Online Monotone Metric Embeddings

Christian Coester
Department of Computer Science, University of Oxford, UK

Yichen Huang
John A. Paulson School of Engineering and Applied Sciences,
Harvard University, Cambridge, MA, USA

Abstract

Metric embeddings into structured spaces, particularly hierarchically well-separated trees (HSTs), are a fundamental tool in the design of online algorithms. In the classical online embedding setting, points arrive sequentially and must be embedded irrevocably upon arrival, resulting in strong distortion lower bounds of $\Omega(\min(n, \log n \log \Delta))$, where n is the number of points and Δ their aspect ratio.

We propose a novel relaxation, *online monotone metric embeddings*, which allows distances between embedded points in the target space to decrease monotonically over time. Such relaxed embeddings remain compatible with many online algorithms. Moreover, this relaxation breaks existing lower bound barriers, enabling embeddings into HSTs with distortion $O(\log^2 n)$.

We also study a dynamic variant, where points may both arrive and depart, seeking distortion guarantees in terms of the maximum number l of simultaneously present points. For traditional embeddings, such bounds are impossible, and this limitation persists even for deterministic monotone embeddings. Surprisingly, probabilistic monotone embeddings allow for $O(l \log l)$ distortion, which is nearly optimal given an $\Omega(l)$ lower bound.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online Algorithms, Metric Embeddings, k-Taxi

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.66

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2604.27059`

Funding *Christian Coester*: Funded by the European Union (ERC, CCOO, 101165139). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.
Yichen Huang: Supported by NSF grant CNS-2107078.

Acknowledgements The authors thank Michael Mitzenmacher for insightful comments on earlier versions of this paper, and the anonymous reviewers for their constructive feedback.

1 Introduction

Metric embeddings are a powerful tool in many online and approximation algorithms. By embedding a metric space into a structured one with small distortion, one can solve algorithmic problems on the simpler metric and then translate solutions back to the original space with bounded loss. Here we focus on embeddings into *hierarchically well-separated trees (HSTs)*, a widely studied target metric space with numerous algorithmic applications [3, 22].

When applying metric embeddings in the context of online algorithms, the traditional approach embeds the entire underlying metric into an HST offline, subsequently solving the problem there [5, 3, 22]. However, the distortion of these embeddings inherently depends on


the size of the metric space M (e.g., at least $\Omega(\log |M|)$ [3]), which can be infinite even for simple metrics like the real line. This motivates the study of *online embeddings*, which only embed points relevant to the request sequence and thus involve only finitely many points.

Online Embeddings and Their Challenges. In an online embedding, points arrive sequentially. Upon arrival of each new point v, the embedding algorithm must irrevocably place v into the target metric without knowledge of future points, and it cannot later move points in this embedding under traditional models [27, 5, 9]. This restriction leads to strong distortion lower bounds: a deterministic lower bound of $2^{\Omega(n)}$ [33] and a randomized lower bound of $\widetilde{\Omega}(\min(n, \log n \log \Delta))$ [27, 5], where n is the length of the sequence and Δ the aspect ratio, i.e., the ratio between the maximum and minimum nonzero distances between points. Since Δ can be unbounded, these bounds are exponentially larger than those in the offline setting, which admits deterministic distortion $O(n)$ and randomized distortion $O(\log n)$.

Monotone Metric Embeddings: A Mild but Powerful Relaxation. Our starting observation is that requiring a fully fixed embedding at each step is often unnecessarily restrictive. In many algorithmic settings, permitting distances between already-embedded points to *decrease* over time is a benign change that would not degrade performance. For example, with a *non-contractive* embedding that never underestimates distances, decreasing distances among existing points only improves the approximation quality. As a consequence, many online metric algorithms remain competitive even when paired with an embedding that occasionally shrinks distances, and we provide a meta-theorem for verifying the compatibility of potential-based algorithms.

This motivates a new model that we call *online monotone metric embeddings*: upon arrival of a new point, previously embedded points may be repositioned, provided that no pairwise distance between already-embedded points increases. This additional flexibility turns out to evade the lower bounds known for the traditional model. Specifically, we achieve probabilistic online monotone embeddings into HSTs with distortion $O(\log^2 n)$, eliminating the dependence on Δ and approaching the offline embedding distortion of $\Theta(\log n)$ [22]. We also obtain tight $\Theta(n)$ bounds on *deterministic* online monotone embeddings, improving exponentially over the strict[1] setting.

Dynamic Embeddings with Removals. Another natural question is whether an online embedding must always maintain an embedding of *all* points ever introduced. In practice, only a small set of *alive* points may matter at any given time. For example, in metrical service systems (i.e., set chasing problems) [13, 34, 19], only the points in the current and previous request sets matter. In k-server or k-taxi problems, points requested long ago intuitively lose relevance over time. Motivated by such scenarios, we examine a *dynamic* embedding model where points may arrive and depart, such that at most l points are alive (arrived and have not departed) at any time. The algorithm needs only to maintain an embedding of the alive set.

Previous literature on the k-server problem has attempted to employ such an approach, with partial success: [14] achieved a polylog(k, Δ)-competitive algorithm for the k-server problem using an evolving HST embedding. Building upon this, [30] aimed to further reduce the competitive ratio to polylog(k) by refining the dynamic embedding, and while this work

[1] We refer to the traditional embedding model, where distances in the target metric must be fixed upon arrival, as the *strict* embedding model.

contained several promising ideas, it was later withdrawn due to a bug in the proof. Despite this interest, a systematic exploration of dynamic embedding models remains limited. We investigate what distortion as a function of l, if any, is achievable in the dynamic embedding setting.

In the strict setting, known results on embedding graphs of pathwidth l [31] imply an *offline* embedding with distortion $O((4l)^{l^3+1})$ into trees. For *strict online* embeddings into HSTs, even such exponential distortion is unachievable: for $l = 3$, the distortion is an unbounded function of n. For online monotone embeddings, the same impossibility still holds in the case of deterministic embeddings. Surprisingly, we show that the combination of monotonicity and randomization removes these barriers, enabling a distortion of $O(l \log l)$. We further establish an almost matching lower bound of $\Omega(l)$ for probabilistic embeddings, which holds even offline. This lower bound may be of independent interest, particularly for future research seeking to narrow the gap in the randomized competitive ratio of the k-server problem. It could inspire new lower bound constructions, or at least constrain possible approaches when aiming for a polylog(k)-competitive algorithm.

In summary, monotone metric embeddings offer a natural relaxation of the online embedding model that enables overcoming known lower bounds, often exponentially. To the best of our knowledge, this notion of "monotone recourse" is new and may have broader applicability to other online problems. We view this conceptual idea as our main contribution. Some of our results follow from relatively simple adaptations of existing techniques, illustrating that monotone recourse can be addressed within known algorithmic frameworks.

1.1 Our Results

We will refer to the notion of an update sequence, formally defined in Definition 10, which specifies the arrival and departure of points. We denote by n the total number of points in the sequence, and by l the *width*, the maximum number of points simultaneously present at any time.

Our embedding results are presented in two categories: the *incremental* setting[2], in which points only arrive and the distortion is analyzed as a function of n, and the *fully dynamic* setting, in which points arrive and depart and the distortion is analyzed as a function of l. Finally, we discuss some applications.

1.1.1 Incremental Setting

We first present improved probabilistic embeddings for the classical incremental model when monotone updates are allowed.

▶ **Theorem 1** (Probabilistic Embedding). *For every $n \in \mathbb{N}$, there exists a probabilistic online monotone embedding of up to n points from any metric space into HSTs with distortion $O(\log^2 n)$. If the points are from an $O(1)$-dimensional normed space, the distortion improves to $O(\log n)$.*

Although Theorem 1 requires prior knowledge of n, we can employ it in our applications (Theorems 7 and 8) even in situations where n is unknown by using a standard guess-and-double approach. For the pure embedding question with unknown n, a modified algorithm

[2] Although our incremental algorithms can accommodate departures, when focusing on an n-dependent bound, having departures only simplifies the task by reducing the number of alive points.

gives distortion $O(\log^2 n \log\log n)$. Note that the $\Omega(\log n)$ lower bound for offline embeddings extends to our setting, so our bounds for constant-dimensional normed spaces are tight, whereas a quadratic gap remains for general metrics.

We also analyze deterministic online monotone embeddings, obtaining tight bounds. Recall that the optimal distortion for deterministic strict online embeddings is exponential in n.

▶ **Theorem 2** (Deterministic Embedding). *For every $n \in \mathbb{N}$, there is a deterministic online monotone embedding of up to n points from any metric space into HSTs with distortion $n-1$ when n is known. Without prior knowledge of n, the distortion is $n \cdot \widetilde{\Theta}(\log n)$.*[3] *Both bounds are tight.*

1.1.2 Fully Dynamic Setting

In the fully dynamic model, points may both arrive and depart. We first present negative results demonstrating that strict dynamic embeddings and deterministic monotone embeddings cannot achieve bounded distortion purely in terms of the width l.

▶ **Theorem 3** (Dynamic Impossibility). *Even for update sequences of width 3, any deterministic strict dynamic embedding into HSTs incurs distortion $\Omega(2^n)$, and any probabilistic strict embedding incurs distortion $\Omega(n)$. Furthermore, any deterministic monotone embedding into HSTs incurs distortion $\Omega(n)$ for update sequences of width 3.*

Despite this deterministic impossibility, randomization surprisingly enables embeddings whose distortion is bounded by the width l:

▶ **Theorem 4** (Probabilistic Dynamic Embedding). *There exists a probabilistic online monotone embedding from any metric space into HSTs with distortion $O(l \log l)$, where l is the width of the sequence. If the points are from an $O(1)$-dimensional normed space, the distortion improves to $O(l)$. These results hold without prior knowledge of l.*

We complement this result with an almost matching lower bound of $\Omega(l)$. Our lower bound is constructed on the line and holds even offline. The proof is short but non-trivial, and may be of interest for future research aimed at closing the gap in the randomized competitive ratio for the k-server problem.

▶ **Theorem 5** (Dynamic Lower Bound). *For every $l \in \mathbb{N}$, there exists an update sequence of width l on the line such that any probabilistic monotone embedding of the sequence into HSTs incurs distortion $\Omega(l)$, even if the entire sequence is known in advance.*

1.1.3 Applications

Finally, we demonstrate some applications of our embeddings for online algorithms. To give a systematic meta-theorem, we consider online algorithms whose analysis is based on a potential function. Note that the competitiveness of an online algorithm is *equivalent* to the existence of a corresponding potential function [6].[4] Intuitively, the potential value quantifies the amount of disadvantage of the online algorithm's configuration. A natural property of potential functions, which is typically satisfied (cf. Section 6.2), is to be *non-decreasing* in

[3] $\widetilde{\Theta}$ hides $\log\log n$ factors.
[4] However, it can be difficult to find an explicit expression for the potential.

distances of the metric space. The following theorem, stated more formally in Theorem 30, shows that this condition suffices in order for an online algorithm to be compatible with our monotone embeddings.

▶ **Theorem 6** (Application, Informal). *Consider an online problem Π on a metric N. If there exists an online monotone embedding from N into a family of metrics $\mathcal{M}$ with distortion λ, and a ρ-competitive algorithm for Π on every metric $M \in \mathcal{M}$ using a potential function that is monotone non-decreasing in distances, then there is a $\rho\lambda$-competitive algorithm for Π on N.*

As examples, we recover a result for k-server in [5], and give new applications to the k-taxi problem.

▶ **Theorem 7** (k-Server). *There is an $O(\log^2 k \log^2 n)$-competitive algorithm for the k-server problem on general metrics and an $O(\log^2 k \log n)$-competitive algorithm on $O(1)$-dimensional normed spaces, where n is the number of requested locations.*

The $O(\log^2 k \log^2 n)$ bound matches the guarantee in [5], where it was obtained by combining an $O(\log^2 k)$-competitive HST algorithm with an online embedding of distortion $O(\log n \log \Delta)$ paired with algorithm combination techniques (applicable to problems that admit a so-called min-operator) and an additional "baseline" algorithm, which allows to replace Δ by n in the overall competitiveness. Our algorithm offers an alternative perspective on this result, where the $O(\log^2 n)$ factor in the competitiveness comes directly from the embedding, without requiring a min-operator or a baseline algorithm.

Similarly, our embedding yields an alternative, more direct method to recover an $O(\log^2 n)$-competitive algorithm (Corollary 31) for the subadditive constrained forest problem [25, 5].

For the k-taxi problem, we obtain the following new result.

▶ **Theorem 8** (k-Taxi). *There is an $O(2^k \log^2 n)$-competitive algorithm for the k-taxi problem on general metrics, and an $O(2^k \log n)$-competitive algorithm for $O(1)$-dimensional normed spaces, where n is the number of requested locations.*

We note that in previous guarantees for the k-taxi problem [17, 11, 15, 26], n refers instead to the total number of points in the underlying metric space (including those that are never requested), which is infinite even for simple metrics like the line; see Section 1.2 for details on previous results.

Our dynamic embedding with distortion $O(l \log l)$ also suggests a potential avenue for addressing the long-standing open question of whether there exists an algorithm for the k-taxi problem whose competitive ratio depends only on k. In particular, Theorem 32 demonstrates that it suffices to maintain online a set of $g(k)$ points such that the optimal *offline* algorithm restricted to these points achieves an $h(k)$-approximation.

1.2 Related Work

Embeddings into HSTs. Offline, any metric of n points can be embedded into an HST with $O(\log n)$ distortion [22]. For the online model, [27] adapted the offline algorithm of [3] to achieve $O(\log n \log \Delta)$ distortion, and [5] provides an almost matching lower bound of $\widetilde{\Omega}(\log n \log \Delta)$ on probabilistic embeddings into HSTs. For a distortion allowed to depend on n only, it becomes $\Omega(n)$ [27]. For metrics with doubling dimension DDIM, [9] achieves $O(\text{DDIM} \log \Delta)$-distortion online embeddings into HSTs.

Online Problems and Algorithms. The k-server problem is one of the most prominent problems in the field of online algorithms. Deterministically, the competitive ratio is $\Theta(k)$ [32, 29]. The paper [14] achieved a randomized competitive ratio of $O(\log^2 k)$ on HSTs, which implies an $O(\log^2 k \log n)$-competitive algorithm on any n-point metric. The reader is referred to [28] for a survey on the k-server problem, and [12] for more recent results. The latter [12] also shows that the approach via HST embeddings is in fact optimal for the related metrical task systems problems on general metrics, despite the distortion loss.

The k-taxi problem is a generalization of the k-server problem [23]. Each request in the k-taxi problem is a pair of points (s, t), requesting a taxi to first come to s and then to t. In the hard version of the problem, the cost is the total distance taxis travel without a customer. This hard version has an $\Omega(2^k)$ lower bound on the competitive ratio for randomized algorithms against adaptive adversaries [17] and a few (incomparable) upper bounds: $O(2^k \log n)$, $O((n \log k)^2 \log n)$, $2^{O(\sqrt{\log k \log \Delta})} \log_\Delta n$, and $O(\log^3 \Delta \log^2(nk\Delta))$ [17, 11, 15, 26], all of which use n for the number of points in the entire metric space. On general metrics (with infinitely many points), competitive algorithms are known only for $k \leq 3$ [18].

When applying online embeddings with online algorithms, [5] explains a framework to bypass the dependency on the aspect ratio Δ at the expense of another $O(\log n)$ factor for problems that admit min operators and belong to a class they call *abstract network design problem*. Our results apply to a class of *metrical request-answer games* that includes abstract network design and other metrical request-answer games.

Online Algorithms with Recourse. The idea of allowing some changes to previous actions is related to the notion of recourse in online and dynamic algorithms, with the focus on bounding the *number* or *cost* of changes that an algorithm makes [8, 7, 35, 9]. Unlike typical models with recourse, where allowing changes introduces a tradeoff or cost, our monotone updates only decrease distances and hence can be viewed as benign flexibility rather than true recourse. We therefore view this as a free monotone relaxation rather than a standard recourse model.

Dynamic Algorithms. The field of *dynamic algorithms* also studies problems whose input is a sequence of arrivals and departures (e.g., of nodes or edges in a graph) while a solution to the current instance has to be maintained [8, 7, 35, 21, 24]. The goal in these problems is typically to minimize update time when computing new solutions, without restrictions on the type of changes allowed when updating the solution. In contrast, we allow only monotone changes but do not impose any restrictions on the running time, as is common in the field of online algorithms.

1.3 Organization

Section 2 defines the notions and the models and provides some useful lemmas. Section 3 explains the main technical ideas and a general framework for our algorithms. Section 4 and 5 prove the main results for the incremental setting and the fully dynamic setting, respectively. Section 6 introduces the metrical request-answer game and proves applications for our embedding. Section 7 contains concluding remarks and highlights future directions. In this version, we focus on probabilistic embeddings from general metrics; embeddings from normed spaces and deterministic monotone embeddings are presented in the full version.

2 Preliminaries

For a set S, let $\mathcal{D}(S)$ be the set of probability distributions over S. We use $\mathbb{P}(\cdot)$ to denote probabilities and $\mathbb{E}[\cdot]$ for expectations. For a sequence $a \in A^*$, the notation $a_{[i,j]}$ denotes the subsequence $a_i, \dots, a_j$.

▶ **Definition 9** (HSTs). *For $\mu \geq 1$, a* μ-hierarchically well-separated tree *(HST) is a metric space whose points are the leaves of a rooted tree T. Each node*[5] *v of T has a weight $\varphi(v) \geq 0$, with $\varphi(v) = 0$ if and only if v is a leaf, and if v is a child of u, then $\varphi(v) \leq \varphi(u)/\mu$. The distance between two leaves u and v is given by $d_T(u,v) = \varphi(\mathrm{lca}(u,v))$, where $\mathrm{lca}(u,v)$ denotes the least common ancestor of u and v.*

Throughout the paper, we refer to an original metric space (X, d_X) and consider embeddings of finite subsets $V \subseteq X$ into HSTs. Our probabilistic embedding will fix $\mu = 2$. Unless specified otherwise, $d(u,v)$ (without a subscript) refers to the distance in X. We write $d_{\max}(V) = \max_{u,v \in V} d(u,v)$ for the diameter of V, and $d_{\min}(V) = \min_{u \neq v \in V} d(u,v)$ for the smallest nonzero distance in V. The *aspect ratio* of V is $\Delta(V) = \frac{d_{\max}(V)}{d_{\min}(V)}$. We may omit the argument V and write Δ when it is clear from context. For two metric spaces $M_1 = (V_1, d_1)$ and $M_2 = (V_2, d_2)$, we say M_1 *dominates* M_2 if for all $u, v \in V_1 \cap V_2$, $d_1(u,v) \geq d_2(u,v)$.

We now define an *update sequence*, which captures both arrivals and departures of points:

▶ **Definition 10** (Update Sequence). *An update sequence σ on a metric (X, d_X) is a sequence of pairs $(v_t, o_t) \in X \times \{+,-\}$. We say the point v_t* arrives *at time t if $o_t = +$, and it* leaves *at time t if $o_t = -$. Let L_t be the set of* alive *points at time t: We set $L_0 = \varnothing$, and then for each $t \geq 1$:*

$$L_t = \begin{cases} L_{t-1} \cup \{v_t\}, & \text{if } o_t = +, \\ L_{t-1} \setminus \{v_t\}, & \text{if } o_t = -. \end{cases}$$

We let n denote the length of the sequence and l denote its width, i.e., $\max_t |L_t|$. We write $V_t = \{v_j : j \leq t, o_j = +\}$ for the set of all points introduced up to time t.

Traditional metric embeddings map to a single fixed target metric. In our framework, we allow the target metric to change over time, provided distances between previously embedded points never increase. Thus, rather than mapping to one fixed metric, we map to a *family* of metrics (in our case, HSTs) in an online manner. Similar to [5], we focus on non-contractive embeddings, so we omit the term "non-contractive" here, and just call them embeddings.

▶ **Definition 11** (Online Monotone Embedding). *Let (X, d_X) be a metric space and let $\mathcal{M}$ be a family of metric spaces. A* (deterministic) online monotone embedding *from (X, d_X) into $\mathcal{M}$ takes inputs from an update sequence σ of length n one by one, and upon receiving σ_t, outputs a metric d_t satisfying the following conditions:*

1. *$M_t = (L_t, d_t) \in \mathcal{M}$;*
2. *d_t dominates d_X on L_t: for all $u, v \in L_t$, $d_X(u,v) \leq d_t(u,v)$;*
3. *d_t is dominated by d_{t-1} on $L_{t-1} \cap L_t$: for all $u, v \in L_{t-1} \cap L_t$, $d_t(u,v) \leq d_{t-1}(u,v)$.*

A probabilistic online monotone embedding is a probability distribution over deterministic ones. Such an embedding has distortion λ *if, for every update sequence σ, every t, and $u, v \in L_t$, $\mathbb{E}[d_t(u,v)] \leq \lambda d_X(u,v)$, where the expectation is taken over the internal randomness of the embedding. We consider the* oblivious adversary *model, i.e., the update sequence does not depend on the outcome of random choices made by the algorithm.*

[5] The original definition in [3] assigns weights to edges instead of nodes; both formulations are equivalent up to a constant factor (see, e.g., [4, 5]).

From Partitions to HST Embeddings

Instead of constructing an HST directly, we follow the established approach in [3, 5] of building it implicitly via partitions.

▶ **Definition 12** (Partition). *A* partition P *of a set of points* V *is a collection of disjoint subsets (called* clusters*) whose union equals* V*. For a point* $v \in V$*, we denote by* $P(v)$ *the cluster containing* v*. A partition of a metric space is* s-bounded *if each cluster has a diameter at most* s*. A probabilistic* s*-bounded partition is a distribution over* s*-bounded partitions.*

▶ **Definition 13** (Smoothness Parameters). *A probabilistic* s*-bounded partition* P *of* V *is said to be* $(\delta, \varepsilon, \gamma)$-smooth *if, for all* $u, v \in V$*,*

$$d(u,v) \le s \implies \mathbb{P}[P(u) \neq P(v)] \le \frac{d(u,v)}{s}\delta, \tag{1}$$

$$d(u,v) \le \varepsilon s \implies \mathbb{P}[P(u) \neq P(v)] \le \frac{d(u,v)}{s}\delta\gamma. \tag{2}$$

Condition (1) resembles the padding parameter δ used in prior literature [1, 5] but is slightly more relaxed. Condition (2) generalizes the notion of ε-enforcing from [3], which required no splitting at all when $d(u,v) \le \varepsilon s$ (i.e. $\gamma = 0$). In the online setting, we cannot strictly enforce this; however, we will ensure a small γ so that the distortion remains unaffected up to constant factors. We write δ-smooth as an abbreviation of $(\delta, 1, 1)$-smooth, in which case we only make use of property (1).

▶ **Definition 14** (Online Monotone Partitions). *An* online monotone partition *on* (X, d_X) *takes an update sequence* σ *of length* n *and, for each* $1 \le t \le n$*, generates a partition* P_t *over* L_t *such that:*

1. P_t *does not depend on* $\sigma_{[t+1,n]}$*.*
2. *For any* $u, w \in L_t \cap L_{t+1}$*, if* $P_t(u) = P_t(w)$*, then* $P_{t+1}(u) = P_{t+1}(w)$*.*

This ensures that clusters cannot be split over time: points in the cluster at time t *remain in the same cluster at time* $t+1$*. (Note that two clusters may merge, and new arrivals can join existing clusters, but splitting a previously formed cluster is disallowed.)*

The following lemma relates the distortion of the embedding to the smoothness of the partitions. A similar lemma that does not allow updates and $\gamma = 0$ was proved in [3] (see also [5]), and we defer the complete proof to the full version. Roughly speaking, each cluster at a scale corresponds to a node of the HST at the same scale. Because no cluster can split over time, the lowest common ancestor of two points will not move to high scales, and thus the distances do not increase. With this lemma, our main focus will be on building online probabilistic partitions with desired smoothness parameters.

▶ **Lemma 15** (HST Construction). *Suppose that for every integer* j*, there is a probabilistic online* 2^j*-bounded monotone partition on* (X, d_X) *for an update sequence* σ *of length* n *and that for each time* $t \le n$*, the probabilistic partition* C_t *is* $(\delta_t^{(j)}, \varepsilon, \gamma)$*-smooth. Then there is an online monotone embedding from* (X, d_X) *into HSTs that, on the same input, achieves a distortion of*

$$O\left(\max_{t \le n}\left(\max_{j \in \mathbb{Z}} \delta_t^{(j)} \log \varepsilon^{-1} + \gamma \sum_{j \in \mathbb{Z}} \delta_t^{(j)}\right)\right).$$

Finally, we introduce the notion of *relevant* scales. We say that a scale (or level) j is *relevant* at time t if the partition P_t is not 0-smooth at scale 2^j. We will use this to provide an upper bound of $\sum \delta_t^{(j)}$. We note a technical lemma for bounding the number of relevant scales, whose proof is also in the full version.

▶ **Lemma 16.** *Let V be a set of n points in a metric space (X, d). Fix $0 < \varepsilon < 1$ and define $S = \{j \in \mathbb{Z} \mid \exists u, v \in V : d(u, v) \in [\varepsilon 2^{j-1}, 2^j)\}$. Then $|S| = O(n \log(1/\varepsilon))$.*

3 Technical Overview

Starting Observations. Equipped with Lemma 15, our goal is to maintain probabilistic online s-bounded smooth partitions at a given scale s. We begin by recalling some techniques used in offline embedding for $(\delta, \varepsilon, \gamma)$-smooth partitions that we make use of, and explaining the challenges in the online setting. Obtaining a reasonable δ-value does not require updates; for instance, the algorithms in [3, 5] guarantee $\delta = O(\log n)$ and can be simulated online. With $O(\log \Delta)$ relevant scales, this gives a distortion of $O(\log n \log \Delta)$.

Offline, one can eliminate the dependency on Δ by constructing $(O(\log n), O(1/n), 0)$-smooth partitions [3, 1]. The idea is to first contract all pairs of points at distance $\leq \varepsilon s$ into single points[6] before partitioning, and expand them back after the partition. This inflates each cluster's diameter by at most $n\varepsilon s$, which contributes an acceptable constant multiplicative factor with $\varepsilon = O(1/n)$.

However, in the online setting, the algorithm may discover only later that two points should have been contracted, yet they were placed in separate clusters at an earlier step. Existing online embeddings thus end up with a distortion either dependent on the aspect ratio or polynomial in n. The aim here is to leverage *monotonicity* to overcome these limitations.

Incremental Setting. Monotone embeddings permit us to correct earlier mistakes by merging clusters containing points at a close distance $\leq \varepsilon s$ that should have been initially contracted. However, repeatedly merging clusters is problematic, as it can increase cluster diameters and violate the s-boundedness property. Therefore, we ultimately might still split some close pairs, contributing to the positive γ in the smoothness.

We will use the partitioning algorithm from [5, 3] as the base partition, which we detail in Section 3.1, and specify some merging strategy to ensure a small γ. The algorithm [5, 3] proceeds by constructing balls of random radii around points, which we call *components*. Each point belongs to the first component that includes it. A carefully chosen distribution of the radii ensures $\max \delta^{(j)} = O(\log n)$ and $\sum \delta^{(j)} \leq n \max \delta^{(j)}$. Recall from Lemma 15 that our final distortion is

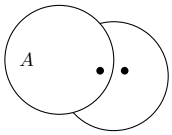
$$O\left(\max_{j \in \mathbb{Z}} \delta_t^{(j)} \log \frac{1}{\varepsilon} + \gamma \sum_{j \in \mathbb{Z}} \delta_t^{(j)}\right) = O(\log n \log \varepsilon^{-1} + \gamma n \log n).$$

Therefore, if we could achieve $\gamma = O(1/n)$ using $\varepsilon = 1/\operatorname{poly}(n)$, the distortion would be bounded by $O(\log^2 n)$.

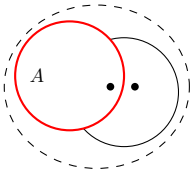
Naturally, we attempt to merge two components whenever they contain points forming a pair with distance at most εs. Crucially, while the multiple merge attempts by a single component are not independent events, the events that *distinct* components simultaneously split close pairs remain independent. Thus, the following surprisingly simple strategy is sufficient, illustrated in Figure 1:

> *Attempt to merge any pair of components containing points within distance εs. Allow the first component that splits a close pair to merge freely with any adjacent component, and reject all other merge attempts not involving this component.*

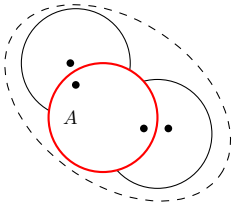
[6] Any chain of points where each adjacent pair is at distance $\leq \varepsilon$ is contracted to a single point.



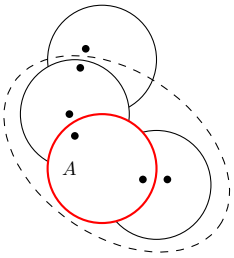
(a) A close pair is separated.



(b) Component A becomes the designated merging component and performs a merge.



(c) One more component merges with A.



(d) Last component not allowed to merge.

Figure 1 The merging strategy.

We call the first component that splits a close pair and is allowed to merge the *designated merging component*. Thus, two close points u, w with $d(u, w) \leq \varepsilon s$ remain eventually separated only if

1. The boundary of a component A splits u and w. The probability that a fixed component splits pair u, w is bounded by $O(\log n) \cdot d(u, w)/s$; and
2. A distinct component B is appointed the designated merging component. A component is the designated merging component only if its boundary splits a close pair u', w' with $d(u', w') \leq \varepsilon s$, which for each pair happens with probability $\leq O(\varepsilon \log n)$. Thus, the overall probability is at most $O(\varepsilon n^2 \log n) = O(\varepsilon \operatorname{poly}(n))$.

Importantly, these two events are independent after fixing the two components since they only rely on independently sampled radii. A union bound over all pairs then gives $\gamma = O(\operatorname{poly}(n)\varepsilon)$. Taking $\varepsilon = n^{-c}$ for a sufficiently large constant c (e.g., $c = 6$ suffices) ensures $\gamma = O(1/n)$, as desired.

The embedding algorithm above requires prior knowledge of n to select the parameter ε. For unknown n, we use phases with guesses $m_i = 2^{2^i}$ and threshold $\epsilon_i = 1/\operatorname{poly}(m_i)$. Instead of one designated merging component overall, we allow one per phase. Since there are only $O(\log \log n)$ phases, the partition retains the same smoothness properties but becomes $O((\log \log n) \cdot s)$-bounded, thus incurring an extra $O(\log \log n)$ factor in contraction. A detailed description of this algorithm appears in the full version.

Fully Dynamic Setting. In the fully dynamic scenario (where points arrive and depart), the monotone updates play a distinct yet crucial role: they allow us to bound the number of relevant scales in terms of the width l. In this setting, we do not make use of a small γ (our $\gamma = 1$). Instead, we only want to bound $\sum \delta^{(j)}$ by controlling the number of relevant scales in which our partitions are not 0-smooth. For strict embeddings, the number of such scales could be as large as $\Omega(n)$: previously arrived (but now departed) points may have caused undesirable separations.

We address this problem via merges. The single-merge strategy for the incremental setting fails here because we no longer have a sufficiently small probability bound (as a function of l) on any particular merge attempt. Thus, we employ the following simple merging strategy:

> *Attempt to merge any pair of components containing points within distance $s/4$. Allow a merge attempt if the merge cannot potentially create a cluster of diameter larger than s.*

Consequently, when alive points in L_t are either within $s/4$ or further than s apart, close pairs attempt merges, and these attempts succeed because distant points cannot interfere with such merges. This approach ensures that at most $O(l)$ scales with non-zero smoothness δ exist at any time, resulting in an overall distortion of $O(l \log l)$.

3.1 Algorithm Outline

For each scale s, we maintain an *online monotone partition* P_t of the set L_t of alive points at time t. We construct P_t in two steps: we maintain a set C_t of *components* and a partition G_t over the set C_t. Each point $u \in L_t$ is assigned to a component $C_t(u) \in C_t$, and the final partition P_t is given by: $P_t(u) = P_t(w)$ if and only if $G_t(C_t(u)) = G_t(C_t(w))$.

For every component c, we ensure that the set of points (potentially not alive) that *may* belong to c has diameter at most $s/4$. It is helpful to think of C_t as a collection of (incomplete) balls and $C_t(u)$ as the ball containing u. To establish $(\delta, \varepsilon, \gamma)$-smoothness, we show:

1. If $d(u,w) \le s$, then $\mathbb{P}(C_t(u) \neq C_t(w)) \le \frac{d(u,w)}{s}\delta$.
2. If $d(u,w) \le \varepsilon s$, then $\mathbb{P}(G_t(C_t(u)) \neq G_t(C_t(w))) \le \frac{d(u,w)}{s}\delta\gamma$.

Intuitively, the set C_t forms our primary partition, and G_t tracks merges among components. We maintain C_t so that $\delta = O(\log n)$. Initially, every component forms a singleton set in G_t, and merges are executed when necessary. Formally, merging two components $C_t(u)$ and $C_t(w)$ means replacing $G_t(C_t(u))$ and $G_t(C_t(w))$ in G_t with their union $G_t(C_t(u)) \cup G_t(C_t(w))$. Our algorithms for the incremental and dynamic cases will share the following construction of C_t, but have different merging strategies.

Smooth Probabilistic Partition C_t. We construct our components C_t by adapting the probabilistic partitions of [3, 5] to accommodate deletions.

Each component in C_t is represented by a triple (c, r, b), where $c \in X$ is the center, $r \in [0, s/8]$ is the radius, and $b \in \mathbb{N}$ is the birth time of the component. We maintain pairwise distinct birth times for components, ensuring uniqueness. A point $u \in X$ *belongs* to the component identified by (c, r, b) if $d(c, u) \le r$ and b is minimal among all components in C_t containing u within their radius. Note that it is possible for the center not to belong to the component. We sometimes refer to a component simply by its center: for $c \in X$, component c refers to the component with center c, and we denote its radius by $r(c)$.

Initially, $C_0 = \varnothing$. When a new point $v \in X$ arrives at time t, we first set $C_t = C_{t-1}$ and then add a new component (v, z, t) to C_t. The radius $z \in [s/16, s/8]$ is independently sampled from the probability distribution $p(z)$ defined as follows. Letting $j = |C_{t-1}| + 1$ denote the number of components after insertion and setting $\chi_j = 2j$, we have:

$$p(z) = \frac{32\,\chi_j^2 \log \chi_j}{s(1-\chi_j^{-2})} \exp\left(-\frac{32z \log \chi_j}{s}\right), \qquad z \in \left[\frac{s}{16}, \frac{s}{8}\right].$$

When a point departs, we remove from C_t any component that contains no alive points.

The following claim establishes that the smoothness parameter satisfies $\delta_t = O(\log l)$. Its proof follows [3, 5], incorporating minor adjustments to account for deletions, and is hence deferred to the full version. The technical choice $\chi_j = 2j$ ensures that $\sum_j \chi_j^{-2} \leq 1$ and $\log(\chi_j) = O(\log l)$, resulting in the claimed smoothness.

▷ Claim 17. If $d(u, w) \leq s$, then at any time t, $\mathbb{P}(C_t(u) \neq C_t(w)) \leq O(\log l) \cdot \frac{d(u,w)}{s}$.

4 Incremental Setting

We now formalise the description in Section 3 and prove an embedding with $O(\log^2 n)$ distortion in the incremental setting, assuming knowledge of n.

▶ **Theorem 18.** *For every $n \in \mathbb{N}$, $s > 0$, and $\varepsilon \leq 1$, given knowledge of n, there is a probabilistic online s-bounded monotone partition of up to n points from any metric that is $(O(\log n), \varepsilon, O(n^5\varepsilon))$-smooth at every time t. Moreover, if no pair of points in V_n has distance in the interval $[s/16, s]$, the partition is 0-smooth at all times.*

Proof. Our initial component partition C_t is constructed as described in Section 3.1, yielding $\delta = O(\log n)$ by Claim 17. We first prove the 0-smooth property.

▷ Claim 19. If no pair of points in V_n has distance in $[s/16, s]$, the partition C_t is 0-smooth at all times t.

Proof. In this scenario, points form natural clusters, each with diameter at most $s/16$, and distinct clusters are separated by distances exceeding s. Hence, the first component created in each group deterministically includes all points in the group, ensuring that the partition is always 0-smooth. ◁

We now specify the merging strategy. We permit at most one component – termed the *designated merging component* – to merge freely with other components whenever they split a pair of points at a distance at most εs. All other merge attempts are rejected.

Formally, we maintain (the center of) the designated component c^*. Initially set to $c^* = \text{NIL}$. We say a component with center c *cuts* a pair $\{u, w\}$ if exactly one of u, w lies inside the (complete) ball defined by its radius:

$$r(c) \in [d(c, u), d(c, w)) \cup [d(c, w), d(c, u)).$$

Note that cut is defined purely about the geometry, and some component cutting a pair is a necessary but insufficient condition for them to belong to different components, since a component created earlier might include both points.

Whenever two points $u, w \in V_t$ at distance $d(u, w) \leq \varepsilon s$ satisfy $C_t(u) \neq C_t(w)$, let c_u and c_w denote the centers of $C_t(u)$ and $C_t(w)$, respectively. We proceed as follows:

1. If $c^* = \text{NIL}$, we assign the designated merging component to be the component among $C_t(u)$ and $C_t(w)$ that cuts $\{u, w\}$. If both components cut $\{u, w\}$, we choose arbitrarily. We then merge $C_t(u)$ and $C_t(w)$ by inserting $G_t(C_t(u)) \cup G_t(C_t(w))$ into G_t and removing $G_t(C_t(u))$ and $G_t(C_t(w))$.
2. If c^* is already set, we merge $C_t(u)$ and $C_t(w)$ only if $c^* \in \{c_u, c_w\}$; otherwise, we reject the merge and do nothing.

Since each component has a diameter at most $s/4$ and any merged component contains a point that is at most εs from the designated merging component, the final clusters will have a diameter at most $3s/4 + 2\varepsilon s \leq s$, satisfying s-boundedness.

▷ Claim 20. For any time t and points $u, w \in V_t$, if $d(u,w) \le \varepsilon s$, then

$$\mathbb{P}(G_t(C_t(u)) \neq G_t(C_t(w))) \le O(n^5 \varepsilon \log n) \cdot \frac{d(u,w)}{s}.$$

Proof. We bound the probability that simultaneously $C_t(u) \neq C_t(w)$ and the designated merging component c^* is assigned elsewhere. For each $c \in V_t$, define the radius set potentially causing c to cut a close pair and be the designated merging component:

$$X(c) = \bigcup_{u',w' \in V_t,\, d(u',w') \le \varepsilon s} [d(c,u'), d(c,w')).$$

By a direct integration of the distribution p, we obtain for all $u', w' \in V_t$:

$$\mathbb{P}(c \text{ cuts } \{u',w'\}) \le \frac{O(\log n) \cdot d(u',w')}{s}, \qquad \mathbb{P}(r(c) \in X(c)) \le O(\varepsilon n^2 \log n).$$

The event $G_t(C_t(u)) \neq G_t(C_t(w))$ can occur only if (i) some component c cuts $\{u,w\}$, and simultaneously, (ii) another distinct component c' is appointed to be the designated merging component c^*, which happens only when $r(c') \in X(c')$. Since radii are independently sampled, after fixing c and c', these events are independent. Thus, by a union bound:

$$\begin{aligned}
\mathbb{P}(G_t(C_t(u)) \neq G_t(C_t(w))) &\le \sum_{c \in V_t} \sum_{c' \neq c} \mathbb{P}(c \text{ cuts } \{u,w\}) \mathbb{P}(r(c') \in X(c')) \\
&\le n^2 \cdot \frac{O(\log n) \cdot d(u,w)}{s} \cdot O(\varepsilon n^2 \log n) \\
&= O(n^5 \varepsilon \log n) \cdot \frac{d(u,w)}{s}.
\end{aligned}$$

◁

Theorem 18 then follows from Claim 17, 19, and 20. ◀

Proof of Theorem 1 (general metric). For each integer j, we apply Theorem 18 with scale $s = 2^j$ and parameter $\varepsilon = n^{-6}$. This produces an online 2^j-bounded partition that is $(\delta_t^{(j)}, n^{-6}, O(n^{-1}))$-smooth at every time t. By Lemma 16, there are at most $O(n)$ scales j for which there exist points $u, v \in V_n$ satisfying $d(u,v) \in [2^j/16, 2^j]$. Theorem 18 guarantees $\delta_t^{(j)} = O(\log n)$ for these scales and $\delta_t^{(j)} = 0$ for all other scales. Hence, we have

$$\max_j \delta_t^{(j)} = O(\log n), \qquad \sum_j \delta_t^{(j)} = O(n \log n).$$

Applying Lemma 15, the total distortion is bounded by

$$O(\log n \log(n^6)) + O(n \log n \cdot n^{-1}) = O(\log^2 n).$$

◀

5 Fully Dynamic Setting

When the width l of the update sequence is bounded, it is natural to expect distortion to depend only on l, independently of the sequence length n. However, this is impossible with traditional strict embeddings: there exist update sequences with width $l = 3$ that incur deterministic lower bounds of $\Omega(2^n)$ and randomized lower bounds of $\Omega(n)$. In fact, the distortion remains unbounded even for deterministic monotone embeddings. The constructions are relatively simple and are deferred to the full version. Despite the negative results, we will show that combined with randomness, monotone embeddings achieve distortion bounded solely by the width l.

5.1 Upper Bound

▶ **Theorem 21.** *For every $l \in \mathbb{N}$ and $s > 0$, there is a probabilistic online s-bounded monotone partition for update sequences of width l that is $O(\log l)$-smooth at every time t. Moreover, if no pair of points $u, w \in L_t$ satisfies $d(u, w) \in [s/4, s]$, the partition P_t at time t is 0-smooth. These results hold without prior knowledge of l.*

Proof. We follow the framework outlined in Section 3.1, constructing C_t as previously described with $\delta = O(\log l)$, and now specify the merge procedure for constructing G_t explicitly.

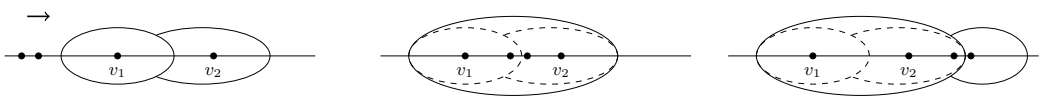
In contrast to the incremental case, where a single merging component sufficed, we allow multiple merges to maintain a bounded number of relevant scales. At each time t, initialize G_t as a copy of G_{t-1}. When a new point v_t arrives, we first update C_t according to Section 3.1. If there exists some point $u \in L_t$ with $d(u, v_t) \leq s/4$ but $C_t(u) \neq C_t(v_t)$, we attempt to merge the components containing u and v_t. The merge succeeds if and only if for every $(c_1, r_1, b_1) \in G_t(C_t(u))$ and $(c_2, r_2, b_2) \in G_t(C_t(v_t))$, $r_1 + r_2 + d(c_1, c_2) \leq s$. This condition ensures the s-boundedness of the partition persists despite potential future arrivals. When a point departs, we remove empty components from C_t, potentially allowing previously rejected merges to now succeed; we thus re-check merge conditions after each departure.

It remains to verify that if no pair of points in L_t has distance in $[s/4, s]$, then P_t is 0-smooth at time t. Indeed, suppose no alive points $u, w \in L_t$ have distance in $[s/4, s]$. Points in L_t naturally form clusters of diameter at most $s/4$, with distances between clusters strictly exceeding s. For any pair u', w' within distance at most $s/4$ belonging initially to different components $C_t(u')$ and $C_t(w')$, the merging condition is satisfied since no distant clusters can obstruct the merge. Thus, all pairs of points within each cluster eventually merge, ensuring P_t is 0-smooth at time t. ◀

Using Lemma 16, at every time t, we have that $\sum \delta_t^{(j)} = O(l \log l)$. Applying Lemma 15 gives the general metric part in Theorem 4. Note we never utilize prior knowledge of l, hence neither does Theorem 4.

5.2 Lower Bound

It might seem tempting to apply similar techniques as in the incremental setting to achieve a distortion of $O(\text{polylog}\, l)$. Unfortunately, this is not possible. To illustrate why, consider the following scenario (see Figure 2): Suppose two points v_1 and v_2 on the line lie in separate components, and two additional close points move through from left to right. At some stage, these two moving points will cause a merge of the intervals containing v_1 and v_2. Due to the monotonicity constraint, this merge cannot be undone, even after the moving points leave, thus preventing future merges that may be necessary when other close points arrive.



(a) Setup. **(b)** A Merge. **(c)** Merge Failure.

Figure 2 Illustration of obstacles in the dynamic setting.

In fact, we show that a linear dependency on l is unavoidable. Notably, this lower bound holds even in an *offline* setting, as it uses a fixed sequence.

▶ **Theorem 5** (Dynamic Lower Bound). *For every $l \in \mathbb{N}$, there exists an update sequence of width l on the line such that any probabilistic monotone embedding of the sequence into HSTs incurs distortion $\Omega(l)$, even if the entire sequence is known in advance.*

Proof. We construct the sequence on the real line, identifying each point with its coordinate. Let $m = 2^l$. Define the following notion: for each integer x, the set of *encompassing points* is given by

$$E(x) = \bigcup_{j=0}^{l} \left\{ \left\lfloor \frac{x}{2^j} \right\rfloor 2^j,\ \left(\left\lfloor \frac{x}{2^j} \right\rfloor + 1 \right) 2^j \right\}.$$

Intuitively, $E(x)$ includes endpoints obtained by recursively halving the segment containing x exactly l times (see Figure 3).

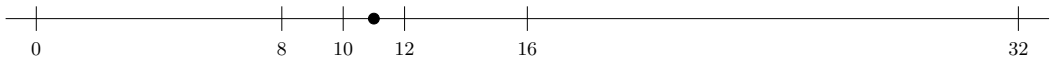


Figure 3 Encompassing points marked as vertical segments.

Our update sequence models the following process, depicted in Figure 4: two moving points travel from 0 to m, arriving and leaving iteratively: initially, points 0 and 1 arrive, then point 0 leaves and point 2 arrives, and so forth. When the moving points are at coordinates $(x, x+1)$, we ensure the points in $E(x)$ are present and remove any points no longer belonging to $E(x)$ (see Figure 4). Clearly, at any given time, the number of alive points is $O(l)$.

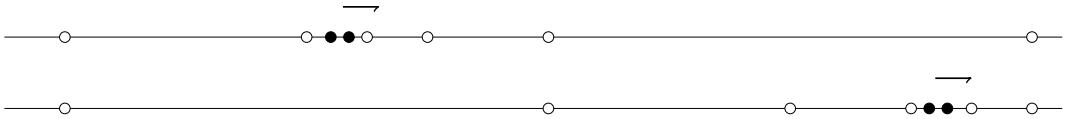


Figure 4 Black points are iterating to the right. White points are encompassing points.

Since the sequence is fixed, we may assume the embedding is non-contractive. By Yao's minimax principle, it suffices to give a distribution over triples (u, v, t) such that every deterministic monotone embedding has expected distortion $\Omega(l)$. Let t_x be the first time x appears as an iterating point, and choose uniformly among triples $(x, x+1, t_{x+1})$. Since all original distances are 1, we write $f(l)$ for the expected embedded distance and aim to prove $f(l) \geq l/2$ by induction. The base case $l = 2$ is trivial.

We now perform the induction as follows. Fix l and $m = 2^l$ and assume $f(j) \geq j/2$ for all $j < l$. Since the embedding is non-contractive, we have $d_{t_m}(0, m) \geq m$. Let p be the largest integer $p < m$ such that $d_{t_p}(0, p) < m$; thus, we must have $d_{t_{p+1}}(0, p+1) \geq m$. As an HST is an ultrametric and $d_{t_{p+1}}(0, p) \leq d_{t_p}(0, p) < m$, it follows directly that $d_{t_{p+1}}(p, p+1) \geq m$. The pair $(p, p+1)$ yields a distortion of m and is selected with probability $1/m$, contributing at least 1 to the expected distortion.

Now, we consider the intervals $[1, p]$ and $[p+1, m]$. Note that we could not invoke the inductive hypothesis immediately, because our instance requires endpoints to exist when the points are iterating. Nevertheless, we can partition the intervals into smaller intervals of length powers of two, in which the encompassing points will be persistent endpoints.

Let p have binary representation $p = \sum_{j=1}^{h} 2^{w_j}$ for distinct integers $w_1 > w_2 > \cdots > w_h$ and $p_i = \sum_{j=1}^{i} 2^{w_j}$ for $i \in [0, h]$. Then the pairs (p_i, p_{i+1}) will persist as encompassing pairs when $t \in [t_{p_i}, t_{p_{i+1}}]$, and the instance forms a smaller replica of size w_{i+1}. Similarly, we decompose the interval $[p+1, m]$. See Figure 5 for an example of the decomposition.

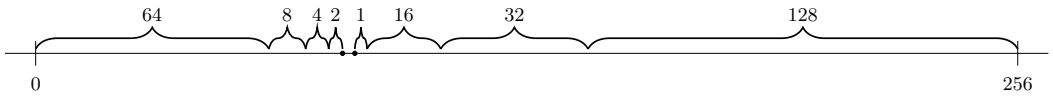


Figure 5 Example of decomposition with $l = 8$, $p = 78$. Not drawn to scale.

We can now safely invoke our inductive hypothesis. For any split point $1 \le p \le m-1$, the binary representations of p and $m-p-1$ together contain each of $2^0, \dots, 2^{l-1}$ exactly once, since the binary representation of $2^l - 1$ is $(111\dots1)_2$ and thus p and $2^l - 1 - p$ would not have 1 on the same bit. For the interval of size 2^j, the tuple $(u, v, t) \sim \mathcal{D}$ falls in that interval with probability 2^{j-l}; conditioned on the pair belonging to the interval, the expected distortion is $f(j)$. Recall we also have a probability of 2^{-l} to pick the pair $(p, p+1)$ whose distance is 2^l. We have

$$f(l) = \sum_{j=0}^{l-1} 2^{j-l} \cdot f(j) + 2^{-l} \cdot 2^l \ge \sum_{j=0}^{l-1} \frac{j}{2^{l-j+1}} + 1 = \frac{l}{2} - 1 + \frac{1}{2^l} + 1 \ge \frac{l}{2}. \qquad \blacktriangleleft$$

6 Applications

6.1 Metrical Request-Answer Games

Throughout this section, we use bold notation such as $\boldsymbol{a}$ for finite sequences. For a sequence $\boldsymbol{a}$ of length m and any $t \le m$, we denote by a_t the tth entry and by $\boldsymbol{a}_{[0,t]}$ the prefix of length t of $\boldsymbol{a}$.

We define a general class of online problems in metric spaces that we call *metrical request-answer games*. It generalizes the notion of request-answer games defined in [6] by introducing dependence on a metric space. The request-answer games in [6] correspond to the special case of our definition where $\alpha_t = 0$.

▶ **Definition 22** (Metrical Request-Answer Game). *A* metrical request-answer game *is defined by a request set R, an answer set A, a metric space (X, d_X), and for each $t \in \mathbb{N}$ a cost function $c_t : R^t \times A^t \to \mathbb{R}_{\ge 0} \cup \{\infty\}$ of the form*[7]

$$c_t(\boldsymbol{r}, \boldsymbol{a}) = \sum_{(u,v) \in X^2} \alpha_t(u, v, \boldsymbol{r}, \boldsymbol{a}) d_X(u, v) + \beta_t(\boldsymbol{r}, \boldsymbol{a}),$$

for some functions $\alpha_t \colon X^2 \times R^t \times A^t \to \mathbb{R}_{\ge 0}$ and $\beta_t \colon R^t \times A^t \to \mathbb{R}_{\ge 0} \cup \{\infty\}$.

Intuitively, $\alpha_t(u, v, \boldsymbol{r}, \boldsymbol{a})$ indicates the number of times an algorithm pays the distance from u to v at step t if it serves the request sequence $\boldsymbol{r}$ with answers $\boldsymbol{a}$, and $\beta_t(\boldsymbol{r}, \boldsymbol{a})$ is some metric-independent cost. In particular, $\beta_t(\boldsymbol{r}, \boldsymbol{a}) = \infty$ allows to specify infeasible answers.

▶ **Definition 23** (Online Algorithm). *A* deterministic online algorithm *ALG for a metrical request-answer game is a sequence of functions $\text{ALG}_t \colon R^t \to A$. Given a request sequence $\boldsymbol{r} \in R^m$, we write $\text{ALG}(\boldsymbol{r}) = (a_1, a_2, \dots, a_m)$ for the sequence of answers selected by ALG, where $a_t = \text{ALG}_t(\boldsymbol{r}_{[0,t]})$ is the answer after the t-th request. The cost of ALG on $\boldsymbol{r}$ is* $\text{cost}_{\text{ALG}}(\boldsymbol{r}) = \sum_{t=1}^{m} c_t(\boldsymbol{r}_{[0,t]}, \text{ALG}(\boldsymbol{r}_{[0,t]}))$. *The optimal cost for the same sequence is* $\text{opt}(\boldsymbol{r}) = \min_{\boldsymbol{a} \in A^m} \sum_{t=1}^{m} c_t(\boldsymbol{r}_{[0,t]}, \boldsymbol{a}_{[0,t]})$.

[7] In fact, as long as c_t is non-negative, concave, and non-decreasing in d_X, our theorem holds.

A randomized online algorithm $\textsc{Alg}$ *is a distribution over deterministic online algorithms* $\textsc{Alg}_x$*. For any request sequence* $\boldsymbol{r}$*, the answer sequence* $\textsc{Alg}(\boldsymbol{r})$ *and hence* $\mathrm{cost}_{\textsc{Alg}}(\boldsymbol{r})$ *become random variables. Algorithm* $\textsc{Alg}$ *is* ρ-competitive *if there exists* $\eta \geq 0$ *such that for every* $\boldsymbol{r}$*,* $\mathbb{E}_x[\mathrm{cost}_{\textsc{Alg}_x}(\boldsymbol{r})] \leq \rho \cdot \mathrm{opt}(\boldsymbol{r}) + \eta$*. We sometimes use* $\mathrm{cost}_{\textsc{Alg}}(\boldsymbol{r})$ *to refer to* $\mathbb{E}_x[\mathrm{cost}_{\textsc{Alg}_x}(\boldsymbol{r})]$*.*

Countless online problems involving metric spaces can be modelled as metrical request-answer games. We give the examples of the k-server and k-taxi problems. More examples, such as the Metrical Task Systems (MTS) [10], Abstract Network Design [5], and Metric Problems with Delay (e.g. [2, 20]), are detailed in the full version.

▶ **Example 24** (k-Server [32] and k-Taxi [23])**.** In the k-taxi problem, there are k taxis located at points of a metric space (X, d_X). At each time t, a request appears, specified by a pair $(x_t, y_t) \in X \times X$ representing a passenger that wants to travel from x_t to y_t. In response, an algorithm must move a taxi to x_t and then to y_t before seeing future requests. The cost is the total distance traveled *without* a passenger on board (i.e., the distance from x_t to y_t is excluded). The k-server problem is the special case where $x_t = y_t$ for each request.

To model k-taxi as a metrical request-answer game, we choose $R = X \times X$, $A = \{1, \dots, k\}$ (assigning numbers to taxis in some fixed way), $\beta_t(\boldsymbol{r}, \boldsymbol{a}) = 0$, and $\alpha_t(u, v, \boldsymbol{r}, \boldsymbol{a}) = 0$ or 1 depending on if a taxi moves from u to v without a passenger in the corresponding step.

For a request sequence $\boldsymbol{r} \in R^t$, let $V(\boldsymbol{r})$ denote the set of relevant points induced by the first t requests, i.e., points whose distance to some point might contribute to the cost:

$$V(\boldsymbol{r}) = \left\{u \in X \mid \exists j \leq t, v \in X,\ \boldsymbol{a} \in A^j : \alpha_j(u, v, \boldsymbol{r}_{[0,j]}, \boldsymbol{a}) + \alpha_j(v, u, \boldsymbol{r}_{[0,j]}, \boldsymbol{a}) > 0\right\}.$$

For example, in the k-taxi problem, $V(\boldsymbol{r})$ is the set of points appearing in the initial configuration and requests of $\boldsymbol{r}$.

6.2 Monotone Potential Functions

We have alluded to the fact that potential functions are usually monotone in distances as a central motivation for online monotone embeddings. In this section, we provide more discussion. We first define the notion of a potential function, which generalizes the definition in [6] to include online algorithms against oblivious adversaries. Recall that $\mathcal{D}(S)$ denotes the set of probability distributions over a set S.

▶ **Definition 25** (Potential Function)**.** *For a metrical request-answer game, a family*

$$\Phi = \{\Phi_t : R^t \times \mathcal{D}(A^t) \times A^t \to \mathbb{R}_{\geq 0}\}_{t \geq 0}$$

is called a potential function for ρ*-competitiveness if the following is true for each* $t \geq 1$*:*

For every $\boldsymbol{r} \in R^t$ *and* $\gamma \in \mathcal{D}(A^{t-1})$*, there exists* $\gamma' \in \mathcal{D}(A^t)$ *such that the marginal distribution of* γ' *over the first* $t-1$ *answers is* γ *and for every* $\boldsymbol{b} \in A^t$ *we have*

$$\mathbb{E}_{\boldsymbol{a} \sim \gamma'}\left[c_t(\boldsymbol{r}, \boldsymbol{a})\right] + \Phi_t(\boldsymbol{r}, \gamma', \boldsymbol{b}) - \Phi_{t-1}(\boldsymbol{r}_{[0,t-1]}, \gamma, \boldsymbol{b}_{[0,t-1]}) \leq \rho \cdot c_t(\boldsymbol{r}, \boldsymbol{b}). \tag{3}$$

Potential functions are a common method of proving the competitiveness of online algorithms. Given a potential function, a corresponding online algorithm can be defined by extending its distribution of answers in each step from γ to some γ' satisfying inequality (3). Taking $\boldsymbol{b}$ to be the answer sequence chosen by some optimal offline algorithm and summing inequality (3) over all time steps shows that the algorithm is ρ-competitive.

Intuitively, the potential function measures the disadvantage of the online algorithm in a given configuration. Typically, this involves a *distance* between the online and offline configurations (e.g., matching or relative entropy), and sometimes *dispersion* of the online configuration, as this causes future uncertainty or mistakes (e.g., pairwise distances among servers of the Double Coverage algorithm [16] for k-server), both of which are monotone. We give a few examples to illustrate the idea.

▶ **Example 26** (Potential for k-Server on Trees)**.** For the k-server problem on trees, [16] gave a k-competitive algorithm, using a potential $\Phi = kM + \Sigma$ where M denotes the value of a minimum matching between the locations of the k online and offline servers and Σ denotes the sum of pairwise distances between the online servers.

▶ **Example 27** (Potential for k-Server on HSTs)**.** For the k-server problem on HSTs, [14] gave an $O(\log^2 k)$-competitive randomized algorithm, using a monotone potential. The full description and proof of monotonicity can be found in the full version.

▶ **Example 28** (Potential for k-Taxi on HSTs)**.** For the k-taxi problem on HSTs, [17] gave a $(2^k - 1)$-competitive randomized algorithm, using a potential equal to $2^k - 1$ times the value of a minimum matching between the locations of the k online taxis and the locations of the k offline taxis. In our notation, we can express this as follows:

For two multisets A and B of locations in an HST T, let $d_T(A, B)$ denote the value of a minimum matching between A and B. For a sequence of requests $\boldsymbol{r} \in R^t$ and answers $\boldsymbol{a} \in A^t$, let $C(\boldsymbol{r}, \boldsymbol{a})$ denote the resulting set of locations of the k taxis. Then $\Phi_t(\boldsymbol{r}, \gamma, \boldsymbol{b}) = (2^k - 1)\mathbb{E}_{\boldsymbol{a}\sim\gamma}[d_T(C(\boldsymbol{r}, \boldsymbol{a}), C(\boldsymbol{r}, \boldsymbol{b}))]$.

▶ **Example 29** (Potential for Constrained Forest on Trees)**.** The Subadditive Constrained Forest Problem (see [25, 5] or our full version for a formal definition) can be trivially solved exactly on trees [5]. Therefore, the constant function 0 is a potential function.

6.3 The Application Framework

We are now ready to state our reduction theorem, which is the formal version of Theorem 6: our evolving embeddings are applicable whenever there is a potential function in the family of target spaces that is non-decreasing in distances.

▶ **Theorem 30.** *Consider a request set R, answer set A, and functions α_t and β_t as in Definition 22 for some fixed ground set X. All metric spaces below have subsets of X as their sets of points. For a metric space $M = (V_M, d_M)$, denote by G_M the associated request-answer game (restricted to request sequences $\boldsymbol{r}$ with $V(\boldsymbol{r}) \subseteq V_M$) and by c_t^M its cost function (as induced by α_t and β_t). Let $\mathcal{R}$ be some set of request sequences such that $|V(\boldsymbol{r})| \le n$ for all $\boldsymbol{r} \in \mathcal{R}$. Let $\mathcal{M}$ be a family of target metrics. Suppose the following holds:*

1. *There is an online monotone embedding from $N = (X, d_N)$ to $\mathcal{M}$ with distortion λ for sequences of n points.*
2. *For each $M \in \mathcal{M}$, there exists a potential function Φ^M for ρ-competitiveness on G_M. Further, the family $\{\Phi^M\}$ is non-decreasing in distances: if M dominates M' (i.e., $d_M(u, v) \ge d_{M'}(u, v)$ for all u, v), then $\Phi_t^M(\boldsymbol{r}, \gamma, \boldsymbol{b}) \ge \Phi_t^{M'}(\boldsymbol{r}, \gamma, \boldsymbol{b})$ for all $t, \boldsymbol{r}, \gamma, \boldsymbol{b}$.*

Then there is a $\rho\lambda$-competitive algorithm for G_N for request sequences in $\mathcal{R}$.

Proof Sketch. Let $\boldsymbol{r} \in \mathcal{R}$ be the request sequence that is revealed online. Note that the sets $V(\boldsymbol{r}_{[0,t]})$ of relevant points for the first t requests are increasing in t. Applying the online monotone embedding, we obtain a metric $M_t = (V(\boldsymbol{r}_{[0,t]}), d_t)$ for each $t = 1, \ldots, m$, where m is the length of $\boldsymbol{r}$.

For fixed $\boldsymbol{M} = (M_1, \dots, M_m)$, we denote by $G_{\boldsymbol{M}}$ the game with cost function $c_t^{M_t}$ at step t. We first define an algorithm $\textsc{Alg}^{\boldsymbol{M}}$ for $G_{\boldsymbol{M}}$ inductively. If $\gamma \in \mathcal{D}(A^{t-1})$ is the distribution of answers before the ith request arrives, then the next answer is chosen to extend the distribution to $\gamma' \in \mathcal{D}(A^i)$ satisfying inequality (3) for the cost function $c_t^{M_t}$ and potential Φ^{M_t}. The overall algorithm $\textsc{Alg}$ for G_N is obtained by taking randomness over $\boldsymbol{M}$.

Bounding the competitive ratio has three steps. First, we show that the total cost of $\textsc{Alg}^{\boldsymbol{M}}$ in $G_{\boldsymbol{M}}$ is bounded by ρ times the offline cost in $G_{\boldsymbol{M}}$. This step uses the monotonicity of the potential function and the embedding. Then, we bound the expected offline cost in $G_{\boldsymbol{M}}$ by the offline cost on G_N times the distortion of $\boldsymbol{M}$. Finally, since the embedding is non-contractive, we bound the cost of $\textsc{Alg}$ for the original game G_N by its cost on $G_{\boldsymbol{M}}$. Combining the three arguments gives the competitive ratio $\rho\lambda$.

The full proof formalizes the above reasoning and appears in the full version. ◀

Together with the monotonicity of Example 29, Theorem 30 immediately implies a competitive algorithm for the subadditive constrained forest problem, recovering a result from [5, 9] where this was proved using a strict online embedding and a min operator to combine with a baseline algorithm to avoid dependence on Δ.

▶ **Corollary 31** (Constrained Forest [5, 9]). *There is an $O(\log^2 n)$-competitive algorithm for the subadditive constrained forest problem on general metrics and an $O(\log n)$-competitive algorithm on $O(1)$-dimensional normed spaces.*

Theorem 8 for the k-taxi problem is almost implied by the monotonicity of Example 28 (the k-taxi potential), Theorem 30 and Theorem 1. The only remaining issue is that our embedding algorithms require prior knowledge of n. We now provide a sketch on how to use a "guess-and-double" technique to handle the unknown-n case. Details and the proof for Theorem 7 (k-server) can be found in the full version.

▶ **Theorem 8** (k-Taxi). *There is an $O(2^k \log^2 n)$-competitive algorithm for the k-taxi problem on general metrics, and an $O(2^k \log n)$-competitive algorithm for $O(1)$-dimensional normed spaces, where n is the number of requested locations.*

Proof Sketch. Let $\boldsymbol{r}$ be the sequence revealed online. Recall $V(\boldsymbol{r}_{[0,t]})$ is the set of points among the initial configuration and the first t requests, and let $n_t = |V(\boldsymbol{r}_{[0,t]})|$. Let opt_t denote the optimal cost of serving the first t requests and $\mathrm{opt} = \mathrm{opt}_{|\boldsymbol{r}|}$.

The execution is divided into phases. In phase i, the algorithm maintains guesses ζ_i and m_i, where ζ_i is intended to be within a constant factor of $\log^2 n \cdot \mathrm{opt}$, and m_i is an upper bound on the number of points. Initially, we use the smallest possible guesses. If, at some time t, we find that $\zeta_i < \log^2 n_t \cdot \mathrm{opt}_t$, we end the current phase, double ζ_i, and update the size guess to $m_i = n_t^2$. One can verify that this ensures $n_t \in [\sqrt{m_i}, m_i]$ for every t in phase i, and hence $O(\log m_i) = O(\log n_t)$.

Within each phase, we run the $O(2^k)$-competitive algorithm in [17] with the HST embedding of $V(\boldsymbol{r})$ using the current value of m_i. By Theorem 30, the cost incurred by the simulated algorithm on the prefix ending at the ith phase boundary t_i is $O(2^k \log^2 m_i) \cdot \mathrm{opt}_{t_i}$, which is at most a constant multiple of $2^k \zeta_i$. The cost of a phase transition is also bounded by the sum of the costs of the old and new simulations on the prefix so far. Thus, the total cost over all phases is $O(2^k \sum_i \zeta_i)$. Since the guesses double from phase to phase, this sum is dominated by the final guess, which is at most $O(2^k \log^2 n) \cdot \mathrm{opt}$. ◀

6.4 Application for Dynamic Embedding

The application of dynamic embeddings can require problem-specific considerations, since the set of "relevant" points may vary for different problems. As an example, we revisit the k-taxi problem.

▶ **Theorem 32.** *The following two statements are equivalent:*

1. *There is an $f(k)$-competitive online algorithm for the k-taxi problem, for some function f.*
2. *It is possible to maintain online an evolving set of at most $g(k)$ "relevant" points, such that the best* offline *algorithm that must have taxis only at relevant points is an $h(k)$-approximation of the unrestricted optimal solution, for some functions g and h.*

Proof. Let S_t denote the set of relevant points and C_t denote the set of online taxis at time t. For $1 \Rightarrow 2$, setting $S_t = C_t$ gives $g(k) = k$ and $h(k) = f(k)$.

For $2 \Rightarrow 1$, when a new request (x_{t+1}, y_{t+1}) arrives, embed at step $t+1$ the set $C_t \cup S_t \cup S_{t+1} \cup \{x_{t+1}, y_{t+1}\}$ into an HST using Theorem 4 and simulate the algorithm of [17] on this HST. To see competitiveness, consider the optimal offline solution $\boldsymbol{b}$ that restricts its taxis to locations in S_t at every time t. Since $|C_t \cup S_t \cup S_{t+1} \cup \{x_{t+1}, y_{t+1}\}| \le 2g(k) + k + 2$, the embedding distortion at each step is at most $O(g(k)\log(g(k)))$. Recall from [17] that the potential function is monotone. Thus, a similar analysis to that in Theorem 30 shows that the algorithm is $O(2^k g(k)\log(g(k)))$-competitive against the restricted offline solution $\boldsymbol{b}$, which implies the overall competitive ratio of $O(2^k h(k) g(k)\log(g(k)))$. ◀

7 Conclusion and Discussion

We present a new framework, online monotone metric embeddings, which allows the embedding to evolve over time, provided that distances between points do not increase. For embeddings into HSTs, we establish $O(\log^2 n)$ distortion from general metrics and tight $O(\log n)$ distortion from $O(1)$-dimensional normed spaces. We also discuss dynamic monotone embeddings and present an $O(l \log l)$ upper bound and a $\Omega(l)$ lower bound for the distortion. Finally, we discuss the conditions under which an algorithm can be combined with our embedding, and illustrate some applications.

While we use monotone potential functions in Theorem 30 as a systematic way to certify compatibility with particular online algorithms, our embedding framework may be applicable more broadly. It suffices to inspect an algorithm's proof and verify that the argument remains valid under monotone distance updates. In this way, future work may be able to directly plug our embeddings into their analyses. We highlight some additional future directions below.

1. What is the tight bound for the distortion of online monotone embeddings into HSTs? Deterministically, our algorithm is optimal. For probabilistic embeddings, the only known lower bound is the $\Omega(\log n)$ offline lower bound. Our algorithms match this on constant-dimensional normed spaces, but leave a quadratic gap in the general case.
2. Are there interesting results for online monotone embedding into metrics other than trees? We mainly focus on HSTs since many online problems have competitive algorithms on HSTs. However, embedding into other metrics is also of interest.
3. Dynamic embedding with limitations on the type of recourse is a field with major potential for many online problems, of which we have only scratched the surface. Dynamic embeddings have been a main candidate for getting a polylog(k)-competitive algorithm for the randomized k-server problem. Our $\Omega(l)$ lower bound imposes limitations on certain approaches. As another example, our Theorem 32 directly suggests pathways for the k-taxi problem.

4. Are there similar characterizations for other online problems with monotone recourse? For example, can improved results be obtained for online matching with monotone recourse, where agents are only willing to change their partner if they prefer the new partner?

References

1 Ittai Abraham, Yair Bartal, and Ofer Neimany. Advances in metric embedding theory. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 271–286, 2006. doi:10.1145/1132516.1132557.

2 Yossi Azar, Arun Ganesh, Rong Ge, and Debmalya Panigrahi. Online service with delay. *ACM Trans. Algorithms*, 17(3), 2021. doi:10.1145/3459925.

3 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS*, 1996.

4 Yair Bartal. Advances in metric Ramsey theory and its applications. *CoRR*, abs/2104.03484, 2021. arXiv:2104.03484.

5 Yair Bartal, Nova Fandina, and Seeun William Umboh. Online probabilistic metric embedding: A general framework for bypassing inherent bounds. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2020.

6 Shai Ben-David, Allan Borodin, Richard M. Karp, Gábor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1), 1994. doi:10.1007/BF01294260.

7 Aaron Bernstein, Aditi Dudeja, and Zachary Langley. A framework for dynamic matching in weighted graphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2021.

8 Sayan Bhattacharya, Monika Henzinger, and Danupon Nanongkai. New deterministic approximation algorithms for fully dynamic matching. In *Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing, STOC*, 2016.

9 Sujoy Bhore, Arnold Filtser, and Csaba D. Tóth. Online duet between metric embeddings and minimum-weight perfect matchings. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2024.

10 Allan Borodin, Nathan Linial, and Michael E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4), 1992. doi:10.1145/146585.146588.

11 Sébastien Bubeck, Niv Buchbinder, Christian Coester, and Mark Sellke. Metrical service systems with transformations. In *12th Innovations in Theoretical Computer Science Conference, ITCS*, 2021.

12 Sébastien Bubeck, Christian Coester, and Yuval Rabani. The randomized k-server conjecture is false! In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC*, 2023.

13 Sébastien Bubeck, Christian Coester, and Yuval Rabani. Shortest paths without a map, but with an entropic regularizer. *SIAM J. Comput.*, 54(5):S22–265, 2025. doi:10.1137/22M1539149.

14 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2018.

15 Niv Buchbinder, Christian Coester, and Joseph Naor. Online k-taxi via double coverage and time-reverse primal-dual. *Math. Program.*, 197(2):499–527, 2023. doi:10.1007/S10107-022-01815-6.

16 Marek Chrobak, Howard Karloof, T. Payne, and Sundar Vishwnathan. New results on server problems. *SIAM Journal on Discrete Mathematics*, 4(2), 1991. doi:10.1137/0404017.

17 Christian Coester and Elias Koutsoupias. The online k-taxi problem. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2019.

18 Christian Coester and Tze-Yang Poon. Online 3-taxi on general metrics. In *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2026. doi:10.1137/1.9781611978971.238.

19 Christian Coester and Alexa Tudose. Chasing small sets optimally against adaptive adversaries. In *53rd International Colloquium on Automata, Languages, and Programming, ICALP*, 2026.

20 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2016.

21 David Eppstein, Zvi Galil, Giuseppe F. Italiano, and Amnon Nissenzweig. Sparsification—a technique for speeding up dynamic graph algorithms. *J. ACM*, 44(5):669–696, 1997. doi:10.1145/265910.265914.

22 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC*, 2003.

23 Amos Fiat, Yuval Rabani, and Yiftach Ravid. Competitive k-server algorithms. In *31st Annual Symposium on Foundations of Computer Science, FOCS*, 1990.

24 G.N. Frederickson. Ambivalent data structures for dynamic 2-edge-connectivity and k smallest spanning trees. In *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*, pages 632–641, 1991. doi:10.1109/SFCS.1991.185429.

25 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

26 Anupam Gupta, Amit Kumar, and Debmalya Panigrahi. Poly-logarithmic competitiveness for the k-taxi problem. In *Proceedings of the 2024 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2024.

27 Piotr Indyk, Avner Magen, Anastasios Sidiropoulos, and Anastasios Zouzias. Online embeddings. In *Proceedings of the 13th International Conference on Approximation, and 14th International Conference on Randomization, and Combinatorial Optimization: Algorithms and Techniques, APPROX/RANDOM*, 2010.

28 Elias Koutsoupias. The k-server problem. *Computer Science Review*, 3(2), 2009. doi:10.1016/J.COSREV.2009.04.002.

29 Elias Koutsoupias and Christos H. Papadimitriou. On the k-server conjecture. *J. ACM*, 42(5), 1995. doi:10.1145/210118.210128.

30 James R. Lee. Fusible HSTs and the randomized k-server conjecture. *2018 IEEE 59th Annual Symposium on Foundations of Computer Science, FOCS*, 2018.

31 James R. Lee and Anastasios Sidiropoulos. Pathwidth, trees, and random embeddings. *Combinatorica*, 33:349–374, 2009. URL: https://api.semanticscholar.org/CorpusID:11371999.

32 Mark Manasse, Lyle McGeoch, and Daniel Sleator. Competitive algorithms for online problems. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing, STOC*, 1988.

33 Ilan Newman and Yuri Rabinovich. Online embedding of metrics. In *17th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT*, 2020.

34 Christos H. Papadimitriou and Mihalis Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991. doi:10.1016/0304-3975(91)90263-2.

35 Christian Wulff-Nilsen. Fully-dynamic minimum spanning forest with improved worst-case update time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, 2017.