LUMÁWIG: Un-bottling the bottleneck distance for zero dimensional persistence diagrams at scale

Paul Samuel P. Ignacio[*]
University of the Philippines Baguio
Baguio City, Philippines 2600

Jay-Anne B. Bulauan
University of the Philippines Baguio
Baguio City, Philippines 2600

David Uminsky
University of Chicago
Chicago, Il, United States 60637

Abstract

We present LUMÁWIG, a novel efficient algorithm to compute dimension zero bottleneck distance between two persistence diagrams of a specific kind which outperforms all other publicly available algorithm in runtime and accuracy. We bypass the overwhelming matching problem in previous implementations of the bottleneck distance, and prove that the zero dimensional bottleneck distance can be recovered from a very small number of matching cases. LUMÁWIG also generally enjoys linear complexity as shown by empirical tests. This allows us to scale TDA to data sets of sizes encountered in machine learning and utilize persistence diagrams in a manner that goes beyond the simple use of the most persistent components[2].

1 Introduction

Topological data analysis (TDA) has gathered significant interest from a wide range of researchers because of its novel approach and use of classical tools from algebraic topology for extracting descriptive features from data. Succinctly, topological data analysis captures and records the persistence (2; 3) of algebraically computable topological signatures, and regards it as a measure of significance for different features embedded in the structure of data. For the zero dimensional case, these signatures correspond to clusters within data that merge based on a filtration of the data points. One of the most common filtration used in practice is the *Rips* filtration where pairs of points are considered merged at a given filtration slice δ when the points are at most δ apart. Hence, as opposed to other filtrations that require additional parameter choices, the Rips filtration only depends on intrinsic distances between data points and reveals the underlying multi-scale connectivity information about natural clusters existing within data. The Rips filtration produces summaries of topological signatures all beginning at the start of the filtration, capturing cluster merging dynamics akin to that observed by hierarchical clustering methods (see Figure 1). This is the setting we will be working on.

The captured topological signatures are recorded in diagrams called *persistence diagrams*, which are a collection of points in the extended plane where the coordinates represent the birth and death times of the recorded features. In these diagrams, points that have multiplicity capture distinct features with the same birth-death profile, and points with infinite persistence capture perpetual features. For

[*]Correspondence: ppignacio@up.edu.ph
[2]An extended version of this paper appears in (1).

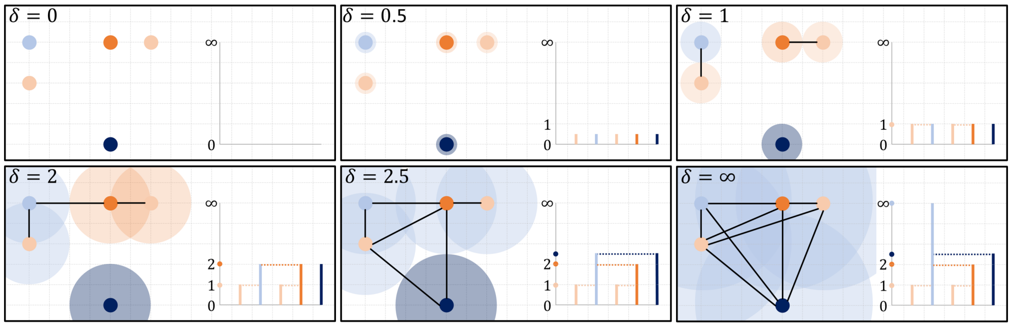


Figure 1: A Rips filtration over a point cloud captures the merging dynamics of clusters evolving from points across multiple scales. The dimension zero persistence diagram produced by this filtration is a set of points positioned along an extended vertical line at the merging heights in the corresponding dendrogram, except for the last point positioned at ∞ representing the eventual single component. The neighborhoods around points are colored by the persistent cluster determined by the elder rule.

diagrams induced by the Rips filtration, the sole constant perpetual feature appears in dimension 0, capturing the eventual single cluster that merges all components (see Figure 1).

Given two persistence diagrams X and Y, the *bottleneck distance* between them is defined as $d_B(X, Y) = \inf_\phi \sup_{x \in X} ||x - \phi(x)||_\infty$ where the infimum is taken over all bijections $\phi : X \sqcup \Delta \to Y \sqcup \Delta$ and Δ is the diagonal. In general terms, the bottleneck distance measures the cost to transform one diagram to another. The first, and for a long time the only, publicly available implementation of the bottleneck distance for persistence diagrams is in the library DIONYSUS, released by Morozov (4) in 2010. DIONYSUS uses a variant of the Hungarian algorithm (5) for the assignment problem.

In 2017, Morozov et al. (6) provided an improved implementation in the library HERA by exploiting geometry. Their approach follows closely the work of Efrat et al. (7). For the sets X_0 and Y_0 of orthogonal projections on the diagonal Δ of points respectively from X and Y, and the sets $U = X \cup Y_0$ and $V = X_0 \cup Y$, they consider the weighted complete bipartite graph $G = (U \sqcup V, U \times V, w)$ where $w : U \times V \to \mathbb{R}_{\geq 0}$ is given by $w(u, v) = ||u - v||_\infty$ if $u \in X$ or $v \in Y$ and $w(u, v) = 0$ otherwise. With this, the bottleneck computation problem can be recast in the following manner: if $G[r]$ is the subgraph of G having all edges e of weight $w(e) \leq r$, then the bottleneck distance of G is the minimal value r such that $G[r]$ contains a perfect matching. Hence the bottleneck distance can be recovered by combining a binary search on the edge weights of G with a test for a perfect matching. For the matching step, they augment the Hopcroft-Karp algorithm (8) by appealing to a near-neighbor data structure (a k-d tree) to search for the best candidate pair for a query point, pruning from the search the subtrees (and hence all other candidates within them) whose enclosing box is further away from the query than the current best candidate. This circumvents the overwhelming matching problem by significantly shrinking down the combination pool to retrieve the best matching. To approximate complexity, they fit curves of the form cn^α and found a best fit with $\alpha = 1.4$. This translates to speed-up from DIONYSUS already by a factor of 400 on diagrams with 2,800 points, and opened opportunities for several works that examine larger (9) or more complex (10; 11) data sets.

We take inspiration from this idea of exploiting the geometry of persistence diagrams to extract computational speed-up. By considering dimension 0 persistence diagrams induced from the Rips filtration, we can approach the problem via a different framework, birthing a new efficient algorithm for computing the bottleneck distance. The key idea is to begin with a specific initial bijection that one can methodically modify to optimize the norm between matched points. This process allows us to identify all possible instances where the bottleneck matching is achieved, and the exact value for the bottleneck distance, significantly bypassing the overwhelming matching step in previous implementations. We remark that while this strategy only works for persistence diagrams of a specific kind—those whose detected signatures all begin at the same time — this class is in no way less significant than diagrams induced from other filtrations. Moreover, in addition to diagrams induced from the above setting, this class also includes diagrams obtained from the output of any hierarchical clustering algorithm applied to point cloud data. Hence, the computational speed-up for the bottleneck distance we obtain benefits the comparison of these diagrams as well. Furthermore, we note that there are other metrics used in the literature to compare persistence diagrams, and we make no preference claim in favor of the bottleneck distance. In fact, it is a good question to ask whether the above strategy can be followed to generate computational speed-up for these metrics as well. We credit Katharine Turner for raising this question first in relation to the Wasserstein distance.

2 Bypassing matchings

We first note that for many practical applications to data analysis of 0-dimensional persistence diagrams, components are assumed to be born at the beginning of the filtration, hence all non-trivial points lie in the vertical axis. Hence, in this case, if δ_x and $\delta_{\phi(x)}$ are the death times respectively for the pair x and $\phi(x)$, then

$$||x - \phi(x)||_\infty = \begin{cases} \max(\delta_x, \delta_{\phi(x)})/2 & \text{if } \phi(x) \in \Delta \\ |\delta_x - \delta_{\phi(x)}| & \text{otherwise.} \end{cases} \tag{1}$$

This suggests that while it is natural to do a point-to-point matching between diagrams, there are cases when we are better off matching a point to the diagonal. For a point $x \in X$ and $\phi(x) \in Y$, this happens precisely when

$$\max(\delta_x, \delta_{\phi(x)}) > 2\min(\delta_{\phi(x)}, \delta_x). \tag{2}$$

Figure 5a in the supplementary materials illustrates this point. Therefore, unless (2) is satisfied, it is our priority to match a non-trivial point in a diagram X with a non-trivial point in another diagram. This supports the interpretation that the bottleneck distance is the cost of transforming one diagram to another.

We now present the LUMÁWIG algorithm. We first induce an ordering of the death times in both diagrams and define a bijection that we can methodically modify to optimize the norm between matched points and recover the desired matching that achieves the bottleneck distance. The proof of Lemma 1 (see supplementary material) provides the basic argument that allows to bypass the overwhelming matching problem. Lemma 2 proceeds in the same manner and identifies all other instances where the bottleneck matching is achieved, and the exact bottleneck distance in each case.

Let X and Y be two 0-dimensional persistence diagrams whose death times are arranged from largest to smallest. Without loss of generality, assume that X has at most as many points as Y has. We remark that this pre-processing is equivalent to considering the bijection ϕ that matches points between X and Y according to the relative ranking of death times from largest to smallest, and where unmatched points in Y are matched to the diagonal. Let $N = length(X)$ and define

$$Z = [z_i]_1^{length(Y)} \text{ where } z_i = \begin{cases} |x_i - y_i| & \text{if } i \le N \\ y_i/2 & \text{otherwise} \end{cases}$$

and $l = \arg\max(Z)$. We then have the following results.

Lemma 1. *If $N < length(Y)$ and $\max(Z) \le y_{N+1}/2$, then $d_B(X,Y) = y_{N+1}/2$ where y_{N+1} is the largest death time of a point in Y matched to the diagonal.*

Lemma 2. *Let ζ be the second largest entry of Z.*
1. If $\max(Z) \le \max(x_l, y_l)/2$, then $d_B(X,Y) = \max(Z)$.
2. If $\zeta < \max(x_l, y_l)/2 < \max(Z)$, then $d_B(X,Y) = \max(x_l, y_l)/2$.
3. If $\zeta \ge \max(x_l, y_l)/2$ and $m \ge l$ for every m such that $z_m \ge \max(x_l, y_l)/2$, then $d_B(X,Y) = \max(x_l, y_l)/2$.
4. If $\zeta \ge \max(x_l, y_l)/2$ and there exists $m < l$ such that $z_m \ge \max(x_l, y_l)/2$, then there exists a bijection τ between X and Y such that one of the three preceding cases holds and where $\max ||x - \tau(x)||_\infty < \max ||x - \phi(x)||_\infty$.

The two Lemmas above provide the theoretical basis for our bypass strategy. Together, they take advantage of the specific form of dimension zero persistence diagrams being considered, and the methodical approach to optimize norms induced by a specific matching. The complete pseudo code for the algorithm is provided as supplementary material.

3 Benchmarking LUMÁWIG against the state-of-the-art

We benchmark LUMÁWIG against the current state-of-the-art implementation of the bottleneck distance in HERA. Specific details about this benchmarking are provided as supplementary material.

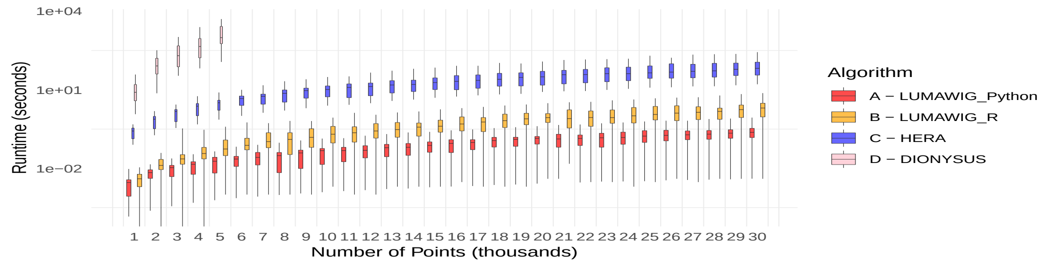
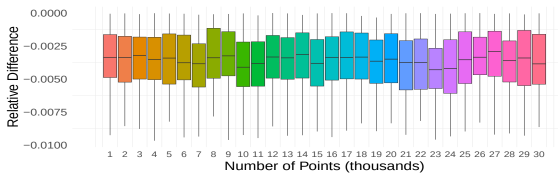


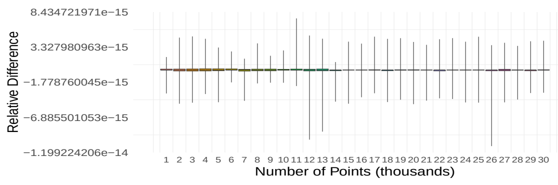
Figure 2: Running time (seconds in $\log$ scale) of LUMÁWIG versus the current state-of-the-art implementation in HERA. Five boxplots for the running time of the original algorithm in DIONYSUS are superimposed for reference.

Figure 2 shows the running time distribution of 100 dimension zero bottleneck distance computations over increasing diagram sizes. Note that the vertical axis is displayed in logarithmic scale. Only five boxplots for the running time of the original algorithm implemented in DIONYSUS are superimposed to provide reference for the state-of-the-art HERA and our two implementations of LUMÁWIG. A quick inspection reveals that both implementations of LUMÁWIG are consistently several orders of magnitude faster than the current state-of-the-art HERA.

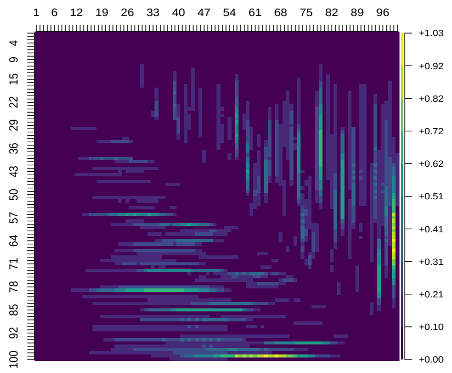
As LUMÁWIG$_{\text{R}}$ yields exact values for the bottleneck distance relative to the original DIONYSUS implementation, we use it as basis in the computation of relative differences in this stage. Figures 3a and 3b show the relative difference in the computed dimension zero bottleneck distance respectively of HERA and LUMÁWIG$_{\text{PY}}$ with respect to that of LUMÁWIG$_{\text{R}}$. Note that HERA consistently overestimates the bottleneck distance with respect to that of LUMÁWIG$_{\text{R}}$. In contrast, relative differences between the two implementations of LUMÁWIG can be attributed to rounding differences.



(a) HERA versus LUMÁWIG$_{\text{R}}$.



(b) LUMÁWIG$_{\text{PY}}$ versus LUMÁWIG$_{\text{R}}$.



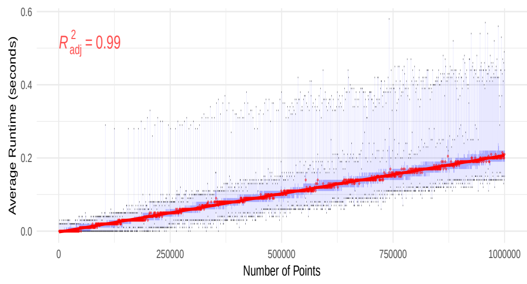
(c) Heatmap of median running time.

Figure 3: (a)-(b) Boxplots of relative differences between the bottleneck computation outputs of the indicated pair of implementations. (c) For $i \leq j$, pixel (i, j) is the median running time to compute the bottleneck distance between diagrams of sizes $1000i$ and $1000j$ with death times uniformly drawn from the interval range $(0, 2000i)$ and $(0, 2000j)$ respectively.

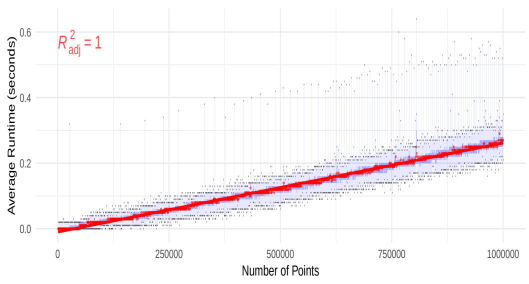
4 Empirical tests for complexity

Figure 3c shows a heat map of the median running time of LUMÁWIG$_{\text{R}}$ over 100 computations per pixel of the bottleneck distance between pairs of persistence diagram with varying number of points. It can be inferred that the best running times happen along the main diagonal and the upper and left portions of the heat map. These correspond respectively to when the diagrams have equal number of points, or when one is overwhelmingly larger than the other. In contrast, regions in the heat map that show increased running times correspond to when a large diagram is compared to another that has about half as many points.

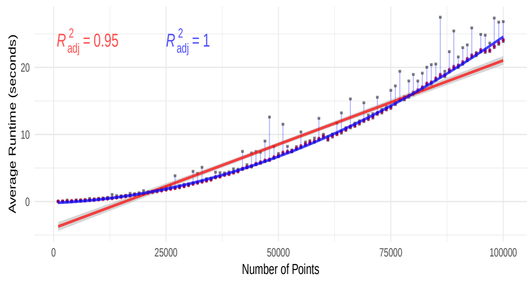
To further investigate the observations above, we examine the performance of LUMÁWIG$_{\text{R}}$ in the computation of dimension zero bottleneck distance in four pairs of settings for size of the diagrams and the range of values the death times are drawn from. The first is when LUMÁWIG$_{\text{R}}$ is tasked to compare two persistence diagrams with the same number of points whose death times are drawn from the same range of values. We calculate the dimension zero bottleneck distance over 100 pairs of persistence diagrams of equal sizes starting from 1000 to 1,000,000 points. Every diagram is simulated in the same manner as the previous experiments. Median running times are then plotted and fitted with a regression curve. Midspread and range for every 100 computations at every unit of 1000 points are superimposed to illustrate the distribution of running times. Figure 4a shows an excellent linear fit ($R^2 = 0.99$) for the running time. We also highlight the observed experimental result that the running time between two diagrams each with 1 million points with death times drawn from the range (0, 2,000,000) averages to between 2 and 3 tenths of a second.



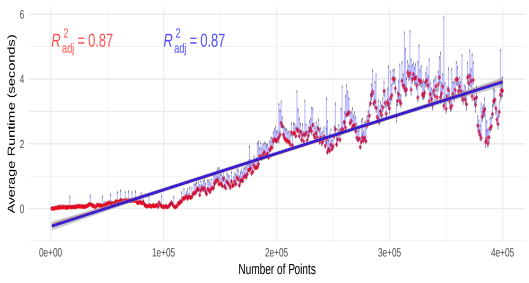
(a) Equal size and range.



(b) Equal size but different range.



(c) Different size but equal range.



(d) Different size and range.

Figure 4: Median running time in the computation of bottleneck distance between two diagrams with varying size and range settings fitted with regression curves. Superimposed are the minimum and maximum running times over the 100-run simulation per unit of 1000 points to illustrate the running time range, and the narrow darker blue band to show the midspread.

The second setting involves two diagrams of the same size but the range of death values for the second is half as wide as the first. In this case, we see in Figure 4b that the running time trend is perfectly fitted with a linear curve. The third setting considers two diagrams where the second has half as many points as the first. We remark that this setting differs from that performed for Figure 3c in that the range where the death times are drawn from for the simulated diagrams in this experiment is the same for the two diagrams. We do this to ensure that any observed significant difference in performance is attributable only to fixed difference in the number of points between the diagrams. As we observe an increased running time for LUMÁWIG$_{\text{R}}$ in this case, we compute only to until there are 100,000 points in the larger diagram. Figure 4c shows two fitted regression curves: a quadratic fit with $R^2 = 1$ and a linear fit with $R^2 = 0.95$. We highlight that even for the case where LUMÁWIG$_{\text{R}}$ evidently takes longer to compute the dimension zero bottleneck distance, a linear model provides a very good fit for the trend.

The final setting is where the second of two diagrams has half as many points with death values drawn from a range half as wide as that for the first. Regression curves are again shown in Figure 4d with linear and quadratic fit both achieving $R^2 = 0.87$.

5 Conclusion

Our new algorithm, LUMÁWIG, outperforms by several orders of magnitude, all currently available implementations of dimension zero bottleneck distance in terms of running time. LUMÁWIG also recovers the exact bottleneck distance produced by DIONYSUS. As LUMÁWIG generally enjoys linear complexity as shown by our empirical tests, we are able to present in this note the first instance, to the best of our knowledge, that the bottleneck distance is used in practice on data of magnitude and scale in the order of up to a million. This opens the opportunity to scale TDA to data sets of sizes encountered in machine learning and utilize persistence diagrams in a manner that goes beyond the simple use of the most persistent components.

Broader Impact

Our motivation for this work is to clear the computational obstruction in the use of bottleneck distance in applications. The English translation of LUMÁWIG is *to extend, broaden, or expand*. Our hope is that this contribution will serve as a catalyst in the further development of TDA that leverages persistence diagrams and the bottleneck distance similar to what has been achieved for persistence landscapes. Even now, a truly comprehensive and holistic treatment of information embedded in dimension zero persistence diagrams has been left unexplored due primarily to the lack of feasible machinery that can handle significant scaling up in data size. We believe that LUMÁWIG is a significant contribution in this direction as it affords a viable tool to process and utilize dimension zero persistence diagrams in comparing evolving connectivity information between larger data sets.

References

[1] Ignacio, P.S., Bulauan, J., Uminsky, D., Lumáwig: An Efficient Algorithm for Dimension Zero Bottleneck Distance Computation in Topological Data Analysis, Algorithms, (2020), 13, 291.

[2] Zomorodian, A., Computing and Comprehending Topology: Persistence and Hierarchical Morse Complexes. Ph.D. thesis, University of Illinois at Urbana-Champaign, (2001).

[3] Edelsbrunner, H., Letscher, D., Zomorodian, A., Topological persistence and simplification. Discrete Comput. Geom., (2002), 28, 511–533.

[4] Morozov, D., DIONYSUS library for computing persistent homology. mrzv.org/software/dionysus.

[5] Munkres, J., Algorithms for the assignment and transportation problems. J. Soc. Industr. Appl. Math. 5, 1 (1957), 32 -38.

[6] Kerber, M., Morozov, D., Nigmetov, A., Geometry Helps to Compare Persistence Diagrams, J. Exp. Algorithmicsm 2017, 22(1), Article 1.4, 20 pages.

[7] Efrat, A., Itai, A., Katz, M., Geometry Helps in Bottleneck Matching and Related Problems, Algorithmica, 2001, 31: 1. https://doi.org/10.1007/s00453-001-0016-8

[8] Hopcroft, J., Karp, R., An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2, 4 (1973), 225-231.

[9] Garin, A., Tauzin, G., A Topological "Reading" Lesson: Classification of MNIST using TDA, In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), IEEE (2019), pp 1551-1556

[10] Weber, E.S., Harding, S.N., Przybylski, L. Detecting Traffic Incidents Using Persistence Diagrams, Algorithms, (2020), 13, 222.

[11] Belchi, F., Pirashvili, M., Conway, J., Bennett, M., Djukanovic, R., Brodzki, J., Lung Topology Characteristics in patients with Chronic Obstructive Pulmonary Disease, Scientific Reports, (2018), 8:5341

[12] Fasy, B., Kim, J., Lecci, F., Maria, C., Introduction to the R package TDA, arXiv:1411.1830

[13] Saul, N. , Tralie, C., Scikit-TDA: Topological Data Analysis for Python, 2019. https://doi.org/10.5281/zenodo.2533369

Supplementary Materials

Proofs of Lemmas 1 and 2

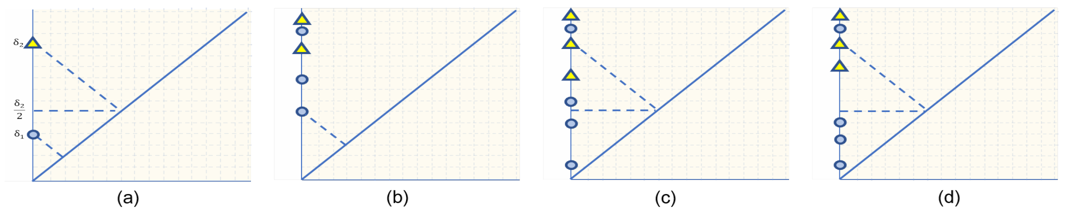


Figure 5: Examples of point-matching between persistence diagrams highlighting the resulting bottleneck distance. Points in each diagrams are shape coded and matched points are color coded. Figure (**a**) illustrates when diagonal matching achieves the bottleneck distance. Figure (**b**) is used in the proof of Lemma 1 and (**c**,**d**) in Lemma 2.

Proof of Lemma 1. For the bijection ϕ corresponding to the pre-processing described above, it follows that

$$\max_{x \in X} ||x - \phi(x)||_\infty = y_{N+1}/2.$$

To see why ϕ achieves the infimum over all bijections between X and Y, note that any other bijection ψ produces a death time for a point in Y matched to the diagonal that is at least as large as $y_{N+1}/2$. Therefore $\max_{x \in X} ||x - \phi(x)||_\infty \leq \max_{x \in X} ||x - \psi(x)||_\infty$. See Figure 5b. ■

Proof of Lemma 2.

1. It follows from our remark immediately after (1) that

   $$\max ||x - \phi(x)||_\infty = \max(Z) \leq \max(x_l, y_l)/2 = \max ||x - \phi'(x)||_\infty$$

   where ϕ' is the bijection that matches both x_l and y_l to the diagonal, and coincides with ϕ otherwise. Figure 5c illustrates this comparison between the two matchings. For any other bijection ψ, if $x' \in X$ such that $|x' - \psi(x')|$ is maximum among all non-trivial matchings, either $\max(Z) \leq |x' - \psi(x')|$, or $\max(x_l, y_l) \leq \max(x', \psi(x'))$. If $N < length(Y)$, then a similar argument as that in Lemma 1 holds. The conclusion now follows.

2. In this case, the same bijection ϕ' in the previous case yields

   $$\max ||x - \phi'(x)||_\infty = \max(x_l, y_l)/2 < \max(Z) = \max ||x - \phi(x)||_\infty.$$

   The same argument in the previous case holds for any other bijection ψ. Hence, the inequality above implies the conclusion.

3. For the bijection ϕ'' that sends x_m and y_m to the diagonal for all such m, and coincides with ϕ otherwise (see Figure 5d), we have that

   $$\max ||x - \phi''(x)||_\infty = \max(x_l, y_l)/2 < \max(Z) = \max ||x - \phi(x)||_\infty.$$

   Again, since the same argument in the first case holds for any other bijection ψ, the previous inequality implies the conclusion.

4. Define the bijection τ that sends x_j and y_j to the diagonal for all $j \geq l$, and coincides with ϕ otherwise. Then we have that

   $$\max ||x - \tau(x)||_\infty < \max(Z) = \max ||x - \phi(x)||_\infty,$$

   Moreover, note that $\max ||x - \tau(x)||_\infty$ depends only on $||x - \tau(x)||_\infty$ for non-trivially matched x and $\tau(x)$. Therefore, we can consider only the subsets X' and Y' respectively of X and Y whose points are non-trivially matched by τ. In this case, $length(X') = length(Y')$ and one of the three previous cases above holds.

The proof is now complete.■

The complete LUMÁWIG pseudo code is given below.

Algorithm 1 LUMÁWIG algorithm for computing 0-dimensional bottleneck distance between two persistence diagrams

```
1: Input: Two dimension zero persistence diagrams $X$ and $Y$ such that $X \neq Y$ and where $X$ has fewer than or as many points as $Y$.
2: Output: The bottleneck distance between $X$ and $Y$.
3: Initialization $d \leftarrow 0$, $X \leftarrow$ death times of points from $X$ sorted from largest to smallest, $Y \leftarrow$ death times of points from $Y$ sorted from largest to smallest, $N = length(X)$, $Z \leftarrow$ vector $[z_i := |x_i - y_i|]_1^N$, $l = \arg\max(Z)$, $d_{temp} = \max(Z)$
4: if $length(X) \neq length(Y)$ and $d_{temp} < y_{N+1}/2$ then
5:     $d = (y_{N+1})/2$;
6: else
7:     while $length(Z) > 1$ do
8:         if Second largest entry of $Z < \max(x_l, y_l)/2 < d_{temp}$ then
9:             $d = \max(x_l, y_l)/2$
10:            $break$
11:        else if Second largest entry of $Z \geq \max(x_l, y_l)/2$ then
12:            if For every $m$ for which $z_m \geq \max(x_l, y_l)/2$, $m \geq l$ then
13:                $d = \max(x_l, y_l)/2$
14:                $break$
15:            else
16:                Trim off all $z_m, x_m, y_m$ for $m \geq l$; update $l$ and $d_{temp}$
17:                if $length(Z) = 1$ then
18:                    $d = \min(d_{temp}, \max(x_l, y_l)/2)$
19:                    $break$
20:                end if
21:            end if
22:        else
23:            $d = d_{temp}$
24:            $break$
25:        end if
26:    end while
27: end if
```

Experimental details for benchmarking

We simulate 100 0-dimensional persistence diagrams using a set of positive numbers as death times uniformly chosen from a range twice as wide as the number of points. We pair each diagram with another simulated diagram with as much as 80% more or fewer points, then compute the bottleneck distance between the pair. The running time of LUMÁWIG (implemented both in PYTHON and R) and HERA are recorded, and the distribution summary of 100 run times for each algorithm is plotted out as a boxplot. For HERA, we follow the experimental setup from (6) and set $\delta = 0.01$. We repeat this process while increasing number of points from 1,000 to 30,000.