Approximation Algorithms for 1-Wasserstein Distance Between Persistence Diagrams

Samantha Chen
University of California at San Diego, La Jolla, CA, USA

Yusu Wang
University of California at San Diego, La Jolla, CA, USA

Abstract

Recent years have witnessed a tremendous growth using topological summaries, especially the persistence diagrams (encoding the so-called persistent homology) for analyzing complex shapes. Intuitively, persistent homology maps a potentially complex input object (be it a graph, an image, or a point set and so on) to a unified type of feature summary, called the persistence diagrams. One can then carry out downstream data analysis tasks using such persistence diagram representations. A key problem is to compute the distance between two persistence diagrams efficiently. In particular, a persistence diagram is essentially a multiset of points in the plane, and one popular distance is the so-called 1-Wasserstein distance between persistence diagrams. In this paper, we present two algorithms to approximate the 1-Wasserstein distance for persistence diagrams in near-linear time. These algorithms primarily follow the same ideas as two existing algorithms to approximate optimal transport between two finite point-sets in Euclidean spaces via randomly shifted quadtrees. We show how these algorithms can be effectively adapted for the case of persistence diagrams. Our algorithms are much more efficient than previous exact and approximate algorithms, both in theory and in practice, and we demonstrate its efficiency via extensive experiments. They are conceptually simple and easy to implement, and the code is publicly available in github.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases persistence diagrams, approximation algorithms, Wasserstein distance, optimal transport

Digital Object Identifier 10.4230/LIPIcs.SEA.2021.14

Supplementary Material *Software (Source Code)*: `https://github.com/chens5/w1estimators.git` archived at `swh:1:dir:03da011d5be7f5de530383a67da81499fb8b195c`

Funding This work is partially supported by National Science Foundation (NSF) via grants OAC-2039794 and IIS- 2050360.

Acknowledgements We want to thank Chen Cai for providing the reddit-binary and ModelNet10 datasets used in the experiments.

1 Introduction

Recent years have witnessed a tremendous growth using topological summaries, especially the persistence diagrams (encoding the so-called persistent homology) for analyzing complex shapes. Indeed, persistent homology is one of the most important development in the field of topological data analysis in the past two decades [11, 10]. Given an object, e.g, a mesh, an image, a point cloud, or a graph, by taking a specific view of how the object evolves (more formally, a filtration of it), persistent homology maps the input, a potentially complex object, to a topological summary, called the persistence diagram, which captures multiscale features of this objects w.r.t. this view. Persistent homology thus provides a unifying way of mapping complex objects to a common feature space: the space of persistence diagrams. One

can then carry out data analysis tasks of the original objects, e.g, clustering or classifying a collection of graphs, in this feature space. Indeed, in the past decade, persistence diagram summaries have been used for a range of applications in various domains, e.g, in material science [5, 14, 23], neuroanatomy [17, 24], graphics [8, 29], medicine /biology [13, 27], etc.

A key component involved in such a persistent-homology based data analysis framework is to put a suitable metric on the space of persistence diagrams, and compute such distances efficiently. One classic distance measure developed for persistence diagrams is the p-th Wasserstein distance, and in practice, a popular choice for p is $p = 1$, i.e, the 1-Wasserstein distance. This paper focuses on developing efficient, practical and light-weight algorithms to approximate the 1-Wasserstein distance for persistence diagrams.

In particular, a persistence diagram consists of a multiset of points in the plane, where each point (b, d) corresponds to the creation and death of some topological feature w.r.t. some specific filtration (view) of the input object. Given two persistence diagrams P and Q, the 1-Wasserstein distance between them, denoted by $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P}, \mathsf{Q})$, is similar to the standard 1-Wasserstein distance (also known as the earth-mover distance) between these two multisets of planar points, but with an important distinction where points are also allowed to be matched to points in the diagonal $\mathcal{L}$ (the line defined by equation $y = x$) in the plane. Intuitively, the topological features associated to points matched to the diagonal are considered as noise.

The 1-Wasserstein distance for persistence diagrams can be computed using the Hungarian algorithm [20] in $O(n^3)$ time where n is the total number of points in the two persistence diagrams. This algorithm is implemented in the widely used Dionysus package [26]. In [18], Kerber et al. develops a more efficient algorithm to approximate the 1-Wasserstein distance between finite persistence diagrams within constant factors. Their algorithm is based on the auction algorithm of Bertsekas [4], but with a geometric twist: given that points in the persistence diagrams are all in the plane, they use the weighted kd-tree to provide more efficient search inside the auction algorithm. In [4], the time complexity of the auction algorithm is stated to be $O(A \cdot n^{1/2} \log(nC))$ where, in the case of persistence diagrams, A is the number of possible pairings of points between persistence diagrams ($A = \Theta(n^2)$ in the worst case) and C is $\max\{||x - y||_q\}$ over all possible pairings between persistence diagrams. While Kerber et al. did not provide an asymptotic time complexity for their approximation algorithm, they provided an empirical estimation of $O(n^{1.6})$ (not true asymptotic time complexity) by using linear regression on the observed running time versus the size of problems. They further show via extensive experiments that their approximation algorithm has a speed-up factor of 50 for small instances to a speed-up factor of 400 for larger instances in comparison to the Hungarian algorithm based implementation.

Related work in optimal transport for Euclidean point sets. As we will formally introduce in Section 2), 1-Wasserstein distance for persistence diagrams can be viewed as the standard 1-Wasserstein distance for discrete planar point sets with special inclusion of points in the diagonal. In what follows, to avoid confusion, we refer to the standard 1-Wasserstein distance between point sets as the *optimal transport (OT)* distance. Starting from [2] and [3], there has been a long line of work to approximate the OT-distance for Euclidean point sets using randomly shifted quadtrees (e.g, [7, 19, 15, 22, 28, 1]). In particular, we consider two such approaches, the L_1-*embedding approach* by [15], and the *flowtree approach* by [1]. The former maps an input point set P to a certain count-vector V^P with the help of a randomly shifted quadtree, and uses the L_1 distance $\|V^P - V^Q\|_1$ between two such count-vectors to approximate the OT-distance between P and Q. The latter also uses a randomly shifted quadtree and embeds input points to quadtree cells. It then shows that a certain distance

computed from an optimal OT-flow induced by the tree metric (which can be computed by a greedy algorithm in linear time) can approximate the OT-distance between the original point sets. Let Δ denote the spread of the union of two input point sets. Both approaches give an $O(\log \Delta)$-approximation of the OT-distance between original point sets, in time $O(n \log \Delta)$.

Recently, the idea of using metric trees to approximate OT-distance has also been extended to a more general unbalanced optimal transport problem (where $|P| \neq |Q|$) in [28]. In [28], Sato et al. develops an $O(n \log^2 n)$ time algorithm to approximate unbalanced optimal transport on tree metrics using dynamic programming.

New work. In practice, for applications such as nearest neighbor search, clustering and classification on large data sets, huge numbers of distance computations will be needed. The time complexity of the aforementioned algorithms for persistence diagrams using the Hungarian algorithm or the geometric variant of the Auction algorithm still causes a significant computational burden. In this paper, we aim to develop *near-linear time* approximation algorithms for the 1-Wasserstein distance between persistence diagrams. Specifically:

- In Section 3, we show how to modify the algorithms of [15] and [1] to approximate the 1-Wasserstein distances between persistence diagrams within the same approximation factor (Theorems 7 and 10). Note that in the literature (e.g, [18]), it is known that $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q})$ between two persistence diagrams can be computed by (i) first augmenting P and Q to be $\widehat{\mathsf{P}} = \mathsf{P} \cup \pi(\mathsf{Q})$ and $\widehat{\mathsf{Q}} = \mathsf{Q} \cup \pi(\mathsf{P})$, respectively, where $\pi(x)$ projects a point x to its nearest neighbor in the diagonal $\mathcal{L}$; and then (ii) compute the OT-distance between $\widehat{\mathsf{P}}$ and $\widehat{\mathsf{Q}}$, although it is important to note that the cost of matching two diagonal points needs to be set to be 0, instead of the standard Euclidean distance. However, this requires the modification of the cost for diagonal points; in addition, this also needs to modify a diagram P depending on which other diagram Q it is to be compared with. We instead develop a modification where such projection is not needed.
- Our modified approaches maintain the simplicity of the original approximation algorithms and are easy to implement. In comparison to approximation for unbalanced optimal transport presented in [28], our modified approaches are specific to persistence diagrams and the data structures needed for both of our approaches are much simpler than those of [28]. Our code is publicly available in github. In Section 4, we present various experimental results of our new algorithms. We show that both are orders of magnitude faster than previous approaches, although at the price of worsened approximation error. However, note that in practice, the approximate factors are rather small, not as large as the worst case approximation factor. We also note that the modified flowtree algorithm achieves a more accurate approximation of the 1-Wasserstein distance for persistence diagrams than the modified L_1-embedding approach empirically, although the latter is significantly faster than the former. However, the L_1-embedding approach is easier to combine with proximity search data structures e.g, locality sensitive hashing (LSH), given that each input persistence diagram is mapped to a vector and the distance computation is the L_1-distance between two such vectors.

2 Preliminaries

In this section, we first introduce the persistence diagrams and the 1-Wassertein distance between them, which is related to the optimal transport distance (standard 1-Wasserstein distance) for Euclidean point sets. We next describe two existing approximation algorithms for optimal transport distance [1, 15] based on the use of randomly shifted quadtrees. Our new algorithms (in section 3) will be based on these two approximation algorithms.

2.1 Persistence Diagrams and 1-Wasserstein distance

We first give a brief introduction of persistent homology and its associated persistence diagram summary. See [10] for a more detailed treatment of these topics. Suppose we are given a topological space X. A filtration of X is a growing sequence of sub-spaces

$$\mathbb{F}: \quad \emptyset = \mathrm{X}_0 \subseteq \mathrm{X}_1 \subseteq \mathrm{X}_2 \subseteq \cdots \mathrm{X}_m = \mathrm{X}$$

which can be viewed as a specific way to inspect X. For example, a popular way to generate a filtration of X is by taking some meaningful descriptor function $f : \mathrm{X} \to \mathbb{R}$ on X, and take the growing sequence of sub-level sets $\mathrm{X}_a := f^{-1}(-\infty, a] = \{x \in \mathrm{X} \mid f(x) \leq a\}$ as a increases to be the filtration. Now given a filtration $\mathbb{F}$, through its course, new topological features (e.g, components, independent loops and voids, which are captured by the so-called homology classes) will sometimes appear and sometimes disappear. The persistent homology encodes the *birth* and *death* of such features in the *persistence diagram* $\mathsf{dgm}\mathbb{F}$. In particular, $\mathsf{dgm}\mathbb{F}$ consists of a *multiset* of points in the plane, that is, a set of points with multiplicities, where each point (b, d) with multiplicity m intuitively means that m independent topological features (homology classes) are created in X_b and killed in X_d. Thus, we also refer to b and d as the *birth-time* and *death-time*. The *persistence* of this feature is $|d - b|$ which is the lifetime of this feature. We refer to points in the persistence diagram as *persistent-points*.

Note that, in general, persistent-points lie above the diagonal $\mathcal{L} = \{(x, x) \mid x \in \mathbb{R}\}$ in the plane. Points closer to the diagonal $\mathcal{L}$ have lower lifetime (persistence) and thus are less important, with a point $(x, x) \in \mathcal{L}$ intuitively meaning a feature with persistence 0.

To compare two persistence diagrams P and Q, intuitively, we wish to find a one-to-one correspondence between their multiset of points (and thus between the features they capture). However, the two sets may be of different cardinality, and we also wish to allow a persistent-point from one diagram to be "noise" and not present in the other diagram, which can be captured by allowing this point $p = (p.x, p.y)$ to be matched to its nearest neighbor projection $\pi(p)$ in $\mathcal{L}$. Let $\pi : \mathbb{R}^2 \to \mathcal{L}$ be this projection, where $\pi(p) := (\frac{p.x+p.y}{2}, \frac{p.x+p.y}{2})$. The following p-th Wasserstein distance essentially captures this intuition [10].

▶ **Definition 1** (p-Wasserstein distance for persistence diagrams)**.** *Given a persistence diagram* P*, its* augmentation $\mathsf{aug}(\mathsf{P})$ *consists of* P *together with all points in* $\mathcal{L}$ *each with infinite multiplicity. Given two persistence diagrams* P *and* Q*, with their augmentations* $\mathsf{aug}(\mathsf{P})$ *and* $\mathsf{aug}(\mathsf{Q})$*, respectively, the p-Wassertein distance between them is*

$$\mathrm{d}^{\mathrm{per}}_{\mathrm{W},\mathrm{p}}(\mathsf{P}, \mathsf{Q}) := \inf_{\mu:\mathsf{aug}(\mathsf{P})\to\mathsf{aug}(\mathsf{Q})} \left(\sum_{p\in\mathsf{aug}(\mathsf{P})} \|p - \mu(p)\|_q^p \right)^{1/p}, \tag{1}$$

where $\mu : \mathsf{aug}(\mathsf{P}) \to \mathsf{aug}(\mathsf{Q})$ *ranges over all possible bijections among the two sets.*

Note that q is used to denote the inner L_p-norm. If $p = \infty$, the ∞-Wasserstein distance is the classic *bottleneck distance* between persistence diagrams [9][18]. In this paper, we are interested in the case when $p = 1$. It turns out that an equivalent definition (which we will use in this paper) is as follows:

▶ **Definition 2** (1-Wasserstein distance for persistence diagrams, version 2)**.** *Given two point sets* A *and* B *in* $\mathbb{R}^2$*, an* augmented (perfect) matching *for them is a subset* $\Gamma \subset \big(A \cup \pi(B)\big) \times \big(B \cup \pi(A)\big)$ *such that (i) each* $a \in A$ *or* $b \in B$ *appears in exactly one pair in* Γ*, and (ii) each* $(a, b) \in \Gamma$ *is of the following three forms: (1)* $a \in A, b \in B$*, (2)* $a \in A, b = \pi(a) \in \pi(A)$*, or (3)* $a = \pi(b) \in \pi(B), b \in B$*.*

Given two persistence diagrams P *and* Q*, the 1-Wasserstein distance between them is:*

$$\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P},\mathsf{Q}) := \min_{\Gamma} \sum_{(p,q)\in\Gamma} \|p-q\|_p, \tag{2}$$

where Γ *ranges over all possible augmented matchings for* P *and* Q*.*

2.2 Relation to optimal transport

Readers may have already noticed the similarity between Definition 1 with the standard p-th Wasserstein distance between two probability measures. To avoid confusion, from now on we refer to 1-Wassertein distance as *optimal transport* so as to differentiate from the use of 1-Wasserstein distance of persistence diagrams.

▶ **Definition 3** (Optimal transport)**.** *Given a finite metric space* (X, d_X) *and two measures* $\mu, \nu \in X \to \mathbb{R}$*, the* optimal transport *between them is*

$$\mathrm{d}_{\mathrm{OT}}(\mu,\nu) := \min_{\tau: X\times X\to\mathbb{R}} \sum_{x,y\in X} \tau(x,y)\cdot d_X(x,y), \tag{3}$$

where τ*, called a* transport plan *or* a flow*, is a measure on* $X \times X$ *whose marginals equal to* μ *and* ν*, respectively; that is,* $\tau(\cdot, Y) = \mu(\cdot)$ *and* $\tau(X, \cdot) = \nu(\cdot)$*.*

Given a multiset of points A in the plane, note that we can view this as a discrete measure supported on points in A, such that for each subset S of A, $\mu_A(S) = \sum_{a\in S} c_a\delta_a$ where c_a is the multiplicity of a in A, while δ_a is the Dirac measure supported at a. Hence in what follows, we sometimes abuse the notations and equate a multiset of points with the discrete measure induced by it, and talk about optimal transport between two multisets of points.

As shown in [18], one can consider $\widehat{\mathsf{P}} := \mathsf{P} \cup \pi(\mathsf{Q})$ and $\widehat{\mathsf{Q}} := \mathsf{Q} \cup \pi(\mathsf{P})$ and modify the Euclidean distance so that $d(x, y) = 0$ for $x, y \in \pi(P) \cup \pi(Q)$ to obtain a modified pseudo-metric space $(\mathbb{R}^2, d)$. In this case, $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P},\mathsf{Q})$ becomes the optimal transport between the discrete measures induced by $\widehat{\mathsf{P}}$ and $\widehat{\mathsf{Q}}$ under this modified pseudo-metric.

We can also relate $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P},\mathsf{Q})$ to the optimal transport between the discrete measures induced by $\widehat{\mathsf{P}}$ and $\widehat{\mathsf{Q}}$ with the following observation (simple proof is in Appendix A):

▶ **Observation 4.** *Let* $\mu_{\widehat{\mathsf{P}}}$ *be the discrete measure induced by* $\widehat{\mathsf{P}}$ *and* $\nu_{\widehat{\mathsf{Q}}}$ *be the discrete measure induced by* $\widehat{\mathsf{Q}}$*. Then* $\mathrm{d}_{\mathrm{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}}) \le 2 \cdot \mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P},\mathsf{Q})$*.*

Given two discrete measures $\mu, \nu \in X \times \mathbb{R}$ on a finite metric space (X, d_X), computing the optimal transport distance can be reduced to finding the optimal min-cost flow on a complete bipartite graph using combinatorial flow algorithms as described in [20]. In our setting later, μ and ν will both be induced by point sets in $\mathbb{R}^2$, and d_X is the standard Euclidean distance.

2.3 Quadtree-based approximation algorithms for optimal transport

In this section, we briefly review two algorithms to approximate the optimal transport for two discrete measures μ and ν. Both of these algorithms use a randomly shifted quadtree, which we introduce first.

Randomly-shifted quadtree. Let $X \subseteq \mathbb{R}^d$ be a finite set of points (for our setting, $d = 2$ for persistence diagrams). To simplify the description, we will assume that the minimum pairwise distance between any two points in X is 1 and that X is contained in $[0, \Delta]^d$ (where Δ, the ratio of the diameter of X over the minimum pairwise distance, is also called the *spread* of X). First, let $H_0 = [-\Delta, \Delta]^d$ be the hypercube with side length 2Δ which is centered at the origin. Now shift H_0 by a random vector whose coordinates are from $[0, \Delta]$ to obtain H. Note that H still encloses X as H has side length 2Δ.

Construct a tree of hypercubes by letting H be associated to the root and halving H along each dimension. Recurse on the resulting sub-hypercubes that contain at least one point from X, and stop when a hypercube contains exactly one point from X. Each leaf node of resulting quadtree T_X contains exactly one point in X, and there are exactly $|X|$ leaves. The resulting quadtree T_X has at most $O(\log(d\Delta))$ levels. To see that T_X has at most $O(\log(d\Delta))$ levels, consider the depth i of some internal node. We know that the hypercube associated with the node has a side length of $\frac{\Delta}{2^i}$ and the distance between any two points in the hypercube, c, is less than or equal to $\frac{\Delta\sqrt{d}}{2^i}$. Then $i \leq \log(d\Delta)$ so there are at most $O(\log(d\Delta))$ levels in T_X. Additionally, the size of T_X is $O(|X| \log(d\Delta))$. It can be constructed in $\tilde{O}(|X| \log(d\Delta))$ time where $\tilde{O}$ includes term polynomial in $\log |X|$. We set the root level as level $\log \Delta + 1$ and subsequent levels are labeled as $\log \Delta, \log \Delta - 1, \ldots$. The weight of each tree edge between level $\ell + 1$ and level ℓ is 2^ℓ. Note that the quadtree cell has side length 2^ℓ at level ℓ.

Approximation algorithm 1: L_1-embedding via T_X. Given two discrete measures μ and ν, let X be the union of their support [1]. Construct the randomly shifted quadtree T_X as described above; X is sometimes omitted from the subscript when its choice is clear from the context. Given a tree node $v \in \mathrm{T}_X$, its level is denoted by $\ell(v)$. We will abuse the notation slightly and use v also to denote the quadtree cell (which is a hypercube of size length $2^{\ell(v)}$). Given a discrete μ, then $\mu(v)$ denotes the total measure of points from μ contained within this quadtree cell, namely, the total size of points with multiplicity counted from μ within this quadtree cell. We can now map μ to a vector V^μ where each index corresponds to a tree node $v \in \mathrm{T}_X$, and $\mathrm{V}^\mu[v]$ has coordinates $2^{\ell(v)}\mu(v)$. Similarly, map ν to vector V^ν. Then Indyk and Thaper [15] showed that $\|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1 = \sum_{v \in \mathrm{T}} 2^{\ell(v)} |\mu(u) - \nu(v)|$ gives an approximation to the optimal transport $\mathrm{d}_{\mathrm{OT}}(\mu, \nu)$ in expectation.

▶ **Theorem 5** ([15]). *Given two discrete measures μ, ν such that $\mathsf{supp}(\mu) \cup \mathsf{supp}(\nu) \subseteq \mathbb{R}^2$ and $s = |\mathsf{supp}(\mu) \cup \mathsf{supp}(\nu)|$, using a randomly shifted quadtree, $\|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1$ can be calculated in time $O(s \log \Delta)$ and there are constants C_1, C_2 such that $C_1 \cdot \mathrm{d}_{\mathrm{OT}}(\mu, \nu) \leq E[\|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1] \leq C_2 \cdot \log \Delta \mathrm{d}_{\mathrm{OT}}(\mu, \nu)$. Here, $E[\cdot]$ stands for the expectation.*

We note that it also turns out that $\|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1$ gives exactly the optimal transport between μ and ν along the tree metric induced by T_X. Specifically, for each $v \in \mathrm{T}_X$, set its weight to be $w(v) = 2^{\ell(v)}$. Then for any $x, x' \in X$, define $d_T(x, x')$ to be the total weight of the unique tree path connecting the quadtree leaf v_x (containing x) and leaf $v_{x'}$ (containing x'). Then the optimal transport between μ and ν w.r.t. metric d_T, denoted by $\mathrm{d}_{\mathrm{OT},\mathrm{d_T}}(\mu, \nu)$, satisfies that $\mathrm{d}_{\mathrm{OT},\mathrm{d_T}}(\mu, \nu) = \|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1$.

[1] Note that in general, X can be a superset of the support of μ and ν. Indeed, if there are a set of m measures and we perform kNN queries for a query measure, it is more convenient to set X as the union of support of all these measures and build only a single quadtree T_X.

Furthermore, we can consider that this optimal transport $\mathrm{d}_{\mathrm{OT},\mathrm{d}_\mathrm{T}}$ is generated by a following *greedy-flow* $f^*_G : X \times X \to \mathbb{R}$: Starting from leaf-nodes, we will match up as many unmatched points $\mu \cap v$ to $\nu \cap v$ as we can within each node v, and pass the remaining unmatched portion to its parent. In general, each tree node $v \in \mathrm{T}$ will have a μ-demand $\widehat{\mu}(v)$ and $\widehat{\nu}(v)$, which collect all unmatched measure from its 2^d child nodes. μ-demand (resp. ν-demand) at a leaf node v is initialized to be $\mu(v)$ (resp. $\nu(v)$). We then match these demand as much as we can and pass on $|\widehat{\mu}(v) - \widehat{\nu}(v)|$ to its parent as unmatched μ-measure, or unmatched ν-measure, whichever is left. Note that a greedy-flow f_G is not unique, but it tuns out that any such greedy-flow (greedy transport plan) gives rise to the optimal transport distance between μ and ν w.r.t. the tree metric d_T (See [16] for more detail): i.e,

$$(\|\mathrm{V}^\mu - \mathrm{V}^\nu\|_1 =)\ \mathrm{d}_{\mathrm{OT},\mathrm{d}_\mathrm{T}}(\mu,\nu) = \sum_{x,x' \in X} f^*_G(x,x') d_T(x,x'). \tag{4}$$

Approximation algorithm 2: Flowtree. The flowtree algorithm by [1] is based on the previous approach. The only modification is that, consider a greedy-flow f^*_G as described above. Instead of using the tree metric d_T to compute the optimal transport distance, the *flowtree estimate* computes the cost of this flow using the standard Euclidean distance:

$$\mathrm{d}^{\mathrm{flow}}_{\mathrm{OT}}(\mu,\nu) = \sum_{x,x' \in X} f^*_G(x,x') \|x - x'\|. \tag{5}$$

Comparing Equation (4) to the above equation, the difference is minor ($d_T(x,x')$ versus $\|x - x'\|$). However, in practice, $\mathrm{d}^{\mathrm{flow}}_{\mathrm{OT}}$ appears to provide a much more accurate estimate to the optimal transport distance $\mathrm{d}_{\mathrm{OT}}(\mu,\nu)$ w.r.t. the Euclidean distance. Unfortunately, unlike $\mathrm{d}_{\mathrm{OT},\mathrm{d}_\mathrm{T}}$, which can be computed as a L_1-distance between two specific vectors, to compute $\mathrm{d}^{\mathrm{flow}}_{\mathrm{OT}}$, we now have to compute a greedy-flow f^*_G explicitly (which can be done linear in the size of quadtree; however conceptually, this is not as simple as L_1-distance). Overall, we have the following result:

▶ **Theorem 6** ([1]). *Given two discrete measures μ, ν such that $\mathsf{supp}(\mu) \cup \mathsf{supp}(\nu) \subseteq \mathbb{R}^2$ and $s = |\mathsf{supp}(\mu) \cup \mathsf{supp}(\nu)|$, using a randomly shifted quadtree, $\mathrm{d}^{\mathrm{flow}}_{\mathrm{OT}}(\mu,\nu)$ can be computed in time $O(s \log \Delta)$ and there are constants C_1, C_2 such that $C_1 \cdot \mathrm{d}_{\mathrm{OT}}(\mu,\nu) \le E[\mathrm{d}^{\mathrm{flow}}_{\mathrm{OT}}(\mu,\nu)] \le C_2 \cdot \log \Delta \cdot \mathrm{d}_{\mathrm{OT}}(\mu,\nu)$. Here, $E[\cdot]$ stands for the expectation.*

3 Approximating 1-Wasserstein distances for persistence diagrams

We now present two algorithms to approximate the 1-Wasserstein distance for persistence diagrams, based on the approximation schemes of optimal transport in Section 2.3. Note that the results here are developed for the L_2 norm and through the equivalence of norms, can be generalized to any L_p norm and only changes the constant factor in the distortion induced by each approximation.

3.1 Approximation algorithms via L_1 embedding

3.1.1 Description of the new quadtree-based L_1-embedding

Let P and Q be two persistence diagrams and let $X = \mathsf{P} \uplus \mathsf{Q}$, the disjoint union of P and Q. In what follows, for simplicity of presentation, we assume that the minimum distance between any two distinct points in X, as well as between any point in X with a point in the

diagonal $\mathcal{L}$, is 1. The latter constraint can be removed with some extra care on handling leaf nodes in the quadtree. Assume w.l.o.g that Δ is a power of 2.

Partition the (randomly shifted) hypercube H described in section 2.3 into grids where the cells have side length $\Delta, \Delta/2, \ldots, 2^i, \ldots, 2, 1, \frac{1}{2}$. Note that each cell at the lowest level can contain at most one point, and if a leaf contains a point then it cannot intersect the diagonal $\mathcal{L}$. Let T_X be the resulting quadtree, where leaves are all cells that contain exactly one point from X. We further use G_i to denote the set of quadtree cells with side-length 2^i (i.e, those in level-i); we refer to G_i as the level-i grid. Note that the size of the quadtree is $O(|X| \log \Delta)$. Additionally, we call a cell a *terminal cell* if it intersects the diagonal $\mathcal{L}$; otherwise, it is *non-terminal*.

Now for each grid G_i, construct a vector $\mathrm{V}_i^{\mathsf{P}}$ with one coordinate per cell, where each coordinate counts the number of points in the corresponding cell. The vector representation V^{P} for P is then the concatenation of all these vectors $2^i \mathrm{V}_i^{\mathsf{P}}$ where 2^i is the cell side length for grid G_i:

$$\mathrm{V}^{\mathsf{P}} = [\frac{1}{2}\mathrm{V}_{-1}^{\mathsf{P}}, \mathrm{V}_0^{\mathsf{P}}, 2\mathrm{V}_1^{\mathsf{P}}, \ldots, 2^i \mathrm{V}_i^{\mathsf{P}}, \ldots,]$$

Construct the vector V^{Q} similarly. We use p_k to denote the value of coordinate k in $\mathrm{V}_i^{\mathsf{P}}$ and q_k to denote the value of coordinate k in $\mathrm{V}_i^{\mathsf{Q}}$. Now, we will describe a *modified-L_1 distance* $|\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T$ for these vectors, which is similar to the L_1 norm. To compute $|\mathrm{V}_i^{\mathsf{P}} - \mathrm{V}_i^{\mathsf{Q}}|_T$, we will define $|p_k - q_k|_T$. There are two cases for the $|p_k - q_k|_T$ to consider:
Case 1: if coordinate k is not associated with a terminal cell, then use $|p_k - q_k|$ for $|p_k - q_k|_T$.
Case 2: if coordinate k is associated with a terminal cell, then set $|p_k - q_k|_T = 0$.
Then we have $|\mathrm{V}_i^{\mathsf{P}} - \mathrm{V}_i^{\mathsf{Q}}|_T = \sum_{k=1}^{|G_i|} |p_k - q_k|_T$, and

$$\widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q}) := |\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T = \sum_{i=-1}^{\log_2 \Delta} 2^i |\mathrm{V}_i^{\mathsf{P}} - \mathrm{V}_i^{\mathsf{Q}}|_T. \tag{6}$$

An equivalent L_1-distance formulation. We introduce the above vector representation and the modified L_1-distance as it is more convenient for later theoretical analysis. However, algorithmically, we wish to have a true L_1-embedding. It turns out that an equivalent formulation is as follows: Let $\widehat{G}_i$ denote the level-i quadtree cells *that do not intersect the diagonal* $\mathcal{L}$. We then compute a vector representation $\widehat{\mathrm{V}}^{\mathsf{P}}$ (resp. $\widehat{\mathrm{V}}^{\mathsf{Q}}$) restricted only to cells in $\bigcup \widehat{G}_i$. In other words, all entries corresponding to cells intersecting the diagonal are ignored in constructing $\widehat{\mathrm{V}}^{\mathsf{P}}$ and $\widehat{\mathrm{V}}^{\mathsf{Q}}$. We then have that

$$\|\widehat{\mathrm{V}}^{\mathsf{P}} - \widehat{\mathrm{V}}^{\mathsf{Q}}\|_1 = |\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T = \widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q}). \tag{7}$$

That is, our quadtree-induced distance $\widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q})$ is a L_1-distance for suitably constructed vectors. Nevertheless, we use the definition as in Equation (6) to simplify proofs later.

It is easy to see that the construction takes the same time as the L_1-embedding approach described in Section 2.3. We now show that the L_1-distance $\widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q})$ approximates the 1-Wasserstein distance $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q})$ for the persistence diagrams. The main results for this L_1-embedding approach are summarized as follows, and we prove the approximation bound in Section 3.1.2.

▶ **Theorem 7.** *Given persistence diagrams P and Q such that $s = |\mathsf{P}| + |\mathsf{Q}|$, we can compute $\widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q}) = |\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T$ in time $O(s \log \Delta)$ using the randomly shifted quadtree. Furthermore, the expected value of $\widehat{d}_{L_1}(\mathsf{P}, \mathsf{Q})$ is an $O(\log \Delta)$-approximation of the 1-Wasserstein distance $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q})$; i.e, there are constants c_1 and c_2 such that $c_1 \cdot \mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q}) \le E[|\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T] \le c_2 \log \Delta \cdot \mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q})$.*

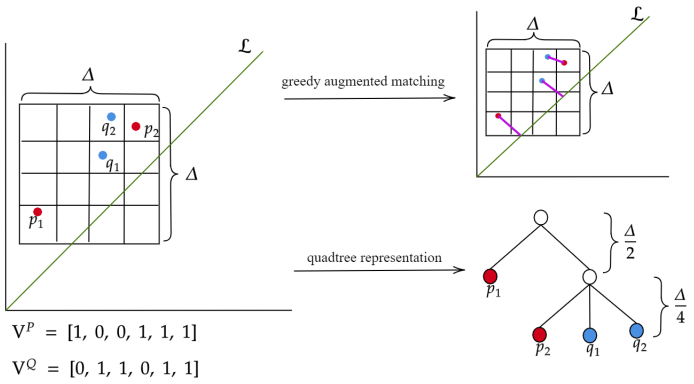


Figure 1 A simple example of the vector and quadtree representations of the persistence diagrams $P = \{p_1, p_2\}$ and $Q = \{q_1, q_2\}$.

3.1.2 Approximation guarantees

Our approximation bound in Theorem 7 follows from Lemmas 8 and 9 below. To prove these lemmas, we will first introduce a *greedy augmented matching*.

Greedy augmented matching $\widehat{\Gamma}$. We construct the following augmented matching (recall Definition 2) $\widehat{\Gamma} \subseteq (\mathsf{P} \cup \pi(\mathsf{Q})) \times (\mathsf{Q} \cup \pi(\mathsf{P}))$ in a *bottom-up greedy manner*: Starting from the level $i = -1$, we will aim to match points in $\mathsf{P} \uplus \mathsf{Q}$ as much as we can within each level G_i. Those remaining unmatched points will then be considered at the next level G_{i+1}: in particular, within each *non-terminal cell* in G_{i+1}, we will match the maximal possible unmatched points in P to unmatched points in Q so far, and pass the remainder unmatched points (which can now only come from either P or Q, but not both) to its parents. Within a *terminal cell* v in G_{i+1}, we match every unmatched point p from $\mathsf{P} \cap v$ and from $\mathsf{Q} \cap v$ to its closest point $\pi(p) \in \mathcal{L}$ in the diagonal. Finally, at the root cell (in level $\log \Delta$), any unpaired points from either P or Q will be paired to the diagonal, as the root is a terminal cell. Note that by construction, at any level i, $|\mathrm{V}_i^{\mathsf{P}} - \mathrm{V}_i^{\mathsf{Q}}|_T$ is exactly the number of points from $\mathsf{P} \uplus \mathsf{Q}$ that could not be matched at level i or below under such a greedy augmented matching and will subsequently need to be matched in grid G_j, $j \geq i+1$. See Figure 1 for a simple illustration.

▶ **Lemma 8.** *There is a constant C such that $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q}) \leq C \cdot |\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T$.*

Proof. Consider the greedy augmented matching $\widehat{\Gamma}$ we described above, induced by pairing points or pairing points with their diagonal projection greedily within the same cells of the grids $G_{-1}, G_0, G_1, \ldots$ in a bottom-up manner. First, given $(p, q) \in \widehat{\Gamma}$, we say that (p, q) is paired in level-i, if the lowest level any quadtree cell containing both p, q is level-i; intuitively, (p, q) are paired greedily in this cell in level-i. We now use $\widehat{\Gamma}_i \subseteq \widehat{\Gamma}$ to denote the set of pairs from level-i. For each level i, let $cost(i) = \sum_{(p,q)\in\widehat{\Gamma}_i} \|p - q\|_2$ be the total cost incurred by all those pairs from $\widehat{\Gamma}_i$, and obviously,

$$\sum_i cost(i) = \sum_{(p,q)\in\widehat{\Gamma}} \|p - q\|_2 \geq \mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q}), \tag{8}$$

where the right inequality holds as $\mathrm{d}_{\mathrm{W},1}^{\mathrm{per}}(\mathsf{P}, \mathsf{Q})$ is the smallest total cost of any augmented matching and the greedy augmented matching $\widehat{\Gamma}$ is an augmented matching.

There are no pairings induced in the grid G_{-1} (of size 1/2) since the minimum inter-point distance is 1 and the minimum distance between a point and its diagonal is 1 as well. Hence we have that $|\mathrm{V}^{\mathsf{P}}_{-1} - \mathrm{V}^{\mathsf{Q}}_{-1}|_T = |\mathsf{P}| + |\mathsf{Q}|$. Since all points from P and Q will remain unpaired in level -1 (i.e, within cells of grid G_{-1}), we then have that there are exactly

$$|\mathsf{P}| + |\mathsf{Q}| - |\mathrm{V}^{\mathsf{P}}_0 - \mathrm{V}^{\mathsf{Q}}_0|_T = |\mathrm{V}^{\mathsf{P}}_{-1} - \mathrm{V}^{\mathsf{Q}}_{-1}|_T - |\mathrm{V}^{\mathsf{P}}_0 - \mathrm{V}^{\mathsf{Q}}_0|_T$$

total number of points from $\mathsf{P} \uplus \mathsf{Q}$ that can either be paired to each other or matched to the diagonal in grid cells G_0. As the maximal distance between any pair of matched points is $2^i\sqrt{2}$ within a cell in G_i, the maximum cost incurred by all matched points in G_0 is $cost(0) \leq \sqrt{2}(|\mathrm{V}^{\mathsf{P}}_{-1} - \mathrm{V}^{\mathsf{Q}}_{-1}|_T - |\mathrm{V}^{\mathsf{P}}_0 - \mathrm{V}^{\mathsf{Q}}_0|_T)$. In general, the maximum cost of the matched points in G_i is

$$cost(i) \leq 2^i \cdot \sqrt{2}(|\mathrm{V}^{\mathsf{P}}_{i-1} - \mathrm{V}^{\mathsf{Q}}_{i-1}|_T - |\mathrm{V}^{\mathsf{P}}_i - \mathrm{V}^{\mathsf{Q}}_i|_T).$$

Hence combining Equation (8), the total cost (i.e, $\sum_{(p,q)\in\widehat{\Gamma}} ||p-q||_2$) of the greedy augmented matching $\widehat{\Gamma}$, is bounded from above by:

$$\sum_{(p,q)\in\widehat{\Gamma}} ||p-q||_2 \leq \sum_{i=0}^{\log_2 \Delta+1} 2^i \cdot \sqrt{2}(|\mathrm{V}^{\mathsf{P}}_{i-1} - \mathrm{V}^{\mathsf{Q}}_{i-1}|_T - |\mathrm{V}^{\mathsf{P}}_i - \mathrm{V}^{\mathsf{Q}}_i|_T) \leq 2\sqrt{2}|\mathrm{V}^P - \mathrm{V}^Q|_T.$$

By the right inequality of Equation (8), the claim then follows. ◀

▶ **Lemma 9.** *There is a constant C' such that the expected value of $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$ is bounded by $E[\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})] = E[|\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T] \leq C' \cdot \log\Delta \cdot \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$.*

Proof. Set $\widehat{\mathsf{P}} = \mathsf{P} \cup \pi(\mathsf{Q})$ and $\widehat{\mathsf{Q}} = \mathsf{Q} \cup \pi(\mathsf{P})$ as before. For a given grid G_i and some coordinate k in $\mathrm{V}^{\mathsf{P}}_i$ and $\mathrm{V}^{\mathsf{Q}}_i$, let p_k be the value of coordinate k in $\mathrm{V}^{\mathsf{P}}_i$ and q_k be the value of coordinate k in $\mathrm{V}^{\mathsf{Q}}_i$. Analogously, let $\mathrm{V}^{\widehat{\mathsf{P}}}$ and $\mathrm{V}^{\widehat{\mathsf{Q}}}$ be the vector representations w.r.t multisets $\widehat{P}$ and $\widehat{Q}$, respectively, and let $\widehat{p}_k$ (resp. $\widehat{q}_k$) be the value of coordinate k in $\mathrm{V}^{\widehat{\mathsf{P}}}_i$ (resp. $\mathrm{V}^{\widehat{\mathsf{Q}}}_i$). Note that $\widehat{p}_k \geq p_k$ and $\widehat{q}_k \geq q_k$.

(i) Now if $\widehat{p}_k > p_k$ or $\widehat{q}_k > q_k$, then there exists at least one point $x \in \pi(\mathsf{P}) \cup \pi(\mathsf{Q})$ in the cell v associated with coordinate k. In other words, this cell v must be a terminal cell, and $|p_k - q_k|_T = |\widehat{p}_k - \widehat{q}_k|_T = 0$.

(ii) Otherwise if the conditions in (i) do not hold, it must be that $\widehat{p}_k = p_k$ and $\widehat{q}_k = q_k$, in which case we also have that $|p_k - q_k|_T = |\widehat{p}_k - \widehat{q}_k|_T$.

Combining (i) and (ii) we then have that $|\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T \leq |\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}|_T$.

On the other hand, by the definition of metric $|\cdot|_T$, we know that $|\widehat{p}_k - \widehat{q}_k|_T \leq |\widehat{p}_k - \widehat{q}_k|$, implying that $|\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}|_T \leq \|\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}\|_1$. Let $\mu_{\widehat{\mathsf{P}}}$ and $\nu_{\widehat{\mathsf{Q}}}$ be the discrete measures induced by $\widehat{\mathsf{P}}$ and $\widehat{\mathsf{Q}}$ respectively. By Theorem 5, there is some constant C such that $E[\|\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}\|_1] \leq C \cdot \log\Delta \cdot \mathrm{d}_{\mathrm{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}})$. From Observation 4, $\mathrm{d}_{\mathrm{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}}) \leq 2 \cdot \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$. Therefore,

$$E[|\mathrm{V}^{\mathsf{P}} - \mathrm{V}^{\mathsf{Q}}|_T] \leq E[|\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}|_T] \leq E[\|\mathrm{V}^{\widehat{\mathsf{P}}} - \mathrm{V}^{\widehat{\mathsf{Q}}}\|_1] \leq 2 \cdot C \cdot \log\Delta \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q}).$$

The lemma then follows. ◀

3.2 Approximation algorithm via flowtree

We now propose an alternative approximation algorithm for $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$. The high level idea is the same as the flowtree algorithm described in Section 2.3: in particular, we first compute the optimal flow for points in P and Q along the randomly shifted quadtree T as constructed

earlier, but now with the modification that a point can be paired to diagonal. It turns out that this leads to the same greedy augmented matching $\widehat{\Gamma}$ we described at the beginning of Section 3.1.2. Then, similar to flowtree, under this greedy augmented matching $\widehat{\Gamma}$, we use the Euclidean distance between a pair of matched points (instead of using the tree-distance as for $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$) to measure the cost of each pair of matched points. This leads to the following *modified flowtree estimate*:

$$\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q}) = \sum_{(p,q)\in\widehat{\Gamma}} ||p-q||_2, \tag{9}$$

and in our second algorithm, we will use $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q})$ as an approximation of the true 1-Wasserstein distance $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$.

From an implementation point of view, unlike $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$, which can be computed as the L_1-distance between two vectors, we now must explicitly compute the greedy augmented matching, $\widehat{\Gamma}$ between P and Q. (Note that this greedy augmented matching was only used in proving the approximation guarantee for $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$, and not needed for its computation.) Computing this greedy augmented matching (and calculating its cost) takes the same time as computing the greedy flow in the original flowtree algorithm 2.3. Furthermore, it is easy to see that in the proof of Lemma 8, we in fact showed that $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q}) \leq \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q}) \leq C\cdot|\mathrm{V}^P-\mathrm{V}^Q|_T$ (see Eqn (8)). Combining this with Theorem 5, we thus obtain the following approximation result for this modified flowtree estimate.

▶ **Theorem 10.** *Given two persistence diagrams* P *and* Q *where* $s=\max(|P|,|Q|)$ *and* Δ *is the spread of point set* $\mathsf{P}\cup\mathsf{Q}$*, we can compute* $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q})$ *in time* $O(s\log\Delta)$ *using a randomly shifted quadtree. Additionally, the expected value of* $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q})$ *is an* $O(\log\Delta)$*-approximation of the 1-Wasserstein distance* $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$*; i.e. there are constants* C_1 *and* C_2 *such that* $C_1\cdot\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q}) \leq E[\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q})] \leq C_2\cdot\log\Delta\cdot\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$.

Remark. We remark that while these two approximation schemes, $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$ and $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1\,F}(\mathsf{P},\mathsf{Q})$, have similar approximation guarantees for $\mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(\mathsf{P},\mathsf{Q})$, in practice, the modified flowtree based approach has much higher accuracy. This is consistent with the performance of flowtree algorithm versus the L_1-embedding approach for general optimal distance [1]. In contrast, the benefit of the L_1 embedding approach is that it is easy to compute $\widehat{d}_{L_1}(\mathsf{P},\mathsf{Q})$. Also, each persistence diagram is now mapped to a vector representation, and the distance is L_1-distance among these vector representations. One could combine this with methods such as locality-sensitive-hashing for more efficient approximate k-nearest neighbor queries. In general, such a L_1-norm also makes $\widehat{d}_{L_1}(\cdot,\cdot)$ potentially more suitable for downstream machine learning pipelines.

4 Experimental results

We evaluate both the runtime and accuracy of the modified flowtree and L_1-embedding against the *Hera* method of [18]. We use the implementation of Hera provided in the GUDHI library [25] for testing. We run the experiments using an Intel Core i7-1065G7 CPU @ 1.30 GHz and 12.0GB RAM. Additionally, the implementations for both the modified flowtree and L_1-embedding were done in C++ (wrapped in python for evaluation) and are based on the code provided in [1].

Datasets. For our experiments, we use both synthetic persistence diagrams, as well as persistence diagrams generated from real data. For synthetic data, we generate two sets of persistence diagrams: called "*synthetic-uniform*" and "*synthetic-Gaussian*", which are generated by a uniform sample and a sample w.r.t. a Gaussian distribution on the birth-death plane to obtain the persistence diagrams, respectively. For real datasets, we use persistence diagrams generated from the so-called Reddit data sets (which is a collection of graphs) [6], and from the ModelNet10 [30] dataset of shapes. Details of these datasets are in Appendix B.

Speed comparison. We compare the running time of our new approximation algorithms with that of Hera [18] – note that we do not directly compare with the exact algorithm by Dionysus, because as reported by [18], Hera is 50 times to 400 times faster than Dionysus. Note that Hera is also an approximation algorithm, and there is a parameter ε to adjust its approximation factor $(1+\varepsilon)$. By setting this parameter to be very small ($\varepsilon = 0.01$), we use the distance computed by Hera as ground truth later when we measure approximation accuracy; see Table 1.

The comparison of the running times of our approaches with that of Hera (for a range of different approximation factors) can be found in Figure 2, which summarizes the runtime for each method on a **log-scale** using both randomly generated diagrams as well as the reddit-binary dataset (real persistence diagrams from graphs). We compare the speed of our modified flowtree and L_1 embedding against Hera where the parameter ε for Hera is set to be 300000. However, note that as one relaxes the approximation parameter ε, the speed of Hera in fact does not improve much (as shown in Figure 2). Thus our speed gains remain no matter which choice of ε we use for Hera. Additionally, the true approximation error for Hera also does not decrease much; see Table 1. To get the approximation error for Hera, we find the true Wasserstein distance by using the `wasserstein_distance` function from the GUDHI library which uses the Python Optimal Transport library [12] and is based on ideas from [21]. The results in Figure 2 indicate that the modified flowtree approach is between 50 and 1000 times faster than Hera and the difference increases as the size of the diagrams increases. Similarly, the L_1 embedding approach is between 150 and 4900 times faster than Hera. Both are order of magnitudes faster than Hera; but the price to pay is that the approximation factor is worse for our approach as shown in Appendix C Figure 3.

Approximation accuracy comparison. To measure the accuracy of the approximation of both the modified flowtree and L_1 embedding approach, we first measure the average relative error and standard deviation of both methods on all datasets using L_1, L_2, and L_∞ as the ground metrics. In particular, given a ground truth distance d and an approximate distance $\tilde{d}$, the relative error is $\rho(d) = \frac{|d-\tilde{d}|}{d}$. As mentioned earlier, we use the output of Hera for $\varepsilon = 0.01$ as ground truth, and compare our approximated distance with that. The results are summarized in Figure 3 and a detailed table of the average relative error and standard deviation is in Table 2. Overall, while our modified flowtree is slower than the L_1-embedding approach, it achieves much better approximation error. For our experiments, we generate a quadtree only once for all persistence diagrams in a given dataset and calculate error for the approximated distances for the single quadtree. However, note that by constructing several quadtrees and averaging the distance estimates or taking the smallest estimated, we could potentially reduce the approximation error.

In addition to the average error of our approximate distances, we can also consider the efficacy of both methods in terms of nearest neighbor search and ranking accuracy. To evaluate nearest neighbor search, we first split the set of persistence diagrams into query

diagrams and candidate diagrams. Then, we measure recall@m accuracy where recall@m is defined as the fraction of queries that have the true nearest neighbor within the top m-ranked candidates returned by the evaluated method. The results are reported in Figure 4. For ranking accuracy, the detailed results are in Appendix C. To summarize, the modified flowtree approach is more accurate than the L_1 embedding approach both in terms of nearest neighbor search and closeness to the true ranking of candidate diagrams for a fixed query diagram. Both approaches appear to have a lower degree of accuracy on diagrams where there a higher proportion of points near the diagonal. This may be due to the increased possibility of erroneously matching points to the diagonal.

In summary, we note that both our new approximation algorithms significantly improve the speed previously best-known Hera algorithm by orders of magnitudes, but with worse approximation factors. Empirically, the approximation factors remain constant despite our theoretical results suggestion an $O(\log \Delta)$-approximation. In particular, the relative approximation error of flowtree is often smaller than 0.50 for L_2 ground metric (see Appendix C Table 2).

Table 1 Comparison of the maximum allowed relative error with the average experimental relative error for Hera on the reddit-binary dataset. The relative error was calculated using the GUDHI library's `wasserstein_distance` function which uses the Python Optimal Transport library to compute exact 1-Wasserstein distance.

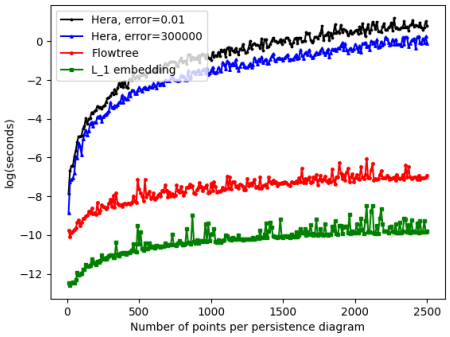
Maximum allowed relative error	Average relative error
0.1	0.00043768
1.0	0.003331
100000	0.0076055
200000	0.0076055

Table 2 Average error and standard deviation for all datasets. We abbreviate the L_1 embedding approach to embd and the flowtree approach to ft.

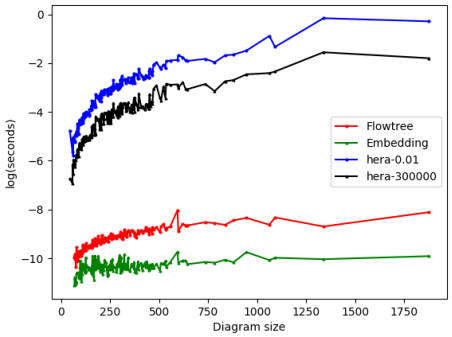
		L_1		L_2		L_∞	
		embd	ft	embd	ft	embd	ft
synthetic-uniform	Avg. Error	2.058	0.2846	3.161	0.2664	4.536	0.2595
	Std. Dev.	1.034	0.3891	1.189	0.3488	1.164	0.3176
synthetic-Gaussian	Avg. Error	1.341	0.3358	2.136	0.2860	3.035	0.2251
	Std. Dev.	0.3647	0.1809	0.4139	0.1519	0.3669	0.1178
reddit-binary	Avg. Error	2.112	0.2899	3.089	0.3080	3.921	0.2854
	Std. Dev.	1.275	0.3859	2.100	0.5126	2.427	0.4801
ModelNet10	Avg. Error	2.189	0.7331	2.438	0.4929	3.061	0.9399
	Std. Dev.	0.9543	0.4132	0.8136	0.3171	1.051	0.4448

5 Concluding remarks

In this paper, we presented two algorithms for fast approximation of the 1-Wasserstein distance between persistence diagrams based on L_1 embedding. While the relative error incurred by both algorithms is higher than that of Hera, the runtime is significantly faster. We also observe that approximation methods introduced are more accurate on persistence diagrams with a lower proportion of points near the diagonal.

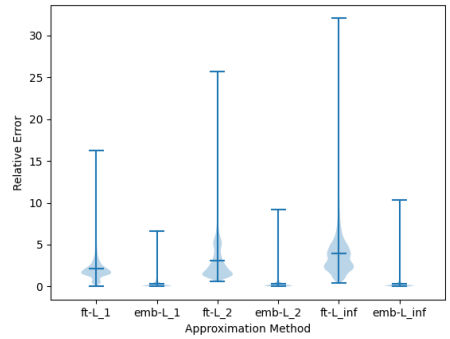


(a) synthetic-uniform.

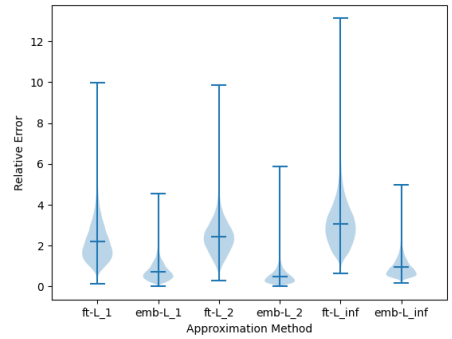


(b) reddit-binary.

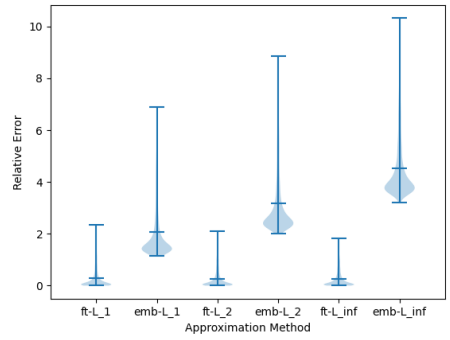
Figure 2 Comparison of the runtimes of HERA, flowtree, and L_1 embedding using both generated and real data with L_2 as the ground metric.



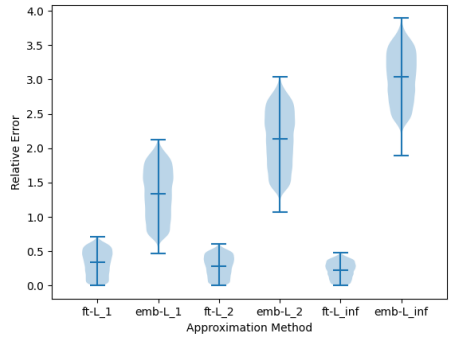
(a) reddit-binary.



(b) ModelNet10.

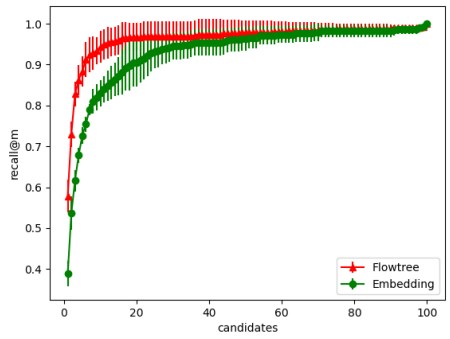


(c) synthetic-uniform.

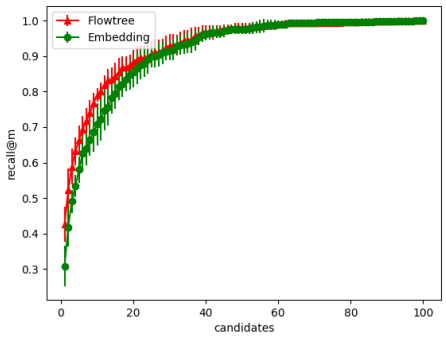


(d) synthetic-Gaussian.

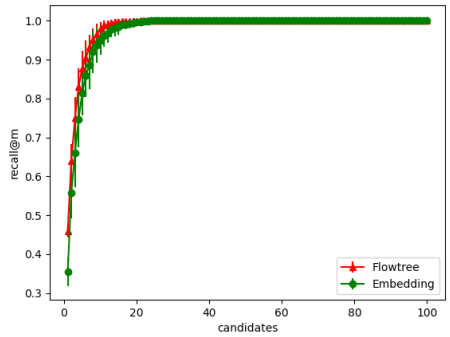
Figure 3 Relative error for flowtree and quadtree approximations over L_1, L_2, and L_∞ ground metrics for all datasets.



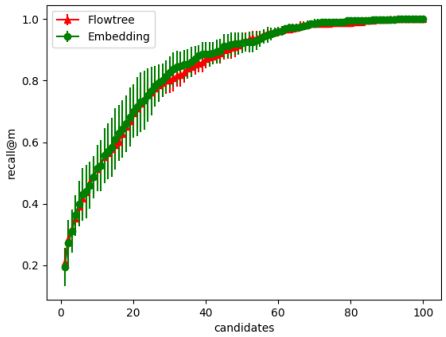
(a) Recall@m accuracy on reddit-binary dataset with L_2 ground metric.



(b) Recall@m accuracy ModelNet10 dataset with L_2 ground metric.



(c) Recall@m accuracy on synthetic-uniform dataset with L_2 ground metric.



(d) Recall@m accuracy on synthetic-Gaussian dataset with L_2 ground metric.

Figure 4 Recall@m accuracy on reddit-binary and ModelNet10 datasets with L_2 ground metric.

In the future, we are interested in using the L_1 embedding described with locality sensitive hashing for sub-linear nearest neighbor search. Additionally, it maybe be possible to use the ideas to compare persistence diagrams under some transformations: e.g, parallel shifting along the diagonal directions (which corresponding to that the input functions generating the persistence diagram is added by a constant term). It will also be interesting to expand this work to perform statistics on the space of persistence diagrams (e.g, computing 1-mean of persistence diagrams under our approximation distances).

References

1 Arturs Backurs, Yihe Dong, Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Scalable nearest neighbor search for optimal transport. *arXiv preprint arXiv:1910.04126*, 2019.

2 Yair Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of 37th Conference on Foundations of Computer Science*, pages 184–193. IEEE, 1996.

3 Yair Bartal. On approximating arbitrary metrices by tree metrics. *STOC*, 98:161–168, 1998.

4 Dimitri Bertsekas. The auction algorithm: A distributed relaxation method for the assignment problem. *Annals of Operations Research*, 14(1):105–123, 1988.

5 Mickaël Buchet, Yasuaki Hiraoka, and Ippei Obayashi. Persistence homology and material and informatics. In Isao Tanaka, editor, *Nanoinformatics*, pages 75–95. Spring Singapore, Singapore, 2018. doi:10.1007/978-981-10-7617-6_5.

6 Chen Cai and Yusu Wang. Understanding the power of persistence pairing via permutation test. *arXiv preprint arXiv:2001.06058*, 2020.

7 Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on theory of computing*, pages 2292–2300. ACM, 2002.

8 Frédéric Chazal, David Cohen-Steiner, Leonidas J. Guibas, Facundo Mémoli, and Steve Y. Oudot. Gromov-hausdorff stable signatures for shapes using persistence. *Computer Graphics Forum*, 28(5):1393–1403, 2009.

9 David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete Comput. Geom.*, 37(1):103–120, 2007.

10 Herbert Edelsbrunner and John Harer. *Computational topology: an introduction*. American Mathematical Society, 2010.

11 Herbert Edelsbrunner, David Letscher, and Afra Zomorodian. Topological persistence and simplification. *Discrete Comput. Gemo.*, 28:511–533, 2002.

12 Rémi Flamary and Nicolas Courty. Pot: Python optimal transport library, 2017. URL: https://pythonot.github.io/.

13 Jennifer Gamble and Giseon Ho. Exploring uses of persistence homology for statistical analysis of landmark-based shape data. *Journal of Multivariate Analysis*, 101(9):2184–2199, 2010.

14 Yasuaki Hiraoka, Takenobu Nakamura, Akihiko Hirata, Emerson G. Escolar, Kaname Matsue, and Yasumasa Nishiura. Hierarchical structures of amorphous solids characterized by persistent homology. *Proceedings of the National Academy of Sciences*, 113(26):7035–7040, 2016. doi:10.1073/pnas.1520877113.

15 Piotr Indyk and Nitin Thaper. Fast image retrieval via embeddings. In *3rd international workshop on statistical and computational theories of vision*, volume 2, page 5, 2003.

16 Bahman Kalantari and Iraj Kalantari. A linear-time algorithm for minimum cost flow on undirected one-trees. *Combinatorics Advances*, pages 217–223, 1995.

17 Lida Kanari, Paweł Dłotko, Martina Scolamiero, Ran Levi, Julian Shillcock, Kathryn Hess, and Henry Markram. A topological representation of branching neuronal morphologies. *Neuroinformatics*, 16(1):3–13, 2018. doi:10.1007/s12021-017-9341-1.

18 Michael Kerber, Dimitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *Journal of Experimental Algorithmics(JEA)*, 22:1–20, 2017.

19 Jon Kleinberg and Eva Tardos. Approximation algorithms for classification problems with pairwise relationships: Metric labeling and markov random fields. *Journal of the ACM (JACM)*, 49(5):616–639, 2002.

20 Harold Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

21 Théo Lacombe, Marco Cuturi, and Steve Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport, 2018. arXiv:1805.08331.

22 Tam Le, Makoto Yamada, Kenji Fukumizu, and Marco Cuturi. Tree-sliced approximation of wasserstein distances. *arXiv preprint arXiv:1902.00342*, 2019.

23 Yongjin Lee, Senja D. Barthel, Paweł Dłotko, S. Mohamad Moosavi, Kathryn Hess, and Berend Smit. Quantifying similarity of pore-geometry in nanoporous materials. *Nat Commun.*, 8:15396, 2017. doi:10.1038/ncomms15396.

24 Yanjie Li, Dingkang Wang, Giorgio Ascoli, Partha Mitra, and Yusu Wang. Metrics for comparing neuronal tree shapes based on persistent homology. *PLOS One*, 12(8):1–24, 2017. doi:10.1371/journal.pone.0182184.

25 Clément Maria, Jean-Daniel Boissonat, Marc Glisse, and Mariette Yvinec. The gudhi library: simplicial complexes and persistent homology, 2014. URL: http://gudhi.gforge.inria.fr/python/latest/index.html.

26 Dimitriy Morozov. Dionysus, 2010. URL: `mrzv.org/software/dionysus`.

27 Ahmet Sacan, Ozgur Ozturk, Hakan Ferhatosmanoglu, and Yusu Wang. Lfm-pro: a tool for detecting significant local structural sites in proteins. *Bioinformatics*, 23(6):709–716, 2007.

28 Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Fast unbalanced optimal transport on tree. *arXiv preprint arXiv:2006.02703*, 2020.

29 Primoz Skraba, Maks Ovsjanikov, Frédéric Chazal, and Leonidas Guibas. Persistence-based segmentation of deformable shapes. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 45–52, 2010. `doi:10.1109/CVPRW.2010.5543285`.

30 Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumentric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

A Additional Proofs

Proof of observation 4. By the optimality of $\mathrm{d_{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}})$, we know that

$$\mathrm{d_{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}}) \leq \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(P,Q) + \sum_{(a,b)\in\Gamma_1} ||\pi(a) - \pi(b)||_q$$

where Γ_1 is the set of $(a,b) \in \Gamma$ that have first form given in definition 2. We know that $||\pi(a) - \pi(b)||_q \leq ||a-b||_q$ so $\mathrm{d_{OT}}(\mu_{\widehat{\mathsf{P}}}, \nu_{\widehat{\mathsf{Q}}}) \leq \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(P,Q) + \sum_{(a,b)\in\Gamma_1} ||\pi(a) - \pi(b)||_q \leq 2 \cdot \mathrm{d}^{\mathrm{per}}_{\mathrm{W},1}(P,Q)$. ◀

B Datasets

We use datasets of synthetic persistence diagrams as well as persistence diagrams generated from real data. For persistence diagrams from real data, we use both graph and shape datasets.

- *Synthetic data*: We generated two sets of persistence diagrams. For the first set of synthetic persistence diagrams, we find a random persistence of size at most s where $s = 10x$ for $x \in \{1, \ldots, 100\}$ by generating points $p_1, \ldots, p_s$. To find each points p_i, we sample $p_i.x$ from a uniform distribution from 0 to 200. We then sample $p_i.y$ from a uniform distribution between x and 300. We will refer to this set of synthetic persistence diagrams as synthetic-uniform. For the second set of synthetic persistence diagrams, we again generate a point p_i by sampling $p_i.x$ from a uniform distribution from 0 to 200. We then sample $p_i.y$ from a Gaussian distribution centered about x with standard deviation 1.0. We will refer to the second set of synthetic diagrams as synthetic-Gaussian.
- *Graphs*: For persistence diagrams generated from graphs, we use the reddit-binary graph dataset, which consists of graphs corresponding to online discussion on Reddit. In each graph, nodes correspond to users and there is an edge between two nodes if at least one of the corresponding users has responded to the other's comments. The data is taken from four popular subreddits: IAmA, AskReddit, TrollXChromosomes, and atheism. Additionally, the persistence diagrams are generated using node degree as the filtration function [6].
- *Shapes*: We use the ModelNet10 [30] dataset to generate persistence diagrams from shapes. ModelNet10 is comprised of 4899 CAD models from 10 object categories. The persistence diagrams are generated using closeness centrality as the filtration function [6].

The statistics of the ModelNet10 and reddit-binary datasets are summarized in Table 3. Note that a persistence point is considered close to the diagonal if its lifetime is less than one-tenth of the lifetime of the point with the largest lifetime.

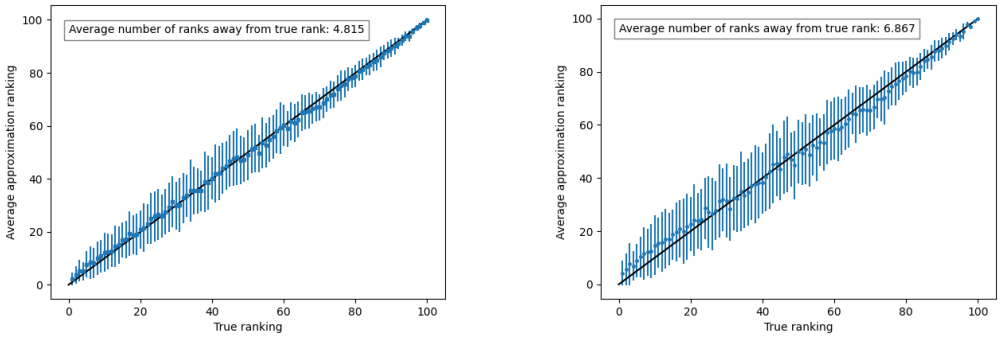
Table 3 Diagram statistics for reddit-binary and ModelNet10 datasets.

	Average # of PD points	Average # of points near diagonal
reddit-binary	278.155	20.245
ModelNet10	40.52	34.345

C Results

To measure the accuracy of the rankings produced by the modified L_1 embedding and flowtree methods, we plot the true ranking of each candidate against the rank of the candidate in the rankings produced by the evaluated method. Note that we will do this using the L_2 norm as the ground metric. The ranking accuracies for the evaluated methods for the reddit-binary dataset and ModelNet10 are summarized in Figures 5, 6, 8, 7. The average number of ranks away from the true rank is less than 10 for both reddit-binary and synthetic-uniform whereas the same metric for synthetic-uniform and ModelNet10 is above ten for both datasets.

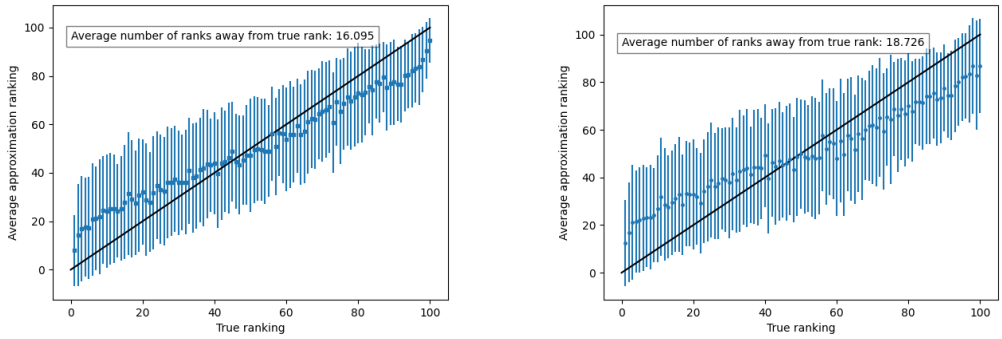
Both the modified flowtree and the L_1 embedding approximations seem to be less effective for estimating nearest neighbor for persistence diagrams where there is a high proportion of points near the diagonal. This may be because a higher proportion of points near the diagonal increases the possibility of erroneously matching points to the diagonal.



(a) Flowtree rankings.

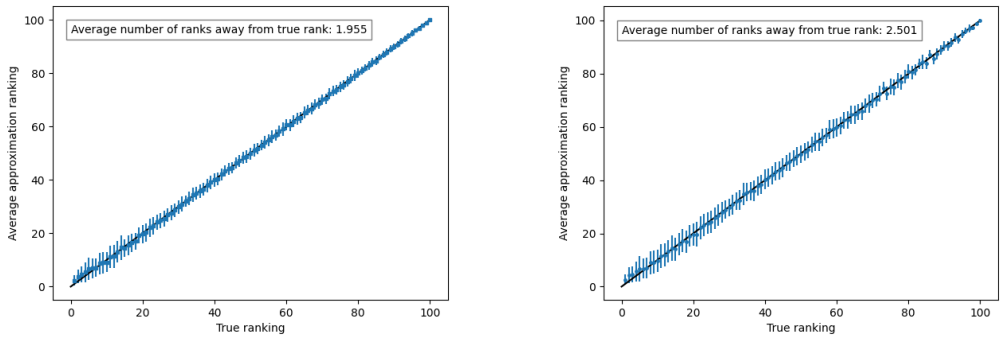
(b) L_1 embedding rankings.

Figure 5 Comparison of rankings generated by the flowtree and L_1 embedding approximations with the true rankings of the candidate diagrams using the reddit-binary datset.



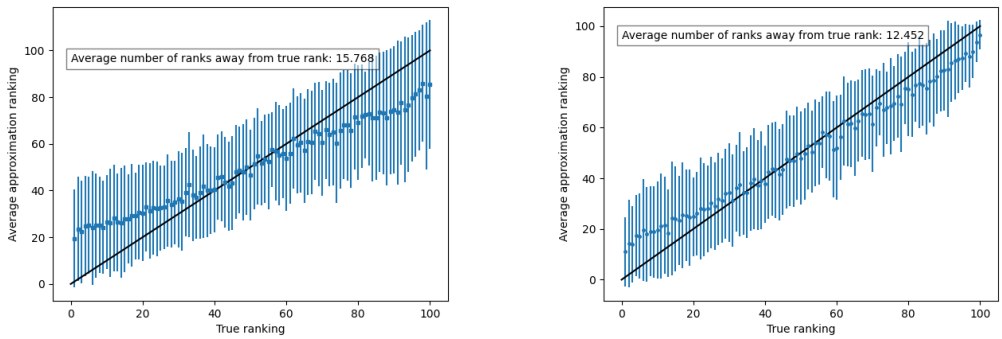
(a) Flowtree rankings. **(b)** L_1 embedding rankings.

Figure 6 Comparison of rankings generated by the flowtree and L_1 embedding approximations with the true rankings of the candidate diagrams using the ModelNet10.



(a) Modified flowtree rankings. **(b)** L_1 embedding rankings.

Figure 7 Comparison of rankings generated by the modified flowtree and L_1 embedding approximations with the true rankings of the candidate diagrams using synthetic-uniform dataset.



(a) Modified flowtree rankings. **(b)** L_1 embedding rankings.

Figure 8 Comparison of rankings generated by the modified flowtree and L_1 embedding approximations with the true rankings of the candidate diagrams using synthetic-Gaussian.