New Streaming Algorithms for High Dimensional EMD and MST*

Xi Chen
xichen@cs.columbia.edu
Columbia University
USA

Rajesh Jayaram
Google Research
USA
rkjayaram@google.com

Amit Levi
University of Waterloo
Canada
amit.levi@uwaterloo.ca

Erik Waingarten
Stanford University
USA
eaw@cs.columbia.edu

ABSTRACT

We study streaming algorithms for two fundamental geometric problems: computing the cost of a Minimum Spanning Tree (MST) of an n-point set $X \subset \{1, 2, \ldots, \Delta\}^d$, and computing the Earth Mover Distance (EMD) between two multi-sets $A, B \subset \{1, 2, \ldots, \Delta\}^d$ of size n. We consider the turnstile model, where points can be added and removed. We give a one-pass streaming algorithm for MST and a two-pass streaming algorithm for EMD, both achieving an approximation factor of $\tilde{O}(\log n)$ and using polylog(n, d, Δ)-space only. Furthermore, our algorithm for EMD can be compressed to a single pass with a small additive error. Previously, the best known sublinear-space streaming algorithms for either problem achieved an approximation of $O(\min\{\log n, \log(\Delta d)\} \log n)$. For MST, we also prove that any constant space streaming algorithm can only achieve an approximation of $\Omega(\log n)$, analogous to the $\Omega(\log n)$ lower bound for EMD.

Our algorithms are based on an improved analysis of a recursive space partitioning method known generically as the Quadtree. Specifically, we show that the Quadtree achieves an $\tilde{O}(\log n)$ approximation for both EMD and MST, improving on the $O(\min\{\log n, \log(\Delta d)\} \log n)$ approximation.

CCS CONCEPTS

• **Theory of computation → Streaming, sublinear and near linear time algorithms**.

KEYWORDS

sketching, streaming, earth-mover's distance, minimum spanning tree

ACM Reference Format:

Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. 2022. New Streaming Algorithms for High Dimensional EMD and MST. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing (STOC '22), June 20–24, 2022, Rome, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3519935.3519979

*Xi Chen is supported by NSF grants CCF-1703925, IIS-1838154, CCF-2106429 and CCF-2107187. Erik Waingarten is supported by the National Science Foundation under Award No. 2002201 and Moses Charikar's Simons Investigator award. This work was carried out while the author Amit Levi was a PhD student at the University of Waterloo.

STOC '22, June 20–24, 2022, Rome, Italy

ACM ISBN 978-1-4503-9264-8/22/06.
https://doi.org/10.1145/3519935.3519979

1 INTRODUCTION

We study two fundamental geometric problems in high-dimensional spaces: the Earth Mover's distance and minimum spanning tree. Let $(\mathcal{X}, d_{\mathcal{X}})$ be a metric space. Given two (multi-)sets $A, B \subset \mathcal{X}$ of size $|A| = |B| = n$, the Earth Mover's distance (EMD) between A and B is

$$\mathsf{EMD}_{\mathcal{X}}(A, B) = \min_{\substack{\text{matching} \\ M \subset A \times B}} \sum_{(a,b) \in M} d_{\mathcal{X}}(a, b).$$

Given a single multi-set $X \subset \mathcal{X}$ of size n, the cost of the minimum spanning tree (MST) of X is

$$\mathsf{MST}_{\mathcal{X}}(X) = \min_{\substack{\text{tree } T \\ \text{spanning} \\ X}} \sum_{(a,b) \in T} d_{\mathcal{X}}(a, b).$$

Computational aspects of EMD and MST consistently arise in multiple areas of computer science [19, 36, 37], such as in computer vision [12, 41], image retrieval [38], biology [34], document similarity [28], machine learning [7, 16, 32], among other areas. Their centrality in both theory and practice has motivated the theoretical study of approximate and sublinear algorithms [1–3, 5, 6, 8, 10, 11, 13, 17, 18, 20, 23, 26, 29, 39, 40, 42, 43] in both low- and high-dimensional settings.

As an illustrative example, an important application for high-dimensional EMD comes from natural language processing, particularly document retrieval and classification. A document can be represented as a collection of vectors in Euclidean space by applying *word embeddings* [30, 35] to each of its words; these embeddings have the property that semantically similar words map to geometrically close vectors. In this context, computing the EMD between the embeddings of two documents yields a natural measure of similarity, aptly termed the *Word Mover's Distance* [28].

In this paper, we study streaming and sketching algorithms for computing EMD and MST. Specifically, we consider the turnstile *geometric streaming model*, introduced by [20], where the algorithm receives the input set $X \subset \mathcal{X}$ via an arbitrarily ordered sequence of insertions and deletions of points $p \in \mathcal{X}$. The goal is for the algorithm to approximate a fixed function of the implicit set of points X in small space, without storing X; ideally, one would hope for space *polylogarithmic* in the number of points in $|X|$. We focus on the high-dimensional Euclidean space, where $\mathcal{X} = \{1, 2, \ldots, \Delta\}^d$, and the distance between points is given by an ℓ_p norm for $p \in [1,2]$. One can always reduce from the case of $\mathbb{R}^d$ to $\{1,\dots,\Delta\}^{d'}$ via standard embeddings. For $p > 2$, there are is a $\Omega(d^{1-2/p})$ lower bounds on the space required to estimate the ℓ_p norm [9]. Therefore, estimating $\mathsf{EMD}_{\ell_p}(\{0\},\{b\})$ and $\mathsf{MST}_{\ell_p}(\{0,b\})$ already requires polynomial-in-d space. In this work, our sketches will use $\mathrm{polylog}(n,d)$ bits.

Prior Work on Sketching and Streaming EMD *and* MST. We briefly survey what is known for streaming and sketching EMD and MST. We emphasize that many aspects of the sketchability and streamability of EMD and MST remain open, and obtaining tight bounds for these tasks, as well as related geometric graph problems, still remains elusive.[1]

Indyk [20], building on work of [13], was the first to formulate dynamic geometric streams and give algorithms for EMD and MST which achieved an $O(d\log\Delta)$-approximation. The result for MST was improved to a $(1+\epsilon)$-approximation in [17], however, the resulting space complexity is exponential in the dimension, making the algorithm suitable only in low-dimensional spaces. For EMD on the plane, [2] gave a $O(1/\epsilon)$ approximation at the cost of a Δ^{ϵ} dependence in the space complexity. The best lower bound on sketching EMD on the plane is due to [5], where they show that one cannot have both a constant bit and constant approximation sketch. If the sketch proceeds by an embedding into ℓ_1, [33] show the approximation must be $\Omega(\sqrt{\log\Delta})$. Parametrizing the approximation in terms of n, [11] gave embeddings of EMD on the plane into ℓ_1 with distortion $O(\log n)$.

For the high-dimensional regime, Andoni, Indyk, and Krauthgamer [3] gave an algorithm for EMD (in fact, an embedding into ℓ_1) with approximation $O(\log n\log(d\Delta))$. Furthermore, building on an ℓ_1-embedding lower bound of [27], they show that any s-bit sketch with approximation $\alpha > 1$ must have $s\alpha = \Omega(\log n)$. For sketching, the approximation of [3] may be improved to $O(\log n\min\{\log n,\log(d\Delta)\})$ by the techniques in [8, 11]. MST has not been formally considered in the high-dimensional regime, although we note that an $O(\log n\min\{\log n,\log(d\Delta)\})$-approximate streaming algorithm readily applies here as well. For lower bounds on streaming high-dimensional MST, nothing was known, and (prior to this work) a constant-bit stream achieving a constant approximation was possible.

1.1 Our Results

In this work, we develop new algorithms and lower bounds for approximating EMD and MST in a stream. Specifically, we show that the approximation factor for these problems can be improved from $O(\log n\cdot\min\{\log n,\log(\Delta d)\})$ to $\tilde{O}(\log n)$. We now state the main results of this paper. In the theorem statements which follow, we consider a fixed setting of n, d and Δ. The metric space consists of points in $[\Delta]^d = \{1,\dots,\Delta\}^d$ with ℓ_p distance for any fixed $p \in [1,2]$. We state the theorems in the *random-oracle model*, i.e., any random bits stored by the algorithm do not factor into the space complexity — we show that storing the random bits would incur at most an additive $d\cdot\mathrm{polylog}(n,\Delta)$ bits of space..[2]

[1]See Open Problems 7 and 49 for sketching EMD in https://sublinear.info/

[2]Also note that to even store a single update $p \in [\Delta]^d$, one requires $\Omega(d\log\Delta)$ bits of space.

Theorem 1.1 (MST Streaming Algorithm). *There exists a turnstile streaming algorithm using at most* $\mathrm{polylog}(n,d,\Delta)$ *bits of space which, given a set $X \subset [\Delta]^d$ of size n, outputs $\widehat{\boldsymbol{\eta}} \in \mathbb{R}$ satisfying*

$$\mathsf{MST}_{\ell_p}(X) \le \widehat{\boldsymbol{\eta}} \le \tilde{O}(\log n)\cdot\mathsf{MST}_{\ell_p}(X)$$

with high probability.

For EMD, our algorithm achieving an $\tilde{O}(\log n)$-approximation requires two passes over the data. This arises from a technical issue in the approach for EMD which is not present in MST. We state the theorem in terms of two-pass streaming algorithms, and then show how to compress the two passes into one, at the cost of an additive error in the approximation.

Theorem 1.2 (EMD Two-Pass Streaming Algorithm). *Given two multi-sets $A, B \subset [\Delta]^d$ of size n there exists a two-pass turnstile streaming algorithm using* $\mathrm{polylog}(n,d,\Delta)$ *bits of space which outputs $\widehat{\boldsymbol{\eta}} \in \mathbb{R}$ satisfying*

$$\mathsf{EMD}_{\ell_p}(A,B) \le \widehat{\boldsymbol{\eta}} \le \tilde{O}(\log n)\cdot\mathsf{EMD}_{\ell_p}(A,B)$$

with high probability.

Theorem 1.3 (EMD One-Pass Streaming Algorithm). *Given two multi-sets $A, B \subset [\Delta]^d$ of size n and any $\epsilon > 0$, there exists a turnstile streaming algorithm using* $O(1/\epsilon)\cdot\mathrm{polylog}(n,d,\Delta)$ *bits of space which outputs $\widehat{\boldsymbol{\eta}} \in \mathbb{R}$ satisfying*

$$\mathsf{EMD}_{\ell_p}(A,B) \le \widehat{\boldsymbol{\eta}} \le \tilde{O}(\log n)\cdot\mathsf{EMD}_{\ell_p}(A,B) + \epsilon d\Delta n.$$

with high probability.

We encourage the reader to think of instances where A and B are size-n subsets of the hypercube $\{0,1\}^d$ with ℓ_1 distance (i.e., $\Delta = 2$ and $p = 1$). This setting captures all the complexity encountered in this work. For $\Delta > 2$ and $p \in (1,2]$, the algorithm first applies an embedding into $\{0,1\}^d$ with ℓ_1.

Regarding the additive error in Theorem 1.3, while an appropriate setting of ϵ may absorb the additive error into relative error, we leave as an open problem whether this additive error may be removed completely in one-pass algorithms. For instance, if the points do not overlap almost always, i.e., when $|A\cap B|/|A\cup B| \le 1-\epsilon_0$, then $\mathsf{EMD}_{\ell_p}(A,B) \ge \epsilon_0 n$, and ϵ may be set to $\epsilon_0/d\Delta$ in order to absorb the additive error into the relative error by increasing the space by a factor of $d\Delta$, and keeping a poly-logarithmic dependence on n. From a practical perspective, the fact that points do not overlap may be a reasonable assumption to make.

All of our streaming algorithms are *linear sketches*, meaning that they store only the matrix-vector product $\mathbf{S}f$ for some randomized $\mathbf{S} \in \mathbb{R}^{k\times n}$, where $f = f_X \in \mathbb{R}^{\Delta^d}$ is the indicator vector (with multiplicity) of X for the case of MST, and $f = f_{A,B} \in \mathbb{R}^{2\cdot\Delta^d}$ is the indicator vector (with multiplicity) of A, B for EMD. Linear sketches are an important class of turnstile streaming algorithms, and have many well-known and studied advantages. For instance, such sketches directly resulted in algorithms for distributed computation such as the MPC model, as well as algorithms for multi-party communication. Our results, therefore, can be applied in a natural way to these models as well.

Improved Analysis of the Quadtree. The prior sketching algorithms are based on a hierarchical partitioning method known as the *Quadtree.*[3] Here, we refer to quadtrees as a generic class of methods that embed points from $\mathcal{X}$ into a randomized tree by recursively partitioning the space. At a high level, the Quadtree algorithm recursively and randomly partitions the space $\mathcal{X}$, which results in a rooted (randomized) tree. Each point in the set X for the case of MST, or $A \cup B$ for EMD, is sent down to a leaf of the tree. From there, a spanning tree or a matching, is constructed in a bottom-up fashion. Each point "walks up the tree" and is greedily connected (in the case of MST), or matched (in the case of EMD) as it encounters other points. This results in a very efficient *offline* (non-sketching) algorithm. The recent work of [8] study the quadtree algorithm explicitly, where they call it "Flowtree," and showed it has favorable practical properties. From a theoretical point-of-view, the approximation incurred by these methods were the bottleneck in prior works for sketching and streaming EMD and MST, here, we improve this analysis of [3, 8] from $O(\log n \min\{\log n, \log(d\Delta)\})$ to $\tilde{O}(\log n)$.

Theorem 1.4 (Quadtree Methods (Informal)). *Given two multi-sets $A, B \subset [\Delta]^d$ of size n, the "Flowtree" algorithm of [8] outputs an $\tilde{O}(\log n)$-approximation to $\mathsf{EMD}_{\ell_1}(A, B)$ with probability at least 0.9. Similarly, given a multi-set $X \subset [\Delta]^d$ of size n, the greedy, bottom-up spanning tree is an $\tilde{O}(\log n)$-approximation to $\mathsf{MST}_{\ell_1}(X)$ with probability at least 0.9.*

Lower bounds for MST.. For lower bounds, [3] shows that any randomized ℓ-bit streaming algorithm distinguishing $\mathsf{EMD}_{\ell_1}(A, B) \geq r$ and $\mathsf{EMD}_{\ell_1}(A, B) \leq r/\alpha$ with probability at least 2/3 must satisfy $\alpha\ell = \Omega(d)$, where the instances used have $d = \log n$. For a qualitative comparison, estimating ℓ_1 norm does admit such $O(1)$-approximation, $O(1)$-bit space streaming algorithms (with public randomness), implying that EMD is a harder problem. We show an analogous lower bound for MST in the streaming model.

Theorem 1.5. *Any randomized ℓ-bit streaming algorithm which can distinguish whether a size-n set $X \subset \{0, 1\}^d$ has $\mathsf{MST}_{\ell_1}(X) \geq nd/3$ or $\mathsf{MST}_{\ell_1}(X) \leq nd/\alpha$ with probability at least 2/3 must satisfy $\ell + \log \alpha = \Omega(\log n/\alpha)$. Moreover, this holds even in the insertion-only model, where points are only added to X in the stream.*

We emphasize that, prior to Theorem 1.5, there were no lower bounds known for streaming MST — not even an $\Omega(1)$ lower bound was known on the approximation of a constant-bit algorithm. We note that [3] actually considers the (stronger) two-party communication setting for EMD, where each player receives one of the sets. The two-party communication game for MST where each player receives half of the set X is insufficient, as there is simple $O(1)$-approximation, constant-bit protocol.[4] Therefore, our theorem will crucially involve the streaming nature of the algorithm.

1.2 Technical Overview

1.2.1 The Main Idea: Tree Embeddings with Data-Dependent Edge Weights. In [20], Indyk described an approach for streaming a variety of graph problems (including MST and EMD) in discrete geometric spaces, leading to $O(d \log \Delta)$-approximations for these problems in the metric space $[\Delta]^d$ with ℓ_1 distance. This approach, later refined in [3], forms the basis of our work, so we give a very high level overview in order to highlight the new ideas. For simplicity, we describe it for EMD, as the high-level picture for MST is similar.

A streaming algorithm for EMD with sets $A, B \subset [\Delta]^d$ of size n may proceed in the following way:

(1) Sample a recursive random partition of the space, broadly referred to as a quadtree, which specifies an embedding of the original space $[\Delta]^d$ into a rooted tree. For example, when $d = 2$, one may sample $\log_2 \Delta$ randomly shifted, nested square grids of side length $\Delta/2, \Delta/4, \ldots, 1$ and arrange them into a rooted tree of depth $\log_2 \Delta + 1$. Each node corresponds to a region of the space, where the root contains the entire space, and the children of a node have regions which partition the region of the parent. The points in A and B are assigned to leaves of this tree, according the regions where points fall, and the quadtree implicitly defines a matching M between A and B given by the natural bottom-up greedy procedure. Having implicitly specified a matching M, the goal of the streaming algorithm will be to approximate the cost of M.

(2) In order to do so, [3, 20] maintains a high-dimensional vector which implicitly encodes the matching M. Specifically, the vector has a coordinate for each edge of the quadtree, and the entry in each coordinate is the number of points from A falling within the region of the child minus the number of points in B falling within the region of the child. Furthermore, the ℓ_1-norm of the vector, where each coordinate of an edge is scaled by some edge weight (for example, by the size of the parent region) gives an approximation of the cost of M. Thus, this gives an ℓ_1-embedding for EMD over $[\Delta]^d$, and known algorithms for streaming the ℓ_1-norm can be applied.

With the above approach in mind, there are two steps involved in showing the approximation guarantee: (i) showing the matching M in Step 1 has approximately optimal cost, and (ii) showing that the appropriate scalings of coordinates reduce approximating the cost of the matching M to an ℓ_1-computation. We note that even though the above presentation is a two-step procedure, [3, 20] do not present it this way. In fact, de-coupling the matching M from the method to approximate the cost of M is an important conceptual contribution which is made explicit in [8], which led us to revisit the EMD problem.

Prior to our work, (i) proceeded by the method of tree embeddings. One assigns the edge weights to the quadtree and interprets it as a tree embedding of the metric $([\Delta]^d, \ell_1)$. By studying the distortion of this embedding, one bounds the cost of M. The edge weights chosen in [20] (building on work of [13]) embed $([\Delta]^d, \ell_1)$ with distortion $O(d \log \Delta)$, which will become the approximation. Refining the approach, [3] show that another choice of edge weights (better suited for high-dimensional spaces) embeds subsets of $([\Delta]^d, \ell_1)$ with bounded average distortion which suffices for an $O(\log n \log(d\Delta))$ bound on the cost of M. Given the bound

[3]The name *Quadtree* is an artifact of the study of the algorithm originally in the planar (two-dimensional) case, in which the algorithm recursively partitions the plane into quadrants. Our Quadtrees, being in high dimensions, will partition space into more than 4 parts at a time. However, since they are the natural generalization of the planar case, it is common to refer to the generic method as Quadtree regardless of dimension.

[4]Intuitively, the players may compute the cost of their MST locally, and compute the distance between two arbitrary points. The sum of these quantities is a 3-approximation to the MST of the entire set.

on M with respect to a fixed tree metric, (ii) is straight-forward: since the fixed tree metric specifies the scalings of the vector, and approximating the cost of M amounts to an ℓ_1-norm computation.

Our main contribution is two-fold. First, we show how to go beyond the distortion argument in (i) to show that the cost of M is a $\tilde{\Theta}(\log n)$-approximation to EMD with probability 0.9. To do so, we study a *data-dependent* notion: instead of fixing the edge weights as in [3, 20], we allow the edge weights to depend on the input $A \cup B$. The use of data-dependent edge weights implies M is actually a better quality matching than what the method of tree embeddings specified. The data-dependent edge weights are (relatively) simple: the weight of an edge (u, v), where u is the parent of v, is the average distance between a randomly sampled point of $A \cup B$ within the region of u and a randomly sampled point of $A \cup B$ within the region of v. However, the fact these data-dependent edge weights yield an improved upper bound on the cost of M constitutes the bulk of the work.

Unfortunately, the introduction of data-dependent edge weights breaks Step 2. Now, approximating the cost of M with the data-dependent weights is no longer as simple as an ℓ_1-computation. The coordinates of the vector remain the same, however, the scaling of each coordinate depends on additional structure of the points. Importantly, data-dependent edge weights do not result in an ℓ_1-embedding, and we cannot use known ℓ_1-sketching algorithns. This takes us to our algorithmic contribution, where we design the sketching algorithms for Step 2 with data-dependent edge weights. More generally, we introduce a two-step template for transforming data-dependent costs in the Quadtree into streaming algorithms. Conceptually, the approach generalizes the well-known ℓ_p sampling problem [4, 24, 25, 31] to *ℓ_p-sampling with meta-data*. For EMD, the high-level idea is the following: first, sample a coordinate of the vector proportional to the ℓ_1-distribution (i.e., the ℓ_1-sampling problem), and second, estimate the data-dependent edge weight for the coordinate sampled (the meta-data), so that we can scale the contribution of that coordinate appropriately.

1.2.2 Implementing Step 1: Quadtree Matching with Data-Dependent Edge Weights. We begin by describing our improved analysis of the randomized space partitioning algorithm, *Quadtree*. For the sake of simplicity, we focus on its analysis in the context of approximating EMD; the same ideas work similarly for MST. We begin by more formally introducing the Quadtree in high-dimensional spaces. In what follows, we focus on the case when the metric space is the hypercube with the Hamming distance, i.e. $A, B \subset \{0,1\}^d$ and $d(p,q) = \|p-q\|_1$ for $p, q \in \{0,1\}^d$. For the approximation, this is without loss of generality: one may embed $(\mathbb{R}^d, \ell_p)$ into $\{0,1\}^d$ by increasing the dimension. The new dimensionality is proportional to the dimension d, $\log n$, and the "aspect ratio" (maximum distance divided by minimum distance); since our space will have poly-logarithmic dependence on the dimension, the embedding introduces a logarithmic dependence on the aspect ratio which will also be incurred in the additive error of Theorem 1.3.

Quadtree. The Quadtree algorithm creates a randomized tree $\mathbf{T}$ with depth $h := \log_2 2d$ by recursively sub-dividing the hypercube $\{0,1\}^d$. Therefore, each node u in $\mathbf{T}$ will be associated with a subcube $S_u \subseteq \{0,1\}^d$, where the root r has $S_r = \{0,1\}^d$. To create these subcubes, each internal node u of $\mathbf{T}$ at depth $j < h$ is labeled with an ordered tuple of 2^j coordinates $(i_1, \ldots, i_{2^j}) \in [d]$ (which are not necessarily distinct), and has 2^{2^j} children. Each of the 2^{2^j} children of u will *uniquely* correspond to one of the 2^{2^j} fixings of the coordinates $(i_1, \ldots, i_{2^j}) \in \{0,1\}^{2^j}$. Specifically, each child v of u is assigned a unique bit-string $(b_1, \ldots, b_{2^j}) \in \{0,1\}^{2^j}$. The child v then corresponds to the subcube $S_v \subseteq S_u$ obtained by fixing the i_t-th coordinate to b_t, for each $t = 1, \ldots, 2^j$. We now describe the procedure for generating a random Quadtree $\mathbf{T}$:

(1) Uniformly sampling a tuple $(i_1, \ldots, i_{2^j}) \in [d]^{2^j}$ of 2^j coordinates independently for each node u at depth $j \in \{0, 1, \ldots, h-2\}$ to use as its label.
(2) Setting $(1, \ldots, d)$ as the label of every node at depth $h-1$.

A Quadtree $\mathbf{T}$ defines a map φ from $\{0,1\}^d$ to leaves of $\mathbf{T}$: $\varphi(p) = v$ if $p \in S_v$. Given A and B, we write A_v and B_v to denote $A \cap S_v$ and $B \cap S_v$ for each node v in $\mathbf{T}$.

Quadtrees in [3, 8]. Both works of *[3, 8]* use a tree structure that is very similar to the Quadtree used in this paper. In particular, they consider a slightly different algorithm which at depth i, samples 2^i coordinates from $[d]$ and divides into 2^{2^i} branches according to settings of $\{0,1\}$ to these 2^i coordinates (instead of each vertex independently sampling 2^i coordinates). For the sake of the analysis in Section 3, there will be no difference between independently sampling coordinates for each vertex in a level, and using the same sampled coordinates for each level. Thus, our analysis apply to trees of *[3, 8]* as well as the Quadtrees defined here.

Depth-greedy Matching from Quadtree. Given a random Quadtree $\mathbf{T}$, one obtains a natural depth-greedy matching as follows: We first map all points in $C = A \cup B$ to leaves of $\mathbf{T}$ using φ. Then, we greedily match points between A and B in a bottom up fashion, by walking each point up the tree level-by-level, and at each node one arbitrarily matches as many of the unmatched points from A and B as possible. Let $\mathbf{M}$ be any depth-greedy matching obtained from $\mathbf{T}$ in this fashion. The goal of our improved analysis of the Quadtree for EMD is to show that

$$\mathsf{EMD}(A,B) \le \textsc{Cost}(\mathbf{M}) \stackrel{\text{def}}{=} \sum_{(a,b)\in\mathbf{M}} \|a-b\|_1 \le \tilde{O}(\log n) \cdot \mathsf{EMD}(A,B)$$

with high probability (over the randomness of $\mathbf{T}$). Note that the first inequality is trivial.

Analysis of Quadtree via Tree Embeddings. Before presenting an overview of our new techniques, it will be helpful to begin with a recap of the analysis of [3] which can be used to show that $\textsc{Cost}(\mathbf{M}) \le O(\log n \log d) \cdot \mathsf{EMD}(A,B)$. The analysis of [3] starts by assigning a weight of $d/2^i$ to each edge from a node at depth i to a node at depth $i+1$ in $\mathbf{T}$. This defines a metric embedding $\varphi : A \cup B \to \mathbf{T}$ by mapping each point to a leaf of $\mathbf{T}$. The choice of edge weights is motivated by the observation that two points $x, y \in \{0,1\}^d$ with $\|x-y\|_1 = d/2^i$ are expected to have their paths diverge for the first time at depth i. If this is indeed the case then $d_{\mathbf{T}}(\varphi(x), \varphi(y))$ would capture $\|x-y\|_1$ up to a constant.

To upperbound $\textsc{Cost}(\mathbf{M})$, one studies the distortion of this embedding. Firstly, for any $\lambda > 1$ and $x, y \in \{0,1\}^d$, it is easy to verify that distances in the tree metric do not contract much:

$$\Pr_{\mathbf{T}}\left[d_{\mathbf{T}}(\varphi(x),\varphi(y)) < \frac{1}{\lambda}\cdot\|x-y\|_1\right] \le \left(1-\frac{\|x-y\|_1}{d}\right)^{1+2+\cdots+2^{\left\lfloor \log_2\left(\frac{\lambda d}{\|x-y\|_1}\right)\right\rfloor}} \le 2^{-\Omega(\lambda)}.$$

Thus by a union bound, for all $x, y \in A \cup B$ we have

$$\|x-y\|_1 \le O(\log n)\cdot d_{\mathbf{T}}(\varphi(x),\varphi(y)) \tag{1}$$

with probability at least $1 - 1/\mathrm{poly}(n)$, which essentially means that we can assume (1) in the worst case. As a result, we have

$$\begin{aligned}\textbf{Cost}(\mathbf{M}) &= \sum_{(x,y)\in\mathbf{M}}\|x-y\|_1 \le O(\log n)\sum_{(x,y)\in\mathbf{M}} d_{\mathbf{T}}(\varphi(x),\varphi(y)) \\ &\le O(\log n)\sum_{(x,y)\in M^*} d_{\mathbf{T}}(\varphi(x),\varphi(y)),\end{aligned}$$

where the last inequality holds for any matching M^* between A and B given that the depth-greedy matching is optimal under the tree metric. Setting M^* to be the optimal matching between A and B under the original ℓ_1 metric, we finish the proof by upperbounding $d_{\mathbf{T}}(\varphi(x),\varphi(y))$ using $O(\log d)\ \|x-y\|_1$. To see this, when $\|x-y\|_1 = \Theta(d/2^j)$, the probability that paths of x, y diverge at level $j-k$ is $\Theta(2^{-k})$ for each k, and when it does, $d_{\mathbf{T}}(\varphi(x),\varphi(y)) = \|x-y\|_1\cdot\Theta(2^k)$. Since $j \le h = O(\log d)$,

$$\begin{aligned}\mathbf{E}\big[d_{\mathbf{T}}(\varphi(x),\varphi(y))\big] &\le \|x-y\|_1 + \sum_{k=0}^{j}\Theta(2^{-k})\cdot\|x-y\|_1\cdot\Theta(2^k) \\ &= O(\log d)\cdot\|x-y\|_1.\end{aligned} \tag{2}$$

Together they yield the aforementioned $O(\log n \log d)\cdot\mathsf{EMD}(A,B)$ upper bound for $\textbf{Cost}(\mathbf{M})$.[5]

Tree Embeddings with Data-dependent Edge Weights. We show how to go beyond the distortion arguments of [3] by studying a tree embedding with data-dependent edge weights. In what follows, for any vertex $u \in \mathbf{T}$, let $C_u = A_u \cup B_u$ be the set of all points which map through u (recall $C = A \cup B$). The weight we assign to each edge (u, v)[6] of $\mathbf{T}$ will no longer be a fixed number $d/2^i$ but

$$\mathrm{avg}_{u,v} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\substack{\mathbf{c}\sim C_u\\ \mathbf{c}'\sim C_v}}\big[\|\mathbf{c}-\mathbf{c}'\|_1\big],$$

i.e., the average distance between a point drawn randomly from C_u and a point drawn randomly from C_v; when $C_v = \emptyset$ we define $\mathrm{avg}_{u,v} = 0$ by default. Let $d^*_{\mathbf{T}}$ denote the tree metric under this new set of weights. Again, the depth-greedy matching $\mathbf{M}$ we are interested in is optimal and the cost of $\mathbf{M}$ under the new tree embedding can be expressed as

$$\textbf{Value}_{\mathbf{T}}(A,B) \stackrel{\text{def}}{=} \sum_{(u,v)\in E_T}\big||A_v|-|B_v|\big|\cdot\mathrm{avg}_{u,v}.$$

where E_T is the set of edges of T.[7] On the one hand, $\textbf{Value}_{\mathbf{T}}(A,B)$ is at least $\textbf{Cost}(\mathbf{M})$ given that for any $x, y \in C$, we always have $\|x-y\|_1 \le d^*_{\mathbf{T}}(\varphi(x),\varphi(y))$ by triangle inequality. On the other hand, $\textbf{Value}_{\mathbf{T}}(A,B)$ is at most $\sum_{(a,b)\in M^*} d^*_{\mathbf{T}}(\varphi(a),\varphi(b))$ for any matching M and in particular, the optimal matching M^* under the ℓ_1 metric. As a result, it suffices to upperbound the cost of M^* under the data-dependent tree embedding by $\tilde{O}(\log n)\cdot\textbf{Cost}(M^*)$ given that $\textbf{Cost}(M^*) = \mathsf{EMD}(A,B)$. To this end it suffices to show that the expectation of $d^*_{\mathbf{T}}(\varphi(a),\varphi(b))$ for any $a, b \in C$ can be bounded from above by $\tilde{O}(\log n)\cdot\|a-b\|_1$.

Inspector Payment. Fix $a, b \in C$. We introduce the following quantity as the *inspector payment* of (a, b) with respect to the Quadtree $\mathbf{T}$. (We imagine the process as first drawing the Quadtree and then an "inspector" who examines the tree to track down a and b, making payments accordingly.) Formally we let $(\mathsf{v}_0(x), \mathsf{v}_1(x), \ldots, \mathsf{v}_h(x))$ denote the root-to-leaf path of x in a Quadtree $\mathbf{T}$. Then

$$\textbf{Pay}_{\mathbf{T}}(a,b) \stackrel{\text{def}}{=} \sum_{i\in[h]}\mathbf{1}\big\{\mathsf{v}_i(a)\neq\mathsf{v}_i(b)\big\}\cdot\left(\mathop{\mathbf{E}}_{\mathbf{c}\sim C_{\mathsf{v}_{i-1}(a)}}\big[\|a-\mathbf{c}\|_1\big]+\mathop{\mathbf{E}}_{\mathbf{c}\sim C_{\mathsf{v}_{i-1}(b)}}\big[\|b-\mathbf{c}\|_1\big]\right). \tag{3}$$

Intuitively, this payment scheme corresponds to an inspector who tracks down a and b from the root of T, and whenever a and b first diverge in the tree at node u, pays for a the average distance between a and a random point drawn from C_v for every node along the u-to-leaf path (including u); the inspector pays for b similarly. It again follows from triangle inequality that $2\cdot\textbf{Pay}_{\mathbf{T}}(a,b)$ is at least $d^*_{\mathbf{T}}(\varphi(a),\varphi(b))$. So it suffices to bound the expectation of $\textbf{Pay}_{\mathbf{T}}(a,b)$ by $\tilde{O}(\log n)\cdot\|a-b\|_1$.

Before giving a sketch of this proof, which is the most challenging part of our Quadtree analysis, we note that the inspector payment (3) depends on the data A and B, as well as the Quadtree $\mathbf{T}$ in two ways. The first is the depth when a and b first diverge, captured by the indicator $\mathbf{1}\{\mathsf{v}_i(a)\neq\mathsf{v}_i(b)\}$. The second is the average distance between a and C_v, which not only depends on a, but also on global properties of $C = A \cup B$. At a high level, incorporating this second aspect is the main novelty, since the average distance between a and C_v is an *average* notion of radii at v. Therefore, if the inspector pays a large amount, then an average point in C_v is far from a (as opposed to the farthest point implied by worst-case radii).

Bounding Inspector Payments. Consider fixed $a, b \in C$ at distance $\|a-b\|_1 = \Theta(d/2^j)$, and we give some intuition behind our upper bound on the expectation of the a-part of the payment:

$$\sum_{i\in[h]}\mathbf{1}\big\{\mathsf{v}_i(a)\neq\mathsf{v}_i(b)\big\}\cdot\mathrm{avg}_{a,i-1},\quad\text{where}\quad \mathrm{avg}_{a,i-1} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim C_{\mathsf{v}_{i-1}(a)}}\big[\|a-\mathbf{c}\|_1\big].$$

We will ignore the indicator random variable $\mathbf{1}\big\{\mathsf{v}_i(a)\neq\mathsf{v}_i(b)\big\}$ and use linearity of expectation to focus on $\mathbf{E}_{\mathbf{T}}[\mathrm{avg}_{a,i}]$. (With the

[5] The reason that this analysis can achieve approximation $O(\min\{\log n, \log d\}\log n)$, as opposed to $O(\log n \log d)$ is that with probability $1 - 1/n$, every $x, y \in A \cup B$ with $\|x-y\|_1 = \Theta(d/2^j)$ diverges at depth after $j - O(\log n)$.

[6] We always use u in (u, v) to denote the parent and v to denote the child.

[7] We remark that one can define an analogous quantity for the case of MST, where given a single set $X \subset [\Delta]^d$, we set $\textbf{Value}_{\mathbf{T}}(X) = \sum_{(u,v)\in E_T}\mathbf{1}(|X_v|)\cdot\mathrm{avg}_{u,v}$, where $\mathbf{1}: \mathbb{R}\to\{0,1\}$ is the indicator function (i.e., $\mathbf{1}(x) = 0$ if and only if $x = 0$). It is this quantity that we will analyze in our results for MST.

indicator random variable, we need to consider the expectation of $\text{avg}_{a,i}$ conditioning on the event that a, b have diverged. The conditioning will not heavily influence the geometric intuition, so we will ignore this for the rest of this overview).

Let $\mathsf{v}_i = \mathsf{v}_i(a)$. Similar to worst-case bounds on radii, $\mathbf{E}_{\mathbf{T}}[\,\text{avg}_{a,i}\,]$ can still be $d/2^i \cdot \Omega(\log n)$. As an example, let i_1 be a relatively large depth and for some small $\epsilon \approx 10^{-6}$, consider a set P_1 of n^ϵ many points at distance $\epsilon \log n \cdot d/2^{i_1}$ around a. Then, at depth i_1 of a random Quadtree $\mathbf{T}$, a point in P_1 traverses down to node v_{i_1} with non-negligible probability, roughly $1/n^{-\epsilon}$. If no other points lie closer to a than those in P_1, then $\mathbf{E}_{\mathbf{T}}[\,\text{avg}_{a,i_1}\,] = d/2^{i_1} \cdot \Omega(\epsilon \log n)$, since it is likely that some points of P_1 make it to v_{i_1} and significantly increase the average distance between a and $C_{\mathsf{v}_{i_1}}$. If this happened on a for every depth i, the inspector would be in trouble, as there are $O(\log d)$ levels and a similar argument to that of worst-case radii would mean a payment of $O(\log d \log n) \cdot \|a - b\|_1$.

However, we claim if the arrangement of P_1 resulted in $\mathbf{E}_{\mathbf{T}}[\,\text{avg}_{a,i_1}\,] = d/2^{i_1} \cdot \Omega(\epsilon \log n)$, the same situation will be a lot more difficult to orchestrate for depth $i_2 \leq i_1 - O(\log\log n)$. In particular, at depth i_2, in order to have $\mathbf{E}_{\mathbf{T}}[\,\text{avg}_{a,i_2}\,] = d/2^{i_2} \cdot \Omega(\epsilon \log n)$, there must be a set of points P_2 at distance $d/2^{i_2} \cdot \Omega(\epsilon \log n)$ which cause avg_{a,i_2} to be large. However, it is no longer enough to have $|P_2| = n^\epsilon$. The reason is that points of P_1 in v_{i_2} will help bring down the average distance. Since points in P_1 are at distance $\epsilon \log n \cdot d/2^{i_1} \ll d/2^{i_2}$ from a, there will oftentimes be $\Omega(n^\epsilon)$ points from P_1 in v_{i_2}. In order to significantly increase the average distance, v_{i_2} must oftentimes have at least $n^\epsilon/\text{polylog}(n)$ points from P_2; otherwise, avg_{a,i_2} will be mostly the average distance between a and points in P_1. Since any given point from P_2 traverses down to v_{i_2} with probability roughly $1/n^\epsilon$, we must have $|P_2| \geq n^{2\epsilon}/\text{polylog}(n)$. This argument can only proceed for at most $O(1/\epsilon)$ depths before $|P_{O(1/\epsilon)}| > 2n$, in which case we obtain a contradiction, since all points are in $A \cup B$.

Generally, in order to increase the average distance between a and C_{v_i} *multiple times* as the depth i goes down, the number of points around a at increasing distances must grow very rapidly. More specifically, we show that if a depth i is "bad," meaning that $\mathbf{E}_{\mathbf{T}}[\,\text{avg}_{a,i}\,] \geq \alpha \cdot d/2^i$ for some $\alpha = \omega(\log\log n)$, then the number of points within a ball of radius $d/(2^i \log n)$ around a and within a larger ball of radius $O(\log n \cdot d/2^i)$ around a must have increased by a factor of $\exp(\Omega(\alpha))$; this means the number of such depths i is at most $((\log n)/\alpha) \cdot \text{poly}(\log\log n)$. Combining this analysis and the fact that a and b must diverge in order to incur payment from the inspector, we obtain our upper bound that the expectation of $\textbf{P}\textsc{ay}_{\mathbf{T}}(a, b)$ is at most $\tilde{O}(\log n) \cdot \|a - b\|_1$.

1.2.3 Implementing Step 2: From Quadtree to Sketching Algorithms. By the prior discussion, after sampling a Quadtree $\mathbf{T}$, we know that the quantity $\textbf{Value}_{\mathbf{T}}(A, B)$ is a $\tilde{O}(\log n)$ approximation of the true cost $\mathsf{EMD}(A, B)$. Specifically, we have:

$$\mathsf{EMD}(A, B) \leq \textbf{Value}_{\mathbf{T}}(A, B) \leq \tilde{O}(\log n) \cdot \mathsf{EMD}(A, B) \qquad (4)$$

Thus, the approach of our sketching algorithm is simply to approximate $\textbf{Value}_{\mathbf{T}}(A, B)$. We will decompose $\textbf{Value}_{\mathbf{T}}(A, B)$ based on its level: $\textbf{Value}_{\mathbf{T}}(A, B) = \sum_{i=1}^{h} \textbf{Value}_{\mathbf{T},i}(A, B)$, where

$$\textbf{Value}_{\mathbf{T},i}(A, B) \stackrel{\text{def}}{=} \sum_{\substack{(u,v) \in E_T \\ \textbf{depth}(u,v)=i}} \big||A_v| - |B_v|\big| \cdot \text{avg}_{u,v}$$

where $\textbf{depth}(e)$ for an edge $e \in \mathbf{T}$ is the depth of the child vertex in e. We will attempt to estimate each $\textbf{Value}_{\mathbf{T},i}(A, B)$ independently for each i, so in what follows we now fix any level $i \in [h]$.

We start with some notation. For any (non-root) vertex $v \in \mathbf{T}$, let $\pi(v)$ be the parent of v in $\mathbf{T}$. We then define the *discrepancy* vector for level i, denoted Δ^i, by $\Delta^i_v = |A_v| - |B_v|$ for every vertex v at depth i of the tree (i.e., Δ^i has a coordinate Δ^i_v for each vertex v at depth i). Next, for any vector $x \in \mathbb{R}^N$ and any $p \geq 0$, we define the ℓ_p distribution $\mathcal{D}_p(x)$ over the coordinates of x via $\mathcal{D}_p(x) = \left(\frac{|x_1|^p}{\|x\|_p^p}, \frac{|x_2|^p}{\|x\|_p^p}, \ldots, \frac{|x_N|^p}{\|x\|_p^p}\right)$ for $p > 0$, and for $p = 0$ we define $\mathcal{D}_0(x)$ to be the uniform distribution over the support of x. Now observe:[8]

$$\textbf{Value}_{\mathbf{T},i}(A, B) = \|\Delta^i\|_1 \cdot \mathop{\mathbf{E}}_{\mathbf{v} \sim \mathcal{D}_1(\Delta^i)} \left[\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}\right]$$

Thus, we can write $\textbf{Value}_{\mathbf{T},i}(A, B)$ as the ℓ_1 norm of Δ^i, multiplied by the expected value of $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ taken over drawing a vertex $\mathbf{v}$ in level i with probability proportional to $|\Delta_{\mathbf{v}}| = \big||A_{\mathbf{v}}| - |B_{\mathbf{v}}|\big|$. Note that the norm $\|\Delta^i\|_1$ can be easily estimated using the ℓ_1 sketches of Indyk [21]. Thus, this simple manipulation motivates the following approach: **(1)** sample a vertex $\mathbf{v}$ from level i from the distribution $\mathcal{D}_1(\Delta^i)$, **(2)** recover the value $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$, **(3)** repeat enough times so that the empirical mean of the variables $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ is a good approximation of the expectation $\mathbf{E}_{\mathbf{v} \sim \mathcal{D}_1(\Delta^i)}\left[\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}\right]$.

For the last step, we note that it will be straightforward to bound the standard deviation of the variable $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ by $O(d \log n/2^i)$, which is within a $O(\log n)$ factor of the error to which we will need to estimate the expectation. Thus, if we can carry out steps **(1)** and **(2)** which sample $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ from the correct distribution, we need only repeat them $\text{polylog}(n)$ times to estimate $\textbf{Value}_{\mathbf{T},i}(A, B)$ to sufficiently small error.

Two-Pass Streaming Algorithms. We first describe how the above two steps can be carried out in *two-passes* over the data-stream. Perhaps unsurprisingly, our approach will be to carry out **(1)** on the first pass, obtaining a set of vertices v sampled from the correct distribution $\mathcal{D}_1(\Delta^i)$, and carry out **(2)** on the second pass, where we recover the actual value of $\text{avg}_{\pi(v),v}$ for the vertices v that were sampled.

More formally, our two-pass streaming algorithm proceeds as follows. First we draw a Quadtree $\mathbf{T}$ (for which we may assume (4) holds) and then for each $i \in [h]$, we estimate $\textbf{Value}_{\mathbf{T},i}(A, B)$ as follows. In the first pass we can estimate $\|\Delta^i\|_1$ to error $(1 \pm 1/2)$ with an ℓ_1-sketch [22], and we also can sample $\mathbf{v} \sim \mathcal{D}_1(\Delta^i)$ via known algorithms for ℓ_1-sampling [4, 24, 25]. Furthermore, once a vertex v is fixed, we may estimate $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ in the second round by a point in $C_{\pi(v)}$ and in C_v (via standard sub-sampling

[8] We remark that for the case of MST, the relevant quantity $\textbf{Value}_{\mathbf{T},i}(X)$ below can be written as $\|\Delta^i\|_0 \cdot \mathbf{E}_{v \sim \mathcal{D}_0(\Delta^i)}\left[\text{avg}_{\pi(v),v}\right]$. Namely, we simply replace the ℓ_1 norm in both the scaling and the distribution by the ℓ_0 norm. Thus, the high-level approach to sketching MST will be similar. However, due to using the ℓ_0 instead of the ℓ_1 norm, an entirely different set of techniques will be required to implement each of the steps.

techniques) and approximating their distance using ℓ_1 sketches. By concurrently repeating this process polylog(n) times, we obtain our desired approximation

The remaining challenge, however, is to produce $\mathbf{v} \sim \mathcal{D}_1(\Delta^i)$ *and* an estimate of $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ *simultaneously* in a single pass over the data. This task is a special case of a problem we call *sampling with meta-data*, since the quantity $\text{avg}_{\pi(\mathbf{v}),\mathbf{v}}$ will be the *meta-data* of the sample $\mathbf{v} \sim \mathcal{D}_1(\Delta^i)$ needed to estimate $\mathbf{Value}_{\mathbf{T},i}(A,B)$.

Sampling with Meta-Data and One-Pass Streaming The key task of sampling with meta-data is the following: for $n,k \in \mathbb{N}$, we are given a vector $x \in \mathbb{R}^n$ and collection of *meta-data* vectors $\lambda_1, \lambda_2, \dots, \lambda_n \in \mathbb{R}^k$, and the goal is to sample $i \in [n]$ with probability $|x_i|/\|x\|_1$ (or more generally, $|x_i|^p/\|x\|_p^p$), and output both i and an approximation $\widehat{\lambda_i} \in \mathbb{R}^k$ of the vector λ_i. The challenge is to solve this problem with a small-space linear sketches of x and the meta-data vectors $\lambda_1, \dots, \lambda_n$. It is not hard to see that sampling with meta-data is exactly the problem we seek to solve for linear sketching of EMD.[9]

Our algorithm builds on a powerful sketching technique known as *precision sampling* [4, 24, 25] for sampling an index $i \in [n]$ proportional to $|x_i|/\|x\|_1$ for a vector $x \in \mathbb{R}^n$ (or more generally, for $|x_i|^p/\|x\|_p^p$, but we focus on $p = 1$). The idea is to produce, for each $i \in [n]$ an independent exponential random variable $\boldsymbol{t}_i \sim \text{Exp}(1)$, and construct a "scaled vector" $\boldsymbol{z} \in \mathbb{R}^n$ with coordinates $\boldsymbol{z}_i = x_i/\boldsymbol{t}_i$. One then attempts to return the index $i_{\max} = \text{argmax}_{i\in[n]} \boldsymbol{z}_i$, since

$$\Pr_{\boldsymbol{t}_1,\dots,\boldsymbol{t}_n \sim \text{Exp}(1)}\left[\underset{i'\in[n]}{\text{argmax}} \frac{|x_{i'}|}{\boldsymbol{t}_{i'}} = i\right] = \frac{|x_i|}{\|x\|_1}.$$

To find the index $i_{\max}$ with a linear sketch, we can use a "heavy-hitters" algorithm, such as the Count-Sketch of [14].[10] Specifically, Count-Sketch with error $\epsilon \in (0,1)$ allows us to recover an estimate $\tilde{\boldsymbol{z}}$ to $\boldsymbol{z}$ satisfying (roughly) $\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|_\infty \le \epsilon\|\boldsymbol{z}\|_2$. Then one can show that $\text{argmax}_{i'\in[n]} |\tilde{\boldsymbol{z}}_{i'}|$ is close to being distributed as $|x_i|/\|x\|_1$.

In order to sample with meta-data, our sketch similarly samples independent exponential $\boldsymbol{t}_1, \dots, \boldsymbol{t}_n \sim \text{Exp}(1)$ and applies a Count-Sketch data structure on $\boldsymbol{z} \in \mathbb{R}^n$, where $\boldsymbol{z}_i = x_i/\boldsymbol{t}_i$, and obtains an estimate $\tilde{\boldsymbol{z}}$ of $\boldsymbol{z}$. In addition, we apply a Count-Sketch data structure with error ϵ for the vector $\boldsymbol{w}$ with coordinates given by the values $\lambda_i/\boldsymbol{t}_i$, namely $\boldsymbol{w}_i = \lambda_i/\boldsymbol{t}_i$ (recall that we are assuming that the meta-data λ_i are scalars for this discussion). From this we obtain an estimate $\tilde{\boldsymbol{w}}$ of $\boldsymbol{w}$. The insight is the following: suppose the sample produced is $i^* \in [n]$, which means it satisfies $\tilde{\boldsymbol{z}}_{i^*} \approx \max_{i\in[n]} |x_i|/\boldsymbol{t}_i$. Then the value $\boldsymbol{t}_{i^*}$ should be relatively small: in particular, one can show that we expect $\boldsymbol{t}_{i^*}$ to be $\Theta(|x_{i^*}|/\|x\|_1)$, so that $\boldsymbol{z}_{i^*} \approx \Theta(\|x\|_1) = \Theta(\|\Delta^i\|_1)$. When this occurs, for each $\ell \in [k]$, the guarantees of Count-Sketch imply that the estimate $\boldsymbol{t}_{i^*} \cdot \tilde{\boldsymbol{w}}_{i^*}^\ell$ satisfies

$$\begin{aligned}|\boldsymbol{t}_{i^*} \cdot \tilde{\boldsymbol{w}}_{i^*} - \lambda_{i^*}| &= \boldsymbol{t}_{i^*} \, |\tilde{\boldsymbol{w}}_{i^*} - \boldsymbol{w}_{i^*}| \\ &\le \epsilon \boldsymbol{t}_{i^*} \|\boldsymbol{w}\|_2 \left(= O\left(\epsilon |x_{i^*}| \cdot \frac{\|\lambda\|_1}{\|x\|_1}\right) \text{ in expectation}\right)\end{aligned}$$

where $\lambda \in \mathbb{R}^n$ is the vector with coordinates given by the meta-data $\lambda_1, \dots, \lambda_n$. In other words, if the size of λ_{i^*} is comparable to $|x_{i^*}|$, and if the ratio $\|\lambda\|_1/\|x\|_1$ of the meta-data norms to the norm of x is bounded, then $\boldsymbol{t}_{i^*}\tilde{\boldsymbol{w}}_{i^*}^\ell$ is a relatively good approximation to λ_{i^*}.

Unfortunately, in our application, the above will not always be the case. In particular, the norm of the meta-data $\|\lambda\|_1$ may be much, even poly(n), larger than $\|x\|_1 = \|\Delta^i\|_1$. Intuitively, the issue is that each coordinate λ_v is a sketch of $\text{avg}_{\pi(v),v}$, which is a function both of the points in $C_{\pi(v)}$ and C_v. Thus, the size of the sketch of $\text{avg}_{\pi(v),v}$ depends on all the points in $C_{\pi(v)}$. Moreover, for *every* other sibling v' of v (meaning that $\pi(v') = \pi(v)$), the sketch of $\text{avg}_{\pi(v'),v'}$ will also have to take into account the same information from $C_{\pi(v)}$. Thus, this information is *duplicated* in the sketches of the meta-data, by a number of times equal to the number of children of $\pi(v)$. This duplication, or repetition of the same information in the sketch, results in a blow-up of the norm of λ so that $\|\lambda\|_1 = \Omega(\kappa \cdot \|\Delta^i\|_1)$, where κ is the maximum number of non-empty children of any parent in level $i-1$. Since κ can be poly(n), this is an non-trivial challenge.

Our solution to this, at a high level, is to develop a *two-step* precision sampling with meta-data algorithm to avoid duplication of meta-data. Instead of sampling the vertex $\mathbf{v}^*$ directly, we first sample a parent $\boldsymbol{u}^*$ from level $i-1$ with probability proportional to the ℓ_1-norm of Δ^i restricted to coordinates corresponding to the children of $\boldsymbol{u}^*$; namely, we sample $\boldsymbol{u}^*$ with probability proportional to $\sum_{v:\pi(v)=\boldsymbol{u}^*} |\Delta_v^i|$. Then, we use the precision sampling sketch which recovered $\boldsymbol{u}^*$ to recover a sketch of the a randomly selected point in $C_{\boldsymbol{u}^*}$. Next, once we have $\boldsymbol{u}^*$, we apply precision sampling with meta-data once more, to sample a child $\mathbf{v}^*$ of $\boldsymbol{u}^*$ proportional to $|\Delta_{\mathbf{v}^*}^i|$, and then recover a sketch of a randomly selected point in $C_{\mathbf{v}^*}$. One can then put the two sketches from $C_{\boldsymbol{u}^*}, C_{\mathbf{v}^*}$ together to estimate $\text{avg}_{\boldsymbol{u}^*,\mathbf{v}^*}$.

To accomplish this two-part precision sampling scheme, we must generate a second set of exponentials $\{\boldsymbol{t}_v\}_v$, one for each child node v at depth i. In order to ensure that the sample produced by the second sketch actually returns a child $\mathbf{v}^*$ of $\boldsymbol{u}^*$, and not a child of some other node, we crucially must scale the vector Δ^i by *both* the child exponentials $\{\boldsymbol{t}_v\}_v$ as well as the parent exponentials $\{\boldsymbol{t}_u\}_u$ from the first sketch. Thus, in the second sketch we analyze the twice-scale vector $\boldsymbol{z}$ with coordinates $\boldsymbol{z}_v = \Delta_v^i/(\boldsymbol{t}_{\pi(v)}\boldsymbol{t}_v)$, and attempt to find the largest coordinate of $\boldsymbol{z}$. Importantly, notice that this makes the scaling factors in $\boldsymbol{z}_v$ no longer independent: two children of the same parent share one of their scaling factors. Thus, executing this plan requires a careful analysis of the behavior of norms of vectors scaled by several non-independent variables with heavy-tailed distributions.

The advantage of this two-part scheme is that now there is no duplication of meta-data, since in the first step there is only one λ_u for each parent u, and in the second step, by conditioning on the parent exponential $\boldsymbol{t}_{\boldsymbol{u}^*}$ being sufficiently small, we ensure that the only meta-data that contributes non-trivially to the error of the

[9] Namely, x is the vector Δ^i, and the meta-data vectors λ_v are $k = \text{polylog}(n)$-dimensional ℓ_1 *sketches* of the values of $\text{avg}_{\pi(v),v}$. In the following discussion, for simplicity we omit the details on the ℓ_1 sketches for $\text{avg}_{\pi(v),v}$, since they proceed via somewhat standard techniques, and instead assume that the meta-data is exactly given by the scalars $\lambda_{\mathbf{v}} \approx \text{avg}_{\pi(v),v}$.

[10] We do not explicitly use count-sketch in our one-pass algorithms, and instead apply a sketching procedure closely inspired by Count-Sketch.

sketch are the λ_v for children v of $\boldsymbol{u}^*$. This allows us, ultimately, to obtain our guarantees for one-pass streaming algorithms for EMD. The case of MST is similar at a high-level, however implementing the two-part precision sampling scheme requires an entirely different set of sketching tools, resulting from the fact that we now need to sample a vertex v from the ℓ_0 distribution $\mathcal{D}_0(\Delta^i)$.

1.3 Organization

Due to space constraints, we focus on the analysis of quadtrees for EMD and MST in the conference version of the paper (i.e., Step 1 as sketched in Section 1.2.2. Proofs of our main results can be found in the full version [15].

2 PRELIMINARIES

Given $n \geq 1$ we write $[n]$ to denote $\{1, \ldots, n\}$. Given a vector $x \in \mathbb{R}^n$ and a real number $t \geq 0$, we define $x_{-t} \in \mathbb{R}^n$ to be the vector obtained by setting the largest $\lfloor t \rfloor$ coordinates of x in magnitude equal to 0 (breaking ties by using coordinates with smaller indices). For $a, b \in \mathbb{R}$ and $\epsilon \in (0,1)$, we use the notation $a = (1 \pm \epsilon)b$ to denote the containment of $a \in [(1-\epsilon)b, (1+\epsilon)b]$.

For convenience, we will assume without loss of generality that d is always a power of 2 and write $h := \log_2 2d = \log_2 d + 1$. Given a node v in a rooted tree T, when v is not the root we use $\pi(v)$ to denote the parent node of v in T.

Next we give a formal definition of Quadtrees used in this paper:

Definition 2.1 (Quadtrees). Fix $d \in \mathbb{N}$. A quadtree is a rooted tree T of depth $h := \log_2 2d$. We say a node v of T is at depth j if there are $j+1$ nodes on the root-to-v path in T (so the root is at depth 0 and its leaves are at depth h). Each internal node v of T at depth $j < h$ is labelled with an ordered tuple of 2^j coordinates $i_1, \ldots, i_{2^j} \in [d]$ (which are not necessarily distinct), and has 2^{2^j} children, each of which we refer to as the $(b_1, \ldots, b_{2^j})$-child of v with $b_1, \ldots, b_{2^j} \in \{0,1\}$. Every node at depth $h-1$ is labelled with $(1, \ldots, d)$. We write E_T to denote the edge set of T. Whenever we refer to an edge $(u,v) \in E_T$, u is always the parent and v is the child. A random quadtree $\mathbf{T}$ is drawn by (1) sampling a tuple of 2^j coordinates uniformly and independently from $[d]$ for each node at depth $j < h-1$ as its label; and (2) use $(1, 2, \ldots, d)$ as the label of every node at depth $h-1$. We use $\mathcal{T}$ to denote this distribution of random quadtrees.

Given a quadtree T, each point $x \in \{0,1\}^d$ induces a root-to-leaf path by starting at the root and repeatedly going down the tree as follows: If the current node v is at depth $j < h$ and is labelled with $(i_1, \ldots, i_{2^j})$, then we go down to the $(x_{i_1}, \ldots, x_{i_{2^j}})$-child of v. We write

$$\mathsf{v}_{0,T}(x), \mathsf{v}_{1,T}(x), \ldots, \mathsf{v}_{h,T}(x)$$

to denote this root-to-leaf path, where each $\mathsf{v}_{j,T}$ is a map from $\{0,1\}^d$ to nodes of T at depth j. We usually drop T from the subscript when it is clear from the context.

Alternatively we define a subcube $S_{v,T} \subseteq \{0,1\}^d$ for each v: The set of the root is $\{0,1\}^d$; If (u,v) is an edge, u is at depth j and is labelled with $i_1, \ldots, i_{2^j}$, and v is the $(b_1, \ldots, b_{2^j})$-child of u, then

$$S_{v,T} = \left\{ x \in S_{u,T} : (x_{i_1}, \ldots, x_{i_{2^j}}) = (b_1, \ldots, b_{2^j}) \right\}.$$

Note that $S_{v,T}$'s of nodes v at the same depth form a partition of $\{0,1\}^n$. The root-to-leaf path for $x \in \{0,1\}^d$ can be equivalently defined as the sequence of nodes v that have $x \in S_{v,T}$.

3 ANALYSIS OF QUADTREES FOR EMD AND MST

Our goal in this section is to obtain expressions based on quadtrees that are good approximations of EMD and MST. They will serve as the starting point of our sketches for EMD and MST later.

3.1 Approximation of EMD using Quadtrees

Fix $n, d \in \mathbb{N}$ and let T be a quadtree of depth $h = \log_2 2d$. Let A and B be two multisets of points from $\{0,1\}^d$ of size n each. For each node v in T, we define

$$A_{v,T} \stackrel{\text{def}}{=} \{a \in A : \mathsf{v}_{i,T}(a) = v\} \quad \text{and} \quad B_{v,T} \stackrel{\text{def}}{=} \{b \in B : \mathsf{v}_{i,T}(b) = v\}.$$

Equivalently we have $A_{v,T} = A \cap S_{v,T}$ and $B_{v,T} = B \cap S_{v,T}$. Let $C_{v,T} = A_{v,T} \cup B_{v,T}$. We give the definition of depth-greedy matchings.

Definition 3.1. Let T be a quadtree. For any $a \in A$ and $b \in B$, let

$$\text{depth}_T(a,b) \stackrel{\text{def}}{=} \text{depth of the least-common ancestor of leaves of } a, b \text{ in } T.$$

The class of *depth-greedy matchings*, denoted by $\mathcal{M}_T(A,B)$, is the set of all matchings $M \subseteq A \times B$ which maximize the sum of $\text{depth}_T(a,b)$ over all pairs $(a,b) \in M$. We write

$$\textbf{Cost}(M) = \sum_{(a,b) \in M} \|a - b\|_1$$

to denote the cost of a matching M between A and B. Recall that $\mathsf{EMD}(A,B)$ is defined as the minimum of $\textbf{Cost}(M)$ over all matchings between A and B.

For each edge $(u,v) \in E_T$, we use $\text{avg}_{u,v,T}$ to denote the average distance between points of $C_{u,T}$ and $C_{v,T}$:

$$\text{avg}_{u,v,T} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\substack{\mathbf{c} \sim C_{u,T} \\ \mathbf{c}' \sim C_{v,T}}} \left[\|\mathbf{c} - \mathbf{c}'\|_1 \right],$$

where both $\mathbf{c}$ and $\mathbf{c}'$ are drawn uniformly at random; we set $\text{avg}_{u,v,T}$ to be 0 by default when $C_{v,T}$ is empty. For notational simplicity, we will suppress T from the subscript when it is clear from the context. We are now ready to define the *value* of (A,B) in a quadtree T:

Definition 3.2. Let T be a quadtree. The value of (A,B) in T is defined as

$$\textbf{\textit{Value}}_T(A,B) \stackrel{\text{def}}{=} \sum_{(u,v) \in E_T} \left| |A_v| - |B_v| \right| \cdot avg_{u,v}. \tag{5}$$

We note that the right-hand side of (5) is data-dependent in two respects: the discrepancy between $|A_v|$ and $|B_v|$ and the average distance $\text{avg}_{u,v}$ between points in C_u and C_v.

Our main lemma for EMD shows that the value of (A,B) in a randomly chosen quadtree $\mathbf{T} \sim \mathcal{T}$ and the cost of any depth-greedy matching are all $\tilde{O}(\log n)$-approximations to $\mathsf{EMD}(A,B)$.

Lemma 3.3 (Quadtree lemma for EMD). *Let (A,B) be a pair of multisets of points from $\{0,1\}^d$ of size n each. Let $\mathbf{T} \sim \mathcal{T}$. Then with probability at least 0.99, every $M \in \mathcal{M}_{\mathbf{T}}(A,B)$ satisfies we have*

$$\mathsf{EMD}(A,B) \le \textbf{\textit{Cost}}(M) \le \mathbf{Value}_{\mathbf{T}}(A,B) \le \tilde{O}(\log n)\cdot \mathsf{EMD}(A,B). \tag{6}$$

We start with the left-most inequality in (6). Indeed we will show that $\mathsf{EMD}(A,B) \le \mathbf{Value}_T(A,B)$ for any quadtree T (Lemma 3.4). To this end we prove that $\mathbf{Cost}(M) \le \mathbf{Value}_T(A,B)$ for any depth-greedy matching between A and B obtained from T; the latter by definition is at least $\mathsf{EMD}(A,B)$.

Lemma 3.4. *Let T be any quadtree. Then $\textbf{\textit{Cost}}(M) \le \mathbf{Value}_T(A,B)$ for any $M \in \mathcal{M}_T(A,B)$.*

Proof: Given an $M \in \mathcal{M}_T(A,B)$ and a pair $(a,b)\in M$, we write v and w to denote the leaves of a and b and use $v = u_1, u_2, \ldots, u_k = w$ to denote the path from v to w in T. By triangle inequality,

$$\begin{aligned}\|a-b\|_1 &\le \mathop{\mathbf{E}}_{\mathbf{c}_i \sim C_{u_i}}\Big[\|a-\mathbf{c}_1\|_1 + \|\mathbf{c}_1-\mathbf{c}_2\|_1 + \cdots + \|\mathbf{c}_k - b\|_1\Big]\\ &= \mathrm{avg}_{u_1,u_2} + \cdots + \mathrm{avg}_{u_{k-1},u_k},\end{aligned}$$

where the equation follows from the fact the label of every node at depth $h-1$ is $(1,2,\ldots,d)$ and thus, all points at a leaf must be identical. Summing up these inequalities over all $(a,b)\in M$ gives exactly $\mathbf{Value}_T(A,B)$ on the right hand side. For this, observe that every M in $\mathcal{M}_T(A,B)$ has the property that, for any edge (u,v) in T, the number of $(a,b)\in M$ such that the path between their leaves contains (u,v) is exactly $\big||A_v|-|B_v|\big|$. ■

Now it suffices to upperbound $\mathbf{Value}_{\mathbf{T}}(A,B)$ by $\tilde{O}(\log n)\cdot \mathsf{EMD}(A,B)$ with probability at least 0.9 for a random quadtree $\mathbf{T} \sim \mathcal{T}$. For this purpose we let $C = A \cup B$ and define an *inspector payment* for any pair of points $a,b \in C$[11] based on a quadtree. Given $a,b\in C$, we let

$$\mathbf{Pay}_T(a,b) \stackrel{\text{def}}{=} \sum_{i\in[h]} \mathbf{1}\big\{\mathrm{v}_i(a) \ne \mathrm{v}_i(b)\big\}\cdot\Big(\mathrm{avg}_{a,i-1} + \mathrm{avg}_{b,i-1}\Big) \tag{7}$$

where

$$\mathrm{avg}_{a,i-1} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim C_{\mathrm{v}_{i-1}(a)}}\big[\|a-\mathbf{c}\|_1\big] \text{ and}$$
$$\mathrm{avg}_{b,i-1} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim C_{\mathrm{v}_{i-1}(b)}}\big[\|b-\mathbf{c}\|_1\big].$$

Intuitively $\mathbf{Pay}_T(a,b)$ pays for the average distance between a (or b) and points in $C_{\mathrm{v}_i(a)}$ (or $C_{\mathrm{v}_i(b)}$) along its root-to-leaf path but the payment only starts *at* the least-common ancestor of leaves of a and b. Note that $\mathbf{Pay}_T(a,b)=0$ trivially if $a=b$.

We show that for any matching M between A and B, the total inspector payment from $(a,b)\in M$ is enough to cover $\mathbf{Value}_T(A,B)$:

Lemma 3.5. *Let T be any quadtree and M be any matching between A and B. Then we have*

$$\mathbf{Value}_T(A,B) \le 2\sum_{(a,b)\in M} \textbf{\textit{Pay}}_T(a,b). \tag{8}$$

[11] While we will always have $a\in A$ and $b\in B$ in this subsection, this more general setting allows us to apply what we prove in this subsection to work on MST later.

Proof: Using the definition of $\mathbf{Value}_T(A,B)$, it suffices to show that

$$\sum_{(u,v)\in E_T} \big||A_v|-|B_v|\big|\cdot \mathrm{avg}_{u,v} \le 2\sum_{(a,b)\in M}\mathbf{Pay}_T(a,b).$$

By triangle inequality (and $\mathrm{avg}_{a,h}=0$ because every point in $C_{\mathrm{v}_h(a)}$ is identical to a)

$$\begin{aligned}&2\cdot \mathbf{Pay}_T(a,b)\\ &\ge \sum_{i\in[h]}\mathbf{1}\big\{\mathrm{v}_i(a)\ne \mathrm{v}_i(b)\big\}\cdot\Big(\mathrm{avg}_{a,i-1}+\mathrm{avg}_{a,i}+\mathrm{avg}_{b,i-1}+\mathrm{avg}_{b,i}\Big)\\ &\ge \sum_{i\in[h]}\mathbf{1}\big\{\mathrm{v}_i(a)\ne \mathrm{v}_i(b)\big\}\cdot\Big(\mathrm{avg}_{\mathrm{v}_{i-1}(a),\mathrm{v}_i(a)}+\mathrm{avg}_{\mathrm{v}_{i-1}(b),\mathrm{v}_i(b)}\Big),\end{aligned}$$

i.e., $2\cdot\mathbf{Pay}_T(a,b)$ is enough to cover $\mathrm{avg}_{u,v}$ for every edge (u,v) along the path between the leaf of u and the leaf of v. The lemma then follows from the following claim: For every edge (u,v) in T, $\big||A_v|-|B_v|\big|$ is at most the number of points $a\in A_v$ such that its matched point in M is not in B_v plus the number of points $b\in B_v$ such that its matched point in M is not in A_v. This follows from the simple fact that every $(a,b)\in M$ with $a\in A_v$ and $b\in B_v$ would get cancelled in $|A_v|-|B_v|$. This finishes the proof of the lemma. ■

By Lemma 3.5 the goal now is to upperbound the total inspector payment by $\tilde{O}(\log n)\cdot\mathsf{EMD}(A,B)$ with probability at least 0.9 over a randomly picked quadtree $\mathbf{T}$. We consider a slight modification of the payment scheme given in (7) which we define next; the purpose is that the latter will be easier to bound in expectation, and most often exactly equal to (7).

Specifically, given any (a,b) with $a,b\in C$ and $i_0\in[0:h-1]$, we let

$$\mathbf{Pay}^*_{i_0,T}(a,b) \stackrel{\text{def}}{=} \sum_{i>i_0}^{h}\mathbf{1}\big\{\mathrm{v}_i(a)\ne\mathrm{v}_i(b)\big\}\cdot\Big(\mathrm{avg}^*_{a,i-1}+\mathrm{avg}^*_{b,i-1}\Big), \tag{9}$$

where

$$\mathrm{avg}^*_{a,i} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim C^*_{a,i}}\big[\|a-\mathbf{c}\|_1\big] \quad\text{and}\quad \mathrm{avg}^*_{b,i} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim C^*_{b,i}}\big[\|b-\mathbf{c}\|_1\big]$$

and $C^*_{a,i}$ contains all points in $C_{\mathrm{v}_i(a)}$ that is not too far away from a:

$$C^*_{a,i} \stackrel{\text{def}}{=} \left\{c\in C_{\mathrm{v}_i(a)} : \|a-c\|_1 \le \frac{10d\log n}{2^i}\right\}.$$

The set $C^*_{b,i}$ is defined similarly. Roughly speaking, points in C that share the same node at depth i are expected to have distance around $d/2^i$ (given they have agreed on 2^i-1 random coordinates sampled so far); this is why we refer to points in $C^*_{a,i}$ as those that are not too far away from a.

The following is the crucial lemma for upperbounding the total expected payment according to an optimal matching M^*. Its proof can be found in the full version [15]. We use it to prove Lemma 3.3.

Lemma 3.6. *For any (a,b) with $a,b\in C$, $a\ne b$ and $i_0\in[0:h-1]$ that satisfies*

$$i_0 \le h_{a,b} \stackrel{\text{def}}{=} \left\lfloor \log_2\left(\frac{d}{\|a-b\|_1}\right)\right\rfloor, \tag{10}$$

we have

$$\mathop{\mathbf{E}}_{\mathbf{T}\sim\mathcal{T}}\left[\textbf{\textit{Pay}}^*_{i_0,\mathbf{T}}(a,b)\right] \le \left(\tilde{O}(\log n) + O(\log\log n)\left(h_{a,b} - i_0\right)\right)\cdot \|a-b\|_1.$$

Proof of Lemma 3.3 assuming Lemma 3.6: Let M^* be an optimal matching between A and B that achieves $\mathsf{EMD}(A,B)$. Let $\mathbf{T}\sim\mathcal{T}$. Then we have from Lemma 3.5 that

$$\mathbf{Value}_{\mathbf{T}}(A,B) \le 2\sum_{\substack{(a,b)\in M^*\\ a\neq b}} \mathbf{Pay}_{\mathbf{T}}(a,b) \tag{11}$$

given that $\mathbf{Pay}_{\mathbf{T}}(a,b) = 0$ when $a=b$. Below we focus on the subset M' of M^* with $(a,b)\in M^*$ and $a\neq b$. For each $(a,b)\in M'$, let

$$0 \le \ell_{a,b} \stackrel{\text{def}}{=} \max\left\{0, h_{a,b} - 2\lceil\log_2 n\rceil\right\} \le h_{a,b}.$$

We show that with probability at least $1-o(1)$ over the draw of $\mathbf{T}$, every $(a,b)\in M'$ satisfies

$$\mathbf{Pay}_{\mathbf{T}}(a,b) = \mathbf{Pay}^*_{\ell_{a,b},\mathbf{T}}(a,b). \tag{12}$$

Combining (11) and (12), we have that with probability at least $1-o(1)$ over the draw of $\mathbf{T}$,

$$\mathbf{Value}_{\mathbf{T}}(A,B) \le 2\sum_{(a,b)\in M'} \mathbf{Pay}^*_{\ell_{a,b},\mathbf{T}}(a,b). \tag{13}$$

By applying Lemma 3.6 to every $(a,b)\in M'$ with $i_0 = \ell_{a,b}$, as well as Markov's inequality, we have that with probability at least 0.99 over $\mathbf{T}$, the right hand side of (13) is at most

$$\tilde{O}(\log n)\sum_{(a,b)\in M'}\|a-b\|_1 = \tilde{O}(\log n)\cdot\mathsf{EMD}(A,B).$$

By a union bound, $\mathbf{Value}_{\mathbf{T}}(A,B) \le \tilde{O}(\log n)\cdot\mathsf{EMD}(A,B)$ with probability at least $.99 - o(1)\ge 0.9$.

It suffices to define an event that implies (12) and then bound its probability. The first part of the event requires that for every pair $(a,b)\in M'$, $\mathsf{v}_i(a) = \mathsf{v}_i(b)$ for every $i: 1\le i\le \ell_{a,b}$. The second part requires that for any two distinct points $x,y\in A\cup B$ (not necessarily as a pair in M^* and not even necessarily in the same set), we have $\mathsf{v}_i(x)\neq\mathsf{v}_i(y)$ for all i with

$$2^i \ge \frac{10d\log n}{\|x-y\|_1}. \tag{14}$$

By the definition of $\mathbf{Pay}^*_{\ell_{a,b},\mathbf{T}}(a,b)$ in (9), the first part of the event makes sure that we don't miss any term in the sum; the second part of the event makes sure that every $C^*_{a,i}$ is exactly the same as $C_{\mathsf{v}_i(a)}$ so that $\mathrm{avg}^*_{a,i} = \mathrm{avg}_{a,i}$ (and the same holds for b) . It follows that this event implies (12).

Finally we show that the event occurs with probability at least $1-o(1)$. First, for every $(a,b)\in M'$, if $\ell_{a,b}=0$ then the first part of the event trivially holds. If $\ell_{a,b}>0$ then $\ell_{a,b} = h_{a,b} - 2\lceil\log n\rceil$. The probability of $\mathsf{v}_i(a)\neq\mathsf{v}_i(b)$ for some $i: 1\le i\le\ell_{a,b}$ is at most

$$1-\left(1-\frac{\|a-b\|_1}{d}\right)^{2^{\ell_{a,b}-1}} \le 2^{\ell_{a,b}}\cdot\frac{\|a-b\|_1}{d} \le \frac{1}{n^2}.$$

Hence, by a union bound over the at most n pairs $(a,b)\in M'$, the first part of the event holds with probability at least $1-o(1)$. Furthermore, for any two distinct points $x,y\in A\cup B$, let

$$\ell^* = \left\lfloor \log_2\left(\frac{10d\log n}{\|x-y\|_1}\right)\right\rfloor.$$

Then $\mathsf{v}_i(x) = \mathsf{v}_i(y)$ for some i that satisfies (14) would imply $\mathsf{v}_{\ell^*}(x) = \mathsf{v}_{\ell^*}(y)$ and $\ell^*\le\log d$ (since $\mathsf{v}_h(x)\neq\mathsf{v}_h(y)$ given $x\neq y$). The event above happens with probability

$$\left(1-\frac{\|x-y\|_1}{d}\right)^{2^{\ell^*-1}} \le \exp(-5\log n) = \frac{1}{n^5}.$$

Via a union bound over at most $(2n)^2$ many pairs of x,y, we have that the second part of the event also happens with probability at least $1-o(1)$. This finishes the proof of the lemma. ∎

3.2 Approximation of MST using Quadtrees

We will follow a similar strategy as we took in the previous subsection for EMD. Given a quadtree T of depth $h=\log_2 2d$, we define similarly $\mathsf{v}_0(x),\ldots,\mathsf{v}_h(x)$ as the root-to-leaf path of $x\in\{0,1\}^d$, and write S_v for each node v at depth i to denote the set of $x\in\{0,1\}^d$ with $\mathsf{v}_i(x)=v$.

Let $X\subseteq\{0,1\}^d$ be a set of n points. We define X_v for each node v in T as $X\cap S_v$, and write L_i for each depth i to denote the set of nodes v at depth i such that $X_v\neq\emptyset$ and will refer to them as nonempty nodes.

We give the definition of depth-greedy spanning trees.

Definition 3.7. Let T be a quadtree, and $X\subset\{0,1\}^d$. For any DFS walk of the quadtree T starting at the root, let $\sigma: [n]\to X$ denote the order of points in X encountered during the walk, so that $\mathsf{v}_h(\sigma(i))$ appears before $\mathsf{v}_h(\sigma(i+1))$ for every $i\in[n-1]$. A depth-greedy spanning tree G obtained from a DFS walk is given by the edges $\{(\sigma(i),\sigma(i+1))\}_{i\in[n-1]}$. The class of *depth-greedy spanning trees*, denoted by $\mathcal{G}_T(X)$, is the set of all spanning trees G of X obtained from a DFS walks down the quadtree T. For any spanning tree G, we write

$$\mathbf{Cost}(G) = \sum_{(a,b)\in E(G)}\|a-b\|_1$$

to denote the cost of a tree G (with $n-1$ edges) spanning points in X. Recall $\mathsf{MST}(X)$ is defined as the minimum of $\mathbf{Cost}(G)$ over all spanning trees G of X.

Similar to the previous subsection, for each edge $(u,v)\in E_T$, we write

$$\mathrm{avg}_{u,v} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\substack{\mathbf{c}\sim X_u\\ \mathbf{c}'\sim X_v}}\left[\|\mathbf{c}-\mathbf{c}'\|_1\right].$$

when $X_v\neq\emptyset$, and $\mathrm{avg}_{u,v}=0$ when $X_v=\emptyset$. Recall $\pi(v)$ denotes the parent node of v in T. We are now ready to define the value of X in a quadtree T and then state the main lemma:

Definition 3.8. Let T be a quadtree. The value of X in T is defined as

$$\textbf{\textit{Value}}_T(X) \stackrel{\text{def}}{=} \sum_{i\in[h]}\mathbf{1}\{|L_i|>1\}\cdot\sum_{v\in L_i} avg_{\pi(v),v}.$$

The main lemma for MST shows that the value of X for a random quadtree $\mathbf{T}\sim\mathcal{T}$ and the cost of any depth-greedy spanning tree $G\in\mathcal{G}_T(X)$ are $\tilde{O}(\log n)$-approximations of $\mathsf{MST}(X)$.

Lemma 3.9 (Quadtree lemma for MST). *Let $X \subseteq \{0,1\}^d$ be a set of size n, and let $\mathbf{T} \sim \mathcal{T}$. Then with probability at least 0.99, for any $G \in \mathcal{G}_{\mathbf{T}}(X)$, we have that*

$$\frac{\mathsf{MST}(X)}{2} \le \frac{\textbf{\textit{Cost}}(G)}{2} \le \mathbf{Value}_{\mathbf{T}}(X) \le \tilde{O}(\log n) \cdot \mathsf{MST}(X).$$

We start with the lower bound:

Lemma 3.10. *Let T be any quadtree and any depth-greedy spanning tree $G \in \mathcal{G}_T(X)$. Then $\mathbf{Value}_T(X) \ge \textbf{\textit{Cost}}(G)/2 \ge \mathsf{MST}(X)/2$.*

Proof: Let w be the least common ancestor of leaves $\mathsf{v}_h(x)$, $x \in X$, and let T^* denote the subtree rooted at w that consists of paths from w to $\mathsf{v}_h(x)$, $x \in X$. Using T^* we can equivalently write

$$\mathbf{Value}_T(X) = \sum_{(u,v)\in E_{T^*}} \mathrm{avg}_{u,v}.$$

For each node $v \in T^*$ (note that $X_v \neq \emptyset$), we define ρ_v to be the center-of-mass of points in X_v:

$$\rho_v \stackrel{\text{def}}{=} \frac{1}{|X_v|} \sum_{x\in X_v} x.$$

By triangle inequality we have $\|\rho_u - \rho_v\|_1 \le \mathrm{avg}_{u,v}$ for every $(u,v) \in E_{T^*}$ and thus,

$$\sum_{(u,v)\in E_{T^*}} \|\rho_u - \rho_v\|_1 \le \mathbf{Value}_T(X).$$

We finish the proof by showing that any depth-greedy spanning tree G of X satisfies

$$\textbf{Cost}(G) \le 2 \sum_{(u,v)\in E_{T^*}} \|\rho_u - \rho_v\|_1.$$

To this end we take a DFS walk of T^* from its root w and let $\sigma\colon [n] \to X$ be the order of points in X under which $\mathsf{v}_h(\sigma(1)), \ldots, \mathsf{v}_h(\sigma(n))$ appear in the walk. Then we set G to be the spanning tree $\{(\sigma(i), \sigma(i+1))\}_{i\in[n-1]}$. For each $i \in [n-1]$, letting $u_1, \ldots, u_r$ be the part of DFS walk from $u_1 = \mathsf{v}_h(\sigma(i))$ to $u_r = \mathsf{v}_h(\sigma(i+1))$, we have from triangle inequality that

$$\|\sigma(i)-\sigma(i+1)\|_1 = \|\rho_{u_1}-\rho_{u_r}\|_1 \le \|\rho_{u_1}-\rho_{u_2}\|_1 + \cdots + \|\rho_{u_{r-1}}-\rho_{u_r}\|_1.$$

The lemma follows from the fact that a DFS walk visits each edge twice. ■

Now it suffices to upper bound $\mathbf{Value}_{\mathbf{T}}(X)$ by $\tilde{O}(\log n) \cdot \mathsf{MST}(X)$ with probability at least 0.9 for a random quadtree $\mathbf{T} \sim \mathcal{T}$. For this purpose, we use the same inspector payment defined in the last subsection (the only change is that the set C is now called X which is a set and has size n instead of $2n$). Recall that for any two points $x, y \in X$, we define

$$\textbf{Pay}_T(x,y) \stackrel{\text{def}}{=} \sum_{i\in[h]} \mathbf{1}\{\mathsf{v}_i(x) \neq \mathsf{v}_i(y)\} \cdot \Big(\mathrm{avg}_{x,i-1} + \mathrm{avg}_{y,i-1}\Big), \tag{15}$$

where

$$\mathrm{avg}_{x,i-1} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim X_{\mathsf{v}_{i-1}(x)}} \big[\|x - \mathbf{c}\|_1\big] \text{ and}$$

$$\mathrm{avg}_{y,i-1} \stackrel{\text{def}}{=} \mathop{\mathbf{E}}_{\mathbf{c}\sim X_{\mathsf{v}_{i-1}(y)}} \big[\|y - \mathbf{c}\|_1\big].$$

Next we show that the total payment from any spanning tree G is enough to cover $\mathbf{Value}_T(X)$.

Lemma 3.11. *Let T be any quadtree and G be any spanning tree of X. Then we have*

$$\mathbf{Value}_T(X) \le 2 \sum_{(x,y)\in E(G)} \textbf{\textit{Pay}}_T(x,y). \tag{16}$$

Proof: Let w be the least common ancestor of leaves $\mathsf{v}_h(x)$, $x \in X$, and let T^* denote the subtree rooted at w that consists of paths from w to $\mathsf{v}_h(x)$, $x \in X$. It suffices to show that

$$\sum_{(u,v)\in E_{T^*}} \mathrm{avg}_{u,v} \le 2 \sum_{(x,y)\in E(G)} \textbf{Pay}_T(x,y).$$

By similar arguments in the proof of Lemma 3.5, $2 \cdot \textbf{Pay}_T(x,y)$ is good enough to cover $\mathrm{avg}_{u,v}$ for every edge along the path between the leaf of x and the leaf of y. The lemma follows by summing over all $(x,y) \in E(G)$ and noting that the $\mathrm{avg}_{u,v}$ of each $(u,v) \in E_{T^*}$ is counted at least once. ■

To upperbound the total inspector payment from an optimal spanning tree by $\tilde{O}(\log n) \cdot \mathsf{MST}(X)$, we similarly consider the modified payment scheme $\textbf{Pay}^*_{i_0,T}(x,y)$ as in (9), replacing C by X. The same Lemma 3.6 applies and we use it to prove Lemma 3.9:

Proof of Lemma 3.9 assuming Lemma 3.6: The lower bound follows from Lemma 3.10. For the upper bound, let G^* be an optimal spanning tree of X and let $\mathbf{T} \sim \mathcal{T}$. By Lemma 3.11 we have

$$\mathbf{Value}_{\mathbf{T}}(X) \le 2 \sum_{(x,y)\in E'(G^*)} \textbf{Pay}_{\mathbf{T}}(x,y), \tag{17}$$

where $E'(G^*)$ denotes the set of edges (x,y) in G^* with $x \neq y$. For each $(x,y) \in E'(G^*)$, let

$$0 \le \ell_{x,y} \stackrel{\text{def}}{=} \max\left\{0, h_{x,y} - 2\lceil \log_2 n\rceil\right\} \le h_{x,y}.$$

By similar arguments as in the proof of Lemma 3.3, we have with probability at least $1 - o(1)$ over the draw of $\mathbf{T}$ that every $(x,y) \in E'(G^*)$ satisfies

$$\textbf{Pay}_{\mathbf{T}}(x,y) = \textbf{Pay}^*_{\ell_{x,y},\mathbf{T}}(x,y) \tag{18}$$

Combining (17) and (18), we have that with probability at least $1 - o(1)$ over the draw of $\mathbf{T}$,

$$\mathbf{Value}_{\mathbf{T}}(X) \le 2 \sum_{(x,y)\in E'(G^*)} \textbf{Pay}^*_{\ell_{x,y},\mathbf{T}}(x,y). \tag{19}$$

By applying Lemma 3.6 to every $(x,y) \in E'(G^*)$ with $i_0 = \ell_{x,y}$, as well as Markov's inequality, we have that with probability at least 0.99 over $\mathbf{T}$, the right hand side of (19) is at most

$$\tilde{O}(\log n) \sum_{(x,y)\in E'(G^*)} \|x - y\|_1 = \tilde{O}(\log n) \cdot \mathsf{MST}(X).$$

By a union bound, $\mathbf{Value}_{\mathbf{T}}(X) \le \tilde{O}(\log n) \cdot \mathsf{MST}(X)$ with probability at least $.99 - o(1) \ge 0.9$. ■

ACKNOWLEDGMENTS

We would like to thank David Woodruff, Ilya Razenshteyn, and Aleksandar Nikolov for illuminating discussions relating to this work.

REFERENCES

[1] Pankaj K. Agarwal and R. Sharathkumar. 2014. Approximation algorithms for bipartite matching with metric and geometric costs. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC '2014)*. 555–564.

[2] Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. 2009. Efficient sketches for earth-mover distance, with applications. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*.

[3] Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. 2008. Earth mover distance over high-dimensional spaces. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '2008)*. 343–352.

[4] Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. 2010. Streaming algorithms from precision sampling. *arXiv preprint arXiv:1011.1263* (2010).

[5] Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. 2015. Sketching and Embedding are Equivalent for Norms. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*. 479–488. Available as arXiv:1411.2577.

[6] Alexandr Andoni, Aleksandar Nikolov, Krzysztof Onak, and Grigory Yaroslavtsev. 2014. Parallel algorithms for geometric graph problems. In *Proceedings of the 46th ACM Symposium on the Theory of Computing (STOC '2014)*.

[7] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 214–223.

[8] Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. 2020. Scalable nearest neighbor search for optimal transport. In *Proceedings of the 37th International Conference on Machine Learning (ICLM '2020)*.

[9] Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, and D. Sivakumar. 2004. An information statistics approach to data stream and communication complexity. *J. Comput. System Sci.* 68, 4 (2004), 702–732.

[10] MohammadHossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, Mohammad-Taghi Hajiaghayi, Raimondas Kiveris, Solvio Lattanzi, and Vahab Mirrokni. 2017. Affinity Clustering: Hierarchical Clustering at Scale. In *Proceedings of Advances in Neural Information Processing Systems 30 (NeurIPS '2017)*.

[11] Arturs Bačkurs and Piotr Indyk. 2014. Better embeddings for planar earth-mover distance over sparse sets. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP '2014)*.

[12] Nicolas Bonneel, Michiel Van De Panne, Sylvain Paris, and Wolfgang Heidrich. 2011. Displacement interpolation using Lagrangian mass transport. In *Proceedings of the 2011 SIGGRAPH Asia Conference*. 1–12.

[13] Moses Charikar. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*. 380–388.

[14] Moses Charikar, Kevin Chen, and Martin Farach-Colton. 2002. Finding frequent items in data streams. *Automata, languages and programming* (2002), 784–784.

[15] Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. 2021. New Streaming Algorithms for High Dimensional EMD and MST. *arXiv: 2111.03528* (2021).

[16] Rémi Flamary, Marco Cuturi, Nicolas Courty, and Alain Rakotomamonjy. 2018. Wasserstein discriminant analysis. *Machine Learning* 107, 12 (2018), 1923–1945.

[17] Gereon Frahling, Piotr Indyk, and Christian Sohler. 2005. Sampling in Dynamic Data Streams and Applications. In *Proceedings of the 21st ACM Symposium on Computational Geometry (SoCG '2005)*.

[18] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. 2012. Approximate Nearest Neighbor: Towards Removing the Curse of Dimensionality. *Theory of Computing* 8, 1 (2012), 321–350.

[19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *Elements of statistical learning: data mining, inference, and prediction*. Springer.

[20] Piotr Indyk. 2004. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC '2004)*.

[21] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM (JACM)* 53, 3 (2006), 307–323.

[22] Piotr Indyk. 2006. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM* 53, 3 (2006), 307–323.

[23] Piotr Indyk and Nitin Thaper. 2003. Fast Color Image Retrieval via Embeddings. In *Workshop on Statistical and Computational Theories of Vision (at ICCV)*.

[24] Rajesh Jayaram and David Woodruff. 2021. Perfect L_p Sampling in a Data Stream. *SIAM J. Comput.* 50, 2 (2021), 382–439.

[25] Hossein Jowhari, Mert Sağlam, and Gábor Tardos. 2011. Tight Bounds for Lp Samplers, Finding Duplicates in Streams, and Related Problems. In *Proceedings of the Thirtieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Athens, Greece) *(PODS '11)*. ACM, New York, NY, USA, 49–58. https://doi.org/10.1145/1989284.1989289

[26] Andrey Boris Khesin, Aleksandar Nikolov, and Dmitry Paramonov. 2019. Preconditioning for the geometric transportation problem. In *Proceedings of the 35th International Symposium on Computational Geometry (SoCG '2019)*.

[27] Subhash Khot and Assaf Naor. 2006. Nonembeddability theorems via Fourier analysis. *Math. Ann.* 334, 4 (2006), 821–852.

[28] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML '2015)*.

[29] Andrew McGregor and Daniel Stubbs. 2013. Sketching earth-mover distance on graph metrics. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. Springer, 274–286.

[30] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of Advances in Neural Information Processing Systems (NIPS '2013)*. 3111–3119.

[31] Morteza Monemizadeh and David P Woodruff. 2010. 1-pass relative-error lp-sampling with applications. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*. SIAM, 1143–1160.

[32] Jonas W Mueller and Tommi Jaakkola. 2015. Principal differences analysis: Interpretable characterization of differences between distributions. In *Advances in Neural Information Processing Systems*. 1702–1710.

[33] Assaf Naor and Gideon Schechtman. 2007. Planar Earthmover is not in L_1. *SIAM J. Comput.* 37, 3 (2007), 804–826. An extended abstract appeared in FOCS'06..

[34] Saul B Needleman and Christian D Wunsch. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology* 48, 3 (1970), 443–453.

[35] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '2014)*. 1532–1543.

[36] Gabriel Peyré and Marco Cuturi. 2019. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning* 11, 5–6 (2019), 355–607.

[37] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 2 (2000), 99–121.

[38] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. 2000. The earth mover's distance as a metric for image retrieval. *International journal of computer vision* 40, 2 (2000), 99–121.

[39] R. Sharathkumar and Pankaj K. Agarwal. 2012. A near-linear time ϵ-approximation algorithm for bipartite geometric matching. In *Proceedings of the 42nd ACM Symposium on the Theory of Computing (STOC '2012)*.

[40] Jonah Sherman. 2017. Generalized preconditioning and undirected minimum cost flow. In *Proceedings of the 28th ACM-SIAM Symposium on Discrete Algorithms (SODA '2017)*.

[41] Justin Solomon, Fernando De Goes, Gabriel Peyré, Marco Cuturi, Adrian Butscher, Andy Nguyen, Tao Du, and Leonidas Guibas. 2015. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–11.

[42] Grigory Yaroslavtsev and Adithya Vadapalli. 2018. Massively Parallel Algorithms and Hardness for Single-Linkage Clustering Under ℓ_p-Distances. In *Proceedings of the 35th International Conference on Machine Learning (ICML '2018)*.

[43] Arman Yousefi and Rafail Ostrovsky. 2014. Improved Approximation Algorithms for Earth-Mover Distance in Data Streams. *arXiv preprint arXiv:1404.6287* (2014).