Comparing Graphs via Persistence Distortion*

Tamal K. Dey, Dayu Shi, and Yusu Wang

Computer Science and Engineering Department, The Ohio State University, USA
tamaldey,shiday,yusu@cse.ohio-state.edu

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.

1998 ACM Subject Classification F.2.2 Geometric problems and computations, G.2.2 Graph algorithms

Keywords and phrases Graph matching, metric graphs, persistence distortion, topological method

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.491

1 Introduction

Many data in science and engineering are drawn from a hidden space which are graph-like, such as the cosmic web [28] 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 [18, 27]. They are graphs where each edge is associated with a length inducing the metric of shortest path distance. The comparison of the representative 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 nodes alignments are considered, we aim for a distance

* This work is partially supported by NSF under grants CCF-0747082, CCF-1064416, CCF-1319406, CCF1318595. See [11] for the full version of this paper.

31st International Symposium on Computational Geometry (SoCG'15).
Editors: Lars Arge and János Pach; pp. 491–506

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 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, 21, 25], for general graphs, it is not known whether the graph isomorphism problem is NP complete or not [17]. 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 [15] for comparisons of various graph isomorphism testing methods.

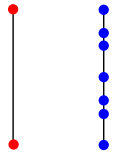
In real world applications, input graphs often suffer from noise and deformation, and it is highly desirable to obtain a *distance* between two input graphs beyond the binary decision of whether they are the same (isomorphic) or not. This is referred to as inexact graph matching in the field of pattern recognition, and various distance measures have been proposed. One line of work is based on graph edit distance which is NP-hard to compute [32]. Many heuristic methods, using for example A^* algorithms, have been proposed to address the issue of high computational complexity, see the survey [16] 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 [19], iterative methods of [24, 30] and probabilistic approach in [31]. Finally, there have been several methods that formulate the optimization problem based on spectral properties of graphs. For example, in [29], 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, 23], the principal eigenvector of a "compatibility" matrix of the input graphs is used to obtain correspondences between input graph nodes. Recently in [22], 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 [20, 26] 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

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

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 previous graph matching algorithms align only graph nodes. Thus 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 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.)

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.

All technical details omitted from this extended abstract due to lack of space can be found in the full version of the paper at [11]. Some preliminary experimental results to demonstrate the use of the persistence-distortion distance are also included in the full version.

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

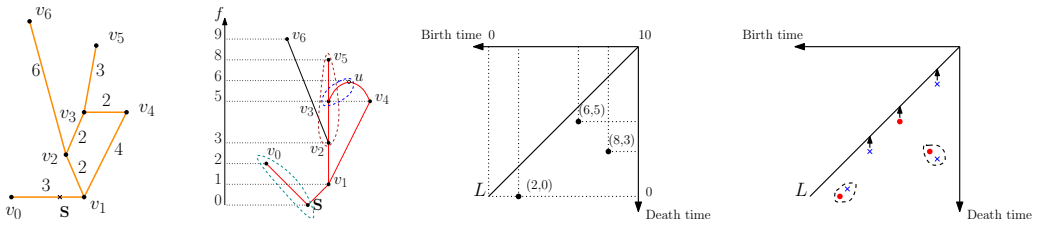


Figure 1 (a) A graph with basepoint $\mathbf{s}$: edge length is marked for each edge. (b) The function $f = d_G(\mathbf{s}, \cdot)$. We also indicate critical-pairs. (c) Persistence diagram $\mathrm{Dg}_0 f$: E.g, the persistence-point $(6, 5)$ is generated by critical-pair (u, v_3). (d) shows a partial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

refer the readers to resources such as [12, 13] 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.

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.

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

[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.

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 partial 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}).

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,

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

▶ 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 (an example is shown in the full version [11]). 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 [20, 4]. Given two metric spaces $\mathcal{X} = (X, d_X)$ and $\mathcal{Y} = (Y, d_Y)$, a *correspondence* 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 [26] 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 (see 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:*

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

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 4.** 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 basepoint moves) and the stability of persistence diagrams.

▶ **Observation 5.** $\mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq \widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2) \leq \mathrm{d_{PD}}(\mathrm{G}_1, \mathrm{G}_2) + \frac{\ell}{2}$*, where* ℓ *is the largest length of any edge in* $E_1 \cup E_2$*.*

▶ **Lemma 6.** *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 [14], 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 $\mathbb{R}^2$ 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 $d_B(\mathrm{P}, \mathrm{Q}) = \hat{d}_B(\mathrm{P}_{\mathbf{s}} \cup Q', \mathrm{Q}_{\mathbf{t}} \cup P')$. Hence $d_B(\mathrm{P}_{\mathbf{s}}, \mathrm{Q}_{\mathbf{t}})$ can be computed by the algorithm of [14] 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 5, $\widehat{\mathrm{d}}_{\mathrm{PD}}(\mathrm{G}_1, \mathrm{G}_2)$ 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 7.** *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

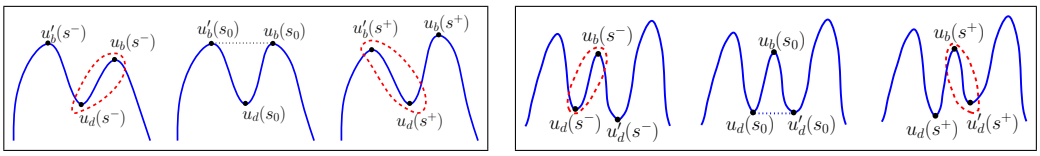


Figure 2 For better illustration of ideas, we use height function defined on a line to show: (a) a max-max critical event at s_0; and (b) a saddle-saddle critical event at s_0.

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[3]. 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.

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 event*. See Figure 2 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. 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 8.** *An up-fork saddle $u \in \mathrm{G}_1$ is necessarily a graph node from V_1 with degree at least 3 unless $u = \mathbf{s}$.*

[3] 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 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, below we track the movement of persistence points ignoring their creation and deletion for the time being.

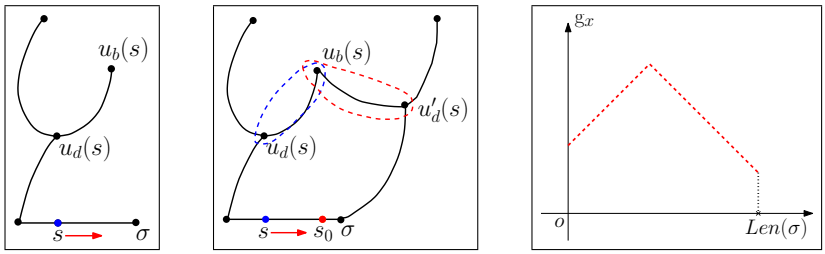


Figure 3 (c) Graph of function $\mathsf{g}_x : [0, \text{Len}(\sigma)] \to \mathbb{R}$.

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 3.

Now for any point $x \in \mathrm{G}_1$, we introduce the function $\mathsf{g}_x : [0, \text{Len}(\sigma)] \to \mathbb{R}$ 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.

▶ **Claim 9.** *For any point $x \in \mathrm{G}_1$, as the basepoint $\mathbf{s}$ moves in an edge $\sigma \in E$, the distance function $\mathsf{g}_x : [0, \text{Len}(\sigma)] \to \mathbb{R}$ 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 3 (c).*

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 PL with at most 2 pieces as shown in Figure 3 (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 10.** *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 11.** *For a fixed persistent point $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$, the corresponding death-time function $\mathsf{d} : [0, \text{Len}(\sigma)] \to \mathbb{R}$ 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 8, $u_{\mathsf{d}}(s)$ 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 $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$, we need to study the movements of its corresponding birth-point (which is a maximum) $u_{\mathsf{b}} : [0, \mathrm{Len}(\sigma)] \to \mathrm{G}_1$ 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 of the distance function $d_{G_1,\mathbf{s}}$, imagine the shortest path tree with $\mathbf{s}$ being the root (source), then any non-tree edge 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 there can be at most one local maximum associated with each edge.

▶ **Lemma 12.** *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*.*

This lemma suggests that we can now associate each local maximum with an edge in E_1, and analyze the changes of such an edge e_{b} containing the birth-point u_{b} (instead of the birth-point itself). Specifically, using approaches similar to the tracking of death-point as in Section 5.1.1, we study, for a fixed edge $e \in E_1$ the function $\mathsf{g}_e : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ where, for any $s \in [0, \mathrm{Len}(\sigma)]$, $\mathsf{g}_e(s)$ 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 9, we have the following characterization for this distance function.

▶ **Proposition 13.** *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'.*

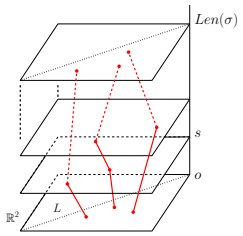
Using argument similar to, but more involved than that of Section 5.1.1, we obtain the following result about the birth-time function, analogous to Proposition 11.

▶ **Proposition 14.** *For a fixed* $(\mathsf{b}(0), \mathsf{d}(0)) \in \mathrm{P}_{\mathbf{s}(0)}$*, the birth-time function* $\mathsf{b} : [0, \mathrm{Len}(\sigma)] \to \mathbb{R}$ *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.*

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 11 and 14, 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 15.** *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}).

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 16.** 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:

$$\vec{d}_H(\mathcal{C}, \mathcal{F}) = \max_{\mathrm{P} \in \mathcal{C}} \min_{\mathrm{Q} \in \mathcal{F}} d_B(\mathrm{P}, \mathrm{Q}) = \max_{\sigma_s \in \mathrm{G}_1} \max_{s \in [1, L_s]} \min_{\sigma_t \in \mathrm{G}_2} \min_{t \in [1, L_t]} F_{\sigma_s, \sigma_t}(s, t). \tag{3}$$

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 distance between G_1 and G_2 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 15, 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).

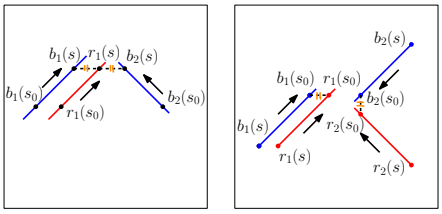
Set $\Omega := [0, L_s] \times [0, L_t]$ and we refer to it as the *s-t domain*. For a point $(s,t) \in \Omega$, the function value $F(s,t)(= F_{\sigma_s,\sigma_t}(s,t)) = d_B(\mathrm{P}_{\mathbf{s}(s)}, \mathrm{Q}_{\mathbf{t}(t)})$ is the bottleneck distance between the set of red and the set of blue 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 $(r(s), b(t)) \in C^*(s,t)$, $d_\infty(r(s), b(t)) \leq d_\infty(r^*(s), b^*(t))$. 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.

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



(case-1) (case-2)

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

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 15, $r : [0, L_s] \to \mathbb{R}^2$ (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 11 and 14, we have the following:

▶ **Proposition 17.** *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'.*

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 17, 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,

▶ **Corollary 18.** *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 $\mathcal{X} = \{X_{(r_1,b_1),(r_2,b_2)} \mid r_1, r_2 \in \mathrm{P}_{\mathbf{s}}, b_1, b_2 \in \mathrm{Q}_{\mathbf{t}}\}$. 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 (see the full version [11] for details):

▶ **Proposition 19.** *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.

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 (3). To do this, we need to further refine the decomposition $\Lambda(\Omega)$ from Proposition 19 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. The proof can be found in the full version [11].

▶ **Theorem 20.** *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.*

5.2.2 Final algorithm and analysis.

We now aim to compute $\vec{d}_H(\mathcal{C}, \mathcal{F})$ using Eqn (3). First, for a fixed edge $\sigma_s \in \mathrm{G}_1$, consider the following *lower-envelop function*

$$\mathcal{L} : [0, L_s] \to \mathbb{R} \quad \text{where} \quad \mathcal{L}(s) \mapsto \min_{\sigma_t \in \mathrm{G}_2} \min_{t \in [1, L_t]} F(s,t), \tag{4}$$

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.

Now for each $\sigma_t \in \mathrm{G}_2$, consider the polygonal decomposition $\widehat{\Lambda}(\Omega)$ as described in Theorem 20. 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 $s \in [1, L_s]$, 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

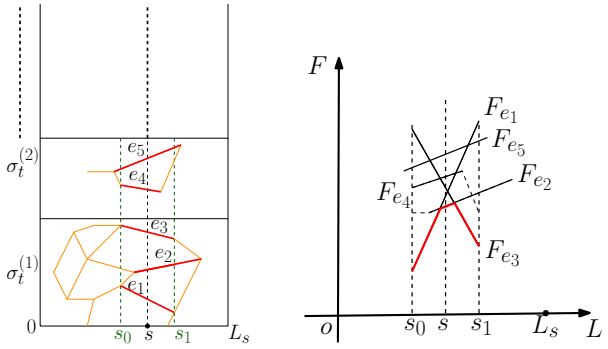


Figure 4 (a) s-t domains for $\sigma_s \in E_1$ and edges $\sigma_t^{(j)} \in E_2$. (b) $\mathcal{L}(s)$ is the lowest value along any F_{e_ℓ}.

$\pi_e : [0, L_s] \to [0, L_t]$ 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 4):

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

There are $O(m)$ edges in G_2, thus by Theorem 20 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 (3), $\vec{d}_H(\mathcal{C}, \mathcal{F}) = \max_{\sigma_s \in \mathrm{G}_1} \max_{s \in [1, L_s]} \mathcal{L}(s)$. Since there are $O(m)$ choices for σ_s, we conclude with the following main result.

▶ **Theorem 21.** *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 Future directions

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 20. 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 [14], which simplifies our original approach based on modifying the algorithm of [14].

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. Ulrich Bauer, Xiaoyin Ge, and Yusu 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. Frédéric Chazal and Jian Sun. Gromov-Hausdorff Approximation of Filament Structure Using Reeb-type Graph. In *Proc. 30th SoCG*, pages 491–500, 2014.
6. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37(1):103–120, 2007.
7. David Cohen-Steiner, Herbert Edelsbrunner, and John Harer. Extending persistence using Poincaré and Lefschetz duality. *Foundations of Computational Mathematics*, 9(1):79–103, 2009.
8. David Cohen-Steiner, Herbert Edelsbrunner, and Dmitriy 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. Tamal K. Dey, Dayu Shi, and Yusu Wang. Comparing graphs via persistence distortion, 2015. arXiv:1503.07414.
12. H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. Amer. Math. Soc., Providence, Rhode Island, 2009.
13. H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete Comput. Geom.*, 28:511–533, 2002.
14. A. Efrat, M. Katz, and A. Itai. Geometry helps in bottleneck matching and related problems. *Algorithmica*, 1:1–28, 2001.
15. P. Foggia, C. Sansone, and M. Vento. A Performance Comparison of Five Algorithms for Graph Isomorphism. In *Proc. of the 10th ICIAP*, Italy, 2001.
16. Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li. A survey of graph edit distance. *Pattern Anal. Appl.*, 13(1):113–129, January 2010.
17. 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.
18. X. Ge, I. Safa, M. Belkin, and Y. Wang. Data skeletonization via Reeb graphs. In *Proc. 25th NIPS*, pages 837–845, 2011.
19. S. Gold and A. Rangarajan. A Graduated Assignment Algorithm for Graph Matching. In *IEEE Trans. on PAMI*, volume 18, pages 377–388, 1996.
20. M. Gromov. *Metric structures for Riemannian and non-Riemannian spaces*. volume 152 of *Progress in Mathematics*. Birkhäuser Boston Inc., 1999.
21. J. E. Hopcroft and J. K. Wong. Linear Time Algorithm for Isomorphism of Planar Graphs (Preliminary Report). In *Proc. of the ACM STOC*, STOC'74, pages 172–184, New York, NY, USA, 1974. ACM.
22. N. Hu, R.M. Rustamov, and L. Guibas. Graph Matching with Anchor Nodes: A Learning Approach. In *IEEE Conference on CVPR*, pages 2906–2913, 2013.
23. M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *IEEE International Conference on ICCV*, pages 1482–1489, 2005.

24 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.

25 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.

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

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

28 T. Sousbie, C. Pichon, and H. Kawahara. The persistent cosmic web and its filamentary structure – II. Illustrations. *Mon. Not. R. Astron. Soc.*, 414:384–403, 2011.

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

30 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.

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

32 Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. Comparing stars: On approximating graph edit distance. *Proc. VLDB Endow.*, 2(1):25–36, August 2009.