Computational Geometry Column 70: Processing Persistence Diagrams as Purely Geometric Objects

Binhai Zhu*

Abstract

In this column, we review the most recent results on processing persistence diagrams as purely geometric objects. (Yes, this is not a typo! While persistence diagrams originally come from computational topology, this column will mainly focus on the geometric parts and the minimal amount of knowledge on computational topology is needed to read this column.) We mainly focus on computing a center persistence diagram under the bottleneck and Wasserstein distances, and also the approximate nearest neighbor query under the bottleneck distance. At the end, we will go over a few open problems as well as some direction for future research, e.g., (approximate) farthest neighbor query under the bottleneck distance.

Keywords: Persistence diagrams, NP-hardness, Bottleneck distance, Wasserstein distance, Approximation algorithm, Nearest neighbor search, Locality-sensitive hashing.

1 Introduction

Persistence diagrams, one of the important tools in computational topology [9, 10, 28], have found various applications in the last decade. For instance, persistent diagrams have been used in geographical information systems [1], in neural science [14], in wireless networks [23], and in prostate cancer research [22], to name a few.

If we somehow ignore the topological meaning of the persistence diagrams, to some extent, each of them is really a set of 2D (topological feature) points above and inclusive of the line $y = x$ in the X-Y plane. More precisely, a topological feature point (b, d) in a persistence diagram, which we will simply call a point henceforth, indicates a topological feature which is born at time b and dies at time d. Hence it is always the case that $b \leq d$. See Figure 1 for an example. Note that a point $q = (t, t)$ on the line $y = x$ can be thought of as a dummy feature point which dies immediately after it is born. Let ℓ be (the set of all points on) the line $y = x$. We use $\mathcal{P} = P \cup \ell$ to represent a persistent diagram throughout this paper, where P is a finite set of 2D points above ℓ. Understanding that each persistent diagram contains ℓ, we simply say that $\mathcal{P}$ has $|P|$ (discrete) points. (In the topological language, points in P are counted with finite multiplicity, e.g., a point appearing three times in P will be counted as three different points, while points on ℓ are counted with infinite multiplicity.)

Then, how to measure the distance between two persistence diagrams $\mathcal{P}$ and $\mathcal{Q}$? Well, it turns out that this question was well-answered: either the bottleneck [4] or the Wasserstein distance [5]; in fact, both with decent stability bounds! These distances are defined formally in the following paragraph. Throughout this paper, for two points $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$, we use $d_p(p_1, p_2)$ to represent the L_p distance between p_1 and p_2, which is $d_p(p_1, p_2) = (|x_1 - x_2|^p + |y_1 - y_2|^p)^{1/p}$,

*Gianforte School of Computing, Montana State University, Bozeman, MT 59717-3880, USA. E-mail: `bhz@montana.edu`

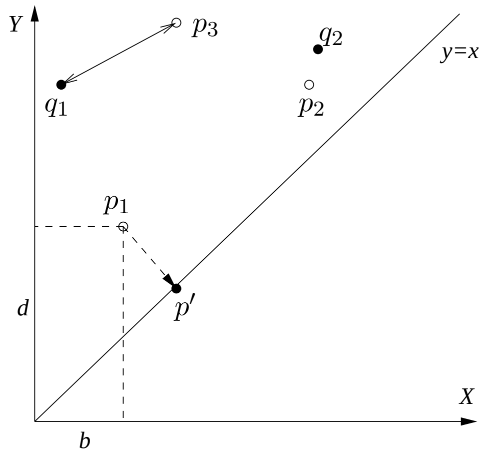

Figure 1: **Two persistence diagrams $\mathcal{P}$ and $\mathcal{Q}$, with feature point sets $P = \{p_1, p_2, p_3\}$ and $Q = \{q_1, q_2\}$. A point $p_1 = (b, d)$ indicates its birth time b and death time d. The projection of p_1 on ℓ (or $y = x$) gives p'. In this case, $d_B(\mathcal{P}, \mathcal{Q}) = d_\infty(p_3, q_1)$.**

for $p < \infty$. When $p = \infty$, $d_\infty(p_1, p_2) = \|p_1 - p_2\|_\infty = \max\{|x_1 - x_2|, |y_1 - y_2|\}$. We will mainly focus on the L_∞ metric.

Given two persistent diagrams $\mathcal{P}$ and $\mathcal{Q}$, each with at most n points, the *bottleneck distance* between them is defined as follows:

$$d_B(\mathcal{P}, \mathcal{Q}) = \inf_{\phi}\{\sup_{x\in\mathcal{P}} \|x - \phi(x)\|_\infty, \phi : \mathcal{P} \to \mathcal{Q} \text{ is a bijection}\}.$$

Similarly, the *p-Wasserstein* distance is defined as

$$W_p(\mathcal{P}, \mathcal{Q}) = \left(\inf_{\phi} \sum_{x\in\mathcal{P}} \|x - \phi(x)\|_\infty^p\right)^{1/p}, \phi : \mathcal{P} \to \mathcal{Q} \text{ is a bijection.}$$

We refer the readers to [9] for further information regarding persistence diagrams. It is worth mention that the Wasserstein distance is coined after Leonid Wasserstein (Leonid Vasserstein in Russian) for the distance between probability distributions [32]. In fact, similar notion of distance between distributions was known as early as in 1781 [20, 25], for the commonly known Monge-Kantorovich transportation problem; also as the *Earth Mover's Distance* nowadays [29].

It looks like these distances are both continuous. However, Edelsbrunner and Harer obtained a beautiful way to discretize them [9]. In the case of the bottleneck distance, this is done through the traditional geometric bottleneck matching (which runs in $O(n^{1.5} \log n)$ time for two point sets both with size n [11]), In fact, it was shown that the multiplicity property of the line $y = x$ can be used to compute the bottleneck matching between two diagrams $\mathcal{P}$ and $\mathcal{Q}$ more conveniently — regardless of their sizes [9]. This can be done as follows. Recall that P and Q are the set of feature points in $\mathcal{P}$ and $\mathcal{Q}$ respectively. Then project points in P (resp. Q) perpendicularly on $y = x$ to have P' (resp. Q'). (See also Figure 1.) It was shown that the bottleneck distance between two diagrams $\mathcal{P}$ and $\mathcal{Q}$ is exactly equal to the bottleneck (bipartite) matching distance, in the L_∞ metric, between $P \cup Q'$ and $Q \cup P'$. Here the weight or cost of an edge $c(u, v)$, with $u \in Q'$ and $v \in P'$, is set to zero; while $c(u, v) = \|u - v\|_\infty$, if $u \in P$ or $v \in Q$. The p-Wasserstein distance can be computed similarly, using a min-sum bipartite matching between $P \cup Q'$ and $Q \cup P'$, with all edge costs raised to c^p. (Kerber, et al. showed that in practice several steps of the bottleneck matching algorithm can be further simplified [21].)

Coming back to reality, as persistence diagrams are used more and more often in applications, a consequence is that practitioners gradually have a database of persistence diagrams computed and collected over certain period of time. It is not uncommon these days that such a database has tens of thousands of persistence diagrams, each with up to several thousands of points. How to process and search these diagrams becomes a new challenge for algorithm designers. In the following two sections, we review two such recent progress, i.e., computing a center persistence diagram under both the bottleneck and Wasserstein distances, and answering (approximate) nearest neighbor queries under the bottleneck distance. We conclude the review in Section 4 with several open problems.

2 Center Persistence Diagram

2.1 Hardness

It is known that prostate cancer develops slowly, and one important part is to determine how the cancer progresses over a long period of time. In [22], the method is to use a persistence diagram for each of the histopathology images (taken over certain period of time, hence image alignment is not feasible). Naturally, for the data collected over some time interval, one could consider representing a corresponding set of persistence diagrams with a center, which could be considered as a *median* persistence diagram summarizing these persistence diagrams. A sequence of such centers over a longer time period would give doctors a rough estimate on how the prostate cancer progresses — the more accurate estimate uses machine learning and is beyond this review. Hence, computing a center persistence diagram is practically meaningful. On the other hand, while the traditional center concept (and the corresponding algorithms) has been used for planar point sets (under the Euclidean distance) [27] and for binary strings (under the Hamming distance) [24]; recently we have also seen its applications in more complex objects, like polygonal chains (under the discrete Frechet distance) [2, 19]. We formally define the problem as follows.

Definition 1. The Center Persistence Diagram Problem under the Bottleneck Distance (CPD-B)

Instance: *A set of m persistence diagrams $\mathcal{P}_1, ..., \mathcal{P}_m$ with the corresponding feature point sets $P_1, ..., P_m$ respectively, and a real value r.*

Question: *Is there a persistence diagram $\mathcal{Q}$ such that $\max_i d_B(\mathcal{Q}, \mathcal{P}_i) \leq r$?*

Recall that $\mathcal{Q} = Q \cup \ell$, where ℓ is the line $y = x$. Note that we could have three versions, depending on how points in Q are selected. If points in Q could be arbitrarily selected, we call the version *continuous*. Otherwise, if points in Q can be selected from P_i's we call the version *discrete*. For the discrete version, we further have two sub-versions: *With Replacement* (i.e., points in Q can be selected with replacement from the set $\cup_{i=1..m} P_i$) and *Without Replacement* (i.e., points in Q can be selected with no replacement from the multiset $\cup_{i=1..m} P_i$). When $m = 2$, all the three versions can be solved in polynomial time with the maximum-flow-based method [26]. Henceforth, we focus on $m \geq 3$. Moreover, since the details for the discrete versions have been covered in [18], we focus more here on the continuous version here. (Note that generic ideas are similar, but details need some twist).

It should be noted that when $m = 3$, this problem is very similar to the NP-complete geometric three-dimensional assignment problem [15, 30], which is to partition three colored and equal-sized point sets into triangles such that the vertices in each triangle are in distinct colors and the maximum perimeter of these triangle is minimized. (Henceforth, we call such a triangle, whose vertices are in different colors, a *cluster*. Apparently, in our problem, the objective is more on minimizing the L_∞ radius of the maximum circumscribing circles of the corresponding clusters.) In all these problems, the reduction is from Planar 3-D Matching (Planar 3DM), which was proved to be NP-complete by Dyer and Frieze [8]. In 3DM, we are given three sets of elements E_1, E_2, E_3 (with $|E_1| = |E_2| = |E_3| = q$) and a set $\mathcal{T}$ of n triples, where $T \in \mathcal{T}$ implies that $T = (t_1, t_2, t_3)$ with $t_i \in E_i$. The problem is to decide whether there is a set S of q triples such that each element in E_i appears exactly once in (the triples of) S [13]. The Planar 3DM incurs an additional constraint: if we embed elements and triples as points in the plane such that there is an edge between an element a and a triple T iff a appears in T, then the resulting graph is planar. An example for Planar 3DM is as follows: $E_1 = \{1, 2\}, E_2 = \{a, b\}, E_3 = \{x, y\}$, and $\mathcal{T} = \{(1, a, x), (2, b, x), (2, b, y), (1, b, y)\}$. The solution is $S = \{(1, a, x), (2, b, y)\}$.

With the nature of bottleneck and Wasserstein distances, where the L_∞ norm is used at the bottom-most level, for the NP-hardness proof the requirement is more demanding here; in fact, we need to make sure that each cluster degenerates to either a horizontal or vertical line segment [18]. Between the discrete and continuous versions for CPD-B we also need some minor twist.

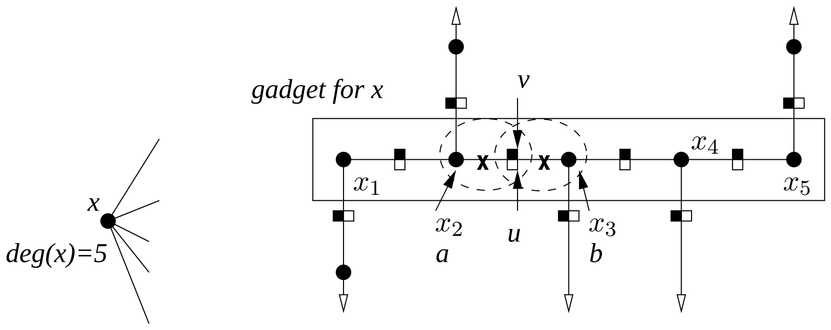

Figure 2: **The gadget for element x. In general, there could be a path of $\bullet$ nodes between x_2 and x_3, hence a and b do not have to be in the form of x_i's (in this figure they do).**

Let an instance for Planar 3DM be given, together with a corresponding planar graph G with $O(n)$ vertices, we first convert it to a planar graph with degree at most 3. This can be done as follows: let an element node x in G be with degree greater than 3 (i.e., $deg(x) = d > 3$), replace x with a path of d nodes $x_1, ..., x_d$, and replace the connection between x and a triple node T by a connection from x_i to T for some i (hence, after this operation, each x_i has degree at most 3, see also Figure 2). We have a resulting planar graph $G' = (V(G'), E(G'))$ with degree at most 3 and with $O(n)$ vertices. Then we construct a rectilinear embedding of G' on a regular rectilinear grid with a unit grid length, where each vertex $u \in V(G')$ is embedded at a grid point and an edge $(u, v) \in E(G')$ is embedded as an intersection-free path between u and v on the grid. Valiant showed that such an embedding can be computed in $O(n^2)$ time [31].

As shown in Figures 2,3 and 4, we use $\bullet$, $\square$ and $\blacksquare$ to denote points in color-1, 2 and 3 respectively. In Figure 2, the (element) gadget for element x is a path of $\bullet$ points. For each valid grid edge connecting two $\bullet$ points (say a and b), we put a color-2 (i.e., $\square$) point u and a color-3 (i.e., $\blacksquare$) point v at the mid-point of edge (a, b). Hence, to form a local optimal cluster with radius 1/4, we either use cluster $\{a, u, v\}$ or $\{b, u, v\}$, whose centers are marked with a cross (i.e. $\times$) in Figures 2-4. This operation is performed on color-2 and color-3 grid edges as well. A *triple gadget* for a triple $T = (x, y, z)$ is constructed by using three points in color-i, $i = 1..3$, at a grid point where the corresponding element gadgets for x, y and z would reach (see Figures 3-4).

In each element gadget for x, we have $3D + 1$ grid points, where D is the number of grid points on the path from x_1 to x_d in the element gadget. We then have the following lemma [18].

Lemma 1. *In an element gadget for x, exactly one x_i is covered by a disk of radius 1/4, centered at a point out of the gadget.*

When x_i is covered by a disk of radius 1/4 centered at a point out of the gadget x, we also say that x_i is *pulled out* of x. (This 'pull-out' concept is similarly defined for a triple gadget.) The center of such a cluster refers to the point marked by $\times$, for example, above x_2 in Figure 3 or below x_3 in Figure 4. We then have the following lemma (adapted from [18], referring Figure 3 and Figure 4).

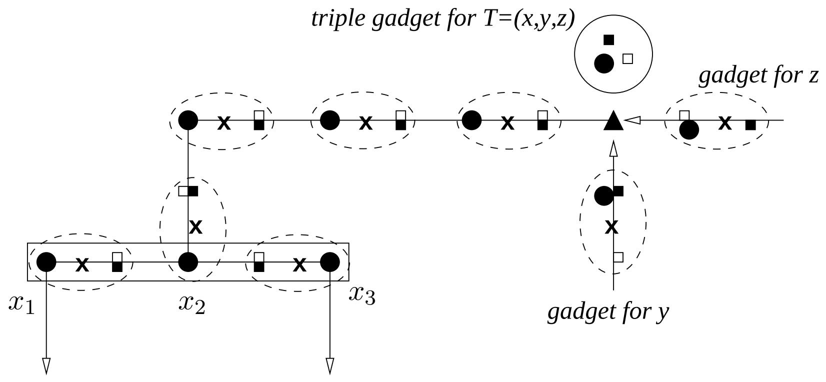

Figure 3: **The triple gadget for $T = \langle x, y, x\rangle$ (represented as ▲, which is really putting three element points on a grid point). In this case the triple $\langle x, y, z\rangle$ is selected in the final solution (assuming operations are similarly performed on y, z). Exactly one of x_i (in this case x_2) is pulled out of the gadget for the element x.**

Lemma 2. *In a triple gadget for $T = (x, y, z)$, to cover the three points representing T using disks of radii at most 1/4, either all the three points are pulled out of the triple gadget T by the three element gadgets respectively, or none is pulled out. In the latter case, these three points can be covered by a disk of radius zero.*

Theorem 1. *The continuous Center Persistence Diagram problem under the bottleneck distance (CPD-B) is NP-hard when $m = 3$ diagrams are given.*

Proof. We obtain our reduction from Planar 3DM [8], which has just been described and can be done in $O(n^2)$ time (with $3Kn$ points used in all three colors, where $K = O(n^2)$). We state the following statement without giving the detailed arguments: Planar 3DM has a solution of q triples if and only if the converted instance with $3Kn$ points can be partitioned into Kn clusters each covered by a disk of radius 1/4; moreover, exactly q clusters are covered by disks of radii zero. Readers can either try it as an exercise, or refer to a similar argument in [18].

To finish the proof for persistence diagrams, we need to move the converted $3Kn$ points far away from $y = x$ (so that the optimal bottleneck distance is not determined by any projected point on $y = x$). Let the diameter of the (union of the) three constructed point sets of different colors (with size $3Kn$) be $\hat{D}$, we translate these points as a whole set rigidly such that all the points are at least $2\hat{D}$ distance away from $y = x$. We then have three persistence diagrams, one for each color. This concludes the proof. $\square$

Note that the above reduction still works if we amplify the radius of the maximum covering disks to below 1/2. Hence, we have the following corollary.

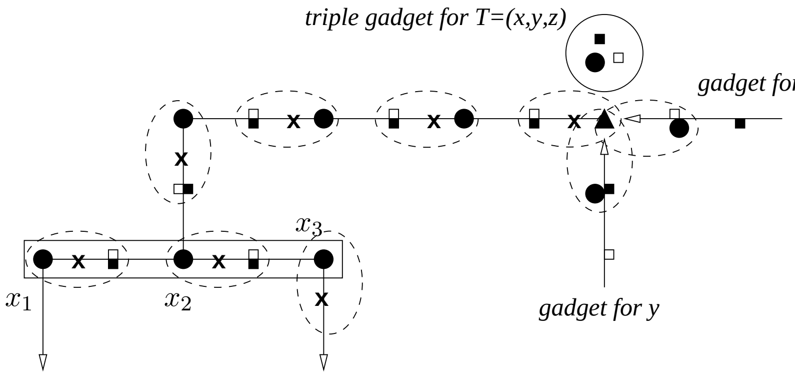

Figure 4: **The triple gadget for $T = \langle x, y, x \rangle$ (represented as ▲, which is really putting three element points on a grid point). In this case the triple $\langle x, y, z \rangle$ would not be selected in the final solution. Note that the black round point in the triple gadget is pulled out by the element x, and the other two points are pulled out similarly by the element y and z.**

Corollary 1. *It is NP-hard to approximate (the optimization version of) Center Persistence Diagram problem under the bottleneck distance for $m \geq 3$ diagrams within a factor $2 - \varepsilon$, for some $\varepsilon > 0$.*

We comment that the above reduction also works for the continuous version of the Center Persistence Diagram problem under the Wasserstein distance, which is defined as follows.

Definition 2. The Center Persistence Diagram Problem under the Wasserstein Distance (CPD-W)

Instance: *A set of m persistence diagrams $\mathcal{P}_1, ..., \mathcal{P}_m$ with the corresponding feature point sets $P_1, ..., P_m$ respectively, and a real value r.*

Question: *Is there a persistence diagram $\mathcal{Q}$ such that* $\max_i W_p(\mathcal{Q}, \mathcal{P}_i) \leq r$*?*

Again, remember that $\mathcal{Q} = Q \cup \ell$, where ℓ is the line $y = x$. Moreover, CPD-W also has three versions as CPD-B. The above reduction does work for the continuous version for CPD-W, but for neither of the discrete versions. The main reason is that each cluster in the proof is either a vertical or a horizontal segment of length 1/2 and the continuous center (marked with $\times$ in Figures 2-4) has the same L_∞ distance of 1/4 to the corresponding three points forming the cluster. Hence the minimum of the maximum Wasserstein distance can be explicitly computed in the continuous case. We summarize the hardness result for CPD-W as follows.

Corollary 2. *It is NP-hard to solve the continuous Center Persistence Diagram problem under the Wasserstein distance for $m \geq 3$ diagrams.*

2.2 Approximation

For approximation algorithms, it is extremely simple. As both d_B and W_p satisfy the triangle inequality, we can simply pick any of the m input diagrams $\mathcal{P}_i, i = 1..m$, as the center — assuming points in $\mathcal{P}_i$ is colored with color-i. (This is similar to [15], even though the objective function there is different.) The difference here is that $\mathcal{P}_i$ could have different sizes, but this can be handled using a generalized projection method. Suppose we select $\mathcal{P}_i$ as our approximate center, a point p of color-i is projected $m-1$ times onto $y = x$ to obtain $m-1$ points of different colors in the color set $\{1, ..., i-1, i+1, ..., m\}$. Moreover, when m is a constant, the running time of this algorithm would be $O(n^{1.5} \log n)$ using the bottleneck distance , where n is the maximum size of the m input persistence diagrams. However, when m is part of the input the running time of the algorithm increases to $O((mn)^{1.5} \log(mn))$ using the bottleneck distance. With the p-Wasserstein distance, of course, the traditional Hungarian method would be used for computing the minimum weight matching, leading to a higher polynomial running time.

Now we finish the analysis of approximation factor for the continuous CPD-W problem. Let $\mathcal{Q}$ be the optimal solution. Then

$$W_p(\mathcal{P}_i, \mathcal{P}_j) \leq W_p(\mathcal{P}_i, \mathcal{Q}) + W_p(\mathcal{Q}, \mathcal{P}_j) \leq 2 \cdot Opt.$$

It is in fact easy to improve this solution practically. For instance, when $m = 3$, for each cluster $\{p_i, p_j, p_k\}$, instead of using p_i as the local center, we could use the Voronoi vertex q, which is the center of the smallest enclosing L_∞-circle for this cluster, as the center for this cluster. But it is unknown yet whether this would give us a better approximation factor.

We summarize our results in the following table.

Table 1: Results for the Center Persistence Diagram problems under the bottleneck (d_B) and Wasserstein (W_p) distances when $m \geq 3$ diagrams are given.

	Hardness	Inapproximability bound	Approximation factor
d_B, with no replacement	NP-complete	$2 - \varepsilon$	2
d_B, with replacement	NP-complete	$2 - \varepsilon$	2
d_B, continuous	NP-hard	$2 - \varepsilon$	2
W_p, with no replacement	?	?	2
W_p, with replacement	?	?	2
W_p, continuous	NP-hard	?	2

3 Approximate Nearest Neighbor Query

In this section, we consider the following problem: given a set of persistent diagrams (PDs) $\Gamma = \{\mathcal{P}_i | i = 1..n\}$, preprocess them with a data structure such that for a query persistent diagram $\mathcal{Q}$ one can return an approximate nearest neighbor (diagram) in Γ from $\mathcal{Q}$ under the bottleneck distance. Recall that $\mathcal{P}_i = P_i \cup \ell$, where ℓ is the line $y = x$. Throughout this section, we assume $\max_i |P_i| = m$ and for any point (a, b) in P_i we have $\max\{|a|, |b|\} \leq M$. In [12], these diagrams are called (M, m)-*bounded*. We assume that $\mathcal{Q}$ is also (M, m)-*bounded*.

Clearly, a brute-force searching over all $\mathcal{P}_i$'s would take $O(nm^{1.5} \log m)$ time, using the geometric bottleneck matching algorithm by Efrat, Itai and Katz [11]. Since n is usually large (sometimes m too, though we could try to reduce m by some preprocessing, like deleting all the noisy features close to ℓ in all the persistence diagrams), this method is not feasible in practice. A second thought could be treating each persistence diagram as a high-dimensional data point and use some standard method like divide-and-conquer. In fact, Clarkson proved that the divide-and-conquer algorithm in metric spaces runs in a time that is exponential in the doubling dimension [3]. (Roughly, in a metric space X with metric d, if the open ball $B(x, r)$, i.e., a ball with center x and radius r respectively, can be covered by at most 2^C balls of radius $r/2$ then the doubling dimension of X is C.) Unfortunately, as shown in (the most recent version of) [12], in the metric space of persistence diagrams under the bottleneck metric, the doubling dimension is infinite. Hence, this method is again not feasible.

Yet another method is to use *locality-sensitive hashing* [17]. In fact, Driemel and Silvestri applied the method on polygonal chains under the discrete Frechet distance [7]. (Roughly, one stores the perturbations of all the vertices in the database of chains, C_i's, such that the perturbation of the query chain C_Q would match with some C_i with a small probability, if the discrete Frechet distance $d_F(C_i, C_Q)$ is small. Then, if one runs this query many times eventually one would get a hit with a high probability.) By adapting the method in [7], we can obtain the following bounds: with $O(2^{12m} n \log n + nm)$ space, given any query PD $\mathcal{Q}$ with at most m feature points, a factor-$8\sqrt{2}$ approximate PD $\mathcal{P}_i$ under the bottleneck distance can be returned in $O(2^{12m} m \log n + 2^{12m} \log^2 n)$ time. However, in this method the key used would be coming from both $\mathcal{P}_i$ and $\mathcal{Q}$ — following the way to compute the bottleneck distance [9], hence it is hard to speed up the search by preprocessing the keys offline before the search starts (let alone building a hierarchical data structure). This drawback was handled as follows [12].

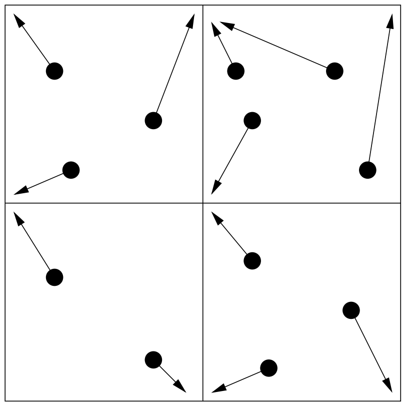

Figure 5: **Snap-rounding a set of feature points to an integer grid. Multiplicity of duplicated grid points must be counted. For this case, the key (reading row by row) is: $\langle 1, 3, 1, 2, 2, 0, 0, 2, 1 \rangle$.**

In [12], a grid-based deterministic hashing (or snap-rounding) method was used. We assume all feature points and grid points are within the box $[-M, M]^2$, note that $-M$ is included as topologically the birth and death times could be negative. (A similar method using such a grid was used by Heffernan and Schirra for point set matching [16]. But in that method multiplicities were ignored.) The idea is as follows:

- At each level $j \geq 0$, an integer grid $G[j]$ is generated where the (base) cell has a length $r_j = 2M/2^j$, and the number of cells is 2^{j+1}.
- At each level j, for each point p in P_i, snap-round p to each of the 4 cell points where p is located (see Figure 5). A vector obtained through this snap-rounding for all the points in P_i gives a key $k_j(P_i)$ for PD $\mathcal{P}_i$. (We here ignore the boundary case when p is on a grid line.)

- At each level j, for each point p in P_i that is at L_∞ distance at most r_j from the diagonal ℓ, additionally allow p to be optionally deleted. (For each of these points we have 5 choices: 4 snap-roundings plus an optional deletion.)

- At each level j, store all the $O(5^m)$ keys generated for each PD.

Let $DS(\mathcal{P}_i, G[j])$ be the set of all snap-roundings of $\mathcal{P}_i$ obtained by additionally allowing $p \in P_i$ within distance at most r_j from the diagonal ℓ to be optionally deleted. Then for $\mathcal{P}_{i'} \in DS(\mathcal{P}_i, G[j])$, we use $k_j(P_{i'})$ to denote the key associated with $\mathcal{P}_{i'}$ at level j.

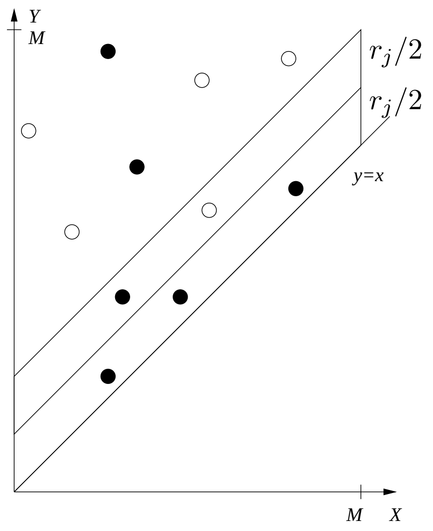

Figure 6: **Handling points close to the diagonal ℓ (or $y = x$) at level-j. Note that if Q is the set of black points, then all the 3 black points in the first $r_j/2$ band above ℓ would be deleted while the only white point in the second band, which is in P_i, has an additional choice of being deleted, besides being snap-rounded to the 4 grid points of the cell containing it.**

For the query diagram $\mathcal{Q}$, a similar (but non-identical!) operation is performed as follows:

- For the query PD $\mathcal{Q}$, at level j, first delete all the points within L_∞ distance at most $r_j/2$ from the diagonal ℓ, let the resulting PD be $\widetilde{\mathcal{Q}}_j$ (we drop the index j sometimes, whenever a specified level is given in the context). Then snap-round each point q in $\widetilde{Q}$ to the nearest grid point, obtain the key $k_j(\widetilde{Q})$ at level-j.

- The searching is done by finding a match between $k_j(\widetilde{Q})$ and $k_j(P_{i'})$, also commonly known as a *collision*[1], in the stored databases of keys.

We state the following central theorem as follows:

Theorem 2. *On grid $G[j]$, after optional deletions on P_i's and mandatory deletions on Q are performed as described in the previous paragraphs. We have*

1. *If there is a $\mathcal{P}_{i'} \in DS(\mathcal{P}_i, G[j])$ such that one of its keys collides with that of $\widetilde{\mathcal{Q}}$, i.e., $k_j(P_{i'}) = k_j(\widetilde{Q})$, then $d_B(\mathcal{P}_i, \mathcal{Q}) \le \frac{3}{2} r_j$.*

[1]We slightly abuse the notation by also saying that the PD's $\mathcal{P}_i$ and $\mathcal{Q}$ *collide* at that level.

2. If $d_B(\mathcal{P}_i, \mathcal{Q}) \leq \frac{1}{2}r_j$, then there is a $\mathcal{P}_{i'} \in DS(\mathcal{P}_i, G[j])$ such that $k_j(P_{i'}) = k_j(\widetilde{Q})$.

This theorem enables us to design a hierarchical data structure Δ_j associated with $G[j]$, for $j = 0, 1, 2, \cdots$, where at each level we store the sorted list of keys and for each key the corresponding subset of PD's. To find a near neighbor of $\mathcal{Q}$ in Γ, we search up to the last level j' where the key for $\mathcal{Q}$ collides with a key k'_j stored at level j' (i.e., the key of $\mathcal{Q}$ has no collision with any key stored in Δ_{j+1}). Any PD associated with k'_j can be returned as an approximate nearest neighbor. The following can be proved and we refer interested readers to [12] for the formal details.

1. (**Hierarchical Collision Lemma**): If a PD $\mathcal{P}_i$ collides with the query PD $\mathcal{Q}$ at level j', then they collide at level j for every $j < j'$.

2. (**Nearest Neighbor Bin Lemma**): Let the PD $\mathcal{P}^* \in \Gamma$ be the nearest neighbor to $\mathcal{Q}$, realizing $\min_{i=1..n} d_B(\mathcal{P}_i, \mathcal{Q})$. Let j be the largest index such that $k_j(\widetilde{Q})$ collides some key in Δ_j. Then there is a snap-rounding $P^*_{j-2} \in DS(\mathcal{P}^*, G[j-2])$ such that $k_{j-2}(P^*_{j-2}) = k_{j-2}(\widetilde{Q}_{j-2})$.

3. (**Nearest Neighbor Approximation Lemma**): If j is the largest index such that $k_j(P_{i'}) = k_j(\widetilde{Q})$, then any $\mathcal{P}_i \in \Gamma$ provides a 6-approximation for $\min_{i=1..n} d_B(\mathcal{P}_i, \mathcal{Q})$, provided that $\mathcal{P}_{i'} \in DS(\mathcal{P}_i, G[j])$ and $k_j(P_{i'})$ is the key for $\mathcal{P}_{i'}$ at level-j.

Note that the Hierarchical Collision Lemma further enables us to run a binary search over $\Delta_0, \Delta_1, \cdots, \Delta_\tau$. (This is not possible if we used an adapted version of Driemel and Silvestri [7].) Also note that the Nearest Neighbor Lemma says that we might not be able to find the true nearest neighbor at Δ_j, but we can find it by a linear search in Δ_j, Δ_{j-1} and Δ_{j-2}. In addition, with some twist, i.e., by identifying the largest level j satisfying the condition that at least $k > 1$ different diagrams with keys colliding with that of $\mathcal{Q}$ at that level, any diagram in Δ_j provides a factor-24 approximation for the kth-nearest neighbor [12]. (Symmetrically, the actual kth-nearest neighbor can be found by a linear search in $\Delta_{j+2}, \Delta_{j+1}, \Delta_j$, Δ_{j-1} and Δ_{j-2}.)

We summarize the main theorem as follows:

Theorem 3. *Given a set of (M, m)-bounded persistence diagrams $\Gamma = \{\mathcal{P}_i | i = 1..n\}$, a τ-level data structure of size $O(n5^m\tau)$ can be constructed such that, for any query diagram $\mathcal{Q}$ (which is also (M, m)-bounded), it can find a 6-approximation of $\min_{i=1..n} d_B(\mathcal{P}_i, \mathcal{Q})$ in $O((m \log n + m^2) \log \tau)$ time.*

We comment that while this (space,query)-bound is significantly better than directly using the method in [7], the space complexity is still exponential in m. In [12], several ideas were proposed to improve the space complexity, but further research is still needed.

4 Open Problems

We close this review with several open problems.

1. Is the discrete CPD-W problem NP-hard when $m \geq 3$? More precisely, under the p-Wasserstein distance, is the discrete version of the Center Persistence Diagram problem NP-hard with at least 3 input diagrams (for either of the 'With Replacement' or 'Without Replacement' cases)?

2. Is it possible to obtain some approximation lower bound (e.g., at least $1 + \varepsilon$) for all the three versions of the CPD-W problem? Note that a related open problem for three-dimensional assignment problem where the objective function is in a min-sum manner was posed by Custic, Klinz and Woeginger a few years ago [6].

3. Is it possible to improve the 2-approximation upper bound for all the three versions of the CPD-W problem?

4. For the approximate nearest neighbor query problem, is it possible to reduce the space complexity to polynomial while still maintaining a constant factor approximation?

5. In some applications, we might need to find persistence diagrams which are far away from a query diagram. Is it possible to design an efficient data structure to answer (approximate) farthest neighbor queries?

Acknowledgments

I thank my collaborators for this series of research: Brittany Fasy, Xiaozhou He, Yuya Higashikawa, Naoki Katoh, Guohui Lin, Zhihui Liu, Samuel Micka, David Millman, Eiji Miyano, Suguru Tamaki, and Junichi Teruyama. Thanks to Adrian Dumitrescu and Wenfeng Lai as well for several helpful comments.

References

[1] M. Ahmed, B. Fasy and C. Wenk. Local persistent homology based distance between maps. In *Proc. 22nd ACM SIGSPATIAL International Conference on Advances in GIS (SIGSPATIAL'14)*, pages 43-52, 2014.

[2] K. Buchin, A. Driemel, J. Gudmundsson, M. Horton, I. Kostitsyna, M. Loffler and M. Struijs. Approximating (k,l)-center clustering for curves. In *Proc. 30th ACM-SIAM Symp. on Discrete Algorithms (SODA'19)*, pages 2922-2938, 2019.

[3] K. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, MIT Press, pages 15-59, 2006.

[4] D. Cohen-Steiner, H. Edelsbrunner and J. Harer. Stability of persistence diagrams. *Disc. and Comp. Geom.*, **37**:103-120, 2007.

[5] D. Cohen-Steiner, H. Edelsbrunner, J. Harer and Y. Mileyko. Lipschitz functions have L_p-stable persistence. *Found. Comput. Math.*, **10**:127-139, 2010.

[6] A. Custic, B. Klinz and G. Woeginger. Geometric versions of the three-dimensional assignment problem under general norms. *Discrete Optimization*, 18:38-55, 2015.

[7] A. Driemel and F. Silvestri. Locality-sensitive hashing of curves. *Proceedings of the 33rd International Symposium on Computational Geometry (SoCG'17)*, pp. 37:1-37:16, 2017.

[8] M. Dyer and A. Frieze. Planar 3DM is NP-complete. *J. Algorithms*, 7:174-184, 1986.

[9] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction.* American Mathematical Soc., 2010.

[10] H. Edelsbrunner, D. Letscher and A. Zomorodian. Topological persistence and simplification. *Disc. and Comp. Geom.*, 28:511-513, 2002.

[11] A. Efrat, A. Itai and M. Katz. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 31(1):1-28, 2001.

[12] B. Fasy, X. He, Z. Liu, S. Micka, D. Millman and B. Zhu. Approximate nearest neighbors in the space of persistence diagrams. *CoRR abs/1812.11257*, Dec, 2018.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[14] C. Giusti, E. Pastalkova, C. Curto and V. Itskov. Clique topology reveals intrinsic geometric structure in neural correlations. *Proceedings of the National Academy of Sciences*, 112(44):13455-13460, 2015.

[15] D. Goossens, S. Polyakovskiy, F. Spieksma and G. Woeginger. The approximability of three-dimensional assignment problems with bottleneck objective. *Optim. Lett.*, 4:7-16, 2010.

[16] P. Heffernan and S. Schirra. Approximate decision algorithms for point set congruence. *Computational Geometry: Theory and Applications*, 4(3):137-156, 1994.

[17] S. Har-Peled, P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321-350, 2012.

[18] Y. Higashikawa, Naoki Katoh, G. Lin, E. Miyano, S. Tamaki, J. Teruyama and B. Zhu. On computing a center persistence diagram. *CoRR abs/1910.01753*, Oct, 2019.

[19] P. Indyk. Approximate nearest neighbor algorithms for Frechet distance via product metrics. In *Proceedings of the 18th Annual ACM Symposium on Computational Geometry (SoCG'02)*, pp. 102-106, 2002.

[20] L. Kantorovich. On the translation of masses. *C.R. (Dokl.) Acad. Sci. URSS*, **37**:1099-226, 1992.

[21] M. Kerber, D. Morozov and A. Nigmetov. Geometry helps to compare persistence diagrams. In *Proceedings of the 18th Workshop on Algorithm Engineering and Experiments (ALENEX'16)*, pp. 103-112, SIAM, 2016.

[22] P. Lawson, J. Schupbach, B. Fasy and J. Sheppard. Persistent homology for the automatic classification of prostate cancer aggressiveness in histopathology images. In *Proc. Medical Imaging: Digital Pathology 2019*, paper 109560G, 2019.

[23] N-K. Le, P. Martins, L. Decreusefond and A. Vergne. Simplicial homology based energy saving algorithms for wireless networks. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 166-172, 2015.

[24] M. Li, B. Ma and L. Wang. On the closest string and substring problems. *J. ACM*, **49**:157-171, 2002.

[25] G. Monge. Mémoire sur la théorie des déblais et des remblais. In *Histoire de l'Académie Royale des Sciences des Paris*, pp. 666-704, 1781.

[26] V.M. Malhotra, M.P. Kumar and S.N. Maheshwari. An O($|V|^3$) algorithm for finding maximum flows in networks. *Info. Process. Lett.*, 7(6):277-278, 1978.

[27] S. Masuyama, T. Ibaraki and T. Hasegawa. The computational complexity of the m-center problems. *Transac. of IEICE.*, E64(2):77-64, 1981.

[28] K. Mischaikow, T. Kaczynski and M. Mrozek. *Computational Homology*, Applied Mathematical Sciences, 157, Springer, 2004.

[29] Y. Rubner, C. Tomasi and L. Guibas. The earth mover's distance as a metric for image retrieval. *Intl. J. of Computer Vision*, **40**:99-121, 2000.

[30] F. Spieksma and G. Woeginger. Geometric three-dimensional assignment problems. *European J. of Operation Research*, 91:611-618, 1996.

[31] L.G. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Computers*, 30(2):135-140, 1981.

[32] L. Wasserstein. Markov processes over denumerable products of spaces describing large systems of automata. *Prob. Inf. Transm.*, **5**:47-52, 1969.