the previous section of decomposing the recursive request sequence into smaller chunks. Apart from this, our construction is similar to the one in [8].

Lemma 18. *There is a constant $\alpha > 0$ such that for any 1-HST (a.k.a. an ultrametric space) $\mathcal{U}' = (U', d)$ there exists a subspace $\mathcal{U} = (U, d_{|U})$ of $\mathcal{U}'$ with $\mathrm{diam}(\mathcal{U}) = \mathrm{diam}(\mathcal{U}')$ and there exists a distribution $\mathcal{D}$ of request sequences in U satisfying the following properties for every initial location $s \in U$:*

- *For any deterministic online algorithm* alg*, and even if an escape price of $2\,\mathrm{diam}(\mathcal{U})$ is available:*
$$\mathbb{E}_{\rho\sim\mathcal{D}}\left[c_{\mathrm{alg},s}(\rho)\right] \ge \mathrm{diam}(\mathcal{U}) \ge \mathbb{E}_{\rho\sim\mathcal{D}}\left[c_{\mathrm{opt},s}(\rho)\right]. \tag{8}$$
- *There exists some $h(\mathcal{U}) \in \mathbb{N}$ such that for all $h \ge h(\mathcal{U})$:*
$$h\cdot\mathrm{diam}(\mathcal{U}) \ge \alpha\cdot\log|U'|\cdot\mathbb{E}_{(\rho_1,\dots,\rho_h)\sim\mathcal{D}^h}\left[c_{\mathrm{opt},s}(\rho_1\rho_2\dots\rho_h)\right]. \tag{9}$$

Before delving into the proof of Lemma 18, we state and prove its consequences, the main results of this section.

Theorem 19. *For all $n \in \mathbb{N}$, in* every *n-point metric space $\mathcal{M}$, $C^{\mathrm{MSS}}_{\mathrm{rand}}(\mathcal{M}, n-1) = \Omega(\log n)$.*

Proof. By Corollary 17, it is sufficient to prove lower bounds for all 1-HSTs. Given an n-point 1-HST $\mathcal{U}' = (U', d)$, Lemma 18 asserts that there exists $h \in \mathbb{N}$ and a probability distribution $\tilde{\rho}$ on request sequences (namely $\rho_1\rho_2\dots\rho_h$ in Inequality 9), such that for every deterministic algorithm alg,

$$\mathbb{E}[c_{\mathrm{alg}}(\rho) : \rho \sim \tilde{\rho}] \ge h\cdot\mathrm{diam}(\mathcal{U}') \ge \alpha\cdot\log n\cdot\mathbb{E}[c_{\mathrm{opt}}(\rho) : \rho\sim\tilde{\rho}],$$

where $\alpha > 0$ is an absolute constant. Therefore, $C^{\mathrm{MSS}}_{\mathrm{distr}}(\mathcal{U}', n) = \Omega(\log n)$, and the proof is concluded by Theorem 3. □

Corollary 20. *For all $k < n \in \mathbb{N}$, in* every *n-point metric space $\mathcal{M}$, $C^{\mathrm{MTS}}_{\mathrm{rand}}(\mathcal{M}) = \Omega(\log n)$ and $C^{\mathrm{kSRV}}_{\mathrm{rand}}(\mathcal{M}) = \Omega(\log k)$*

Proof. From Proposition 8, noting that the k-server problem in $\mathcal{M}$ is at least as hard as it is in any subspace of size $k+1$. □

These universal lower bounds are asymptotically tight due to the matching upper bounds known in some special metrics.

We may assume without loss of generality that all the internal node weights in the HST representation of $\mathcal{U}'$ are at least 1 (otherwise, just scale all the weights uniformly). By first rounding weights to powers of 2 and then contracting all edges whose incident vertices have the same weight, we may further assume without loss of generality (losing a factor 2 in the constant α) that the ratio between the weights of two adjacent internal nodes is always of the form 2^i for some integer $i \ge 1$. (In particular, the modified HST is a 2-HST.)

We prove Lemma 18 by induction on $n := |U'|$. The constant α will be determined later. For $n = 1$ the lemma is trivial. Suppose now that $n \ge 2$. In Section 5.1 we construct inductively the subspace

$\mathcal{U}$ and the probability distribution $\mathcal{D}$. In Section 5.2 we prove Inequality (8). Due to space limitations, the proof of Inequality (9) is deferred to the full version of our paper.

5.1 Construction

Let $\mathcal{U}'_1, \mathcal{U}'_2, \ldots$ be the ultrametric spaces corresponding to the subtrees rooted at children of the root of $\mathcal{U}'$, and let $n_i := |U'_i|$, where $\mathcal{U}'_i = (U'_i, d_{|U'_i})$. We sort the subtrees so that $n_1 \geq n_2 \geq \ldots$. Due to [8, Proposition 11] (see also the full version of our paper), any such sequence of numbers satisfies one of the following two cases: either $\sqrt{n_1} + \sqrt{n_2} \geq \sqrt{n}$, or there exists $\ell \geq 3$ such that $\ell \cdot \sqrt{n_\ell} \geq \sqrt{n}$. In the former case, we let $\ell := 2$ and we call this the *binary case*. The latter case is called the *balanced case* because the proof will invoke the same lower bound $\alpha \log n_\ell$ for each of the first ℓ subtrees.

In the balanced case, if additionally $\log \ell \geq 2\alpha n_\ell$, then we set $U_i = \{x_i\}$ for all $i \in \{1, 2, \ldots, \ell\}$, where $x_i \in U'_i$ is an arbitrary point. We call this the special case of the balanced case the *uniform case*. In all other cases, we set $U_i \subseteq U'_i$ to be the set of points of the subspace $\mathcal{U}_i$ of $\mathcal{U}'_i$ that the induction hypothesis stipulates, for all $i \in \{1, 2, \ldots, \ell\}$. Finally, we let $U := \bigcup_{i=1}^{\ell} U_i$. Notice that in the uniform case, $\mathcal{U}$ is an ℓ-point uniform metric space.

For convenience, we will use in this proof the convention that each request specifies the set of points where the algorithm must *not* be, i.e., the complement of the sets we used earlier. This means that if we issue an inductive request sequence in some U_i, then any algorithm located in some U_j for $j \neq i$ does not incur any cost. Note that the entire proof is written in a manner that restricts the request sequence to the $\mathcal{U}$ subspace. To apply the lower bound to the original $\mathcal{U}'$ space, we need under the above convention to add to each request in the sequence all the points in $U' \setminus U$.

If $|U_i| \geq 2$ (so that $\operatorname{diam}(\mathcal{U}_i) > 0$), let $\mathcal{D}_i$ be the distribution on request sequences in U_i induced by the induction hypothesis. In this case, we call a *chunk in* U_i a concatenation of $\frac{\operatorname{diam}(\mathcal{U})}{2\operatorname{diam}(\mathcal{U}_i)}$ independent random request sequences from $\mathcal{D}_i$. We call each such sequence from $\mathcal{D}_i$ a *subchunk*. If $|U_i| = 1$, we define a *chunk in* U_i to be either the empty request sequence or a single request to the singleton point in U_i, chosen with probabilitiy $1/2$ each. A random request sequence in $\mathcal{D}$ is obtained by repeating the following 2ℓ times independently: Choose $i \in \{1, \ldots, \ell\}$ uniformly at random, then issue a chunk in U_i. See Figure 4 for an illustration of these definitions.

5.2 Proof of inequalities (8)

For the first inequality, it suffices to show that we can charge expected cost at least $\frac{\operatorname{diam}(\mathcal{U})}{2\ell}$ to each chunk. Note that if the algorithm invokes the escape price of $2\operatorname{diam}(\mathcal{U})$, the true cost on all subsequent chunks would be 0. But in this case we charge only cost $\operatorname{diam}(\mathcal{U})$ instead of the full escape price $2\operatorname{diam}(\mathcal{U})$ to the chunk on which the escape price is invoked. We can then use the uncharged escape price to charge $\frac{\operatorname{diam}(\mathcal{U})}{2\ell}$ to each subsequent chunks. This charging scheme does not overcharge the escape price, and guarantees that we can charge the claimed amount even to chunks starting after the algorithm has already escaped. It remains

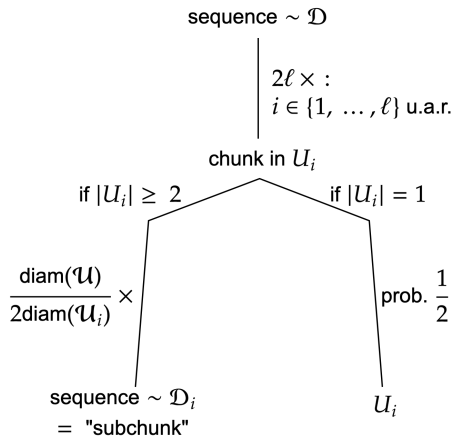

Figure 4: Construction of sequences in $\mathcal{D}$ from chunks and subchunks in the universal lower bound.

to consider chunks that are issued when the algorithm has not yet escaped.

Let $\mathcal{U}_i$ be the subtree where the online algorithm alg is located before a chunk is issued. With probability $\frac{1}{\ell}$, the chunk is issued in U_i, so it suffices to show that conditioned on this being the case, alg has to pay at least $\operatorname{diam}(\mathcal{U})/2$ in expectation on this chunk. If $|U_i| = 1$, this is trivial (as alg has to move elsewhere in U). If $|U_i| \geq 2$, we argue that we can charge an expected cost of $\operatorname{diam}(\mathcal{U}_i)$ to each of the $\frac{\operatorname{diam}(\mathcal{U})}{2\operatorname{diam}(\mathcal{U}_i)}$ subchunks of the chunk. If alg stays inside U_i during the chunk, we charge to each subchunk the cost that is actually suffered on it.

Now suppose that alg invokes the escape price $2\operatorname{diam}(\mathcal{U})$ or switches to some U_j with $j \neq i$. We call the or-combination of these events the *escape-or-switch* event. Such a move incurs a one-time cost of $\operatorname{diam}(\mathcal{U})$. The reason is that this is the cost of switching from U_i to U_j. Recall that in the case of escape we charge only $\operatorname{diam}(\mathcal{U})$ of the actual escape cost $2\operatorname{diam}(\mathcal{U})$ to this chunk, and the true cost on the remaining subchunks would be 0. Of this cost $\operatorname{diam}(\mathcal{U})$, we charge $2\operatorname{diam}(\mathcal{U}_i)$ as escape cost to the subchunk where the escape-or-switch happens, and we charge $\operatorname{diam}(\mathcal{U}_i)$ to all remaining at most $\frac{\operatorname{diam}(\mathcal{U})}{2\operatorname{diam}(\mathcal{U}_i)} - 1$ subchunks belonging to this chunk (whose true cost would have been 0). So, in total we charge at most $\frac{\operatorname{diam}(\mathcal{U})}{2} + \operatorname{diam}(\mathcal{U}_i) \leq \operatorname{diam}(\mathcal{U})$ for an escape-or-switch, and in particular we do not overcharge.

Given this charging scheme, the induction hypothesis implies that the expected cost charged to each subchunk is at least $\operatorname{diam}(\mathcal{U}_i)$. As there are $\frac{\operatorname{diam}(\mathcal{U})}{2\operatorname{diam}(\mathcal{U}_i)}$ subchunks in the chunk, the expected cost of a chunk in the subtree $\mathcal{U}_i$ where alg is located at the start of the chunk is at least $\operatorname{diam}(\mathcal{U})/2$, as desired. This completes the proof of the first inequality in (8).

To see the second inequality in (8), consider the following algorithm (which is actually an online algorithm and therefore shows that our analysis for the first inequality was tight): Let $\mathcal{U}_i$ be the subtree where the algorithm is currently located. If $|U_i| \geq 2$,

then the algorithm stays in U_i and plays according to the induction hypothesis. If $|U_i| = 1$, then the algorithm switches to a different U_j only if U_i is requested. For this algorithm, a chunk in the subtree $\mathcal{U}_i$ where the algorithm is located incurs expected cost $\mathrm{diam}(\mathcal{U})/2$. As it happens twice in expectation during a random $\rho \sim \mathcal{D}$ that a chunk is issued in the subtree where the algorithm is located, the second inequality in (8) follows.

The proof of inequality (9), which proceeds separately for the uniform, balanced non-uniform, and binary case, is deferred to the full version of our paper.

ACKNOWLEDGEMENTS

We wish to thank Yossi Azar, Yair Bartal, and Manor Mendel for pointing out relevant references, and the anonymous reviewers of STOC '23 for useful suggestions.

REFERENCES

[1] A. Andoni, M. S. Charikar, O. Neiman, and H. L. Nguyen. Near linear lower bound for dimension reduction in ℓ_1. In *Proc.of the 52nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 315–323, 2011.
[2] B. Awerbuch, Y. Bartal, and A. Fiat. Heat & Dump: Competitive distributed paging. In *Proc. of the 34th Ann. IEEE Symp. on Foundations of Computer Science*, pages 22–31, 1993.
[3] N. Bansal, N. Buchbinder, A. Madry, and J. Naor. A polylogarithmic-competitive algorithm for the k-server problem. In *Proc. of the 52nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 267–276, October 2011.
[4] N. Bansal, N. Buchbinder, and J. Naor. A primal-dual randomized algorithm for weighted paging. In *Proc. of the 48th Ann. IEEE Symp. on Foundations of Computer Science*, pages 507–517, 2007.
[5] N. Bansal, N. Buchbinder, and J. Naor. Towards the randomized k-server conjecture: A primal-dual approach. In *Proc. of the 21st Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 40–55, 2010.
[6] N. Bansal and C. Coester. Online metric allocation and time-varying regularization. In *Proc. of the 30th Ann. European Symp. on Algorithms*, pages 13:1–13:13, 2022.
[7] Y. Bartal, A. Blum, C. Burch, and A. Tomkins. Polylog(n)-competitive algorithm for metrical task systems. In *Proc. of the 29th Ann. ACM Symp. on Theory of Computing*, pages 711–719, 1997.
[8] Y. Bartal, B. Bollobas, and M. Mendel. A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In *Proc. of the 42nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 396–405, 2001.
[9] Y. Bartal, A. Fiat, and Y. Rabani. Competitive algorithms for distributed data management. In *Proc. of the 24th Ann. ACM Symp. on Theory of Computing*, pages 39–50, 1992.
[10] Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric Ramsey-type phenomena. In *Proc. of the 35th Ann. ACM Symp. on Theory of Computing*, pages 463–472, June 2003.
[11] W. Bein, M. Chrobak, and L. L. Larmore. The 3-server problem in the plane. *Theor. Comput. Sci.*, 289(1):335–354, 2002.
[12] M. Bienkowski, J. Byrka, C. Coester, and Ł. Jeż. Unbounded lower bound for k-server against weak adversaries. In *Proc. of the 52nd Ann. ACM Symp. on Theory of Computing*, pages 1165–1169, 2020.
[13] A. Blum, H. J. Karloff, Y. Rabani, and M. E. Saks. A decomposition theorem and bounds for randomized server problems. In *Proc. of the 33rd Ann. IEEE Symp. on Foundations of Computer Science*, pages 197–207, 1992.
[14] A. Borodin, N. Linial, and M. E. Saks. An optimal online algorithm for metrical task systems. In *Proc. of the 19th Ann. ACM Symp. on Theory of Computing*, pages 373–382, 1987.
[15] B. Brinkman and M. Charikar. On the impossibility of dimension reduction in l_1. In *Proc. of the 44th Ann. IEEE Symp. on Foundations of Computer Science*, pages 514–523, 2003.
[16] S. Bubeck, C. Coester, and Y. Rabani. Shortest paths without a map, but with an entropic regularizer. In *Proc. of the 63rd Ann. IEEE Symp. on Foundations of Computer Science*, 2022.
[17] S. Bubeck, M. B. Cohen, J. R. Lee, and Y.-T. Lee. Metrical task systems on trees via mirror descent and unfair gluing. In *Proc. of the 30th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 89–97, 2019.
[18] S. Bubeck, M. B. Cohen, Y.-T. Lee, J. R. Lee, and A. Madry. k-server via multiscale entropic regularization. In *Proc. of the 50th Ann. ACM Symp. on Theory of Computing*, pages 3–16, 2018.
[19] M. Chrobak, H. J. Karloff, T. Payne, and S. Vishwanathan. New results on server problems. *SIAM J. Discret. Math.*, 4(2):172–181, 1991.
[20] M. Chrobak and L. L. Larmore. An optimal on-line algorithm for k-servers on trees. *SIAM J. Comput.*, 20(1):144–148, 1991.
[21] C. Coester. Personal communication, 2022.
[22] C. Coester and E. Koutsoupias. The online k-taxi problem. In *Proc. of the 51st Ann. ACM Symp. on Theory of Computing*, pages 1136–1147, 2019.
[23] C. Coester and E. Koutsoupias. Towards the k-server conjecture: A unifying potential, pushing the frontier to the circle. In *Proc. of the 48th Int'l Colloq. on Automata, Languages, and Programming*, pages 57:1–57:20, 2021.
[24] C. Coester and J. R. Lee. Pure entropic regularization for metrical task systems. In *Proc. of the 32nd Conf. on Learning Theory*, pages 835–848, 2019.
[25] J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.
[26] A. Fiat, D. P. Foster, H. J. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan. Competitive algorithms for layered graph traversal. In *Proc. of the 32nd Ann. IEEE Symp. of Foundations of Computer Science*, pages 288–297, 1991.
[27] A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *J. Alg.*, 12(4):685–699, 1991.
[28] A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. In *Proc. of the 32nd Ann. ACM Symp. on Theory of Computing*, pages 725–734, May 2000.
[29] A. Fiat, Y. Rabani, and Y. Ravid. Competitive k-server algorithms. In *Proc. of the 31st Ann. IEEE Symp. on Foundations of Computer Science*, pages 454–463, 1990.
[30] Z. Huang and H. Zhang. Deterministic 3-server on a circle and the limitation of canonical potentials, 2022.
[31] I. A. Ibragimov. A central limit theorem for a class of dependent random variables. *Theory Probab. Appl.*, 8(1):83–89, 1963. [*Teor. Veroyatnost. i Primenen.*, 8(1):89–94, 1963].
[32] M. Imase and B. M. Waxman. Dynamic Steiner tree problem. *SIAM J. Discret. Math.*, 4:369–384, 1991.
[33] S. Irani and S. S. Seiden. Randomized algorithms for metrical task systems. In *Proc. of the 4th Int'l Workshop on Algorithms and Data Structures*, volume 955 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 1995.
[34] A. R. Karlin, M. S. Manasse, L. A. McGeoch, and S. S. Owicki. Competitive randomized algorithms for nonuniform problems. *Algorithmica*, 11(6):542–571, 1994.
[35] H. J. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion planning algorithms. In *Proc. of the 23rd Ann. ACM Symp. on Theory of Computing*, pages 278–288, 1991.
[36] A. P. Kosoresow. *Design and Analysis of Online Algorithms for Mobile Server Applications*. PhD thesis, Stanford University, 1996.
[37] E. Koutsoupias. The k-server problem. *Comput. Sci. Rev.*, 3(2):105–118, 2009.
[38] E. Koutsoupias and C. H. Papadimitriou. On the k-server conjecture. In *Proc. of the 26th Ann. ACM Symp. on Theory of Computing*, pages 507–511, 1994.
[39] E. Koutsoupias and C. H. Papadimitriou. The 2-evader problem. *Inf. Process. Lett.*, 57(5):249–252, 1996.
[40] J. R. Lee. Fusible HSTs and the randomized k-server conjecture. In *Proc. of the 59th Ann. IEEE Symp. on Foundations of Computer Science*, pages 438–449, 2018.
[41] J. R. Lee and A. Naor. Embedding the diamond graph in l_p and dimension reduction in l_1. *Geometric and Functional Analysis*, 14(4):745–747, 2004.
[42] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proc. of the 20th Ann. ACM Symp. on Theory of Computing*, pages 322–333, 1988.
[43] M. Mendel and A. Naor. Maximum gradient embeddings and monotone clustering. In *Proc. of the 10th Int'l Conf. on Approximation Algorithms for Combinatorial Optimization Problems, and the 11th Int'l Conf. on Randomization and Computation*, pages 242–256, 2007.
[44] I. Newman and Y. Rabinovich. A lower bound on the distortion of embedding planar metrics into Euclidean space. In *Proc. of the 18th Ann. Symp. on Computational Geometry*, pages 94–96, 2002.
[45] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. of the 16th Int'l Colloq. on Automata, Languages and Programming*, pages 610–620, 1989.
[46] H. Ramesh. On traversing layered graphs on-line. In *Proc. of the 4th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 412–421, 1993.
[47] S. S. Seiden. Unfair problems and randomized algorithms for metrical task systems. *Inf. Comput.*, 148(2):219–240, 1999.
[48] S. S. Seiden. A general decomposition theorem for the k-server problem. In *Proc. of the 9th Ann. European Symposium on Algorithms*, volume 2161 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2001.
[49] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. In *Proc. of the 16th Ann. ACM Symp. on Theory of Computing*, 1984.

Received 2022-11-07; accepted 2023-02-06