Comparing Graphs via Persistence Distortion

Tamal K. Dey*, Dayu Shi, Yusu Wang*

Abstract

Metric graphs are ubiquitous in science and engineering. For example, many data are drawn from hidden spaces that are graph-like, such as the cosmic web. A metric graph offers one of the simplest yet still meaningful ways to represent the non-linear structure hidden behind the data. In this paper, we propose a new distance between two finite metric graphs, called the persistence-distortion distance, which draws upon a topological idea. This topological perspective along with the metric space viewpoint provide a new angle to the graph matching problem. Our persistence-distortion distance has two properties not shared by previous methods: First, it is stable against the perturbations of the input graph metrics. Second, it is a *continuous* distance measure, in the sense that it is defined on an alignment of the underlying spaces of input graphs, instead of merely their nodes. This makes our persistence-distortion distance robust against, for example, different discretizations of the same underlying graph.

Despite considering the input graphs as continuous spaces, that is, taking all points into account, we show that we can compute the persistence-distortion distance in polynomial time. The time complexity for the discrete case where only graph nodes are considered is much faster. We also provide some preliminary experimental results to demonstrate the use of the new distance measure.

1 Introduction

Many data in science and engineering are drawn from a hidden space which are graph-like, such as the cosmic web [27] and road networks [1, 5]. Furthermore, as modern data becomes increasingly complex, understanding them with a simple yet still meaningful structure becomes important. Metric graphs equipped with a metric derived from the data can provide such a simple structure [17, 26]. They are graphs where each edge is associated with a length inducing the metric of shortest path distance. The comparison of the repersentative metric graphs can benefit classification of data, a fundamental task in processing them. This motivates the study of metric graphs in the context of matching or comparison.

To compare two objects, one needs a notion of distance in the space where the objects are coming from. Various distance measures for graphs and their metric versions have been proposed in the literature with associated matching algorithms. We approach this problem with two new perspectives: (i) We aim to develop a distance measure which is both meaningful and stable against metric perturbations, and at the same time amenable to polynomial time computations. (ii) Unlike most previous distance measures which are *discrete* in the sense that only graph node alignments are considered, we aim for a distance measure that is *continuous*, that is, alignment for all points in the underlying space of the metric graphs are considered.

Related work. To date, the large number of proposed graph matching algorithms fall into two broad categories: exact graph matching methods and inexact graph matching (distances between graphs) methods. The exact graph matching, also called the graph isomorphism problem, checks whether there is a bijection

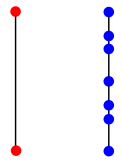
*Department of Computer Science and Engineering, The Ohio State University, Columbus, OH, USA. Emails: tamaldey, shiday, yusu@cse.ohio-state.edu

between the node sets of two input graphs that also induces a bijection in their edge sets. While polynomial time algorithms exist for many special cases, e.g., [2, 20, 24], for general graphs, it is not known whether the graph isomorphism problem is NP complete or not [16]. Nevertheless, given the importance of this problem, there are various exact graph matching algorithms developed in practice. Usually, these methods employ some pruning techniques aiming to reduce the search space for identifying graph isomorphisms. See [14] for comparisons of various graph isomorphism testing methods.

In real world applications, input graphs often suffer from noise and deformation, and it is highly desirable to obtain a *distance* between two input graphs beyond the binary decision of whether they are the same (isomorphic) or not. This is referred to as inexact graph matching in the field of pattern recognition, and various distance measures have been proposed. One line of work is based on graph edit distance which is NP-hard to compute [31]. Many heuristic methods, using for example A^* algorithms, have been proposed to address the issue of high computational complexity, see the survey [15] and references within. One of the main challenges in comparing two graphs is to determine how "good" a given alignment of graph nodes is in terms of the quality of the pairwise relations between those nodes. Hence matching two graphs naturally leads to an integer quadratic programming problem (IQP), which is a NP-hard problem. Several heuristic methods have been proposed to approach this optimization problem, such as the annealing approach of [18], iterative methods of [23, 29] and probabilistic approach in [30]. Finally, there have been several methods that formulate the optimization problem based on spectral properties of graphs. For example, in [28], the author uses the eigendecomposition of adjacency matrices of the input graphs to derive an expression of an orthogonal matrix which optimizes the objective function. In [9, 22], the principal eigenvector of a "compatibility" matrix of the input graphs is used to obtain correspondences between input graph nodes. Recently in [21], Hu et. al proposed the general and descriptive *Laplacian family signatures* to build the compatibility matrix and model the graph matching problem as an integer quadratic program.

New work. Different from previous approaches, we view input graphs as *continuous* metric spaces. Intuitively, we assume that our input is a finite graph $G = (V, E)$ where each edge is assigned a positive length value. We now consider G as a metric space $(|G|, d_G)$ on the underlying space $|G|$ of G, with metric d_G being the shortest path metric in $|G|$. Given two metric graphs G_1 and G_2, a natural way to measure their distance is to use the so-called Gromov-Hausdorff distance [19, 25] to measure the metric distortion between these two metric spaces. Unfortunately, it is NP-hard to even approximate the Gromov-Hausdorff distance for graphs within a constant factor[1]. Instead, we propose a new metric, called the *persistence-distortion distance* $\mathrm{d_{PD}}(G_1, G_2)$, which draws upon a topological idea and is computable in polynomial time with techniques from computational geometry. This provides a new angle to the graph comparison problem, and our distance has several nice properties:

(1) The persistence-distortion distance takes all points in the input graphs into account, while all previous graph matching algorithms align only graph nodes. Hence our persistence-distortion distance is insensitive to different discretization of the same graph: For example, the two geometric graphs on the right are equivalent as metric graphs, and thus the persistence-distortion distance between them is zero.



(2) In Section 3, we show that our persistence-distortion distance $\mathrm{d_{PD}}(G_1, G_2)$ is stable w.r.t. changes to input metric graphs as measured by the Gromov-Hausdorff distance. For example, the two geometric graphs on the right have small persistence-distortion distance. (Imagine that they are the reconstructed road networks from noisy data sampled from the same road systems.)

[1]This result has been very recently obtained by two groups of researchers independently: Agarwal, Fox and Nath from Duke Univ., and Sidiropoulos and Wang from Ohio State Univ.

(3) Despite that our persistence-distortion distance is a *continuous* measure which considers all points in the input graphs, we show in Section 5 that it can be computed in polynomial time ($O(m^{12} \log m)$ where m is the total complexity of input graphs). We note that the *discrete* version of our persistence-distortion distance, where only graph nodes are considered (much like in previous graph matching algorithms), can be computed much more efficiently in $O(n^2 m^{1.5} \log m)$ time, where n is the number of graph nodes in input graphs.

Finally, we also provide some preliminary experimental results to demonstrate the use of the persistence-distortion distance.

2 Notations and Proposed Distance Measure for Graphs

Metric graphs. A metric graph is a metric space (M, d) where M is the underlying space of a finite 1-dimensional simplicial complex. Given a graph $G = (V, E)$ and a weight function $\mathrm{Len} : E \to \mathbb{R}^+$ on its edge set E (assigning length to edges in E), we can associate a metric graph $(|G|, d_G)$ to it as follows. The space $|G|$ is a geometric realization of G. Let $|e|$ denote the image of an edge $e \in E$ in $|G|$. To define the metric d_G, we consider the arclength parameterization $e : [0, \mathrm{Len}(e)] \to |e|$ for every edge $e \in E$ and define the distance between any two points $x, y \in |e|$ as $d_G(x, y) = |e^{-1}(y) - e^{-1}(x)|$. This in turn provides the length of a path $\pi(z, w)$ between two points $z, w \in |G|$ that are not necessarily on the same edge in $|G|$, by simply summing up the lengths of the restrictions of this path to edges in G. Finally, given any two points $z, w \in |G|$, the distance $d_G(z, w)$ is given by the minimum length of any path connecting z to w in $|G|$.

In what follows, we do not distinguish between $|\cdot|$ and its argument and write (G, d_G) to denote the metric graph $(|G|, d_G)$ for simplicity. Furthermore, for simplicity in presentation, we abuse the notations slightly and refer to the metric graph as $G = (V, E)$, with the understanding that (V, E) refers to the topological graph behind the metric space (G, d_G). Finally, we refer to any point $x \in G$ as a point, while a point $x \in V$ as a *graph node*.

Background on persistent homology. The definition of our proposed distance measure for two metric graphs relies on the so-called persistence diagram induced by a scalar function. We refer the readers to resources such as [11, 12] for formal discussions on persistent homology and related developments. Below we only provide an intuitive and informal description of the persistent homology induced by a function under our simple setting.

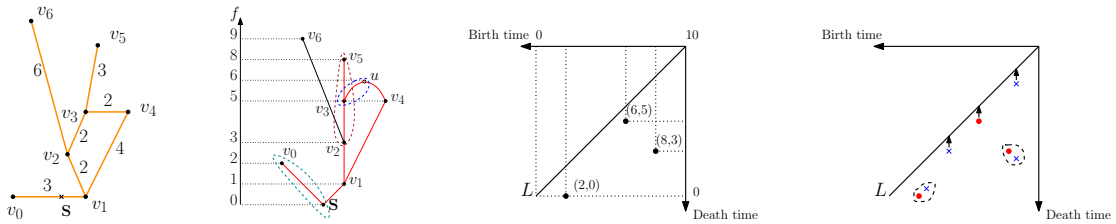
Let $f : X \to \mathbb{R}$ be a continuous real-valued function defined on a topological space X. We want to understand the structure of X from the perspective of the scalar function f: Specifically, let $X^\alpha := \{x \in X \mid f(x) \geq \alpha\}$ denote the *super-level set*[2] of X w.r.t. $\alpha \in \mathbb{R}$. Now as we sweep X top-down by decreasing the α value, the sequence of super-level sets connected by natural inclusion maps gives rise to a *filtration of X induced by f*:

$$X^{\alpha_1} \subseteq X^{\alpha_2} \subseteq \cdots \subseteq X^{\alpha_m} = X, \quad \text{for } \alpha_1 > \alpha_2 > \cdots > \alpha_m. \tag{1}$$

We track how the topological features captured by the so-called homology classes of the super-level sets change. In particular, as α decreases, sometimes new topological features are "born" at time α, that is, new families of homology classes are created in $\mathrm{H}_k(X^\alpha)$, the k-th homology group of X^α. Sometimes, existing topological features disappear, i.e, some homology classes become trivial in $\mathrm{H}_k(X^\beta)$ for some $\beta < \alpha$. The *persistent homology* captures such *birth* and *death* events, and summarizes them in the so-called *persistence diagram* $\mathrm{Dg}_k(f)$. Specifically, $\mathrm{Dg}_k(f)$ consists of a set of points $\{(\alpha, \beta) \in \mathbb{R}^2\}$ in the plane, where each (α, β) indicates a homological feature created at time α and killed at time β.

[2] In the standard formulation of persistent homology of a scalar field, the *sub-level set* $X_\alpha = \{x \in X \mid f(x) \leq \alpha\}$ is often used. We use super-level sets which suit the specific functions that we use.

In our setting, the domain X will be the underlying space of a metric graph G. The specific function that we use later is the geodesic distance to a fixed basepoint $\mathbf{s} \in G$, that is, we consider $f : G \to \mathbb{R}$ where $f(x) = d_G(\mathbf{s}, x)$ for any $x \in G$. We are only interested in the 0th-dimensional persistent homology ($k = 0$ in the above description), which simply tracks the connected components in the super-level set as we vary α.



(a) (b) (c) (d)

Figure 1: (a) A graph with basepoint $\mathbf{s}$: edge length is marked for each edge. (b) The function $f = d_G(\mathbf{s}, \cdot)$. We also indicate critical-pairs. (c) Persistence diagram $\mathrm{Dg}_0 f$: E.g, the persistence-point $(6, 5)$ is generated by critical-pair (u, v_3). (d) shows a partial matching between the red points and blue points (representing two persistence diagrams). Some points are matched to the diagonal L.

Figure 1 gives an example of the 0-th persistence diagram $\mathrm{Dg}_0(f)$ with the basepoint $\mathbf{s}$ in edge (v_0, v_1). As we sweep the graph top-down in terms of the geodesic function f, a new connected component is created as we pass through a *local maximum* u_{b} of the function $f = d_G(\mathbf{s}, \cdot)$. A local maximum of f, such as u in Figure 1 (b), is not necessarily a graph node from V. Two connected components in the super-level set can only merge at an *up-fork saddle* u_{d} of the function f: The up-fork saddle u_{d} is a point such that within a sufficiently small neighborhood of u_{d}, there are at least two branches incident on u_{d} with function values larger than u_{d}. Each point (b, d) in the persistence diagram is called a *persistence point*, corresponding to the creation and death of some connected component: At time b, a new component is created in X^{b} at a local maximum $u_{\mathsf{b}} \in G$ with $f(u_{\mathsf{b}}) = \mathsf{b}$. At time d and at an up-fork saddle $u_{\mathsf{d}} \in G$ with $f(u_{\mathsf{d}}) = \mathsf{d}$, this component merges with another component created earlier. We refer to the pair of points $(u_{\mathsf{b}}, u_{\mathsf{d}})$ from the graph G as the *critical-pair* corresponding to persistent point (b, d). We call b and d the *birth-time* and *death-time*, respectively. The plane containing the persistence diagram is called the *birth-death plane*.

Finally, given two finite persistence diagrams $\mathrm{Dg} = \{p_1, \ldots, p_\ell \in \mathbb{R}^2\}$ and $\mathrm{Dg}' = \{q_1, \ldots, q_k \in \mathbb{R}^2\}$, a common distance measure for them, the *bottleneck distance* $d_B(\mathrm{Dg}, \mathrm{Dg}')$ [6], is defined as follows: Consider Dg and Dg$'$ as two finite sets of points in the plane (where points may overlap). Call $L = \{(x, x) \in \mathbb{R}^2\}$ the *diagonal* of the birth-death plane.

Definition 1 *A* partial matching C *of* Dg *and* Dg$'$ *is a relation* $C : (\mathrm{Dg} \cup L) \times (\mathrm{Dg}' \cup L)$ *such that each point in* Dg *is either matched to a unique point in* Dg$'$*, or mapped to its closest point (under* L_∞*-norm) in the diagonal* L*; and the same holds for points in* Dg$'$*. See Figure 1 (d). The bottleneck distance is defined as* $d_B(\mathrm{Dg}, \mathrm{Dg}') = \min_C \max_{(p,q)\in C} \|p - q\|_\infty$*, where* C *ranges over all possible partial matchings of* Dg *and* Dg$'$*. We call the parital matching that achieves the bottleneck distance* $d_B(\mathrm{Dg}, \mathrm{Dg}')$ *as the* bottleneck matching.

Proposed persistence-distortion distance for metric graphs. Suppose we are given two metric graphs (G_1, d_{G_1}) and (G_2, d_{G_2}). Let (V_1, E_1) and (V_2, E_2) denote the node set and edge set for G_1 and G_2, respectively. Set $n = \max\{|V_1|, |V_2|\}$ and $m = \max\{|E_1|, |E_2|\}$.

Choose any point $\mathbf{s} \in \mathrm{G}_1$ as the base point, and consider the shortest path distance function $d_{G_1,\mathbf{s}} : \mathrm{G}_1 \to \mathbb{R}$ defined as $d_{G_1,\mathbf{s}}(x) = d_{G_1}(\mathbf{s}, x)$ for any point $x \in \mathrm{G}_1$. Let $\mathrm{P}_{\mathbf{s}}$ denote the 0-th dimensional persistence diagram $\mathrm{Dg}_0(d_{G_1,\mathbf{s}})$ induced by the function $d_{G_1,\mathbf{s}}$. Define $d_{G_2,\mathbf{t}}$ and $\mathrm{Q}_{\mathbf{t}}$ similarly for any base point $\mathbf{t} \in \mathrm{G}_2$

for the graph G_2. We map the graph G_1 to the set of (infinite number of) points in the space of persistence diagrams $\mathbb{D}$, denoted by $\mathcal{C} := \{\mathrm{P}_{\mathbf{s}} \mid \mathbf{s} \in \mathrm{G}_1\}$. Similarly, map the graph G_2 to $\mathcal{F} := \{\mathrm{Q}_{\mathbf{t}} \mid \mathbf{t} \in \mathrm{G}_2\}$.

Definition 2 *The* persistence-distortion distance between G_1 and G_2, *denoted by* $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2)$, *is the Hausdorff distance* $d_H(\mathcal{C}, \mathcal{F})$ *between the two sets* $\mathcal{C}$ *and* $\mathcal{F}$ *where the distance between two persistence diagrams is measured by the bottleneck distance. In other words,*

$$\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) = d_H(\mathcal{C}, \mathcal{F}) = \max\{ \max_{\mathrm{P}\in\mathcal{C}} \min_{\mathrm{Q}\in\mathcal{F}} d_B(\mathrm{P}, \mathrm{Q}),\ \max_{\mathrm{Q}\in\mathcal{F}} \min_{\mathrm{P}\in\mathcal{C}} d_B(\mathrm{P}, \mathrm{Q}) \}.$$

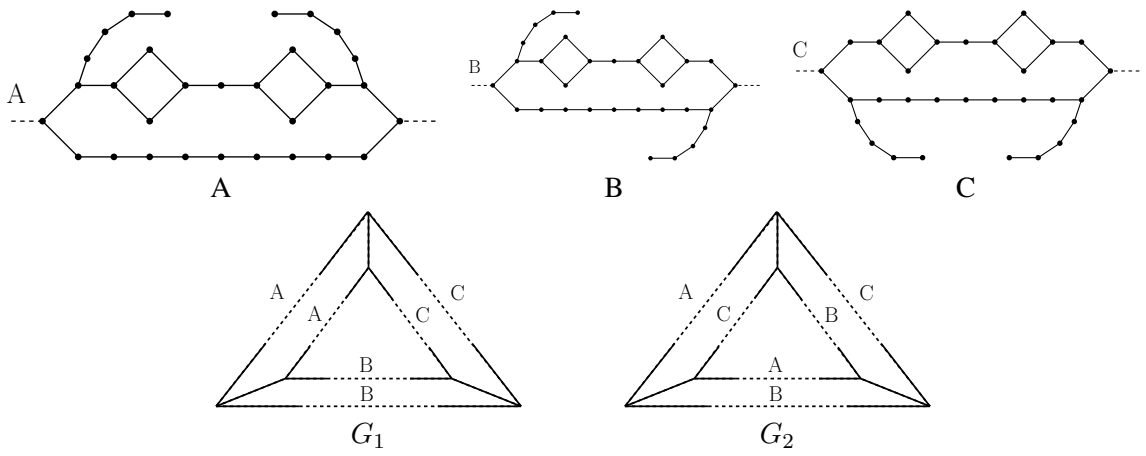


Figure 2: In the top row, we show three components A, B and C, that will be used to construct the two input metric graphs G_1 and G_2 in the bottom row. All edges are of length 1. Each of these compnent has the property, that from a basepoint outside these components, their persistence diagram w.r.t. the geodesic distance remains the same (and hence cannot be distinguished). Graphs G_1 and G_2 are not isomorphic. However, they are mapped to the same image set in the space of persistence diagrams, and hence the persistence-distortion distance between them is zero.

Remark. (1) We note that if two graphs are isomorphic, then $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) = 0$. The inverse unfortunately is not true. (See Figure 2 for an example where two graphs have $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) = 0$, but they are not isomorphic.) Hence $\mathrm{d_{PD}}$ is a pseudo-metric (it inherits the triangle-inequality property from the Hausdorff distance). (2) While the above definition uses only the 0-th persistence diagram for the geodesic distance functions, all our results hold with the same time complexity when we also include the *1st-extended persistence diagram* [7] or equivalently *1st-interval persistence diagram* [10] for each geodesic distance function $d_{G_1,\mathbf{s}}$ (resp. $d_{G_2,\mathbf{t}}$).

3 Stability of persistence-distortion distance

Gromov-Hausdorff distance. There is a natural way to measure metric distortion between metric spaces (thus for metric graphs), called the Gromov-Hausdorff distance [19, 4]. Given two metric spaces $\mathcal{X} = (X, d_X)$ and $\mathcal{Y} = (Y, d_Y)$, a *correspondance* between $\mathcal{X}$ and $\mathcal{Y}$ is a relation $\mathcal{M} : X \times Y$ such that (i) for any $x \in X$, there exists $(x, y) \in \mathcal{M}$ and (ii) for any $y' \in Y$, there exists $(x', y') \in \mathcal{M}$. The *Gromov-Hausdorff* distance between $\mathcal{X}$ and $\mathcal{Y}$ is

$$\mathrm{d_{GH}}(\mathcal{X}, \mathcal{Y}) = \frac{1}{2} \inf_{\mathcal{M}} \max_{(x_1,y_1),(x_2,y_2)\in\mathcal{M}} |d_X(x_1, x_2) - d_Y(y_1, y_2)|, \tag{2}$$

where $\mathcal{M}$ ranges over all correspondences of $X \times Y$. The Gromov-Hausdorff distance is a natural measurement for distance between two metric spaces; see [25] for more discussions. Unfortunately, so far, there is no efficient (polynomial-time) algorithm to compute nor approximate this distance, even for special metric spaces – In fact, it has been recently shown that even the *discrete* Gromov-Hausdorff distance for metric trees (where only tree nodes are considered) is NP-hard to compute, as well as to approximate within a constant factor (recall footnote 1). In contrast, as we show in Section 4 and 5, the persistence-distortion distance can be computed in polynomial time.

On the other hand, we have the following stability result, which intuitively suggests that the persistence-distortion distance is a weaker relaxation of the Gromov-Hausdorff distance. The proof of this theorem leverages a recent result on measuring distances between the Reeb graphs [3] and can be found in the full version.

Theorem 3 (Stability) $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq 6\mathrm{d_{GH}}(\mathrm{G}_1, \mathrm{G}_2)$.

By triangle inequality, this also implies that given two metric graphs G_1 *and* G_2 *and their perturbations* G_1' *and* G_2'*, respectively, we have that:*

$$\mathrm{d_{PD}}(\mathrm{G}_1', \mathrm{G}_2') \leq \mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) + 6\mathrm{d_{GH}}(\mathrm{G}_1, \mathrm{G}_1') + 6\mathrm{d_{GH}}(\mathrm{G}_2, \mathrm{G}_2').$$

Proof of Theorem 3. The remainder of this section devotes to the proof of the above theorem.

Given two input metric graphs (G_1, d_{G_1}) and (G_2, d_{G_2}), set $\delta = \mathrm{d_{GH}}(\mathrm{G}_1, \mathrm{G}_2)$ to be the Gromov-Hausdorff distance between G_1 and G_2. Assume that the correspondence $\mathcal{M}^*$ achieves this metric distortion distance δ [3]. Now for any point $\mathbf{s} \in \mathrm{G}_1$, there must exist $\mathbf{t} \in \mathrm{G}_2$ such that $(\mathbf{s}, \mathbf{t}) \in \mathcal{M}^*$. We will now show that $d_B(\mathrm{P}_\mathbf{s}, \mathrm{Q}_\mathbf{t}) \leq 6\delta$. Symmetrically, we show that for any $\mathbf{t} \in \mathrm{G}_2$, there is $(\mathbf{s}, \mathbf{t}) \in \mathcal{M}^*$ such that $d_B(\mathrm{P}_\mathbf{s}, \mathrm{Q}_\mathbf{t}) \leq 6\delta$. Since such an $\mathbf{t}$ can be found for any point $\mathbf{s} \in \mathrm{G}_1$, and symmetrically, such an $\mathbf{s}$ can be found for any $\mathbf{t} \in \mathrm{G}_2$, it then follows that the Hausdorff distance between $\mathcal{C}$ and $\mathcal{F}$ is bounded from above by 6δ, proving the theorem.

We will prove $d_B(\mathrm{P}_\mathbf{s}, \mathrm{Q}_\mathbf{t}) \leq 6\delta$ for $(\mathbf{s}, \mathbf{t}) \in \mathcal{M}^*$ with the help of another distance, the so-called functional-distortion distance between two Reeb graphs recently introduced in [3]. We recall its definition below.

The functional-distortion distance is defined between two graphs G_1 and G_2, with a function $f : \mathrm{G}_1 \to \mathbb{R}$ and $g : \mathrm{G}_2 \to \mathbb{R}$ defined on each of them, respectively. First, we define the following (pseudo-)metric on the input graphs as induced by f and g, respectively. (Note, that these metrics are different from the path-length distance metrics d_{G_1} and d_{G_2} that input graphs already come with.) Specifically, given two points $x_1, x_2 \in \mathrm{G}_1$, define $d_f(x_1, x_2) = \min_{\pi : x_1 \rightsquigarrow x_2} \mathrm{height}(\pi)$, where π ranges over all paths in G_1 from x_1 to x_2, and $\mathrm{height}(\pi) = \max_{x \in \pi} f(x) - \min_{x \in \pi} f(x)$ is the maximum f-function value difference for points from the path π. Define the metric d_g for G_2 similarly.

Now given two continuous maps $\phi_\rightarrow : \mathrm{G}_1 \to \mathrm{G}_2$ and $\phi_\leftarrow : \mathrm{G}_2 \to \mathrm{G}_1$, we consider the following *continuous matching*, which is a correspondence induced by a pair of *continuous maps*:

$$\mathcal{M}_c(\phi_\rightarrow, \phi_\leftarrow) = \{(x, \phi_\rightarrow(x)) \mid x \in \mathrm{G}_1\} \bigcup \{(\phi_\leftarrow(y), y) \mid y \in \mathrm{G}_2\}. \tag{3}$$

The distortion induced by $\phi_\rightarrow$ and $\phi_\leftarrow$ is defined as:

$$\mathcal{D}(\phi_\rightarrow, \phi_\leftarrow) = \sup_{(x,y),(x',y') \in \mathcal{M}_c(\phi_\rightarrow, \phi_\leftarrow)} \frac{1}{2} |d_f(x, x') - d_g(y, y')|. \tag{4}$$

[3] It is possible that δ is achieved in the limit, in which case, we consider a sequence of ε-correspondences whose corresponding metric distortion distance converges to δ as ε tends to 0. For simplicity, we assume that δ can be achieved by the correspondence $\mathcal{M}^*$.

The *functional-distortion distance* between two metric graphs (G_1, d_f) and (G_2, d_g) is defined as:

$$\mathrm{d_{FD}}(\mathrm{G}_1, \mathrm{G}_2) = \inf_{\phi_\rightarrow, \phi_\leftarrow} \max\{\mathcal{D}(\phi_\rightarrow, \phi_\leftarrow), \max_{x \in \mathrm{G}_1} |f(x) - g \circ \phi_\rightarrow(x)|, \max_{y \in \mathrm{G}_2} |f \circ \phi_\leftarrow(y) - g(y)|\}, \tag{5}$$

where $\phi_\rightarrow$ and $\phi_\leftarrow$ range over all continuous maps between G_1 and G_2. It is shown in [3] that

Theorem 4 ([3]) *$d_B(\mathrm{Dg}_0 f, \mathrm{Dg}_0 g) \leq \mathrm{d_{FD}}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g))$, where $\mathrm{Dg}_0 f$ and $\mathrm{Dg}_0 g$ denote the 0th-dimensional persistence diagram induced by the function $f : \mathrm{G}_1 \to \mathbb{R}$ and $g : \mathrm{G}_2 \to \mathbb{R}$, respectively.*

In what follows, we show that $\mathrm{d_{FD}}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g)) \leq 6\delta$ for $f = d_{G_1,\mathbf{s}}$ and $g = d_{G_2,\mathbf{t}}$. Note that in this case $\mathrm{Dg}_0 f = \mathrm{P}_\mathbf{s}$ and $\mathrm{Dg}_0 g = \mathrm{Q}_\mathbf{t}$. Combining with Theorem 4, this then implies $d_B(\mathrm{P}_\mathbf{s}, \mathrm{Q}_\mathbf{t}) \leq 6\delta$.

Remark. We note that Theorem 4 extends to the case where we consider the 1st-extended persistence diagrams for f and g, respectively, in which case the constant in front of $\mathrm{d_{FD}}$ will change from 1 to 3. In other words, if we include the 1st-extended persistence diagrams in our definitions of the persistence-distortion distance, then Theorem 3 still holds with a slightly worst constant 18 (instead of 6).

Lemma 5 *$\mathrm{d_{FD}}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g)) \leq 6\delta$ for $f = d_{G_1,\mathbf{s}}$ and $g = d_{G_2,\mathbf{t}}$.*

Proof: We will use Theorem A.1 of [3] to prove the above result. In particular, Theorem A.1 of [3] states that $\mathrm{d_{FD}}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g)) \leq 3 d_{fGH}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g))$, where the so-called *functional Gromove-Hausdorff distance* $d_{fGH}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g))$ is a more restricted version of the Gromov-Hausdorff distance, defined as follows:

$$\begin{aligned} &d_{fGH}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g)) \\ &\quad = \inf_{\mathcal{M}} \max\Big\{ \frac{1}{2} \max_{(x_1,y_1),(x_2,y_2) \in \mathcal{M}} |d_X(x_1, x_2) - d_Y(y_1, y_2)|, \max_{(x,y) \in \mathcal{M}} |f(x) - g(y)| \Big\}, \end{aligned}$$

where $\mathcal{M}$ ranges over all correspondences between graphs G_1 and G_2. Compared to the definition of the Gromov-Hausdorff distance in Eqn (2), the functional Gromov-Hausdorff distance has an extra condition that the functional difference $|f(x) - g(y)|$ between a pair of corresponding pair of points $x \in G_1$ and $y \in G_2$ should also be small.

Intuitively, there is a constant factor of 3 in the relation between $\mathrm{d_{FD}}$ and d_{fGH} because the definition of Gromov-Hausdorff distance allows all possible correspondances, while the definition of $\mathrm{d_{FD}}$ only allows those induced by a pair of *continuous* maps.

Given the optimal correspondence $\mathcal{M}^*$ for the Gromov-Hausdorff, it is easy to see that $|f(x) - g(y)| = |d_{G_1,\mathbf{s}}(x) - d_{G_2,\mathbf{t}}(y)| \leq 2\delta$ for any pair $(x, y) \in \mathcal{M}^*$. It then follows that

$$d_{fGH}((\mathrm{G}_1, d_f), (\mathrm{G}_2, d_g)) \leq 2\delta.$$

Combining this with Theorem A.1 of [3], the lemma then follows. ∎

We remark that one can in fact modify the proof of Theorem A.1 of [3] directly to obtain a better constant in Lemma 5.

Putting everything together we obtain Theorem 3.

4 Discrete PD-Distance

Suppose we are given two metric graphs $(\mathrm{G}_1 = (V_1, E_1), d_{G_1})$ and $(\mathrm{G}_2 = (V_2, E_2), d_{G_2})$, where the shortest distance metrics d_{G_1} and d_{G_2} are induced by lengths associated with the edges in $E_1 \cup E_2$. As a simple warm-up, we first compute the following discrete version of persistence-distortion distance where only graph nodes in V_1 and V_2 are considered:

Definition 6 *Let* $\hat{\mathcal{C}} := \{\mathrm{P}_v \mid v \in V(\mathrm{G}_1)\}$ *and* $\hat{\mathcal{F}} := \{\mathrm{Q}_u \mid u \in V(\mathrm{G}_2)\}$ *be two discrete sets of persistence diagrams. The* discrete persistence-distortion distance *between* G_1 *and* G_2*, denoted by* $\widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$*, is given by the Hausdorff distance* $d_H(\hat{\mathcal{C}}, \hat{\mathcal{F}})$.

We note that while we only consider graph nodes as base points, the local maxima of the resulting geodesic function may still occur in the middle of an edge. Nevertheless, for a fixed base point, each edge could have at most one local maximum, and its location can be decided in $O(1)$ time once the shortest-path distance from the base point to the endpoints of this edge are known. The observation below follows from the fact that geodesic distance is 1-Lipschitz (as the basedpoint moves) and the stability of persistence diagrams.

Observation 7 $\mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq \widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq \mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) + \frac{\ell}{2}$*, where* ℓ *is the largest length of any edge in* $E_1 \cup E_2$.

Lemma 8 *Given metric graphs* $\mathrm{G}_1 = (V_1, E_1)$ *and* $\mathrm{G}_2 = (V_2, E_2)$*,* $\widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ *can be computed in* $O(n^2 m^{1.5} \log m)$ *time, where* $n = \max\{|V_1|, |V_2|\}$ *and* $m = \max\{|E_1|, |E_2|\}$.

Proof: For a given base point $\mathbf{s} \in V_1$ (or $\mathbf{t} \in V_2$), computing the shortest path distance from $\mathbf{s}$ to all other graph nodes, as well as the persistence diagram $\mathrm{P}_{\mathbf{s}}$ (or $\mathrm{Q}_{\mathbf{t}}$) takes $O(m \log n)$ time. Hence it takes $O(mn \log n)$ total time to compute the two collections of persistence diagrams $\widehat{\mathcal{C}} = \{\mathrm{P}_{\mathbf{s}} \mid \mathbf{s} \in V(\mathrm{G}_1)\}$ and $\widehat{\mathcal{F}} = \{\mathrm{Q}_{\mathbf{t}} \mid \mathbf{t} \in V(\mathrm{G}_2)\}$.

Each persistence diagram $\mathrm{P}_{\mathbf{s}}$ has $O(m)$ number of points in the plane – it is easy to show that there are $O(m)$ number of local maxima of the geodesic function $d_{G_1,\mathbf{s}}$ (some of which may occur in the interior of graph edges). Since the birth time b of every persistence point (b, d) corresponds to a unique local maximum u_{b} with $f(u_{\mathsf{b}}) = \mathsf{b}$, there can be only $O(m)$ points (some of which may overlap each other) in the persistence diagram $\mathrm{P}_{\mathbf{s}}$.

Next, given two persistence diagrams $\mathrm{P}_{\mathbf{s}}$ and $\mathrm{Q}_{\mathbf{t}}$, we need to compute the bottleneck distance between them. In [13], Efrat et al. gives an $O(k^{1.5} \log k)$ time algorithm to compute the optimal bijection between two input sets of k points P and Q in the plane such that the maximum distance between any mapped pair of points $(p, q) \in P \times Q$ is minimized. This distance is also called the bottleneck distance, and let us denote it by $\hat{d}_B$. The bottleneck distance between two persistence diagrams $\mathrm{P}_{\mathbf{s}}$ and $\mathrm{Q}_{\mathbf{t}}$ is similar to the bottleneck distance $\hat{d}_B$, with the extra addition of diagonals. However, let P' and Q' denote the vertical projection of points in $\mathrm{P}_{\mathbf{s}}$ and $\mathrm{Q}_{\mathbf{t}}$, respectively, onto the diagonal L. It is easy to show that $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{Q}_{\mathbf{t}}) = \hat{d}_B(\mathrm{P}_{\mathbf{s}} \cup Q', \mathrm{Q}_{\mathbf{t}} \cup P')$. Hence $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{Q}_{\mathbf{t}})$ can be computed by the algorithm of [13] in $O(m^{1.5} \log m)$ time. Finally, to compute the Hausdorff distance between the two sets of persistence diagrams $\widehat{\mathcal{C}}$ and $\widehat{\mathcal{F}}$, one can check for all pairs of persistence diagrams from these two sets, which takes $O(n^2 m^{1.5} \log m)$ time since the $|\widehat{\mathcal{C}}| \leq n$ and $|\widehat{\mathcal{F}}| \leq n$. The lemma then follows. ∎

By Observation 7, $\widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ only provides an approximation of $\mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ with an additive error as decided by the longest edge in the input graphs. For unweighted graphs (where all edges have length 1), this gives an additive error of 1. This in turns provides a factor-2 approximation of the continuous persistence-distortion distance, since $\mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ is necessarily an integer in this setting.

Corollary 9 *The discrete persistence-distortion distance provides a factor-2 approximation of the continuous persistence-distortion distance for two graphs* G_1 *and* G_2 *with unit edge length; that is,* $\mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq \widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq 2\mathrm{d}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$.

One may add additional (steiner) nodes to edges of input graphs to reduce the longest edge length, so that the discrete persistence-distortion distance approximates the continuous one within a smaller additive error. But it is not clear how to bound the number of steiner nodes necessary for approximating the continuous distance within a multiplicative error, even for the case when all edges weights are approximately 1. Below we show how to directly compute the continuous persistence-distortion distance *exactly* in polynomial time.

5 Computation of Continuous Persistence-distortion Distance

We now present a polynomial-time algorithm to compute the (continuous) persistence-distortion distance between two metric graphs $(\mathrm{G}_1 = (V_1, E_1), d_{G_1})$ and $(\mathrm{G}_2 = (V_2, E_2), d_{G_2})$. As before, set $n = \max\{|V_1|, |V_2|\}$ and $m = \max\{|E_1|, |E_2|\}$. Below we first analyze how points in the persistence diagram change as we move the basepoint in G_1 and G_2 continuously.

5.1 Changes of persistence diagrams

We first consider the scenario where the basepoint $\mathbf{s}$ moves within a fixed edge $\sigma \in E_1$ of G_1, and analyze how the corresponding persistence diagram $\mathrm{P}_{\mathbf{s}}$ changes. Using notations from Section 2, let $(u_{\mathsf{b}}, u_{\mathsf{d}})$ be the critical-pair in G_1 that gives rise to the persistence point $(\mathsf{b}, \mathsf{d}) \in \mathrm{P}_{\mathbf{s}}$. Then u_{b} is a maximum for the distance function $d_{G_1,\mathbf{s}}$, while u_{d} is an up-fork saddle for $d_{G_1,\mathbf{s}}$. We call u_{b} and u_{d} from G_1 the *birth point* and *death point* w.r.t. the persistence-point (b, d) in the persistence diagram.

As the basepoint $\mathbf{s}$ moves to $\mathbf{s}' \in \sigma$ within ε distance along the edge σ for any $\varepsilon \geq 0$, the distance function is perturbed by at most ε; that is, $\|d_{G_1,\mathbf{s}} - d_{G_1,\mathbf{s}'}\|_\infty \leq \varepsilon$. By the Stability Theorem of the persistence diagrams [6], we have that $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{P}_{\mathbf{s}'}) \leq \varepsilon$. Hence as the basepoint $\mathbf{s}$ moves continuously along σ, points in the persistence diagram $\mathrm{P}_{\mathbf{s}}$ move continuously[4]. We now analyze how a specific point (b, d) may change its trajectory as $\mathbf{s}$ moves from one endpoint v_1 of $\sigma = (v_1, v_2) \in E_1$ to the other endpoint v_2.

Specifically, we use the arc-length parameterization of σ for $\mathbf{s}$, that is, $\mathbf{s} : [0, \mathrm{Len}(\sigma)] \to \sigma$. For any object $X \in \{\mathsf{b}, \mathsf{d}, u_{\mathsf{b}}, u_{\mathsf{d}}\}$, we use $X(s)$ to denote the object X w.r.t. basepoint $\mathbf{s}(s)$. For example, $(\mathsf{b}(s), \mathsf{d}(s))$ is the persistence-point w.r.t. basepoint $\mathbf{s}(s)$, while $u_{\mathsf{b}}(s)$ and $u_{\mathsf{d}}(s)$ are the corresponding pair of local maximum and up-fork saddle that give rise to $(\mathsf{b}(s), \mathsf{d}(s))$. We specifically refer to $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ and $\mathsf{d} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ as the *birth-time function* and the *death-time function*, respectively. By the discussion from the previous paragraph, these two functions are continuous.

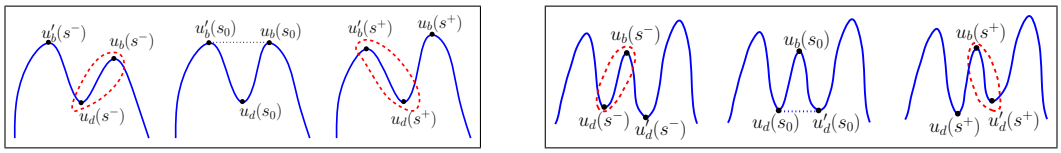


Figure 3: For better illustration of ideas, we use height function defined on a line to show: (a) a max-max critical event at s_0; and (b) a saddle-saddle critical event at s_0.

Critical events. To describe the birth-time and death-time functions, we need to understand how the corresponding birth-point and death-point $u_{\mathsf{b}}(s)$ and $u_{\mathsf{d}}(s)$ in G_1 change as the basepoint $\mathbf{s}$ varies. Recall that as $\mathbf{s}$ moves, the birth-time and death-time change continuously. However, the critical points $u_{\mathsf{b}}(s)$ and $u_{\mathsf{d}}(s)$ in G_1 may (i) stay the same or move continuously, or (ii) have discontinuous jumps. Informally, if it is case (i), then we show below that we can describe $\mathsf{b}(s)$ and $\mathsf{d}(s)$ using a piecewise linear function with $O(1)$ complexity. Case (ii) happens when there is a *critical event* where two critical-pairs (u_b, u_d) and (u'_b, u'_d) swap their pairing partners to (u_b, u'_d) and (u'_b, u_d). Specifically, at a critical event, since the birth-time and death-time functions are still continuous, it is necessary that either $d_{G_1,\mathbf{s}}(u_b) = d_{G_1,\mathbf{s}}(u'_b)$ or $d_{G_1,\mathbf{s}}(u_d) = d_{G_1,\mathbf{s}}(u'_d)$; we call the former a *max-max critical event* and the latter a *saddle-saddle critical*

[4]There could be new persistence points appearing or current points disappearing in the persistence diagram as $\mathbf{s}$ moves. Both creation and deletion necessarily happen on the diagonal of the persistence diagram as $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{P}_{\mathbf{s}'})$ necessarily tends to 0 as $\mathbf{s}'$ approaches $\mathbf{s}$. Nevertheless, for simplicity of presentation, for the time being, we describe the movement of persistence points ignoring their creation and deletion. Such creation and deletion will be addressed later in Section 5.1.3.

event. See Figure 3 for an illustration. It turns out that the birth-time function $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ (resp. death-time function d) is a piecewise linear function whose complexity depends on the number of critical events, which we analyze below.

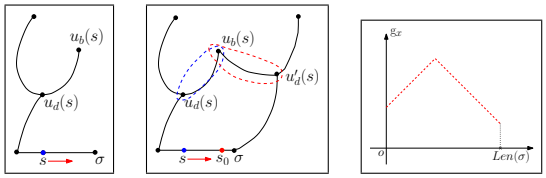
5.1.1 The death-time function $\mathsf{d} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$

The analysis of death-time function is simpler than that of the birth-time function; so we describe it first. Note, given that $d_{G_1,\mathbf{s}}$ is the geodesic distance to the base point $\mathbf{s}$, a merging of two components at an up-fork saddle cannot happen in the interior of an edge, unless at the basepoint $\mathbf{s}$ itself.

Observation 10 *An upper-fork saddle $u \in \mathrm{G}_1$ is necessarily a graph node from V_1 with degree at least 3 unless $u = \mathbf{s}$.*

To simplify the exposition, we omit the case of $u = \mathbf{s}$ (which is an easier case) in our discussions below. Since the up-fork saddles now can only be graph nodes, as the basepoint $\mathbf{s}(s)$ moves, the death-point $u_{\mathsf{d}}(s)$ either (case-1) stays at the same graph node, or (case-2) switches to a different up-fork saddle u'_{d} (i.e, a saddle-saddle critical event); see Figure 4.

Now for any point $x \in \mathrm{G}_1$, we introduce the function $\mathsf{g}_x : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ which is the distance function from x to the moving basepoint $\mathbf{s}(s)$ for $s \in [0, L_\sigma]$; that is, $\mathsf{g}_x(s) := d_{G_1,\mathbf{s}(s)}(x)$. Intuitively, as the basepoint $\mathbf{s}(s)$ moves along σ, the distance from $\mathbf{s}(s)$ to a fixed point x either increases or decreases at unit speed, until it reaches a point where the shortest path from $\mathbf{s}(s)$ to x changes discontinuously. We have the following observation.



(case-1) (case-2) (c)

Figure 4: (c) Graph of function $\mathsf{g}_x : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$.

Claim 11 *For any point $x \in \mathrm{G}_1$, as the basepoint $\mathbf{s}$ moves in an edge $\sigma \in E$, the distance function $\mathsf{g}_x : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ defined as $\mathsf{g}_x(s) := d_{G_1,\mathbf{s}(s)}(x)$ is a piecewise linear function with at most 2 pieces, where each piece has slope either '1' or '-1'. See Figure 4 (c).*

Proof: Let v_1 and v_2 be the two endpoints of the edge σ where the basepoint $\mathbf{s}$ lies in. For a fixed point $x \in G_1$, first consider the shortest path tree T_x with x being the source point (root). If the edge σ is a tree edge in the shortest path tree T_x, then as $\mathbf{s}$ moves from v_1 to v_2, the shortest path from x to $\mathbf{s}$ changes continuously and the distance $d_{G_1}(x, \mathbf{s})$ increases or decreases at unit speed. In this case, the function g_x contains only one linear piece with slope either '1' (if $\mathbf{s}$ is moving towards x) or '-1' (if $\mathbf{s}$ is moving away from x).

Otherwise, the shortest distance to $\mathbf{s}(s)$ from x will be the shorter of the shortest distance to v_i plus the distance from v_i to $\mathbf{s}(s)$, for $i = 1$,or 2 and $s \in [0, \mathrm{Len}(\sigma)]$. That is,

$$\mathsf{g}_x(s) = \min\{d_{G_1}(x, v_1) + s, d_{G_1}(x, v_2) + \mathrm{Len}(\sigma) - s\}.$$

The two functions in the above equation are linear with slope '1' and '-1', respectively. The graph of g_x is the lower envelop of the graphs of these two linear functions, and the claim thus follows.

We note that the break point of the function g_x, where it changes to a different linear function, happens at the value s_0 such that $d_{G_1}(x, v_1) + s_0 = d_{G_1}(x, v_2) + \mathrm{Len}(\sigma) - s_0$, and it is easy to check that $\mathbf{s}(s_0)$ is a local maximum of the distance function g_x. ■

As $\mathbf{s}(s)$ moves, if the death-point $u_{\mathsf{d}}(s)$ stays at the same up-fork saddle u, then by the above claim, the death-time function d (which locally equals g_u) is a piecewise linear function with at most 2 pieces.

Now we consider (case-2) when a saddle-saddle critical event happens: Assume that as s passes value s_0, $u_{\mathsf{d}}(s)$ switches from a graph node u to another one u'. At the time s_0 when this swapping happens, we have that $d_{G_1,\mathbf{s}(s_0)}(u) = d_{G_1,\mathbf{s}(s_0)}(u')$. In other words, the graph for function g_u and the graph for function $\mathsf{g}_{u'}$ intersect at s_0. Before s_0, d follows the graph for the distance function g_u, while after time s_0, u_{d} changes its identity to u' and thus the movement of d will then follow the distance function $\mathsf{g}_{u'}$ for $s > s_0$. Since the function g_x is piecewise-linear (PL) with at most 2 pieces as shown in Figure 4 (c) for any point $x \in \mathrm{G}_1$, the switching for a fixed pair of nodes u and u' can happen at most once (as the graph of g_u and that of $\mathsf{g}_{u'}$ intersect at most once). Overall, since there are $|V_1| \leq n$ graph nodes, we conclude that:

Lemma 12 *As* $\mathbf{s}$ *moves along* σ*, there are* $O(n^2)$ *number of saddle-saddle critical events in the persistence diagram* $\mathrm{P}_{\mathbf{s}}$.

For our later arguments, we need a stronger version of the above result. Specifically, imagine that we track the trajectory of the death-time d for a persistence pair (b, d).

Proposition 13 *For a fixed persistent point* $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$*, the corresponding death-time function* $\mathsf{d} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ *is piecewise linear with at most* $O(n)$ *pieces, and each linear piece has slope either '1' or '-1'. This also implies that the function* d *is 1-Lipschitz.*

Proof: By Observation 10, $u_{\mathsf{d}}(s)$ is always a graph node from V_1. For any node u, recall $\mathsf{g}_u(s) = d_{G_1,\mathbf{s}(s)}(u)$. As described above, $\mathsf{d}(s)$ will follow certain g_u with $u = u_{\mathsf{d}}(s)$ till the identify of $u_{\mathsf{d}}(s)$ changes at a saddle-saddle critical event between u with another up-fork saddle u'. Afterwards, $\mathsf{d}(s)$ will follow $\mathsf{g}_{u'}$ till the next critical event. Since each piece of g_v has slope either '1' or '-1', the graph of d consists of linear pieces of slope '1' or '-1'. Note that this implies that the function d is a 1-Lipschitz function.

On the other hand, for a specific graph node $u \in V$, each linear piece in g_u has slope '1' or '-1'. This means that one linear piece in g_u can intersect the graph of d at most once for $s \in [0, \mathrm{Len}(\sigma)]$ as d is 1-Lipschitz. Hence the graph of g_u can intersect the graph of d at most twice; implying that the node u can appear as $u_{\mathsf{d}}(s)$ for at most two intervals of s values. Thus the total descriptive complexity of d is $O(|V_1|) = O(n)$, which completes the proof. ■

5.1.2 The Birth-time Function $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$

To track the trajectory of the birth-time b of a persistence pair $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_0$, we study the movements of its corresponding birth-point (which is a maximum) $u_{\mathsf{b}} : [0, \mathrm{Len}(\sigma)] \to G$ in the graph. However, unlike up-fork saddles (which must be graph nodes), maxima of the distance function $d_{G_1,\mathbf{s}}$ can also appear in the interior of a graph edge. Roughly speaking, in addition to degree-1 graph nodes, which must be local maxima, imagine the shortest path tree with $\mathbf{s}$ as the root (source), then for any non-tree edge, it will generate a local maximum of the distance function $d_{G_1,\mathbf{s}}$. (Recall the maximum u in Figure 1 (b), which lies in the interior of edge (v_3, v_4).) Nevertheless, the following result states that there can be at most one local maximum associated with each edge.

Lemma 14 *Given an arbitrary basepoint* $\mathbf{s}$*, a maximum for the distance function* $d_{G_1,\mathbf{s}} : \mathrm{G}_1 \to \mathbb{R}$ *is either a degree-1 graph node, or a point* v *with at least two shortest paths to the basepoint* $\mathbf{s}$ *which are disjoint in a small neighborhood around* v.

Furthermore, there can be at most one maximum of $d_{G_1,\mathbf{s}}$ *in each edge in* E_1.

Proof: Consider the shortest path tree T of G_1 rooted at $\mathbf{s}$. All degree-1 graph nodes in V will be tree leaves, and each of them is thus a local maximum for the distance function $d_{G_1,\mathbf{s}}$. For each such maximum, we associate it with the unique tree edge incident on it.

Now take a maximum v which is not a degree-1 graph node. Set k to be the number of branches incident on v in a sufficiently small neighborhood of v: $k = 2$ if v is in the interior of an edge of E_1, and $k = degree(v) \geq 2$ if v is a graph node. Since $d_{G_1,\mathbf{s}}(v)$ is larger than the distance from basepoint $\mathbf{s}$ to any other point in the neighborhood of v, and since the distance function is continuous, there must exist at least k different shortest paths from $\mathbf{s}$ to v, each one coming from a different branch around v (and thus disjoint in a small neighborhood around v).

Furthermore, for each of the $E_1 - V_1 + 1$ number of edges not in the shortest path tree T rooted at $\mathbf{s}$, say $e = (w_1, w_2)$, it must contain one local maximum for the distance function $d_{G_1,bp}$. Indeed, by property of shortest path distance, we know that $d_{G_1,\mathbf{s}}(w_1) \leq d_{G_1,\mathbf{s}}(w_2) + \text{Len}(e)$ and $d_{G_1,\mathbf{s}}(w_2) \leq d_{G_1,\mathbf{s}}(w_1) + \text{Len}(e)$. If the equality does not hold in either of these two relations, then as we move x from the endpoint with lower distance value, say w_1, to w_2 along e, the shortest distance must first increase and then decrease, meaning that there is a local maximum in the interior of e. Specifically, the local maximum happens at the point $v \in e$ such that $d_{G_1,\mathbf{s}}(w_1) + \|w_1 - v\| = d_{G_1,\mathbf{s}}(w_2) + \|v - w_2\|$, and there are two shortest paths from $\mathbf{s}$ to v, one passing through w_1 and the other passing through w_2. If the equality holds for one of them, say $d_{G_1,\mathbf{s}}(w_2) = d_{G_1,\mathbf{s}}(w_1) + \text{Len}(e)$, then w_2 may or may not be a local maximum.

Overall, each edge in G, whether it is a tree edge or non-tree edge in T, will produce at most one local maximum for the distance function $d_{G_1,\mathbf{s}}$. The claim the follows. ■

As the basepoint $\mathbf{s}$ moves, the position of the local maximum within an edge may stay or may move *continuously* along the edge e. The above claim states that for a fixed basepoint, there can be at most one maximum in an edge $e \in E_1$. Hence instead of tracking u_b (which could move continuously), we now associate the identity of u_b with the *birth-edge* $e_\mathsf{b} \in E_1$ that contains u_b, and tracks the changes of the birth-edge $e_\mathsf{b} : [0, \text{Len}(\sigma)] \to E_1$ as the basepoint moves: In particular, as $s \in [0, \text{Len}(\sigma)]$ changes, $e_\mathsf{b}(s)$ could remain the same edge, or it could change to a different one. We now investigate each of these two cases.

Case 1: e_b does not change. For a fixed edge $e \in E_1$ we introduce the function $\mathsf{g}_e : [0, \text{Len}(\sigma)] \to \mathbb{R}$ where, for any $s \in [0, \text{Len}(\sigma)]$, $\mathsf{g}_e(s)$ is the distance from the basepoint $\mathbf{s}(s)$ to the unique maximum (if it exists) in e; $\mathsf{g}_e(s) = +\infty$ if the distance function $d_{G_1,\mathbf{s}(s)}$ does not have a local maximum in e. We refer to the portion of g_e with finite value as *well-defined*. Intuitively, the function g_e serves as the same role as the distance function g_x in Section 5.1.1, and similar to Claim 11, we have the following characterization for this distance function.

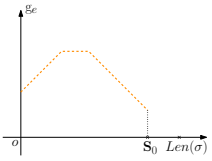


Figure 5: Graph of function $\mathsf{g}_e : [0, \text{Len}(\sigma)] \to \mathbb{R}$.

Proposition 15 *For any edge $e \in E_1$, the well-defined portion of the function g_e is a piecewise-linear function with $O(1)$ pieces, where each piece is of slope '1', '-1' or '0'. See Figure 5 for an illustration.*

Proof: We assume that $e \neq \sigma$; the case $e = \sigma$ is simpler to handle. Let $\mathbf{m}(s) \in e = (w_1, w_2)$ be the maximum for distance function $d_{G_1,\mathbf{s}(s)}$ w.r.t. basepoint $\mathbf{s}(s) \in \sigma = (v_1, v_2)$. Note that $\mathsf{g}_e(s) = d_{G_1,\mathbf{s}(s)}(\mathbf{m}(s))$. From the proof of Lemma 14, we know that

$$d_{G_1,\mathbf{s}(s)}(\mathbf{m}(s)) = d_{G_1,\mathbf{s}(s)}(w_1) + \|w_1 - \mathbf{m}(s)\| = d_{G_1,\mathbf{s}(s)}(w_2) + \|w_2 - \mathbf{m}(s)\| \quad (6)$$

$$= \mathsf{g}_{w_1}(s) + \|w_1 - \mathbf{m}(s)\| = \mathsf{g}_{w_2}(s) + \|w_2 - \mathbf{m}(s)\|. \quad (7)$$

(Recall that as introduced in Section 5.1.1, $\mathsf{g}_x : [0, \text{Len}(\sigma)] \to \mathbb{R}$ is defined as $\mathsf{g}_x(s) = d_{G_1,\mathbf{s}(s)}(x)$.) Conversely, a point $\mathbf{m}(s)$ in the interior of e satisfying the equation above must be a local maximum of the distance function $d_{G_1,\mathbf{s}(s)}$. By Claim 11, as s varies, g_{w_1} (resp. g_{w_2}) is a piecewise linear function with at most two pieces of slope '1' or '-1'.

(1) If at $s \in [0, \mathrm{Len}(\sigma)]$, the slopes of functions g_{w_1} and g_{w_2} are the same (i.e, as s increases, $d_{G_1,\mathbf{s}(s)}(w_1)$ and $d_{G_1,\mathbf{s}(s)}(w_2)$ both increase or both decrease at the same speed), then by Eqn (7), the local maximum $\mathbf{m}(s)$ remains the same as s moves. Hence $\mathsf{g}_e(s) = d_{G_1,\mathbf{s}(s)}(\mathbf{m}(s))$ follows a linear function with the same slope as g_{w_1} (and g_{w_2}) which is either '1' or '-1'.

(2) If at s, the slopes of g_{w_1} and g_{w_2} are not the same, (i.e, as s increases, $d_{G_1,\mathbf{s}(s)}(w_1)$ and $d_{G_1,\mathbf{s}(s)}(w_2)$ change in the opposite manner), then in order for Eqn (7) to hold, $\mathbf{m}(s)$ moves at the same speed as $\mathbf{s}(s)$. In this case, $\mathsf{g}_e(s)$ remains the same value, that is, g_e is a linear (in fact, constant) function with slope '0'.

Now decompose $[0, \mathrm{Len}(\sigma)]$ into maximal intervals such that within each interval, g_{w_1} and g_{w_2} each can be described by a single linear function. By Claim 11, there can be at most three such intervals. Within each interval, if a maximum exists in edge e, then the function g_e (which is the distance to this maximum) can be described by a linear function of slope '1', '-1' or '0', as described by the two cases above.

Finally, note that in (2) above, as $\mathbf{m}(s)$ moves along e, it is possible that $\mathbf{m}(s)$ coincides with one of its endpoint say w_1. After that, Eqn (7) cannot hold and the local maximum moves out of edge e – Indeed, one can verify that after that, the edge e becomes a tree edge in the shortest path tree rooted at $\mathbf{s}(s)$. In other words, afterwards, g_e is no longer well-defined. Within a single maximal interval of $[0, \mathrm{Len}(\sigma)]$ as described above, such event can happen at most once for each of w_1 and w_2. Overall, the well-defined portion of g_e consists $O(1)$ linear functions of slope '1', '-1' or '0'. ■

We remark that we can actually obtain a stronger characterization for the function g_e, which states that the well-defined portion has to be connected, and consists of at most three pieces with a graph as shown in Figure 5 (any piece can be degenerate). However, the above proposition suffices for our later arguments.

Case 2: e_{b} changes from edge e to e'. The change of the identity of e_{b} could be due to that the local maximum u_{b} moves continuously from e to a neighboring edge e' that shares an endpoint with e. Alternatively, it could be caused by a max-max type critical event: Specifically, let u_{d} be the up-fork saddle currently paired with the current birth point $u_{\mathsf{b}} = u \in e_{\mathsf{b}}$ generating the birth time b of (b, d) in the persistence diagram. At a max-max critical event, the up-fork saddle changes its pairing partner from $u_{\mathsf{b}} = u \in e_{\mathsf{b}}$ to another maximum u' in edge e'. Afterwards, the identify of e_{b} corresponding to the birth-time b will change to e'. At the time s_0 when the swapping happens, $d_{G_1,\mathbf{s}(s_0)}(u) = d_{G_1,\mathbf{s}(s_0)}(u')$. It then follows that $\mathsf{g}_e(s_0) = \mathsf{g}_{e'}(s_0)$; that is, s_0 corresponds to an intersection point between the graph of the function g_e and that of the function $\mathsf{g}_{e'}$. Since the function g_e consists of $O(1)$ linear pieces for any e, there are $O(1)$ intersection points between a pair of e and e' from E_1. We thus have:

Lemma 16 *There are $O(m^2)$ max-max critical events as the basepoint $\mathbf{s}$ moves along a fixed edge $\sigma \in E$.*

As in the case of tracking the death-time function d, our later analysis requires a stronger result bounding the descriptive complexity of the birth-time function $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$, starting from a birth-time $\mathsf{b}(0)$ from a fixed persistence pair $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$. In particular, we have the following proposition:

Proposition 17 *For a fixed $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$, the birth-time function $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$, tracking birth-time $\mathsf{b}(0)$, is piecewise linear with at most $O(m)$ pieces, and each linear piece has slope either '1', '-1', or '0'. Note that this also implies that the function b is 1-Lipschitz.*

Proof: We track the edge $e_{\mathsf{b}}(s)$ containing the maximum $u_{\mathsf{b}}(s)$ that gives rise to the birth-time $\mathsf{b}(s)$ for $s \in [0, \mathrm{Len}(\sigma)]$. As described above, $\mathsf{b}(s)$ will follow g_e for $e = e_{\mathsf{b}}(s)$ till e_{b} changes its identity to a new edge e'. Afterwards, $\mathsf{b}(s)$ will follow $\mathsf{g}_{e'}$ till next time e_{b} changes identity. By Proposition 15, b thus consists of a set linear linear functions, each of slope '1', '-1', or '0'. Note that this also implies that b is a 1-Lipschitz function.

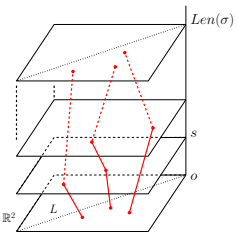
We now bound the descriptive complexity of b. Note that any break-point between two consecutive linear pieces in b of different slopes necessarily involve at least one linear piece of slope either '1' or '-1'. So we

can charge the number of break-points to the number of non-constant linear pieces in b. On the other hand, consider any non-constant piece from the function g_e: This piece can appear in the graph of the function b at most once, because b is 1-Lipschitz, and g_e has slope either '1' or '-1'. Since there are $O(m)$ edges in E_1, there are $O(m)$ non-constant linear pieces from all functions g_e, with $e \in E_1$, which implies that there are only $O(m)$ number of breakpoints in b. This proves the lemma. ■

Remark. The readers may have the following question: Recall that the function g_e could contain portions which are not well-defined. Suppose at some point, $e_{\mathsf{b}} = e$ and b is following the graph of g_e. What if we reach the endpoint s_0 of the well-defined portion of g_e? We note that when this happens, as detailed in the proof of Proposition 15, the corresponding maximum u_{b} currently is an endpoint say w_1 of e, and as the basepoint continues to change, either, $u_{\mathsf{b}}(s)$ moves to a neighboring edge e' of e incident on w_1; or, w_1 was a up-saddle prior to s_0 and at time s_0, the max $u_{\mathsf{b}} = w_1$ cancel with $u_{\mathsf{d}} = w_1$ (which we describe in Section 5.1.3 below). Overall, as the Stability Theorem guarantees, b is necessarily a continuous function.

5.1.3 Tracking the persistence pair $(\mathsf{b}, \mathsf{d}) : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}^2$.

Now consider the space $\Pi_\sigma := [0, \mathrm{Len}(\sigma)] \times \mathbb{R}^2$, where $\mathbb{R}^2$ denotes the birth-death plane: We can think of Π_σ as the stacking of all the planes containing persistence diagrams $\mathrm{P}_{\mathbf{s}(s)}$ for all $s \in [0, \mathrm{Len}(\sigma)]$. Hence we refer to Π_σ as the *stacked persistence-space*. For a fixed persistence pair $(\mathsf{b}, \mathsf{d}) \in \mathrm{P}_{\mathbf{s}(s)}$, as we vary $s \in [0, \mathrm{Len}(\sigma)]$, it traces out a *trajectory* $\pi = \{(s, \mathsf{b}(s), \mathsf{d}(s)) \mid s \in [0, [\mathrm{Len}(\sigma)]\} \in \Pi_\sigma$, which is the same as the "vines" introduced by Cohen-Steiner et al. [8]. By Propositions 13 and 17, the trajectory π is a polygonal curve with $O(n+m) = O(m)$ linear pieces. See the right figure for an illustration, where there are three trajectories in the stacked persistence diagrams.



In general, a trajectory (a vine) could appear or terminate within the range $(0, \mathrm{Len}(\sigma))$. Specifically, as we track a specific point in the persistence diagram, it is possible that the pair of critical points giving rise to this persistent-point may coincide and cease to exist afterwards. In this case, the corresponding trajectory (vine) hits the diagonal of the persistence diagram (since as the two critical points coincide with $u_{\mathsf{b}} = u_{\mathsf{d}}$, we have that $\mathsf{b} = \mathsf{d}$) and terminates. The inverse of this procedure indicates the creation of a new trajectory. Nevertheless, we can show that there can be $O(n+m) = O(m)$ total number of trajectories in the stacked persistence diagrams (whether they span the entire range of $s \in [0, \mathrm{Len}(\sigma)]$ or not). We conclude with the following result.

Theorem 18 *Let $\sigma \in E_1$ be an arbitrary edge from the metric graph (G_1, d_{G_1}). As the basepoint $\mathbf{s}$ moves from one endpoint to another endpoint of σ by $\mathbf{s} : [0, \mathrm{Len}(\sigma)] \to \sigma$, the persistence-points in the persistence diagram $\mathrm{P}_{\mathbf{s}(s)}$ of the distance function $d_{G_1, \mathbf{s}(s)}$ form $O(m)$ number of trajectories in the stacked persistence-space Π_σ. Each trajectory is a polygonal curve of $O(m)$ number of linear segments.*

A symmetric statement holds for metric graph (G_2, d_{G_2}).

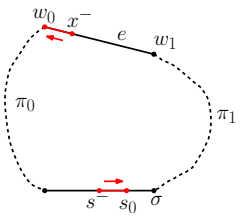
Proof of Theorem 18. As the basepoint $\mathbf{s}(s)$ moves along an edge σ with $s \in [0, \mathrm{Len}(\sigma)]$, we can think of the distance function $d_{G_1, \mathbf{s}(s)}$ as a time-varying function with time range $[0, \mathrm{Len}(\sigma)]$. For a general time-varying function, as we track a specific point in the persistence diagram [8], it is possible that the pair of critical points giving rise to this persistent-point may coincide and cease to exist afterwards. As already mentioned above, in this case, the trajectory (the vine) hits the diagonal of the persistence diagram (since as the two critical points coincide $u_{\mathsf{b}} = u_{\mathsf{d}}$, we have that $\mathsf{b} = \mathsf{d}$) and terminates. The inverse of this procedure indicates the creation of a new trajectory. Hence a trajectory in the stacked persistence-diagrams may not span the entire range $[0, \mathrm{Len}(\sigma)]$.

We claim that there can be only $O(m)$ number of trajectories in the stacked persistence diagram. In particular, first, note that at time $s = 0$, there can be $O(n + m) = O(m)$ number of persistence-points in the persistence diagram $\mathrm{P}_{\mathbf{s}(0)}$ for basepoint $s(0)$. This is because that for a fixed basepoint, by Lemma 14, there can be only $O(n + m)$ number of local maxima for the distance function $d_{G_1,\mathbf{s}(0)} : \mathrm{G}_1 \to \mathbb{R}$, thus generating $O(m)$ number of persistence-points in the persistence diagram. As a result, there can be at most $O(m)$ trajectories spanning through the entire range $[0, \mathrm{Len}(\sigma)]$.

We next bound the number of trajectories not spanning the entire range. That is, these are the trajectories created or terminated at some time in $(0, \mathrm{Len}(\sigma))$. For any such trajectory, assume without loss of generality that it tracks a persistence-point (b, d), and terminates at time s_0. (The case of creation of a new trajectory is symmetric.) At this point, it is necessary that the two critical points u_{b} (a local maximum) and u_{d} (a up-fork saddle) coincide. By Observation 10, the death-point u_{d} must be a graph node, say $w_0 \in V_1$. Hence $u_{\mathsf{b}} = w_0$ as well; that is, w_0 is also a maximum of the distance function $d_{G_1,\mathbf{s}(s_0)}$. We show that for a fixed graph node w_0, such a scenario can happend at most once.

Lemma 19 *For a fixed graph node $w_0 \in V_1$, the birth-point u_{b} and death-point u_{d} can coincide at w_0 at most once as s varies in the range $[0, \mathrm{Len}(\sigma)]$.*

Proof: As above, assume the trajectory hits the diagonal of the persistence diagram at time s_0, and w_0 is the corresponding coincided birth- and death-points. Suppose at $s^- < s_0$ infinitesimally close to s_0, the corresponding local maximum $x^- = u_{\mathsf{b}}(s^-)$ comes from edge e incident on w_0. Assume without loss of generality that s^- is sufficiently close to s_0 such that there is no critical event of any kind and the local maximum $u_{\mathsf{b}}(s)$ approach continuously to w_0 as s tends to s_0 (i.e, for $s \in (s^-, s_0)$).



Let w_1 be the other endpoint of e. Since x^- is a maximum of $d_{G_1,\mathbf{s}(s^-)}$, by Lemma 14, there are two shortest paths from $\mathbf{s}(s^-)$ to x^- passing through w_0 and w_1, which we denote by π_0 and π_1, respectively. We show that π_0 and π_1 in fact are disjoint other than at their endpoints x^- and $\mathbf{s}(s^-)$. See the right figure for an illustration.

Indeed, consider the shortest path tree T^- rooted at $\mathbf{s}(s^-)$, and let z be the common ancestor of w_0 and w_1; z is necessarily a graph node of V_1 unless $z = \mathbf{s}(s^-)$. If z is a graph node, then as s varies, the distance to z either increases or decreases. However, the shortest path distance from z to w_0 and to w_1 remain the same. Hence the distance functions $\mathrm{g}_{w_0} = d_{G_1,\mathbf{s}(s)}(w_0)$ and $\mathrm{g}_{w_1} = d_{G_1,\mathbf{s}(s)}(w_1)$ either both increase or both decrease (because the shortest distance to them is the shortest distance to z plus the shortest distance from z to each of them). However, this falls into case (1) in the proof of Proposition 15, which means that the local maximum necessarily remains the same at x^- as s moves from s^- to s_0, and will not move to w_0. Contradiction. As such, z must be $\mathbf{s}(s^-)$. In other words, the two shortest paths π_1 and π_2 meet only at $\mathbf{s}(s^-)$ and x^-: Their concatenation form a simple loop C where x^- and $\mathbf{s}(s^-)$ are a pair of *antipodal points* along this loop (i.e, they bisect C). As s moves to s_0, its corresponding local maximum $u_{\mathsf{b}}(s)$ remains the antipodal point of $\mathbf{s}(s)$ and moves towards w_0, and w_0 is the antipodal point of $\mathbf{s}(s_0)$.

In other words, let v_1, v_2 denote the two endpoints of the edge σ where the basepoint $\mathbf{s}$ lies in. Since w_0 is the antipodal point of $\mathbf{s}(s_0)$, we have that $d_{G_1}(v_1, w_0) + s_0 = d_{G_1}(v_2, w_0) + \mathrm{Len}(\sigma) - s_0$. Hence there is only one possible value for s_0. This proves the lemma. ∎

It then follows that there can be at most $O(n)$ number of trajectories not spanning the entire time range $[0, \mathrm{Len}(\sigma)]$ (created or terminated in the stacked persistence diagrams). Putting everything together, we have that there are at most $O(n + m) = O(m)$ trajectories in the stacked persistence diagrams as the basepoint $\mathbf{s}$ moves in an edge $\sigma \in E_1$. Combining this with Propositions 13 and 17, Theorem 18 then follows.

5.2 Computing $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2)$

Given a pair of edges $\sigma_s \in \mathrm{G}_1$ and $\sigma_t \in \mathrm{G}_2$, as before, we parameterize the basepoints $\mathbf{s}$ and $\mathbf{t}$ by the arc-length parameterization of σ_s and σ_t; that is: $\mathbf{s} : [0, L_s] \to \sigma_s$ and $\mathbf{t} : [0, L_t] \to \sigma_t$ where $L_s = \mathrm{Len}(\sigma_s)$ and $L_t = \mathrm{Len}(\sigma_t)$. We now introduce the following function to help compute $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2)$:

Definition 20 *The* bottleneck distance function $F_{\sigma_s,\sigma_t} : \Omega \to \mathbb{R}$ *is defined as* $F_{\sigma_s,\sigma_t}(s,t) \mapsto d_B(\mathrm{P}_{\mathbf{s}(s)}, \mathrm{Q}_{\mathbf{s}(t)})$. *For simplicity, we sometimes omit* σ_s, σ_t *from the subscript when their choices are clear from the context.*

Recall that $\mathcal{C} = \{\mathrm{P}_{\mathbf{s}} \mid \mathbf{s} \in \mathrm{G}_1\}$, $\mathcal{F} = \{\mathrm{Q}_{\mathbf{t}} \mid \mathbf{t} \in \mathrm{G}_2\}$, and by Definition 2:

$$\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) = \max\{\max_{\mathrm{P}\in\mathcal{C}} \min_{\mathrm{Q}\in\mathcal{F}} d_B(\mathrm{P}, \mathrm{Q}),\ \max_{\mathrm{P}\in\mathcal{F}} \min_{\mathrm{P}\in\mathcal{C}} d_B(\mathrm{P}, \mathrm{Q})\ \}.$$

Below we focus on computing $\vec{d}_H(\mathcal{C}, \mathcal{F}) := \max_{\mathrm{P}\in\mathcal{C}} \min_{\mathrm{Q}\in\mathcal{F}} d_B(\mathrm{P}, \mathrm{Q})$, and the treatment of $\vec{d}_H(\mathcal{F}, \mathcal{C}) := \max_{\mathrm{P}\in\mathcal{F}} \min_{\mathrm{P}\in\mathcal{C}} d_B(\mathrm{P}, \mathrm{Q})$ is symmetric. It is easy to see:

$$\vec{d}_H(\mathcal{C}, \mathcal{F}) = \max_{\mathrm{P}\in\mathcal{C}} \min_{\mathrm{Q}\in\mathcal{F}} d_B(\mathrm{P}, \mathrm{Q}) = \max_{\sigma_s\in\mathrm{G}_1} \max_{s\in[1,L_s]} \min_{\sigma_t\in\mathrm{G}_2} \min_{t\in[1,L_t]} F_{\sigma_s,\sigma_t}(s,t). \tag{8}$$

In what follows, we present the descriptive complexity of F_{σ_s,σ_t} for a fixed pair of edges $\sigma_s \in \mathrm{G}_1$ and $\sigma_t \in \mathrm{G}_2$ in Section 5.2.1, and show how to use it to compute the persistence-distortion in Section 5.2.2.

5.2.1 One pair of edges $\sigma_s \in \mathrm{G}_1$ and $\sigma_t \in \mathrm{G}_2$.

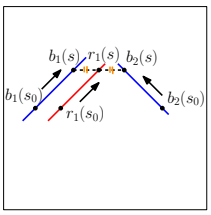
Recall that we call the plane containing the persistence diagrams as the birth-death plane, and for persistence-points in this plane, we follow the literature and measure their distance under the L_∞-norm (recall Definition 1). From now on, we refer to persistence-points in $\mathrm{P}_{\mathbf{s}(s)}$ as *red points*, while persistence-points in $\mathrm{Q}_{\mathbf{t}(t)}$ as *blue points*. As s and t vary, the red and blue points move in the birth-death plane. By Theorem 18, the movement of each red (or blue) point traces out a polygonal curve with $O(m)$ segments (which are the projections of the trajectories from the stacked persistence diagrams onto the birth-death plane).

Set $\Omega := [0, L_s] \times [0, L_t]$ and we refer to it as the *s-t domain*. For a point $(s,t) \in \Omega$, the function value $F(s,t) (= F_{\sigma_s,\sigma_t}(s,t)) = d_B(\mathrm{P}_{\mathbf{s}(s)}, \mathrm{Q}_{\mathbf{t}(t)})$ is the bottleneck distance between the set of red and the set of blue points (with the addition of diagonals) in the birth-death plane. To simplify the exposition, in what follows we ignore the diagonals from the two persistence diagrams and only consider the bottleneck matching between red and blue points.

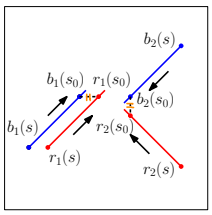
Let $r^*(s) \in \mathrm{P}_{\mathbf{s}(s)}$ and $b^*(t) \in \mathrm{Q}_{\mathbf{t}(t)}$ be the pair of red-blue points from the bottleneck matching between $\mathrm{P}_{\mathbf{s}(s)}$ and $\mathrm{Q}_{\mathbf{t}(t)}$ such that $d_\infty(r^*(s), b^*(t)) = d_B(\mathrm{P}_{\mathbf{s}(s)}, \mathrm{Q}_{\mathbf{t}(t)})$. We call $(r^*(s), b^*(t))$ *the bottleneck pair (of red-blue points) w.r.t.* (s,t). As s and t vary continuously, red and blue points move continuously in the birth-death plane. The distance between any pair of red-blue points change continuously. The bottleneck pair between $\mathrm{P}_{\mathbf{s}(s)}$ and $\mathrm{Q}_{\mathbf{t}(t)}$ typically remains the same till certain *critical values* of the parameters (s,t).

Characterizing critical (s,t) values. Given (s,t), consider the optimal bottleneck matching $C^*(s,t) : \mathrm{P}_s \times \mathrm{Q}_t$. For any corresponding pair $(r(s), b(t)) \in C^*(s,t)$, $d_\infty(r(s), b(t)) \le d_\infty(r^*(s), b^*(t))$. Suppose $r^*(s) = r_1(s)$ and $b^*(t) = b_1(t)$. As (s,t) varies in Ω, the bottleneck pair $(r^*(s), b^*(t))$ may change only when:

- *(case-1)*: $(r_1(s), b_1(t))$ ceases to be a matched pair in the optimal matching $C^*(s,t)$; or
- *(case-2)*: $(r_1(s), b_1(t))$ is still in C^*, but another matched pair $(r_2(s), b_2(t))$ becomes the bottleneck pair.



(case-1)



(case-2)

At the time (s_0, t_0) that either cases above happens, it is necessary that there are two red-blue pairs, one of which being (r_1, b_1), and denoting the other one by (r_2, b_2), such that $d_\infty(r_1(s_0), b_1(t_0)) = d_\infty(r_2(s_0), b_2(t_0))$. (For case-1, we have that either $r_2 = r_1$ or $b_2 = b_1$.) Hence all critical (s,t) values are included in those (s,t) values for which two red-blue pairs of persistence-points acquire equal distance in the birth-death plane. Let

$$X_{(r_1,b_1),(r_2,b_2)} := \{(s,t) \mid d_\infty(r_1(s), b_1(t)) = d_\infty(r_2(s), b_2(t))\}$$

denote the set of *potential critical (s,t)-values generated by* (r_1, b_1) *and* (r_2, b_2). To describe $X_{(r_1,b_1),(r_2,b_2)}$, we first consider, for a fixed pair of red-blue points (r, b), the distance function $D_{r,b} : [0, L_s] \times [0, L_t] \to \mathbb{R}$ defined as the distance between this pair of red and blue points in the birth-death plane, that is, $D_{r,b}(s,t) := d_\infty(r(s), b(t))$ for any $(s,t) \in \Omega$.

In particular, recall that by Theorem 18, $r : [0, L_s] \to \mathbb{R}^2$ (resp. $b : [0, L_t] \to \mathbb{R}^2$) is continuous and piecewise-linear with $O(m)$ segments. In other words, the range $[0, L_s]$ (resp. $[0, L_t]$) can be decomposed to $O(m)$ intervals such that within each interval, r moves (resp. b moves) along a line in the birth-death plane with fixed speed. Hence combining Propositions 13 and 17, we have the following:

Proposition 21 *The s-t domain Ω can be decomposed into an $O(m) \times O(m)$ grid such that, within each of the $O(m^2)$ grid cell, $D_{r,b}$ is piecewise-linear with $O(1)$ linear pieces, and the partial derivative of each piece w.r.t. s or w.r.t. t is either '1', '-1', or '0'.*

Proof: Let $\mathcal{I}_s$ (resp. $\mathcal{I}_t$) denote the decomposition of $[0, L_s]$ (resp. $[0, L_t]$) into $O(m)$ intervals within each of which the red (persistence) point $r \in \mathrm{P_s}$ (resp. the blue persistence point $b \in \mathrm{Q_t}$) moves along a line in the birth-death plane. In fact, by Propositions 13 and 17, we also have that the birth-coordinate $r.x$ for the red point r either increases or decreases at the unit speed (w.r.t. the parameter s), and the death-coordinate $r.y$ of r either increases or decreases at the unit speed, or is stationary. Similar statements hold for the blue point t. Since $D_{r,b}(s,t) = d_\infty(r(s), b(t)) = \max\{|r.x(s) - b.x(t)|, |r.y(s) - b.y(t)|\}$, it follows that for a fixed interval $I_1 \in \mathcal{I}_s$ and $I_2 \in \mathcal{I}_t$, $D_{r,b} : I_1 \times I_2 \to \mathbb{R}$ is piecewise-linear function with $O(1)$ linear pieces, where the partial derivative of each piece w.r.t. s or to t is either '1', '-1', or '0'. ■

Given two pairs of red-blue pairs (r_1, b_1) and (r_2, b_2), the set $X_{(r_1,b_1),(r_2,b_2)}$ of potential critical (s,t) values generated by them corresponds to the intersection of the graph of D_{r_1,b_1} and that of D_{r_2,b_2}. By overlaying the two $O(m) \times O(m)$ grids corresponding to D_{r_1,b_1} and D_{r_2,b_2} as specified by Proposition 21, we obtain another grid of size $O(m) \times O(m)$ and within each cell, the intersection of the graphs of D_{r_1,b_1} and D_{r_2,b_2} has $O(1)$ complexity. Hence, we have:

Corollary 22 *The set $X_{(r_1,b_1),(r_2,b_2)} \subseteq \Omega$ consists of a set of polygonal curves in the s-t domain Ω with $O(m^2)$ total complexity.*

Consider the arrangement $Arr(\Omega)$ of the set of curves in $\mathcal{X} = \{X_{(r_1,b_1),(r_2,b_2)} \mid r_1, r_2 \in \mathrm{P_s}, b_1, b_2 \in \mathrm{Q_t}\}$. Since there are altogether $O(m^4) \times O(m^2) = O(m^6)$ segments in $\mathcal{X}$, we have that the arrangement $Arr(\Omega)$ has $O(m^{12})$ complexity; that is, there are $O(m^{12})$ number of vertices, edges and polygonal cells. However, this arrangement $Arr(\Omega)$ is more refined than necessary. Specifically, within a single cell $c \in Arr(\Omega)$, the *entire* bottleneck matching C^* does not change. By a much more sophisticated argument, we can prove the following:

Proposition 23 *There is a planar decomposition $\Lambda(\Omega)$ of the s-t domain Ω with $O(m^8)$ number of vertices, edges and polygonal cells such that as (s,t) varies within in each cell $c \in \Lambda(\Omega)$, the pair of red-blue persistence points that generates the bottleneck pair (r^*, b^*) remains the same.*

Furthermore, the decomposition $\Lambda(\Omega)$, as well as the bottleneck pair (r^*, b^*) associated to each cell, can be computed in $O(m^{9.5} \log m)$ time.

Proof: First, consider the decomposition of Ω into maximal cells within each of which the bottleneck pair does not change its identity. We refer to each such cell as a *fixed-bottleneck-pair cell*. Consider such a cell c and assume that within this cell c the bottleneck pair is $(r^*, b^*) = (r_1, b_1)$. The boudary of c is a polygonal curve γ, each linear segment of which corresponds to potential critical (s,t)-values where the red-blue pair (r_1, b_1) has equal distance with some other red-blue pair, say (r_2, b_2). Each vertex, say v in this boundary curve γ is where two segments meet, say one corresponding to (r_1, b_1) and (r_2, b_2), and the other corresponding to (r_1, b_1) and (r_3, b_3).

The vertices in γ are of two types: (Type-1): $(r_2, b_2) = (r_3, b_3)$ where v is also a vertex in a polygonal curve from $X_{(r_1,b_1),(r_2,b_2)}$; (Type-2): remaining case where $v = (s_0, t_0)$ represents the moment the red-blue pair (r_1, b_1) has distance equal to that of the two other red-blue pairs: (r_2, b_2) and (r_3, b_3).

Intuitively, the segment of the boundary curve γ, where (r_1, b_1) and (r_2, b_2) have equal distances, represents the boundary between two cells c and c' such that (i) the bottleneck pair is (r_1, b_1) when $(s, t) \in c$, (ii) as we cross this curve, the bottleneck pair may change to (r_2, b_2) when $(s, t) \in c'$, and (iii) along this curve, there is ambiguity and the bottleneck matching can be either (r_1, b_1) or (r_2, b_2).

Type-1 vertices are vertices from the same polygonal curve of where (r_1, b_1) and (r_2, b_2) are at equal distances. Type-2 vertices are where this curve meets another curve representing the (s,t)-values where (r_1, b_1) and (r_3, b_3) are at equal distances. Hence Type-2 vertices represent the places where *three* fixed-bottleneck-pair cells meet.

By Corollary 22, there are $O(m^4) \times O(m^2) = O(m^6)$ number of Type-1 vertices. We now show that the number of Type-2 vertices is $O(m^8)$ and we can compute all Type-2 vertices in $O(m^{9.5} \log m)$ time.

Note that each Type-2 vertex is induced by three red points (r_1, r_2, r_3) and three blue points (b_1, b_2, b_3). First, enumerate all $O(m^6)$ possible triples of red-blue pairs. For each triple (r_1, b_1), (r_2, b_2) and (r_3, b_3), consider the graphs of functions D_{r_1,b_1}, D_{r_2,b_2}, and D_{r_3,b_3}. The intersection of all three graphs are a super-set for Type-2 vertices generated by (r_1, b_1), (r_2, b_2) and (r_3, b_3). It follows from Proposition 21 that there are $O(m^2)$ intersection points of the three graphs – Specifically, we overlay the three $O(m) \times O(m)$ grids as specified by Proposition 21, and within each cell of the resulting grid which is still of size $O(m) \times O(m)$, each function D_{r_i,b_i} has $O(1)$ complexity, and thus they produce $O(1)$ intersection points. For each intersection point, we spend $O(m^{1.5} \log m)$ time using the modified algorithm of [13] to compute its bottleneck matching, and check whether this is a valid Type-2 vertex or not. Altogether, since there are $O(m^6)$ triples we need to check, there can be $O(m^8)$ Type-2 vertices, and they can be identified in $O(m^{9.5} \log m)$ time. Let Σ denote the resulting set of Type-2 vertices.

With Σ computed, we next construct the decomposition $\Lambda(\Omega)$ of Ω into fixed-bottleneck-pair cells. To do this, we simply scan all vertices in Σ from left to right. For each vertex $v \in \Sigma$ corresponding to the three red-blue pairs (r_1, b_1), (r_2, b_2) and (r_3, b_3), we know that locally, there are three branches from v: one from X_{r_1,b_1,r_2,b_2}, one from X_{r_1,b_1,r_3,b_3} and one from X_{r_2,b_2,r_3,b_3}. We simply trace each such curve till we meet another vertex from Σ. Now consider the graph whose nodes are Type-2 vertices, and arcs are polygonal curves connecting them. We can use any graph traversal strategy (such as BFS) to traverse all arcs and thus connecting nodes. The total time is

$$O(\text{Time to traverse the graph}) + O(\text{Time to trace out all arcs}).$$

Since this graph is planar with $O(m^8)$ vertices, there are $O(m^8)$ arcs as well. Hence O(Time to traverse the graph) $= O(m^8)$. The time to trace an arc from one Type-2 vertex to the other is proportional to the complexity of this polygonal curve. We charge this time to the number of interior Type-1 vertices in this arc, as well as the two boundary Type-2 vertices of this arc. By Corollary 22, the polygonal curves from $\mathcal{X}$ has $O(m^2) \times O(m^4) = O(m^6)$ total complexity. Hence the total time to trace out all arcs is also bounded by $O(m^8)$. Putting everything together, we have that, once the Type-2 vertices are computed, we can construct $\Lambda(\Omega)$ in time $O(m^8)$. This completes the proof. ∎

Our goal is to compute the bottleneck distance function $F : \Omega \to \mathbb{R}$ introduced at the beginning of this subsection where $F(s,t) \mapsto d_B(\mathrm{P}_{\mathbf{s}(s)}, \mathrm{Q}_{\mathbf{t}(t)}) = d_\infty(r^*(s), b^*(t))$, so as to further compute persistence-distortion distance using Eqn (8). To do this, we need to further refine the decomposition $\Lambda(\Omega)$ from Proposition 23 to another decomposition $\widehat{\Lambda}(\Omega)$ as described below so that within each cell, the bottleneck distance function F_{σ_s,σ_t} can be described by a single linear function.

Theorem 24 *For a fixed pair of edges $\sigma_s \in \mathrm{G}_1$ and $\sigma_t \in \mathrm{G}_2$, there is a planar polygonal decomposition $\widehat{\Lambda}(\Omega)$ of the s-t domain Ω of $O(m^{10})$ complexity such that within each cell, the bottleneck distance function F_{σ_s,σ_t} is linear. Furthermore, one can compute this decomposition $\widehat{\Lambda}(\Omega)$ as well as the function F_{σ_s,σ_t} in $O(m^{10} \log m)$ time.*

Proof: By Proposition 23, given any cell $c \in \Lambda(\Omega)$, the bottleneck pair (r^*, b^*) remains the same. In other words, let $r_c = r^*$ and $b_c = b^*$ for any $(s,t) \in c$. We have $F(s,t) = d_\infty(r_c(s), b_c(t))(= D_{r_c,b_c}(s,t))$ for $(s,t) \in c$. Let $A_{r_c,b_c}(\Omega)$ be the decomposition of Ω such that within each cell of A_{r_c,b_c}, the function D_{r_c,b_c} is a linear function. By Proposition 21, A_{r_c,b_c} consists of $O(m^2)$ cells, edges and vertices. Hence we can further decompose (refine) the cell c to be the intersection of c with A_{r_c,b_c}. We perform this refinement for each cell $c \in \Lambda(\Omega)$, and denote the resulting decomposition as $\widehat{\Lambda}(\Omega)$. By construction, the bottleneck distance F within each cell of $\widehat{\Lambda}(\Omega)$ is a single linear function.

Next, we bound the complexity of $\widehat{\Lambda}(\Omega)$. First, note that the number of newly added vertices within the interior of a cell $c \in \Lambda(\Omega)$ is bounded from above by $O(m^2)$, since each such vertex is a vertex from A_{r_c,b_c}. While there can be $O(m^8)$ number of cells in $\Lambda(\Omega)$, there can only be $O(m^2)$ choices of bottleneck pairs (r^*, b^*)s. Hence the total number of vertices in the interior cells in $\Lambda(\Omega)$ is $O(m^4)$.

What remains is to bound the number of vertices along edges of $\Lambda(\Omega)$. To this end, notice that each edge $e \in \Lambda(\Omega)$ has two incident cells c_1 and c_2. Any newly added vertex in e must be either an intersection between e with some edge in $A_{r_{c_1},b_{c_1}}$, or with some edge in $A_{r_{c_2},b_{c_2}}$. Hence the total number of such vertices on e is $O(m^2)$. Since there are $O(m^8)$ edges in $\Lambda(\Omega)$, the total number of newly added vertices is at most $O(m^{10})$. Thus the complexity of $\widehat{\Lambda}(\Omega)$ is $O(m^{10})$.

Finally, the refined decomposition $\widehat{\Lambda}(\Omega)$ can be computed in $O(m^{10} \log m)$ time. Specifically, first, it takes $O(m^{9.5} \log m)$ time to compute $\Lambda(\Omega)$ by Proposition 23. Next, for each cell c with k number of boundary edges, it takes $O((k+m^2) \log m)$ time to compute the intersection $c \cap A_{r_c,b_c}$. Summing over all cells in $\Lambda(\Omega)$ gives the claimed time complexity. ■

5.2.2 Final algorithm and analysis.

We now aim to compute $\vec{d}_H(\mathcal{C}, \mathcal{F})$ using Eqn (8). First, for a fixed edge $\sigma_s \in \mathrm{G}_1$, consider the following *lower-envelop function*

$$\mathcal{L} : [0, L_s] \to \mathbb{R} \text{ where } \mathcal{L}(s) \mapsto \min_{\sigma_t \in \mathrm{G}_2} \min_{t \in [0, L_t]} F(s,t), \quad (9)$$

where recall L_s and L_t denote the length of edge σ_s and σ_t respectively. The reason behind the name "lower-envelop function" will become clear shortly.

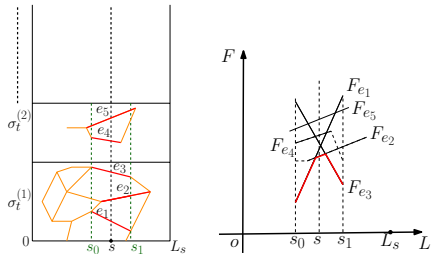


Figure 6: (a) s-t domains for $\sigma_s \in E_1$ and edges $\sigma_t^{(j)} \in E_2$. (b) $\mathcal{L}(s)$ is the lowerest value along any F_{e_ℓ}.

Now for each $\sigma_t \in \mathrm{G}_2$, consider the polygonal decomposition $\widehat{\Lambda}(\Omega)$ as described in Theorem 24. Since within each cell the bottleneck distance function F is a linear piece, we know that for any s, the extreme of $F(s,t)$ for all possible $t \in [0, L_t]$ must come from some edge in $\widehat{\Lambda}(\Omega)$. In other words, to compute the function $\min_{t \in [0,L_t]} F(s,t)$ at any $s \in [0, L_s]$, we only need to inspect the function F restricted

to edges in the refined decomposition $\widehat{\Lambda}(\Omega_{\sigma_s,\sigma_t})$ for the s-t domain $\Omega_{\sigma_s,\sigma_t} = [0, L_s] \times [0, L_t]$. Take any edge e of $\widehat{\Lambda}(\Omega_{\sigma_s,\sigma_t})$, define $\pi_e : [0, L_s] \to [0, L_t]$ such that $(s, \pi_e(s)) \in e$. Now denote by the function $F_e : [0, L_s] \to \mathbb{R}$ as the projection of F onto the first parameter $[0, L_s]$; that is, $F_e(s) := F(s, \pi_e(s))$. Let $E_{\sigma_s} := \{e \in \widehat{\Lambda}(\Omega_{\sigma_s,\sigma_t}) \mid \sigma_t \in \mathrm{G}_2\}$ be the union of edges from the refined decompositions of the s-t domain formed by σ_s and any edge σ_t from G_2. It is easy to see that (see Figure 6):

$$\mathcal{L}(s) = \min_{e \in E_{\sigma_s}} F_e(s); \text{ that is, } \mathcal{L} \text{ is the lower-envelop of linear functions } F_e \text{ for all } e \in E_{\sigma_s}.$$

There are $O(m)$ edges in G_2, thus by Theorem 24 we have $|E_{\sigma_s}| = O(m^{11})$. The lower envelop $\mathcal{L}$ of $|E_{\sigma_e}|$ number of linear functions (linear segments), is a piecewise-linear function with $O(|E_{\sigma_s}| = O(m^{11})$ complexity and can be computed in $O(|E_{\sigma_s}| \log |E_{\sigma_s}|) = O(m^{11} \log m)$ time. Finally, from Eqn (8), $\vec{d}_H(\mathcal{C}, \mathcal{F}) = \max_{\sigma_s \in \mathrm{G}_1} \max_{s \in [0, L_s]} \mathcal{L}(s)$. Since there are $O(m)$ choices for σ_s, we conclude with the following main result.

Theorem 25 *Given two metric graphs (G_1, d_{G_1}) and (G_2, d_{G_2}) with n total vertices and m total edges, we can compute the persistence-distortion distance $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ between them in $O(m^{12} \log n)$ time.*

We remark that if both input graphs are metric trees, then we can compute their persistence-distortion distance more efficiently in $O(n^8 \log n)$ time.

6 Preliminary Experiments

We show two sets of preliminary experimental results. The first experiment aims to demonstrate the stability of the proposed persistence-distortion distance, by showing that the persistence-distortion distance between a graph and a noisy sample of it remains stable w.r.t. the noise added. In the second experiment, we apply our persistence-distortion distance to compare a set of surface models, using simply the 1-skeleton of their mesh models, and show that this distance is robust again non-rigid but near-isometric deformations (such as different poses between humans, or between wolfs and horses), while it still can differentiate different models. In both experiments, to improve the efficiency, we only compute the persistence diagrams to a subset of graph nodes of input graphs, and obtain an even coarser version of the discrete persistence-distortion distance for input graphs.

We also point out that in our experiments, we compute the 0-th *zigzag persistence diagram* for each basepoint. However, we observe little difference in results if only the 0-th standard persistence diagrams are used.

Experiment 1. The first experiment aims to demonstrate the stability of the proposed persistence-distortion distance. Specifically, we consider a set of noisy points P sampled from a hidden graph $G = (V, E)$, and compute the the Rips complex $\mathcal{R}^r(P)$ of P as an approximation of the hidden graph G. The hidden graph G taken in this case is a part of the Athens road network. We obtain a noisy sample P_ε by perturbing each point from a discrete sample of G uniformly randomly within error ε. We then build a Rips complex $\mathcal{R}^{r(\varepsilon)}(P_\varepsilon)$ with parameter $r(\varepsilon) = \frac{3\varepsilon}{2}$. We then treat the 1-skeleton $\mathcal{R}_1^{r(\varepsilon)}$ of $\mathcal{R}^{r(\varepsilon)}(P_\varepsilon)$ as a metric graph, and this metric graph $\mathcal{R}_1^{r(\varepsilon)}$ offers a noisy approximation of the hidden graph G. Examples of G and $\mathcal{R}_1^{r(\varepsilon)}$s are shown in Figure 7 (a), (b) and (c).

To speedup the computation, we compute only the persistence diagrams at a set of δ-sparse subsamples $Q \subset V$ and $Q' \subset P_\varepsilon$, and only use points in Q and Q' as basepoints. Specifically, Q is obtained by the following randomized procedure: Take a random permutation of V. Process each node $v_i \in V$ in this order. We add v_i into Q only if its distance to current points in Q is larger than δ. Q' is obtained from P_ε in a similar

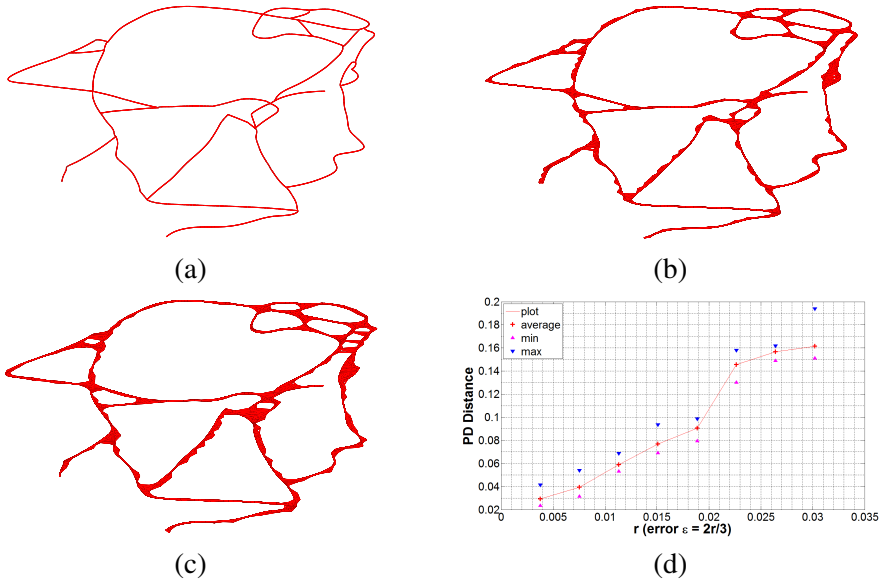


Figure 7: (a) Hidden graph (Athens road map) G. (b) and (c) noisy 1-skeleton of Rips complex $\mathcal{R}^r$ for $r = 0.1$ and $r = 0.13$ respectively. (d) The growth of the persistence-distortion distance $\mathrm{d_{PD}}(\mathcal{R}_1^r, G)$ w.r.t. the parameter r in the Rips complex $\mathcal{R}^r$. Note that the noise level is $\varepsilon = \frac{2r}{3}$. The vertical range (two triangle-points) shows the max and min $\mathrm{d_{PD}}$ values for 10 different re-samples (with noise) (i.e, for each noise level, we take 10 sets of samples) – the middle curve is the average $\mathrm{d_{PD}}$ values of these 10 sets.

manner. We use a random order so as to further demonstrate the robustness against different discretization. By Theorem 3, one can show that this incurs at most 12δ error in the estimation of $\mathrm{d_{PD}}(\mathcal{R}_1^{r(\varepsilon)}, G)$. In our experiments, the size of the subsmapled sets Q and Q' are usually between 150 and 200 points. The time required for computing the discrete persistence-distortion distance using Q and Q' as basepoints, is observed to be from $20 \sim 30$ seconds.

In Figure 7 (d), we show the growth of the persistence-distortion distance $\mathrm{d_{PD}}(\mathcal{R}_1^{r(\varepsilon)}, G)$ with respect to the change of the noise level ε; recall that the parameter $r(\varepsilon) = \frac{3\varepsilon}{2}$. We note that $\mathrm{d_{PD}}(\mathcal{R}_1, G)$ grows roughly proportionally to the noise level, demonstrating its stability. We note that there is a small jump of $\mathrm{d_{PD}}(\mathcal{R}_1, G)$ from $r(\varepsilon) = 0.02$ to $r(\varepsilon) = 0.025$. This is because when $r(\varepsilon)$ increases, small loops (1st homology features) get created in the top-right part of the graph (from Figure 7 (b) to (c)). This shows that our persistence-distortion distance captures such small topological changes.

Experiment 2. In the second experiment, we apply our persistence-distortion distance to compare surface meshes of different geometric models, some of which are different poses of the same object. The set of surface models are shown in Figure 8. For each surface model, we take the 1-skeleton $K_i = (V_i, E_i)$ of its surface mesh as input. Like in the first experiment, we also compute only the persistence diagrams at a set of δ-sparse subsamples $Q_i \subset V_i$ from input surface mesh constructed by a randomized decimation procedure. Again by Theorem 3, one can show that this incurs at most 12δ error in the estimation of $\mathrm{d_{PD}}(K_i, K_j)$.

Figure 9 shows the matrix of pairwise persistence-distortion distance between all pairs of models. Because the subsamples are generated by a randomized procedure, the resulting persistence-distortion distance for the same two graphs may be non-zero (as the basepoints chosen may be different). Nevertheless, note that distance values at the diagonal are usually small, implying that persistence-distortion is stable against different discretization of the same graph.

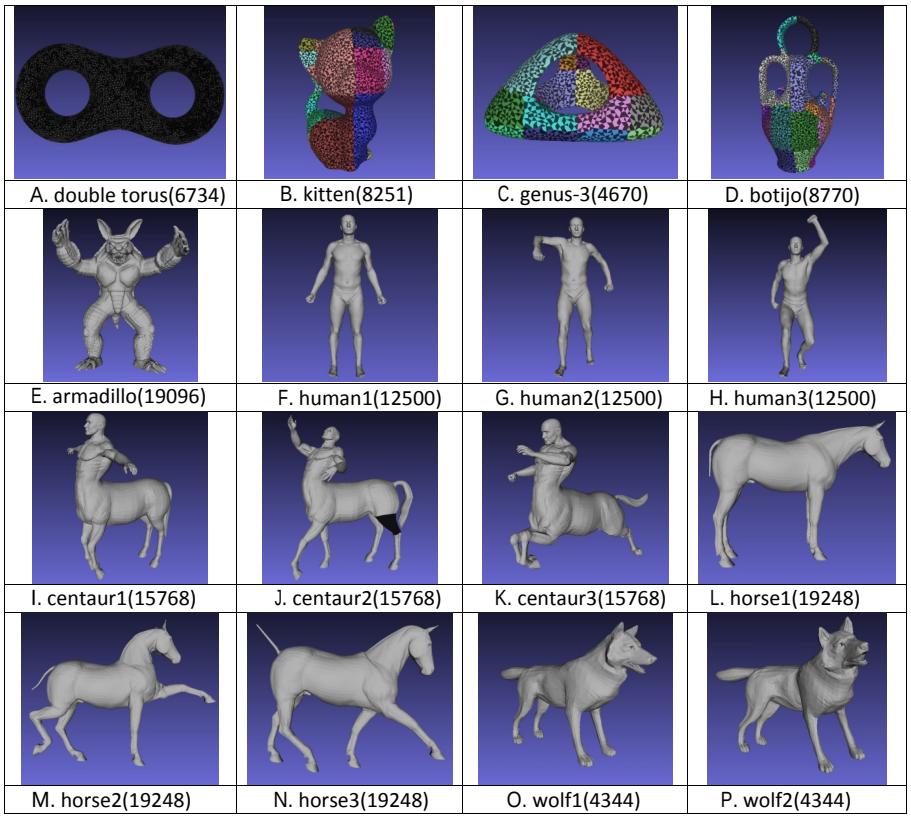


Figure 8: Models used for comparison.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0.0265	0.2402	0.2905	0.2653	0.3038	0.3673	0.3601	0.3400	0.2706	0.2533	0.2430	0.2425	0.2397	0.2492	0.2435	0.2414
B	0.2402	0.0416	0.3235	0.2652	0.2643	0.2739	0.2695	0.2722	0.2697	0.2500	0.2360	0.2387	0.2349	0.2425	0.2405	0.2404
C	0.2905	0.3235	0.0417	0.2619	0.4011	0.3985	0.3928	0.3921	0.4013	0.4013	0.3892	0.3851	0.3928	0.3924	0.3947	0.3995
D	0.2653	0.2652	0.2619	0.0374	0.2761	0.2750	0.2769	0.2794	0.2775	0.2767	0.2789	0.2802	0.2782	0.2792	0.2806	0.2811
E	0.3038	0.2643	0.4011	0.2761	0.0468	0.1727	0.2033	0.2015	0.1276	0.1254	0.1490	0.1188	0.1133	0.1137	0.1253	0.1270
F	0.3673	0.2739	0.3985	0.2750	0.1727	0.0324	0.0605	0.0778	0.2083	0.1893	0.1814	0.2156	0.1898	0.1890	0.1753	0.1685
G	0.3601	0.2695	0.3928	0.2769	0.2033	0.0605	0.0410	0.0650	0.2078	0.1893	0.1857	0.1937	0.1881	0.2035	0.1761	0.1687
H	0.3400	0.2722	0.3921	0.2794	0.2015	0.0778	0.0650	0.0522	0.2075	0.1893	0.1830	0.2021	0.1878	0.2040	0.1777	0.1689
I	0.2706	0.2697	0.4013	0.2775	0.1276	0.2083	0.2078	0.2075	0.0488	0.0686	0.1036	0.1398	0.1607	0.1631	0.1630	0.1839
J	0.2533	0.2500	0.4013	0.2767	0.1254	0.1893	0.1893	0.1893	0.0686	0.0520	0.0918	0.1440	0.1389	0.1175	0.1599	0.1643
K	0.2430	0.2360	0.3892	0.2789	0.1490	0.1814	0.1857	0.1830	0.1036	0.0918	0.0440	0.1211	0.1355	0.1206	0.1389	0.1545
L	0.2425	0.2387	0.3851	0.2802	0.1188	0.2156	0.1937	0.2021	0.1398	0.1440	0.1211	0.0415	0.0939	0.0800	0.1236	0.0971
M	0.2397	0.2349	0.3928	0.2782	0.1133	0.1898	0.1881	0.1878	0.1607	0.1389	0.1355	0.0939	0.0439	0.0657	0.1020	0.1016
N	0.2492	0.2425	0.3924	0.2792	0.1137	0.1890	0.2035	0.2040	0.1631	0.1175	0.1206	0.0800	0.0657	0.0493	0.1080	0.1059
O	0.2435	0.2405	0.3947	0.2806	0.1253	0.1753	0.1761	0.1777	0.1630	0.1599	0.1389	0.1236	0.1020	0.1080	0.0503	0.0649
P	0.2414	0.2404	0.3995	0.2811	0.1270	0.1685	0.1687	0.1689	0.1839	0.1643	0.1545	0.0971	0.1016	0.1059	0.0649	0.0488

Figure 9: Pairwise persistence-distortion distances between models.

From the matrix in Figure 9, we can see that models from the same group (such as human1, human2 and human3) have very small persistence-distortiondistances among them (darker colors for smaller values). Furthermore, models from similar groups (such as between wolves and horses) have persistence-distortion distances smaller than those between dissimilar groups (such as between wolves and double-torus). This demonstrates that our persistence-distortion distance is a reasonable measure for differentiating surface models.

The number of vertices of an input mesh for each model is shown in brackets in Figure 8 after the model name. The size of the subsample of a graph is usually kept between 200 and 300. The time for computing the persistence-distortiondistance is typically less than 10 seconds. For the exceptional case involving two armadillos where the input graphs have large sizes, the running time is around 20 seconds.

We also remark that it is possible to take simply the 1-skeleton of the Rips complex constructed from the point samples V_i of a surface mesh instead of using the surface mesh itself. We expect to obtain similar results though the complex size will most likely be larger.

7 Conclusions and Future Directions

In this paper, we proposed a new way to measure distance between metric graphs, called the persistence-distortion distance. This distance is developed based on a topological idea, and provides a new angle to the metric graph comparison problem. The proposed persistence-distortion distance is stable with respect to metric distortion, and align the underlying space of input graphs (instead of just graph nodes). Despite that we consider all points in input graphs, we present a polynomial time algorithm to compute the persistence-distortion distance.

The time complexity for computing the (continuous) persistence-distortion distance is high. A worthwhile endeavor will be to bring it down with more accurate analysis. In particular, the geodesic distance function (to a basepoint) in the graph has many special properties, some of which we already leverage. It will be interesting to see whether we can further leverage these properties to reduce the bound on the decomposition $\widehat{\Lambda}(\Omega)$ as used in Theorem 24. Developing efficient approximation algorithms for computing the persistence-distortion distance is also an interesting question. Also, the special case of metric trees is worthwhile to investigate. Notice that even discrete tree matching is still a hard problem for unlabeled trees, i.e, when no correspondences between tree nodes are given.

Acknowledgment. We thank anonymous reviewers for very helpful comments, including the suggestion that $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{Q}_{\mathbf{t}})$ can be computed directly using the algorithm of [13], which simplifies our original approach based on modifying the algorithm of [13]. This work is partially supported by NSF under grants CCF-0747082, CCF-1064416, CCF-1319406, CCF1318595.

References

[1] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. *Int. J. Comput. Geom. Appl.*, pages 305–325, 2012.

[2] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison Wesley, 1974.

[3] U. Bauer, X. Ge, and Y. Wang. Measuring distance bewteen Reeb graphs. In *Proc. 30th SoCG*, pages 464–473, 2014.

[4] D. Burago, Y. Burago, and S. Ivanov. *A course in metric geometry*. volume 33 of *AMS Graduate Studies in Math*. American Mathematics Society, 2001.

[5] F. Chazal and J. Sun. Gromov-Hausdorff Approximation of Filament Structure Using Reeb-type Graph. In *Proc. 30th SoCG*, pages 491–500, 2014.

[6] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.

[7] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.

[8] D. Cohen-Steiner, H. Edelsbrunner, and D. Morozov. Vines and vineyards by updating persistence in linear time. In *Proc. 22nd SoCG*, pages 119–126, 2006.

[9] T. Cour, P. Srinivasan, and J. Shi. Balanced Graph Matching. In *Advances in Neural Information Processing Systems 19*, pages 313–320. MIT Press, 2007.

[10] T. K. Dey and R. Wenger. Stability of critical points with interval persistence. *Discrete Comput. Geom.*, 38:479–512, 2007.

[11] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2009.

[12] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.

[13] A. Efrat, M. Katz, and A. Itai. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 1:1–28, 2001.

[14] P. Foggia, C. Sansone, and M. Vento. A Performance Comparison of Five Algorithms for Graph Isomorphism. In *Proc. of the 10th ICIAP*, Italy, 2001.

[15] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, Jan. 2010.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: a guide to the theory of NP-completeness*. W. H. Freeman & Co, New York, NY, USA, 1990.

[17] X. Ge, I. Safa, M. Belkin, and Y. Wang. Data skeletonization via Reeb graphs. In *Proc. 25th NIPS*, pages 837–845, 2011.

[18] S. Gold and A. Rangarajan. A Graduated Assignment Algorithm for Graph Matching. In *IEEE Trans. on PAMI*, volume 18, pages 377–388, 1996.

[19] M. Gromov. *Metric structures for Riemannian and non-Riemannian spaces*. volume 152 of *Progress in Mathematics*. Birkhäuser Boston Inc., 1999.

[20] J. E. Hopcroft and J. K. Wong. Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report). In *Proc. of the ACM STOC*, pages 172–184, 1974.

[21] N. Hu, R. Rustamov, and L. Guibas. Graph Matching with Anchor Nodes: A Learning Approach. In *IEEE Conference on CVPR*, pages 2906–2913, 2013.

[22] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on ICCV*, pages 1482–1489, 2005.

[23] M. Leordeanu, M. Hebert, and R. Sukthankar. An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. In *Proc. NIPS*. Springer, December 2009.

[24] E. M. Luks. Isomorphism of Graphs of Bounded Valence Can be Tested in Polynomial Time. *Journal of Computer and System Sciences*, 25(1):42–65, 1982.

[25] F. Mémoli. On the use of Gromov-Hausdorff Distances for Shape Comparison. In *Symposium on Point Based Graphics*, pages 81–90, 2007.

[26] U. Ozertem and D. Erdogmus. Locally defined principal curves and surfaces. *Journal of Machine Learning Research*, 12:1249–1286, 2011.

[27] T. Sousbie, C. Pichon, and H. Kawahara. The persistent cosmic web and its filamentary structure – II. Illustrations. *Monthly Notices of the Royal Astronomical Society*, 414:384–403, 2011.

[28] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. In *IEEE Trans. on PAMI*, volume 10, pages 695–703, 1998.

[29] B. J. van Wyk and M. A. van Wyk. A pocs-based graph matching algorithm. In *IEEE Trans. on PAMI*, volume 26, pages 1526–1530, 2004.

[30] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. In *IEEE Conference on CVPR*, pages 1–8, June 2008.

[31] Z. Zeng, A. K. H. Tung, J. Wang, J. Feng, and L. Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2(1):25–36, Aug. 2009.