AIMS Mathematics, 10(7): 17179–17231.
DOI:10.3934/math.2025769
Received: 08 April 2025
Revised: 14 June 2025
Accepted: 24 June 2025
Published: 31 July 2025

https://www.aimspress.com/journal/Math

Research article

A finitely stable edit distance for merge trees

Matteo Pegoraro*

Department of Mathematics, KTH, Stockholm 10044, Sweden

* **Correspondence:** Email: matteope@kth.se.

Abstract: In this paper, we defined a novel edit distance for merge trees, which we argued to be suitable for a broad range of applications. Relying also on some technical results contained in other works, we investigated its stability properties, which ended up being analogous to the ones of the 1-Wasserstein distance between persistence diagrams. We tested and compared our metric against the interleaving distance in several simulations and case studies, highlighting the trade-off between stability and sensitivity when choosing the appropriate metric for a given data analysis problem, much alike the bias-variance trade-off in statistical modeling. In the appendix, we also compared our metric with other edit distances appearing in the literature, with both theoretic and practical considerations.

Keywords: topological data analysis; merge trees; interleaving distance; edit distance; binary optimization
Mathematics Subject Classification: 05C05, 05C10, 55N31, 62R40

1. Introduction

Topological data analysis (TDA) is a particular set of techniques within the field of applied topology which aim at including topological information into data analysis pipelines. Topological information is usually understood in terms of generators of homology groups [26] with coefficients in some field. With persistent homology, these generators are extracted along a filtration of topological spaces to capture the shape of the initial datum, typically a function or a point cloud, at "different resolutions" [19]. To proceed with the analysis, this ordered family of vector spaces is then represented with a topological summary. There are many different kinds of topological summaries such as persistence diagrams [18] (PDs), persistence images [1], persistence silhouettes [12], and persistence landscapes [7]. Each of these summaries live in a space with different properties and purposes: for instance, persistence diagrams are highly interpretable and live in a metric space; persistence landscapes are embedded in a linear space of functions, but the embedding is not closed under linear combinations; and persistence images are instead obtained as vectors in $\mathbb{R}^n$, making them suitable for many machine

learning techniques.

Along with the aforementioned summaries, there are also tree-shaped objects called merge trees. Merge trees arise naturally within the framework of TDA when dealing with zero dimensional homology groups, as they capture the merging structure of path connected components along a filtration of topological spaces. Originally, such objects stem out of Morse theory [31] as a topological summary related to Reeb graphs [5,46] and are frequently used for data visualization purposes [56]. Analogously, other different but related kinds of trees like hierarchical clustering dendrograms [32] or phylogenetic trees [23] have also been used extensively in statistics and biology to infer information about a fixed set of labels. When considered as unlabeled objects, however, both clustering dendrograms and phylogenetic trees can be obtained as particular instances of merge trees. The uprising of TDA has propelled works aiming at using merge trees and Reeb graphs as topological summaries in data analysis contexts and, thus, developing metrics and frameworks to analyze populations of such objects.

Previous works on merge trees

The works that have been dealing with the specific topic of merge trees can be divided into two groups: the first group is more focused on the definition of a suitable metric structure to compare merge trees, and the second is more focused on the properties of merge trees and their relationships with PDs [3,15,29]. The first group, in turns, splits into a) works dealing with the interleaving distance between merge trees and other metrics with very strong theoretical properties [4, 8, 16, 21, 25, 49], and b) works defining edit distances focused on computational efficiency [40, 47, 48, 53, 54] at the cost of sacrificing stability properties and trying to mitigate the resulting problems with pre-processing and other computational solutions. Among the aforementioned edit distances, only [53], in its *unconstrained* formulation, which shares some similarities with our approach, satisfies some stability properties, as we prove in this paper.

Main contributions

This paper introduces a novel edit distance for merge trees with the following key contributions:

1) We show that the stability properties of the metric are akin to the ones of the 1-Wasserstein distance for persistence diagrams;

2) We compare our metric with the interleaving distance between merge trees through theoretical analysis and empirical case studies, exploring the trade-off between stability and sensitivity in topological data analysis. We also propose an analogy which compares such trade-off to the *bias-variance* trade-off in statistical modeling.

As we have already mentioned, the metric we propose shares some similarities with the one developed in [53]. More formally, consider $f : X \to \mathbb{R}$ and $g : X \to \mathbb{R}$, and call, with a temporary abuse of notation, $\mathrm{DB}(f, g)$ the distance between the merge trees associated to f and g, as defined in [53]; similarly, call $d_E(f, g)$ the edit distance between the merge trees associated to f and g that we define in this paper. In Proposition 7, we prove that:

$$| \mathrm{DB}(f, g) - d_E(f, g) | = | \max f - \max g | . \tag{1.1}$$

We point out that the metric in [53] and our metric, which has appeared also in earlier papers and preprints (see, for instance, [10, 35]), have both been developed independently. On top of that, despite these strong relationships, there are some very important differences between the two metrics, as we detail in Appendix A.2. For instance, for any $k \in \mathbb{R}$, $\mathrm{DB}(f, f + k) = 0$, while $d_E(f, f + k) = \mid k \mid$. In particular, Theorem 3 does not hold for DB.

The main characteristics of our metric are the following:

- The metric we propose has stability properties analogous to the ones of 1-Wasserstein distance between PDs. In particular, the 1-Wasserstein distance (W_1) and the bottleneck distance between PDs (d_B) enjoy the following relationship for every pair of diagrams D, D':

$$d_B(D, D') \leq W_1(D, D') \leq (\dim(D) + \dim(D'))d_B(D, D'), \tag{1.2}$$

  with $\dim(D_i)$ being the cardinality of the diagrams D_i. Similarly, we prove that the edit distance that we define (d_E) and the interleaving distance between merge trees (d_I) satisfy:

$$d_I(T, T') \leq d_E(T, T') \leq 2(\mathrm{size}(T) + \mathrm{size}(T'))d_I(T, T'), \tag{1.3}$$

  with $\mathrm{size}(T_i)$ being the number of edges of the merge tree T_i. We point out that the rightmost inequality is obtained by [39], which investigates some statistical properties of the metric we define here. The leftmost inequality, instead, is proven in Theorem 3. In Section 6, we argue that Eqs (1.2) and (1.3) make W_1 and d_E better suited for general data analysis purposes, compared to their *universal* counterparts, that is, d_B and d_I, framing such choice as some kind of bias-variance trade-off. Section 7 supports these claims with examples and simulations. Table 2, instead, highlights that almost no other edit distance for merge trees is stable. In Appendix A, we give examples on how the metrics in [40,47,48,54] fail to be stable: only [53], which was already shown to have some good behavior on some datasets [52], satisfies some stability properties, which we prove with Proposition 7. Note that such inequality does not involve d_I and, along the same lines, DB doesn't satisfy Eq (1.3). See Appendix A.2 and Figure 10 for more details.

- The metric we propose has computational complexity close to the one of the classical edit distance between unlabeled trees [27]. Call D_E such edit distance. Table 1 compares the computational costs of D_E and d_E in terms of binary linear programming (BLP), showing that d_E differs from D_E only by two *log* factors in the number of variables. Table 2 shows that none of the metrics in [40,47,48,53,54] is cheaper than D_E. However, for all those metrics, the authors have proposed a polynomial time algorithm to compute an upper bound. These algorithms are obtained in similar ways: [28, 45, 57] contain some upper bounds for D_E, which can also be regarded as a metrics on their own. Such "approximated" edit distances have polynomial time complexity. The authors of [40,47,48,53,54] frame their metrics as edit distances and then resort to these variations/upper bounds to obtain feasible algorithms. We highlight that, via Eq (1.1), a constrained upper bound of d_E with polynomial time complexity can be found using the algorithm in [53] and the heuristics developed by [55] can be employed also to compute our metric. We leave the exploration of these methods to future works.

Table 1. Computational costs of the classical edit distance for unlabeled trees [27] and our edit distance for merge trees, with $N = \text{size}(T)$ and $M = \text{size}(G)$.

	BLP Problems	Variables	Constraints
$D_E(T, G)$	$O(M \cdot N)$	$O(M \cdot N)$	$O(\log_2(M) \cdot \log_2(N))$
$d_E(T, G)$	$O(M \cdot N)$	$O(M \cdot \log_2(M) \cdot N \cdot \log_2(N))$	$O(\log_2(M) \cdot \log_2(N))$

Table 2. A summary of the Edit Distances for Merge Trees. The column "Metric" states which distances satisfy the triangular inequality. The column "Stability" underlines that d_E and DB are the only edit distances satisfying some stability properties, with N and M being the number of edges in the considered merge trees. The column "Computational Cost" compares the computational cost of the other edit distances, with D_E: $\sim D_E$ standing for the cost being equivalent, up to nuances, to the one of D_E; while $\geq D_E$ stands for the cost being, in general, higher.

	Metric	Stability	Comput. Cost
d_E	Yes	$d_I \leq d_E \leq 2(N + M)d_I$	$\geq D_E$
Wass [40, 47]	Yes	No	$\sim D_E$
BDI [54]	No	No	$\geq D_E$
DB [53]	Yes	$\text{DB} \leq 2(N + M) \parallel f - g \parallel_\infty$	$\geq D_E$
Local [48]	Yes	No	$\sim D_E$

Connections with other works

As previously mentioned, this work builds upon results established in separate papers. Specifically:

- [36] introduces an edit distance for weighted trees, which we briefly recall in Section 3, and then extend to merge trees in Section 4. As the results in that section demonstrate, this extension is far from straightforward and involves substantial additional work.
- [39] provides the righthand side of Eq (1.1). While none of the mathematical results in the present manuscript directly rely on this inequality, its validity is essential for the broader soundness of our proposed metric—particularly with regard to the stability properties discussed in Section 6.

Outline

This paper is organized as follows: Section 2 and Section 3 contain the preliminary definitions needed in the paper, which are collected from previous works in the field. In the latter one, in particular, we review the definition of an edit distance between weighted trees, which we use in Section 4 to define a metric structure for merge trees. Section 5 introduces the interleaving distance between merge trees, which is fundamental for the discussion contained in Section 6, regarding the stability properties of d_E. Section 7 contains several simulations and case studies in which the practical consequences of the considerations in Section 6 are extensively showcased. In Section 8, we draw some conclusions and point to future works.

The Appendix contains a detailed comparison (both practical and theoretic) with the metrics for merge trees which we mention in the introduction (Appendix A). Appendix B contains the proofs of

the results in the manuscript.

2. Preliminary definitions-merge trees

We introduce merge trees coherently with most of the scientific literature dealing with such topics: a combinatorial object, which we call tree structure, with a monotone increasing function defined on its vertices (monotone w.r.t. a partial ordering of the vertices).

Definition 1. A tree structure T is given by a set of vertices V_T and a set of edges $E_T \subset V_T \times V_T$ which form a connected rooted acyclic graph. We indicate the root of the tree with r_T. We say that T is finite if V_T is a finite set. The degree of a vertex $v \in V_T$ is the number of edges which have that vertex as one of the endpoints, and is called $\deg_T(v)$. Any vertex with an edge connecting it to the root is its child and the root is its parent: this is the first step of a recursion which defines the parent and children relationship for all vertices in V_T. The vertices with no children are called leaves or taxa and are collected in the set L_T. The relation $child < parent$ generates a partial order on V_T. The edges in E_T are identified in the form of ordered couples (a, b) with $a < b$. A subtree of a vertex v, called $sub_T(v)$, is the tree structure whose set of vertices is $\{x \in V_T \mid x \leq v\}$.

In Definition 1, we introduce the term tree structure to differentiate such objects from the generic word "tree", which we will use to indicate either a tree structure, or a weighted tree or a merge tree, when the context allows to lighten the notation.

Given a tree structure T, identifying an edge (v, v') with its lower vertex v gives a bijection between $V_T - \{r_T\}$ and E_T, that is, $E_T \cong V_T - \{r_T\}$, as sets. Given this bijection, we often use E_T to indicate the vertices $v \in V_T - \{r_T\}$, and vice versa, to simplify the notation.

Now, we want to identify merge trees independently of their vertex set.

Definition 2. Two tree structures T and T' are isomorphic if there exists a bijection $\eta : V_T \to V_{T'}$ that induces a bijection between the edges sets E_T and $E_{T'}$: $(a, b) \mapsto (\eta(a), \eta(b))$. Such η is an isomorphism of tree structures.

Now, we can give the definition of a merge tree.

Definition 3. A merge tree is a finite tree structure T with a monotone increasing height function $h_T : V_T \to \mathbb{R} \cup \{+\infty\}$ and such that: 1) $\deg_T(r_T) = 1$, 2) $h_T(r_T) = +\infty$, 3) $h_T(v) \in \mathbb{R}$ for every $v < r_T$. Two merge trees (T, h_T) and $(T', h_{T'})$ are isomorphic if T and T' are isomorphic as tree structures and the isomorphism $\eta : V_T \to V_{T'}$ is such that $h_T = h_{T'} \circ \eta$. Such η is an isomorphism of merge trees. We use the notation $(T, h_T) \cong (T', h_{T'})$. The set of all merge trees up to isomorphism is called $\mathcal{MT}$.

With some slight abuse of notation, we set $\max h_T = \max_{v \in V_T | v < r_T} h_T(v)$ and $\arg\max h_T = \max\{v \in V_T \mid v < r_T\}$. Note that, given (T, h_T) merge tree, there is only one edge of the form (v, r_T), and we have $v = \arg\max h_T$.

Definition 4. Given a tree structure T, we can eliminate a degree two vertex, connecting the two adjacent edges which share such vertex as endpoint. Suppose we have two edges $e = (v_1, v_2)$ and $e' = (v_2, v_3)$, with $v_1 < v_2 < v_3$, and suppose v_2 is of degree two. Then, we can remove v_2 and merge e and e' into a new edge $e'' = (v_1, v_3)$. This operation is called the *ghosting* of the vertex. Its inverse transformation is called the *splitting* of an edge.

Consider a merge tree (T, h_T) and obtain T' by ghosting a vertex of T. Then, $V_{T'} \subset V_T$, and, thus, we can define $h_{T'} := h_{T|V_{T'}}$.

We can now state the following definition.

Definition 5. Merge trees are equal up to degree 2 vertices if they become isomorphic after applying a finite number of ghostings or splittings. We write $(T, h_T) \cong_2 (T', h_{T'})$. The set of all merge trees up to degree 2 vertices is called $\mathcal{MT} / \sim_2$.

Merge trees are usually employed to give a combinatorial representation of persistent sets [16, 34] which are generated via usual TDA pipelines applying the functor π_0 on filtrations of topological spaces. Going through such details is not essential for defining and understanding the metric d_E; however, it is necessary to introduce the interleaving distance between merge trees and discuss the stability properties of both metrics. Therefore, we will postpone their discussion to Section 5.

3. Preliminary definitions-weighted trees edit distance

In this section, we introduce the last pieces of notation we need to define the merge tree edit distance. In Section 3.1, we report how [36] builds a metric for weighted trees, and in Section 3.2, we recall from [36] the definition of mapping, a combinatorial object which will be used throughout the manuscript.

3.1. Weighted trees and edits

Definition 6. A tree structure T with a weight function $w_T : E_T \to \mathbb{R}_{>0}$ is called a weighted tree. Isomorphisms of weighted trees are defined as in Definition 3. The set of all weighted trees up to isomorphism is called $(\mathcal{T}, \mathbb{R}_{\geq 0})$.

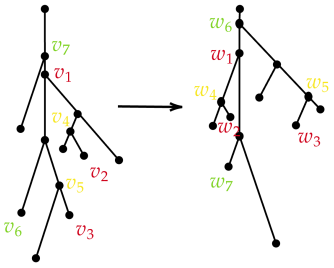
Given a weighted tree (T, w_T), we can modify its edges $E_T \cong V_T - \{r_T\}$ with a sequence of the following edit operations (see Figure 1):

- We call shrinking of a vertex/edge a change of the weight function. The new weight function must be equal to the previous one on all edges, apart from the "shrunk" one, whose weight can become bigger or smaller. In other words, for an edge e, this means changing the value $w_T(e)$ with another value in $\mathbb{R}_{>0}$.
- A deletion is an edit with which a vertex/edge is deleted from the tree structure. Consider an edge (v_1, v_2). The result of deleting v_1 is a new tree structure, with the same vertices apart from v_1 (the lower one), and with the parent of the deleted vertex which gains all of its children. The inverse of the deletion is the insertion of an edge along with its lower vertex. We can insert an edge at a vertex v specifying the name of the new child of v, the children of the newly added vertex (that can be either none, or any portion of the children of v), and the value of the weight function on the new edge.
- Lastly, we can remove or add degree two vertices via the ghosting and splitting edits, which have already been defined in Definition 4. The weight function is obtained by summation of the merged edges in case of ghostings. In case of splittings, it must be arbitrarily defined, with the condition

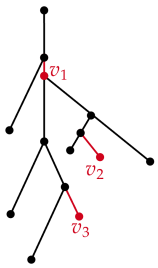
that splitting an edge and then ghosting the degree two vertices obtained must restore the original edge weight.

The set of all weighted trees considered up to ghostings and splittings is called $(\mathcal{T}_2, \mathbb{R}_{\geq 0})$.

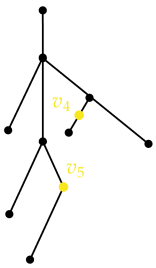
A weighted tree T can be edited to obtain another weighted tree, on which one can apply a new edit to obtain a third tree and so on. Any finite sequence of edits is called edit path. See Figure 1 for an example of an edit path. The set of finite edit paths between T and T' is called $\Gamma(T, T')$. We use functional notations to refer to edits and edits paths, even though each edit transforms a single tree, and it is not defined on the whole space. In particular, we represent an edit defined on a weighted tree T as a function $e : \{T\} \to (\mathcal{T}, \mathbb{R}_{\geq 0})$, so that $e(T)$ is the weighted tree we obtain by transforming T with e. Given an edit path $\gamma = \{e_1, e_2\}$ with two edits, we write $\gamma(T) := e_2(e_1(T)) = e_2 \circ e_1(T)$ to identify the weighted tree obtained applying e_1 on T and e_2 on $e_1(T)$. Analogously for paths with a higher number of edits. This notation will help us simplify some of the upcoming definitions.



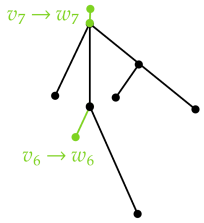
(a) Starting and target weighted trees, with highlighted vertices involved in the edit path.



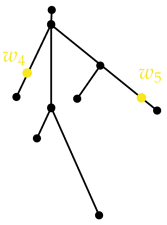
(b) Deletions of v_1, v_2, v_3, in red.



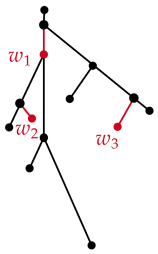
(c) Ghostings of v_4, v_5, in yellow.



(d) Shrinkings of v_6 and v_7 to match the weight, respectively, of w_6 and w_7, in green.



(e) Splitting edges via w_4 and w_5, in yellow.



(f) Insertions of w_1, w_2, w_3, in red.

Figure 1. (b)→(e) form an edit path made from the left weighted tree in (a) to the right one. The edit path can be represented with a mapping Section 3.2 consisting of the couples $(v_i, \mathfrak{D})$ for all the red vertices in (b), $(v_i, \mathfrak{G})$ for all the yellow vertices in (c), (v_i, w_i) for all the vertices associated via the green color in (d), $(\mathfrak{G}, w_i)$ for all the yellow vertices in (e), and $(\mathfrak{D}, w_i)$ for all the red vertices in (f).

The cost of the edit operations is defined as follows:

- The cost of shrinking an edge is equal to the absolute value of the difference of the two weights;

- For any deletion/insertion, the cost is equal to the weight of the edge deleted/inserted;
- The cost of ghostings and splittings is zero.

The cost of an edit path is the sum of the costs of its edit operations. Putting all the pieces together, the edit distance d_E between weighted trees is defined as $d_E(T, T') = \inf_{\gamma\in\Gamma(T,T')} cost(\gamma)$. In [36], it is proved that d_E is a metric on the space of weighted trees considered up to degree two vertices.

3.2. Mappings

Given an edit path between two weighted trees, its cost is often invariant up to many permutations of the edits. To better work in such an environment, we start considering paths up to some permutation of the edits. Objects called mappings, as defined in [36], help us in doing this, as well as making the metric d_E more tractable. For this reason now we report their definition. Symbols $\mathfrak{D}$ and $\mathfrak{G}$ are used to indicate "deletion" and "ghosting".

A *mapping* between T and T' is a set $M \subset (E_T \cup \{\mathfrak{D}, \mathfrak{G}\}) \times (E_{T'} \cup \{\mathfrak{D}, \mathfrak{G}\})$ satisfying:

(M1) Consider the projection of the Cartesian product $(E_T \cup \{\mathfrak{D}, \mathfrak{G}\}) \times (E_{T'} \cup \{\mathfrak{D}, \mathfrak{G}\}) \to (E_T \cup \{\mathfrak{D}, \mathfrak{G}\})$; we can restrict this map to M obtaining $\pi_T : M \to (E_T \cup \{\mathfrak{D}, \mathfrak{G}\})$. The maps π_T and $\pi_{T'}$ are surjective on E_T and $E_{T'}$, i.e., $E_T \subset \mathrm{Im}(\pi_T)$ and $E_{T'} \subset \mathrm{Im}(\pi_{T'})$;

(M2) π_T and $\pi_{T'}$ are injective on $M \cap (E_T \times E_{T'})$;

(M3) Given (a, b) and $(c, d) \in M \cap (V_T \times V_{T'})$, $a > c$, if, and only if, $b > d$;

(M4) If $(a, \mathfrak{G}) \in M$ (or analogously $(\mathfrak{G}, a)$), then after applying all deletions of the form $(v, \mathfrak{D}) \in M$, the vertex a becomes a degree 2 vertex. In other words, let $child(a) = \{b_1, .., b_n\}$. Then, there is exactly one i such that for all $j \neq i$, for all $v \in V_{sub(b_j)}$, we have $(v, \mathfrak{D}) \in M$; and there is one, and only one, c such that $c = \max\{x < b_i \mid (x, y) \in M \text{ for any } y \in V_{T'}\}$.

We call $\mathrm{Mapp}(T, T')$ the set of all mappings between T and T'. We may refer to edges which appear in the couples in $M \cap (V_T \times V_{T'})$ as the *coupled* or *matched* edges/vertices.

As in [36], we use the properties of $M \in \mathrm{Mapp}(T, T')$ to parametrize a set of edit paths starting from T and ending in T', which are collected under the name γ_M.

- γ_d^T is a path made by the deletions to be done on T, that is, the couples $(v, \mathfrak{D})$, executed in any order. So, we obtain $T_d^M = \gamma_d^T(T)$, which, instead, is well-defined and does not depend on the order of the deletions. Similarly, we define $\gamma_d^{T'}$ as a path made by the deletions to be done on T', that is, the couples $(\mathfrak{D}, w)$, executed in any order, and obtain $T_d'^M = \gamma_d^{T'}(T')$.
- One then proceeds editing T_d^M by ghosting all the vertices $(v, \mathfrak{G})$ in M, in any order, getting a path γ_g^T and the weighted tree $T_M := \gamma_g^T \circ \gamma_d^T(T)$. As before, we can do an analogous procedure on $T_d'^M$, ghosting all the vertices $(\mathfrak{G}, w)$ in M, in any order, and building a path $\gamma_g^{T'}$ along with the weighted tree $T'_M := \gamma_g^{T'} \circ \gamma_d^{T'}(T')$.
- Since all the remaining points in M are coupled, the two weighted trees T'_M and T_M must be isomorphic as tree structures. This is guaranteed by the properties of M. So, one can shrink T_M onto T'_M, and the composition of the shrinkings, executed in any order, is an edit path γ_s^T.

By construction, $\gamma_s^T \circ \gamma_g^T \circ \gamma_d^T(T) = T'_M$ and $(\gamma_d^{T'})^{-1} \circ (\gamma_g^{T'})^{-1} \circ \gamma_s^T \circ \gamma_g^T \circ \gamma_d^T(T) = T'$, where the inverse of an edit path is thought as the composition of the inverses of the single edit operations, taken in the inverse order.

Lastly, we call γ_M the set of all possible edit paths of the form $(\gamma_d^{T'})^{-1} \circ (\gamma_g^{T'})^{-1} \circ \gamma_s^T \circ \gamma_g^T \circ \gamma_d^T$, obtained by changing the order in which the edit operations are executed inside γ_d, γ_g, and γ_s. Even if γ_M is a set of paths, its cost is well-defined:

$$cost(M) := cost(\gamma_M) = cost(\gamma_d^T) + cost(\gamma_s^T) + cost(\gamma_d^{T'}).$$

See Figure 1 for an example of a mapping between weighted trees. We conclude this section by recalling that [36, Main Theorem] proves that given two weighted trees T and T', for every finite edit path $\gamma \in \Gamma(T, T')$, there exists a mapping $M \in \mathrm{Mapp}(T, T')$ such that $cost(M) \leq cost(\gamma)$.

Lastly, we consider $M_2(T, T') \subset \mathrm{Mapp}(T, T')$ defined as follows.

Definition 7. [36] A mapping $M \in \mathrm{Mapp}(T, T')$ has maximal ghostings if the following hold: $(v, \mathfrak{G}) \in M$ if, and only if, v is of degree 2 after the deletions in T, and, similarly, $(\mathfrak{G}, w) \in M$ if, and only if, w is of degree 2 after the deletions in T'.

A mapping $M \in \mathrm{Mapp}(T, T')$ has minimal deletions if the following hold: $(v, \mathfrak{D}) \in M$ implies that neither v nor $parent(v)$ are of degree 2 after applying all the other deletions in T, and, similarly, $(\mathfrak{D}, w) \in M$ implies that neither w nor $parent(w)$ are of degree 2 after applying all the other deletions in T'.

We collect all mappings with maximal ghostings and minimal deletions in the set $M_2(T, T')$.

Lemma 1. [36]

$$\min\{cost(M) \mid M \in \mathrm{Mapp}(T, T')\} = \min\{cost(M) \mid M \in M_2(T, T')\}.$$

4. Merge trees edit distance

In this section, we finally exploit the notation established in the previous sections to obtain a (pseudo) metric for merge trees.

4.1. Truncation operators

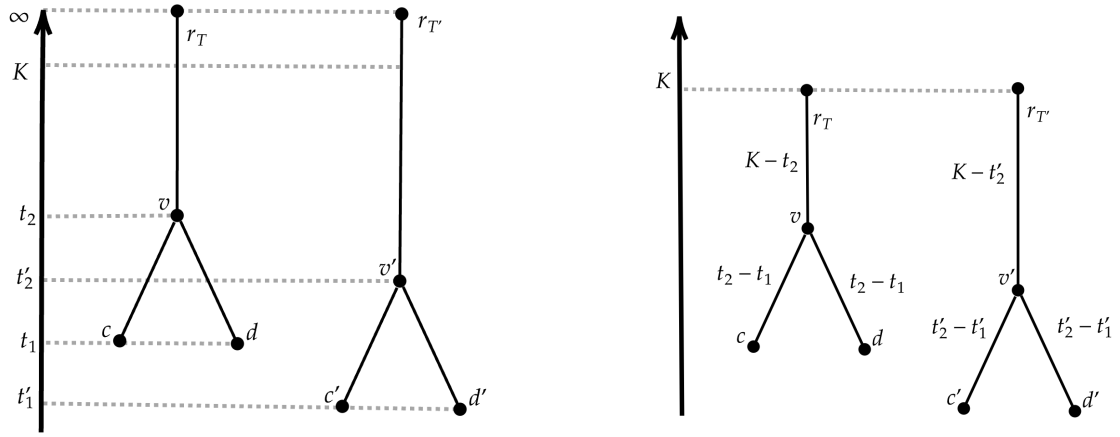
First, we need to bridge between merge trees and weighted trees, in order to induce a metric on merge trees by means of the edit distance defined in Section 3. The general idea is that we want to truncate the edge going to infinity of a merge tree to obtain a weighted tree.

Starting from a merge tree (T, h_T), it is quite natural to turn the height function h_T into a weight function w_T via the rule $w_T((v, v')) := h_T(v') - h_T(v)$. The monotonicity of h_T guarantees that w_T takes values in $\mathbb{R}_{>0}$. However, we clearly have an issue with the edge (v, r_T) as $h_T(r_T) = +\infty$. To solve this issue we need some novel tools; see Figure 2. First, consider the set of merge trees $\mathcal{MT}$ and build the subset $\mathcal{MT}_K := \{(T, h_T) \in \mathcal{MT} \mid \max h_T \leq K\}$, for some $K \in \mathbb{R}$. Then, define the truncation operator at height K as follows:

$$\begin{aligned} \mathrm{Tr}_K : \mathcal{MT}_K &\longrightarrow (\mathcal{T}, \mathbb{R}_{\geq 0}), \\ (T, h_T) &\mapsto (T, h_K) \mapsto (T, w_T), \end{aligned} \tag{4.1}$$

with $h_K(v) = h_T(v)$ if $v < r_T$ and $h_K(r_T) = K$. Then, we set $w_T((v, v')) = h_K(v') - h_K(v)$. To avoid $w_T((v, r_T)) = 0$, if $\max h_T = K$, we take $(T, h_T) \mapsto (T', h_{T'}) \mapsto (T', w_{T'})$ with T' obtained from T via the removal of r_T from V_T and (v, r_T) from E_T. The map $h_{T'}$ is $h_{T|V_{T'}}$. Then, $w_{T'}((v, v')) := h_{T'}(v') - h_{T'}(v)$.

In other words, with Tr_K we are fixing some height K, truncating the edge $(\arg\max h_T, r_T)$ at height K, and then obtaining a positively weighted tree, as in Figure 2. To go back with $(\mathrm{Tr}_K)^{-1}$, we "hang" a weighted tree at height K and extend the edge (v, r_T) to $+\infty$. We formally report these ideas in the following proposition, which we state without proof.



(a) Two merge trees T and T'.

(b) A graphical representation of the weighted trees $\mathrm{Tr}_K(T)$ and $\mathrm{Tr}_K(T')$. Roughly speaking, the merge trees T and T' are recovered sending the two roots back to infinity.

Figure 2. A graphical representation of the truncation operator.

Proposition 1. The operator $\mathrm{Tr}_K : \mathcal{MT}_K \to (\mathcal{T}, \mathbb{R}_{\geq 0})$ can be inverted via $(T, w_T) \mapsto \mathrm{Tr}_K^{-1}((T, w_T)) = (T', h_{T'})$ with the following notation: the tree structure T' is obtained from T via adding $r_{T'}$ to V_T and $(r_T, r_{T'})$ to E_T and ghosting r_T if it becomes a degree 2 vertex. Then, we have $h_{T'}(r_T) = K$ (if it is not ghosted, i.e., if it is of order greater than 2), $h_{T'}(r_{T'}) = +\infty$ and, recursively, for $(v, v') \in E_T$, $h_{T'}(v) = h_{T'}(v') - w_T((v, v'))$. Clearly, $\mathrm{Tr}_K^{-1}(\mathrm{Tr}_K((T, h_T))) \cong (T, h_T)$. Thus, $\mathcal{MT}_K \cong (\mathcal{T}, \mathbb{R}_{\geq 0})$ as sets for every $K \in \mathbb{R}$. Moreover, $\mathrm{Tr}_K((T, h_T)) \sim_2 \mathrm{Tr}_K((T', h_{T'}))$ if, and only if, $(T, h_T) \sim_2 (T', h_{T'})$.

4.2. Edit distance for merge trees

Consider $(T, h_T), (T', h_{T'}) \in \mathcal{MT}$, and select K such that $(T, h_T), (T', h_{T'}) \in \mathcal{MT}_K$. Let $(G, w_G) = \mathrm{Tr}_K((T, h_T))$ and $(G', w_{G'}) = \mathrm{Tr}_K((T', h_{T'}))$. Lastly, set:

$$d_K((T, h_T), (T', h_{T'})) := d_E((G, w_G), (G', w_{G'})).$$

Despite being a promising and natural pseudo metric, d_K is not defined on the whole $\mathcal{MT}$ and, on top of that, it is not clear if its value depends on the K we choose. The following result solves these issues and proves that we can use d_K to define a metric on $\mathcal{MT}$.

Proposition 2. (Truncation) Take (T, w_T) and $(T', w_{T'})$ weighted trees. Suppose r_T and $r_{T'}$ are of order 1 and take two splitting operations induced by the following edges replacements: $\{(v, r_T)\} \to$

$\{(v, v'), (v', r_T)\}$ and $\{(w, r_{T'})\} \to \{(w, w'), (w', r_{T'})\}$, with which we obtain the weighted trees (G, w_G) and $(G', w_{G'})$. Suppose that $w_G((v', r_T)) = w_{G'}((w', r_G))$. Then, $d_E(T, T') = d_E(sub_G(v'), sub_{G'}(w'))$.

We now exploit Proposition 2 to define the edit distance between merge trees.

Theorem 1. (Merge tree edit distance) Given two merge trees $(T, h_T), (T', h_{T'})$ such that $(T, h_T), (T', h_{T'}) \in \mathcal{MT}_K$ and $(T, h_T), (T', h_{T'}) \in \mathcal{MT}_{K'}$, then $d_K(T, T') = d_{K'}(T, T')$.

Thus, for any couple of merge trees in $\mathcal{MT}$, we can define the merge tree edit distance

$$d_E((T, h_T), (T', h_{T'})) := d_K((T, h_T), (T', h_{T'}))$$

for any $K \geq \max\{\max h_T, \max h_{T'}\}$.

For ease of notation, we call d_E both the edit distance between weighted trees and the one between merge trees. The arguments of the distances should clarify which of the two metrics we are referring to. By Proposition 1, we also have the following corollary of Theorem 1.

Corollary 1. The distance d_E is a pseudo metric on $\mathcal{MT}$ and a metric on $\mathcal{MT}/\sim_2$: given two merge trees (T, h_T) and $(T', h_{T'})$, $d_E((T, h_T), (T', h_{T'})) = 0$ if, and only if, $(T, h_T) \sim_2 (T', h_{T'})$.

Remark 1. Theorem 1 and Corollary 1 have a series of important implications. First, they say that $(\mathcal{MT}_K/\sim_2, d_E)$ is isometric and isomorphic to $(\mathcal{T}_2, d_E)$, and, thus, if we have a subset of merge trees contained in $\mathcal{MT}_K/\sim_2$, for some K, we can map them in $(\mathcal{T}_2, d_E)$ via Tr_K and carry out our analysis there. Second, suppose we are given a merge tree T'' with $\max h_{T''} > K$. For any two merge trees T, T' with $K \geq \max h_T, \max h_{T'}$, we can consider $K' \geq \max h_{T''}$ and compute $d_E(T, T'') = d_{K'}(T, T'')$ and $d_E(T', T'') = d_{K'}(T', T'')$. However, we do not have to compute again $d_{K'}(T, T')$, for we have $d_E(T, T') = d_{K'}(T, T') = d_K(T, T')$.

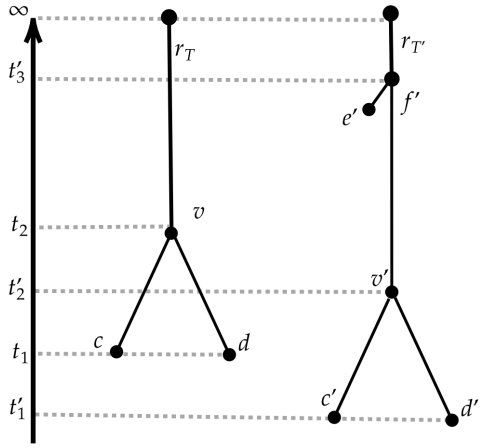
See Appendix A.1 for some examples. We close this section with the following remark.

Remark 2. A more naive approach could have been to model each merge tree as a triplet $(G, w_G, h_T(v)) \in (\mathcal{T}, \mathbb{R}_{\geq 0}) \times \mathbb{R}$, with $v = \arg\max h_T$ and $G = sub_T(v)$, i.e., to record the height of the last merging point in T and then remove (v, r_T) from E_T. Then, one could define a pseudo metric on $\mathcal{MT}$ via the rule:

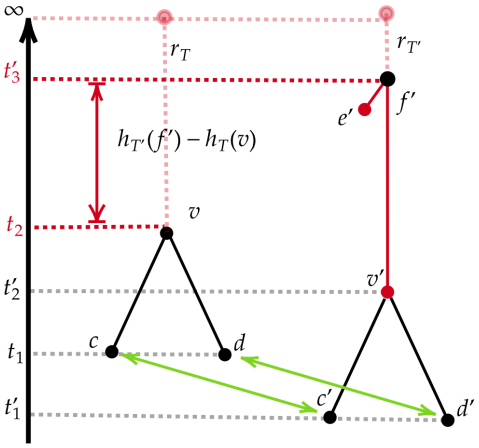
$$d((T, h_T), (T', h_{T'})) = \mid h_T(v) - h_{T'}(v') \mid + d_E((G, w_G), (G', w_{G'})),$$

but this leads to unstable behaviors, as in Figure 3.

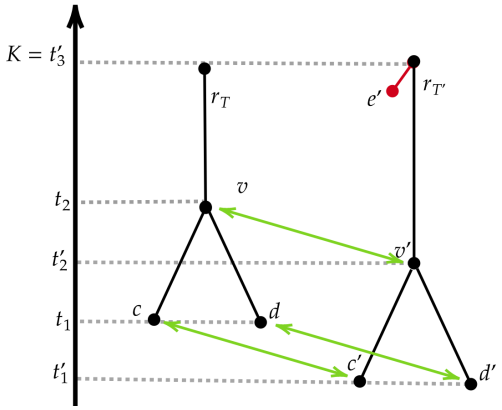
By definition, computing d_E for merge trees amounts to computing d_K, and so, we can exploit the algorithm presented in [36] to compute d_E for weighted trees.



(a) Two merge trees T (left) and T' (right).



(b) The trees T (left) and T' (right) in Figure 3(a) represented as triplets of the form $(G, w_G, h_T(v))$ (see Remark 2) with the (dotted) edges to infinity being removed to obtain weighted trees, and the height of the highest finite vertices being recorded as a real number.



(c) The weighted trees $\mathrm{Tr}_K(T)$ (left) and $\mathrm{Tr}_K(T')$ (right) with $K = t'_3$, obtained from the merge trees T and T' as in Figure 3(a).

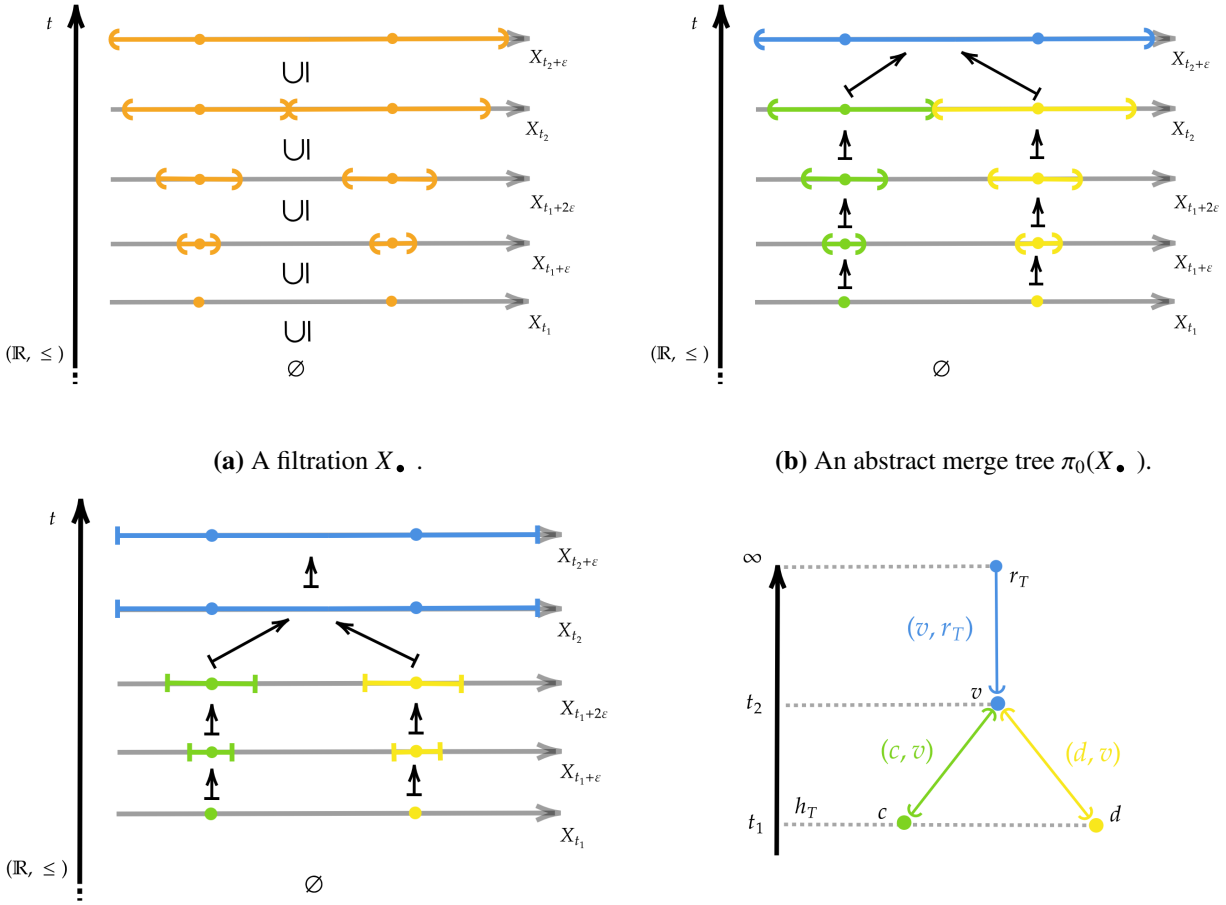
Figure 3. This example illustrates a situation where the metric defined in Remark 2 exhibits unstable behavior, while the metric d_K remains well-behaved. In the diagrams, red edges represent deletions, and green arrows indicate the pairing between remaining edges. The weight differences between paired edges are assumed to be negligible. In Figure 3(a), transforming the left merge tree into the right one involves accounting for the height differences $|h_T(v) - h_{T'}(f')|$ and the deletion costs associated with the red edges. In contrast, in Figure 3(c), the cost of the depicted mapping consists only of the (negligibly small) weight differences between the paired edges and the deletion of the short edge $(e', r_{T'})$.

Proposition 3. (Computational complexity [36]) Let T and T' be two merge trees with full binary tree structures with $\#E_T = N$ and $\#E_{T'} = M$. Then, $d_E(T, T')$ can be computed by solving $O(N \cdot M)$ BLP problems with $O(N \cdot \log(N) \cdot M \cdot \log(M))$ variables and $O(\log_2(M) + \log_2(N))$ constraints.

As anticipated in Section 1, the classical edit distance can be computed by solving $O(N \cdot M)$ BLP problems with $O(N \cdot M)$ variables and $O(\log_2(M) + \log_2(N))$ constraints [27]. In Appendix A.2, we show that we can borrow from [53,55] heuristics to speed up the computations and algorithms to obtain a polynomial time upper bound.

5. Abstract merge trees and the interleaving distance

Following [16,34], we introduce persistent sets. Figure 4 illustrates some of the objects we introduce in this section.



(a) A filtration $X_\bullet$.

(b) An abstract merge tree $\pi_0(X_\bullet)$.

(c) An abstract merge tree $\pi_0(X_\bullet)$.

(d) The merge tree $\mathcal{M}(\pi_0(X_\bullet))$ with $\pi_0(X_\bullet)$ as in Figure 4(c).

Figure 4. On the first line, we see an example of a filtration along with its abstract merge tree. In the bottom line, there is an abstract merge tree and the associated merge tree. The colors are used throughout the plots to highlight the relationships between the different objects.

Definition 8. [16] A filtration of topological spaces is a (covariant) functor $X_\bullet : \mathbb{R} \to \mathbf{Top}$ from the poset $(\mathbb{R}, \leq)$ to **Top**, the category of topological spaces with continuous functions, such that: $X_t \to X_{t'}$, for $t < t'$, are injective maps.

Example 1. Given a real valued function $f : X \to \mathbb{R}$, the *sublevel set* filtration is given by $X_t =$

$f^{-1}((-\infty, t])$ and $X_{t<t'} = i : f^{-1}((-\infty, t]) \hookrightarrow f^{-1}((-\infty, t'])$.

Example 2. Given a finite set $C \subset \mathbb{R}^n$, the *Čech* filtration is given by $X_t = \bigcup_{c\in C} B_t(c)$ with $B_t(c) = \{x \in \mathbb{R}^n \mid\mid\mid c - x \mid\mid\mid < t\}$. As before, $X_{t<t'} = i : \bigcup_{c\in C} B_t(c) \hookrightarrow \bigcup_{c\in C} B_{t'}(c)$.

Given a filtration $X_\bullet$, we can compose it with the functor π_0 sending each topological space into the set of its path connected components. We recall that, according to standard topological notation, $\pi_0(X)$ is the set of the path connected components of X, and given a continuous function $q : X \to Y$, $\pi_0(q) : \pi_0(X) \to \pi_0(Y)$ is defined as:

$$U \mapsto V \text{ such that } q(U) \subset V.$$

Definition 9. [9, 14] A persistent set is a functor $S : \mathbb{R} \to \mathbf{Sets}$. In particular, given a filtration of topological spaces $X_\bullet$, the persistent set of components of $X_\bullet$ is $\pi_0 \circ X_\bullet$. A (one-dimensional persistence module) is a functor $S : \mathbb{R} \to \mathbf{Vec}_\mathbb{K}$ with values in the category of vector spaces $\mathbf{Vec}_\mathbb{K}$.

By endowing a persistent set with the discrete topology, every persistence set can be seen as the persistence set of components of a filtration. Thus, a general persistent set S can be written as $\pi_0(X_\bullet)$, for some filtration $X_\bullet$.

Based on the notion of constructible persistent sets found in [16, 34], one then builds the following objects.

Definition 10. [37] An abstract merge tree (AMT) is a persistent set $S : \mathbb{R} \to \mathbf{Sets}$ such that there is a finite collection of real numbers $\{t_1 < t_2 < \ldots < t_n\}$ which satisfy:

- $S(t) = \emptyset$ for all $t < t_1$;
- $S(t) = \{\star\}$ for all $t > t_n$;
- if $t, t' \in (t_i, t_{i+1})$, with $t < t'$, then $S(t < t')$ is bijective.

The values $\{t_1 < t_2 < \ldots < t_n\}$ are called critical values of the tree, and there is always a minimal set of critical values [37]. We always assume to be working with such a minimal set.

If $S(t)$ is always a finite set, S is a finite abstract merge tree.

We report a result summarizing the relationship between abstract merge trees and merge trees. The main consequence of such a result is that merge trees considered up to degree 2 vertices are an appropriate discrete tool to represent the information contained in AMTs. Figures 4(a) and 4(b) can help the reader going through the following proposition.

Proposition 4. [37] The following hold:

1) We can associate to a regular abstract merge tree $R(\pi_0(X_\bullet))$, a merge tree without degree 2 vertices $\mathcal{M}(R(\pi_0(X_\bullet)))$;
2) We can associate to a merge tree (T, h_T), a regular abstract merge tree $\mathcal{F}((T, h_T))$. Moreover, we have $\mathcal{M}(\mathcal{F}((T, h_T))) \cong_2 (T, h_T)$ and $\mathcal{F}(\mathcal{M}(R(\pi_0(X_\bullet)))) \cong_{a.e.} \pi_0(X_\bullet)$;
3) Given two abstract merge trees $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$, $\mathcal{M}(R(\pi_0(X_\bullet))) \cong \mathcal{M}(R(\pi_0(Y_\bullet)))$ if, and only if, $\pi_0(X_\bullet) \cong_{a.e} \pi_0(Y_\bullet)$;
4) Given two merge trees (T, h_T) and $(T', h_{T'})$, we have $\mathcal{F}((T, h_T)) \cong \mathcal{F}((T, h_T))$ if, and only if, $(T, h_T) \cong_2 (T', h_{T'})$.

Remark 3. For the sake of brevity, we don't report the explicit construction of $\mathcal{M}$ and $\mathcal{F}$, which can be found in [16,37]. Intuitively, $\mathcal{M}$ gives a discretization of an abstract merge tree $\pi_0(X_\bullet)$, by sending it in the "smallest" graph (in terms of vertices) needed to describe the maps $\pi_0(X_{t\leq t'})$, together with the induced function defined on the vertices. In particular, this can be done without using degree 2 vertices, which are superfluous. Viceversa, $\mathcal{F}$ associates to a merge tree a functor which is roughly an algebraic equivalent of its geometric realization, where one can actually consider all the individual points on the edges. Clearly, degree two vertices are lost in this procedure.

We adapt the definition of the interleaving distance between merge trees, which is originally stated by [4] with a different notation.

Definition 11. (Adapted from [4] and [17]) Given $X_\bullet$ filtration and $\varepsilon > 0$, we define $X^\varepsilon_\bullet$ as $X^\varepsilon_t := X_{t+\varepsilon}$ and $X^\varepsilon_{t\leq t'} := X_{t+\varepsilon\leq t'+\varepsilon}$. Coherently with the literature on Reeb graphs, we define the ε-smoothing operator $\mathcal{S}_\varepsilon$ as: $\mathcal{S}_\varepsilon(\pi_0(X_\bullet)) := \pi_0(X^\varepsilon_\bullet)$. Note that $\mathcal{S}_\varepsilon$ also acts on natural transformations $\alpha : \pi_0(X_\bullet) \to \pi_0(Y_\bullet)$: by setting $\mathcal{S}_\varepsilon(\alpha_t) := \alpha_{t+\varepsilon}$, one obtains $\mathcal{S}_\varepsilon(\alpha) : \mathcal{S}_\varepsilon(\pi_0(X_\bullet)) \to \mathcal{S}_\varepsilon(\pi_0(Y_\bullet))$. Lastly, we have the natural transformation $i^\varepsilon_{\pi_0(X_\bullet)} : \pi_0(X_\bullet) \to \mathcal{S}_\varepsilon(\pi_0(X_\bullet))$ given by $(i^\varepsilon_{\pi_0(X_\bullet)})_t := \pi_0(X_{t\leq t+\varepsilon}) : \pi_0(X_t) \to \pi_0(X_{t+\varepsilon})$.

Definition 12. (Adapted from [4]) Take two abstract merge trees $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$. Two natural transformations $\alpha : \pi_0(X_\bullet) \to \mathcal{S}_\varepsilon(\pi_0(Y_\bullet)), \beta : \pi_0(Y_\bullet) \to \mathcal{S}_\varepsilon(\pi_0(X_\bullet))$ are ε-compatible if:

- $\mathcal{S}_\varepsilon(\beta) \circ \alpha = i^{2\varepsilon}_{\pi_0(X_\bullet)}$;
- $\mathcal{S}_\varepsilon(\alpha) \circ \beta = i^{2\varepsilon}_{\pi_0(Y_\bullet)}$.

Unfolding the definitions, this means:

- $\beta_{t+\varepsilon} \circ \alpha_t = \pi_0(X_{t\leq t+2\varepsilon})$
- $\alpha_{t+\varepsilon} \circ \beta_t = \pi_0(Y_{t\leq t+2\varepsilon})$.

Then, the interleaving distance between $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$ is:

$$d_I(\pi_0(X_\bullet), \pi_0(Y_\bullet)) = \inf\{\varepsilon > 0 \mid \exists \alpha, \beta \ \varepsilon\text{-compatible}\}.$$

We also say that $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$ are $d_I(\pi_0(X_\bullet), \pi_0(Y_\bullet))$-interleaved.

To lighten the notation, we may also write $d_I(T, T')$, implying $d_I(\pi_0(X_\bullet), \pi_0(Y_\bullet))$, with $T = \mathcal{M}(\pi_0(X_\bullet))$ and $T' = \mathcal{M}(\pi_0(Y_\bullet))$.

Remark 4. We have introduced the interleaving distance with this notation, because we make use of smoothing operators in Proposition 6. The name smoothing operator comes from the interleaving distance between Reeb graphs. Introducing Reeb graphs and such distance to motivate our definition is outside the scope of this work. However, for the reader familiar with such notions, we point out the following facts. Merge trees can be obtained as Reeb graphs of the epigraph of a function $f : X \to \mathbb{R}$ with the projection on $\mathbb{R}$: consider $f : X \to \mathbb{R}$ and define $\Gamma_f = \{(x,t) \in X \times \mathbb{R} \mid f(x) \leq t\}$. Then, one can take the projection to $\mathbb{R}$ and obtain: $F : \Gamma_f \to \mathbb{R}$. Then, $F^{-1}(t) = \{(x,t) \mid f(x) \leq t\} = f^{-1}((-\infty, t])$.

Thus, the merge tree of f is equivalently represented by the Reeb cosheaf [17] $U \mapsto \pi_0(F^{-1}(U))$. The smoothed version of this Reeb cosheaf, as defined in [17], is induced by $F_\varepsilon^{-1}(t) := F^{-1}((t-\varepsilon, t+\varepsilon)) = f^{-1}((-\infty, t+\varepsilon])$, which implies that the smoothed Reeb cosheaf corresponds to the smoothed merge tree in the sense that we define. This also implies that the interleaving distance between merge trees can be induced by the one between Reeb graphs.

6. Finite stability and stability-Sensitivity trade-off

We now explore some key relationships between formal properties of metrics and practical applications of merge trees.

Merge trees are trees which are used to represent data and, considering populations of such objects, to explore the variability of a dataset. Thus, any metric which is employed on merge trees (and on any data representation in general) needs to measure the variability between merge trees in a "sensible" way, coherently with the application considered. A formal way to assess such properties often comes in the form of continuity results of the operator mapping data into representations, which, in the TDA's literature, are called stability results. In particular, a metric d between persistence modules or merge trees is referred to as *stable* if the operator mapping functions to modules/merge trees (via the sublevel set filtration) is 1-Lipschitz continuous. That is, $d(S_f, S_g) \leq \| f-g \|_\infty$, with S_f, S_g being the persistence modules/merge trees representing f, g. This holds, for instance, for the interleaving distance between persistence modules [11] and merge trees.

Among stable distances, interleaving distances are often used as a benchmark to study the stability of metrics, as such distances are *universal* for persistence modules and merge trees: for any other stable metric d between persistence modules or merge trees, we always have $d(S_f, S_g) \leq d_I(S_f, S_g)$. Thus, a good behavior in terms of interleaving distance implies a good handling of pointwise noise between functions and so also interpretability of the metric. In it also worth mentioning that other kinds of universal properties have also been considered in literature [8, 30].

Being the edit distance a summation of the costs of local modifications of trees, we expect that d_E cannot be bounded from above by the interleaving distance between merge trees, as such metric in some sense, measures only the biggest modification needed to optimally match two merge trees. Thus, a suitable stability condition, which we name finite stability, would be for ε-interleaved trees to be obtained one from the other just by means of edits with cost ε, and with edit paths whose number of operations is bounded by the sizes of the trees. As a consequence, the cost of the local modifications we need to match the two merge trees goes to 0 as their interleaving distance gets smaller and smaller, as is the case for Wasserstein metrics between PDs.

Definition 13. Given a persistence module $S : \mathbb{R} \to \mathbf{Vec}_\mathbb{K}$, we define its dimension as $\dim(S) := \#PD(S)$, i.e., the number of points in its persistence diagram (if it exists). When S is generated on a field $\mathbb{K}$ by an abstract merge tree $\pi_0(X_\bullet)$, we have $\dim(S) := \#PD(S) = \#L_T$, with $(T, h_T) = \mathcal{M}(\pi_0(X_\bullet))$. In this case, we use $\dim(T)$ to refer to $\dim(S)$. We also fix the notation $\text{size}(T) := \#E_T$. Note that $\text{size}(T) \leq 2\dim(T)$. Lastly, we define the space of merge trees with uniform upper bound on their size:

$$\mathcal{MT}^N = \{T \in \mathcal{MT} \mid \text{size}(T) \leq N\}.$$

With these pieces of notation, we can introduce the notion of *finitely stable* metrics.

Definition 14. A metric d for merge trees is finitely stable if there is $C > 0$ such that for every pair of ε-interleaved AMTs $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$, upon setting $T = \mathcal{M}(\pi_0(X_\bullet))$ and $T' = \mathcal{M}(\pi_0(Y_\bullet))$, we have:

$$d(T, T') \leq C(\dim(T) + \dim(T'))\varepsilon.$$

The name "finite" stability is motivated by the following immediate result, which we state without proof.

Proposition 5. Let d be a finitely stable metric between merge trees and let $\varphi : (X, d_X) \to (\mathcal{MT}, d_I)$ be a Lipschitz operator from a metric space (X, d_X) to the space of merge trees considered with the interleaving distance. Then, for every $N \in \mathbb{N}$, the following operator is Lipschitz:

$$\varphi_{|\varphi^{-1}(\mathcal{MT}^N)} : (\varphi^{-1}(\mathcal{MT}^N), d_X) \to (\mathcal{MT}^N, d).$$

In view of these definitions, we can rewrite the following theorem from [39].

Theorem 2. [39] If there are α, β ε-compatible maps between two AMTs $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$, then there exist a mapping M between $T = \mathcal{M}(\pi_0(X_\bullet))$ and $T' = \mathcal{M}(\pi_0(Y_\bullet))$ such that $cost_M((a, b)) \leq 2\varepsilon$ for every $(a, b) \in M$. Since $\text{size}(T) \leq 2\dim(T)$ and $\text{size}(T') \leq 2\dim(T')$, d_E is finitely stable.

Now, we prove a complementary result, which gives the complete picture of the stability properties of d_E, allowing the comparison with Wasserstein distances between PDs. Note that in the remaining part of the manuscript, to lighten the notation, we often deliberately confuse the AMT $\pi_0(X_\bullet)$ and the associated merge tree $T = \mathcal{M}(\pi_0(X_\bullet))$.

Theorem 3. We always have $d_I(T, T') \leq d_E(T, T')$.

The bound given by Theorem 3 is tight.

Example 3. Consider $f : I \to \mathbb{R}$ and $g(x) = f(x) + k$ for some fixed $k \in \mathbb{R}$, and let $X_\bullet : (\mathbb{R}, \leq) \to \mathbf{Top}$ be the sublevel set filtration of f and $Y_\bullet : (\mathbb{R}, \leq) \to \mathbf{Top}$ of g (see Section 5). Let (T, h_T) and $(T', h_{T'})$ be the merge trees representing $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$, and we have $d_I(T, T') = d_E(T, T') = k$.

Putting together Theorem 3 and Theorem 2, we obtain the following inequalities.

Corollary 2.

$$d_I(T, T') \leq d_E(T, T') \leq 2(\text{size}(T) + \text{size}(T'))d_I(T, T'). \tag{6.1}$$

Equation (6.1) is the one that better summarizes the stability properties of d_E. Moreover, inequalities similar to the ones expressed by Eq (6.1) relate also the bottleneck and the 1-Wasserstein distances between PDs. We recall that the points of a persistence diagram are often called *persistence pairs* and that, given two persistence diagrams D and D', the expression of the p-Wasserstein distance between them is the following:

$$W_p(D, D') = \left(\inf_{\gamma} \sum_{x \in D} \| x - \gamma(x) \|_\infty^p \right)^{1/p},$$

where γ ranges over the functions partially matching points between diagrams D and D', and matching the remaining points of both diagrams with the line $y = x$ on the plane (for more details, see [13]).

The case $p = \infty$ is usually referred to as the bottleneck distance d_B. Thus, for every pair of persistence diagrams D, D', we have:

$$d_B(D, D') \leq W_1(D, D') \leq (\dim(D) + \dim(D'))d_B(D, D'). \tag{6.2}$$

The similarity between Eqs (6.1) and (6.2) qualifies d_E as an analogous for merge trees of the 1-Wasserstein distance for PDs. As they have analogous stability properties, w.r.t. the universal distance of, respectively, merge trees and PDs. Those properties are in line with our expectations: when editing ε-interleaved merge trees, we need to produce a small local modification for each vertex of the merge tree and then add up all the contributions. In the same way, with the 1-Wasserstein distance, one measures the difference between ε-interleaved diagrams aggregating the small discrepancies between all persistence pairs.

Roughly speaking, we can say that the left side of Eqs (6.1) and (6.2) guarantees that d_E and W_1 always have at least the same discriminatory power compared to their universal counterparts, while the right side guarantees interpretability. In fact, via the universal properties of the bottleneck and interleaving distances, we can replace d_I and d_B with $\| f - g \|_\infty$ in the right side of Eqs (6.1) and (6.2). In this way, we can describe the behavior of d_E and W_1 in terms of the operators mapping functions into topological summaries.

Universal distances like the interleaving and the bottleneck ones satisfy such strong stability properties because, in some sense, they measure the maximum cost of the modifications that we make on the considered objects. In other words, they are very stable because they are heavily insensitive: one could add an infinite amount of modifications to the considered objects, smaller than the biggest one (like infinite points close enough to the diagonal of PDs), without these changes affecting the distance. That is, they are very poor in discriminating between different objects, and not just when such modifications are small, as shown in Section 7. This has clear repercussions on data analysis pipelines. More formally, the topology induced by d_E is strictly finer than the one induced by d_I: on one hand, for any $T' \in \mathcal{MT}$ and $C > 0$, the following hold:

$$\{T \in \mathcal{MT} \mid d_E(T, T') < C\} \subset \{T \in \mathcal{MT} \mid d_I(T, T') < C\}.$$

On the other hand, it is not difficult to build a tree T and a sequence $\{T_n\}_{n\in\mathbb{N}}$ such that $d_I(T_n, T) \to 0$ and $d_E(T_n, T) \to +\infty$. See also the upcoming example.

Example 4. Let $h_n : [0, 1] \to \mathbb{R}$ be $h_n(x) = 1/n \cdot \sin(2\pi n^2 x)$ and $f : [0, 1] \to \mathbb{R}$ be $f(x) = x$. Let T be the merge tree of (the sublevel set filtration of) f and T_n the merge tree of $f_n := f + h_n$. We have $d_I(T, T_n) \leq 1/n$ while $d_E(T, T_n) \sim O(n)$. Similarly, let D be the PD of f and D_n the PD of f_n. We have $d_B(D, D_n) \leq 1/n$ while $W_1(D, D_n) \sim O(n)$.

These considerations reinforce the fact that, in general, d_E is potentially better than d_I at discriminating trees, as it has more open sets to separate objects, just as W_1 is better than d_B at distinguishing between persistence diagrams.

We can see the problem of balancing stability and sensitivity as a bias-variance trade-off: a very stable metric captures reduced variability, potentially "underfitting" the data in order to be less susceptible to noise, while a more sensitive metric will be more affected by noise, with the risk of capturing too much variability and "overfitting" the data.

On paper, universal metrics can be more safely used when we need to work with topological summaries extracted from noisy data. Consider, for instance, the following statistical model, which

is standard in functional data analysis and regression problems [42]. Take an iid sample $\{(x_i, y_i)\}_{i=1}^n$ from the model $\mathcal{Y} \mid \mathcal{X} = x \sim f(x) + \varepsilon$, with ε being a noise random variable with zero mean and finite variance, $\mathcal{X}, \mathcal{Y}$ being two real valued random variables, and $f : [a, b] \to \mathbb{R}$ a smooth function. Suppose we can build an estimate $\hat{f}_n$ of $f(\cdot) = \mathbb{E}(\mathcal{Y} \mid \mathcal{X} = \cdot)$ from $\{(x_i, y_i)\}_{i=1}^n$. Having $d_I(T_f, T_{\hat{f}_n}) \leq \| f - \hat{f}_n \|_\infty$, implies that $\hat{f}_n$ can be a very naive estimate of f: as long as the estimate has a low point wise error, we know that $T_{\hat{f}_n}$ is a good estimate of T_f in terms of d_I. Instead, if $\hat{f}_n$ is a very rough estimate of f (e.g., the piecewise affine interpolation of $\{(x_i, y_i)\}_{i=1}^n$), it could be a problem for d_E and W_1, as $\hat{f}_n$ is likely to have many ancillary/noisy oscillations, which can blow up size$(T_{\hat{f}_n})$ and dim$(PD(\hat{f}_n))$, so that $T_{\hat{f}_n}$ and $PD(\hat{f}_n)$ can be very poor approximations of T_f and $PD(f)$ in terms of d_E and W_1, respectively. However:

1) As showcased in Section 7.2, there could be situations where the sensitivity of d_E makes it a more effective data analysis tool w.r.t. d_I, even in the presence of considerable levels of noise;
2) As shown in [39], one can solve the problem of obtaining estimates $\hat{f}_n$ which make $T_{\hat{f}_n}$ a good estimator of T_f also in terms of d_E. That is, the problem of denoising data, i.e., estimating topological summaries, can be successfully tackled even for more sensitive metrics, and one is not constrained to use universally stable metrics. Of course, the same applies if the analysts and the field experts can assume that the data generating process is affected by noise only in negligible terms.

We want to add some last details on this topic. In particular, [39] considers only functions of the form $f : [a, b] \to \mathbb{R}$, which means that the problem of estimating merge trees from noisy data is far from being solved, leaving room for novel developments. For instance, we believe that smoothing procedures can be designed working directly on topological summaries, with very general assumptions on the data-generating models, exploiting results like the following.

Proposition 6. Consider T and $\{T_n\}_{n\in\mathbb{N}}$ merge trees such that $d_I(T, T_n) \to 0$, and let d be a finitely stable distance between merge trees. For any $\delta > 0$, there exists $\varepsilon > 0$ (depending also on T) such that, for $T'_n := \mathcal{S}_\varepsilon(T_n)$, we have:

1) $d_I(T_n, T'_n) \leq \delta$ for all n;
2) $\lim_{n\to\infty} d(T, T'_n) \leq \delta$.

To better understand the implications of Proposition 6, consider the following setting. Fix a merge tree T and suppose T_n is an estimate of T obtained from noisy data, with n being some sampling parameter of the data. Suppose that, as n grows, we can get better and better estimates of T in terms of the interleaving distance, i.e., $d_I(T, T_n) \xrightarrow{n} 0$ (possibly, with high probability, see [6] for similar results obtained for persistence diagrams). For instance, in the functional model presented in the previous paragraphs, n would be the number of points which we sample from the model, and T_n could be the merge tree obtained from a suitable kernel smoothing of the sampled points. As a consequence of Proposition 6, once we fix an error tolerance $\delta > 0$, we know there is $\varepsilon_T^\delta > 0$ such that the ε_T^δ-smoothed merge trees $T'_n := \mathcal{S}_{\varepsilon_T^\delta}(T_n)$ converge to T in terms of d_E up to an error of δ.

Of course these considerations are far from being satisfactory: since T is unknown, one needs to estimate ε_T^δ, and, moreover, it would be desirable to find a way to let $\delta \to 0$. Still, the level of generality of these considerations motivates further research in this direction.

Summarizing, it is possible for the analysts to build statistically grounded pipelines involving metrics like d_E and W_p, which are much more sensitive, and whose discriminative power is arguably more useful in many data analysis scenarios compared to their universal counterparts. It is to the point that, to the best of our knowledge, Wasserstein metrics are more frequently used in applications than the bottleneck one. On the other hand, if data is heavily corrupted by noise and cannot be effectively denoised, this will affect the results obtained with W_p and d_E much more than if the analysis had been carried out with d_B and d_I. Nonetheless, even in such situations, W_p and d_E may still prove to be more effective, as showcased in Section 7.2.

The final choice of a metric to be used in a data analysis should be guided by many factors, adding to the stability-sensitivity discussion any specific knowledge about the data and the whole tree-generating pipeline. To such extent, being able to choose between different tools, each with its own well-understood characteristics, is certainly something to strive for.

7. Simulations and case studies

In this section, we take a practical look at the discussion presented in Section 6, comparing d_E and the interleaving distance in simulated scenarios and real-data applications. Notably, there are just a few computational procedures for d_I, which are either unreliable or very expensive (see [38]). This also suggests that its popularity is primarily driven by its strong theoretical properties, which, while valuable, may not translate directly into practical utility, as for the bottleneck distance between persistence diagrams.

To compute d_I, we resort to the computational framework presented in [38], which proposes mixed linear programming procedures to find upper and lower bounds to the interleaving distance. We call such bounds, respectively, d_u and d_l. In particular,

1) We use the upper bound d_u as an estimate of d_I. Thus, in the upcoming formulas, tables, etc., we will always write d_u to highlight that we are considering a proxy of d_I, but, to lighten the notation, in the main text we will deliberately confound the interleaving distance and its estimate;

2) We use the lower bound d_l and the bottleneck distance between persistence diagrams to find the potential error range of the computed upper bound, exploiting the fact that $d_B \leq d_I$ [4]. More precisely, we check the relative discrepancy:

$$\Delta := (d_u - \max\{d_l, d_B\})/d_u. \tag{7.1}$$

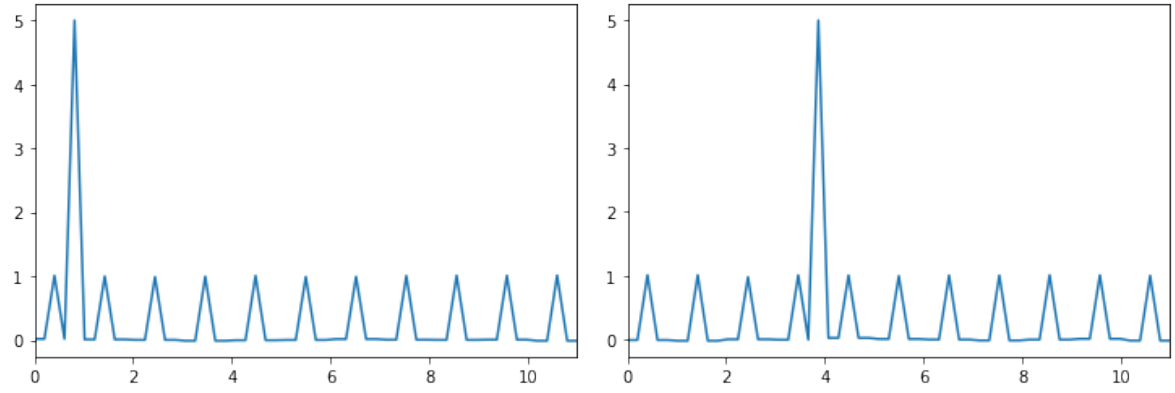
   Such quantity, in fact, gives an upper bound on the relative difference between d_I and d_u, as $d_u - d_I \leq \Delta \cdot d_u$. We stress that this is not an error, but an upper bound on the error $d_u - d_I$;

3) We compare the sup norms between the considered functions with d_u, to check if it shows instances of unstable behavior, i.e., $d_u(T_f, T_g) > \| f - g \|_\infty$.

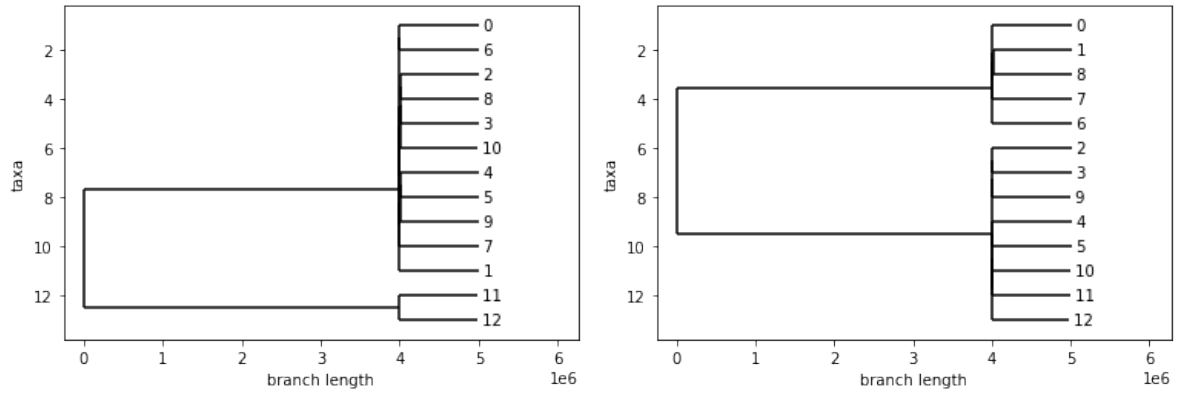
At the end of every section, apart from Section 7.1, in which the computations can be hand checked, we comment on these quantities to strengthen our findings.

7.1. Toy example

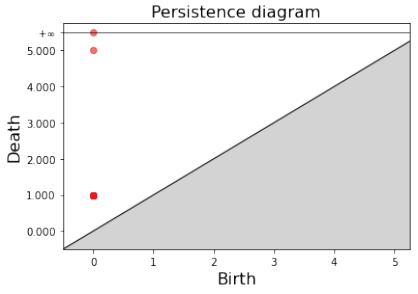
We start illustrating the practical differences between d_I and d_E with a carefully designed toy example that highlights some key properties and behaviors of the metrics. Refer also to Figure 5.



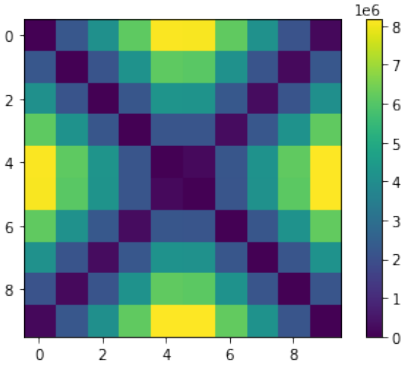
(a) The functions f_0, f_3 belonging to the simulated dataset decribed in Section 7.



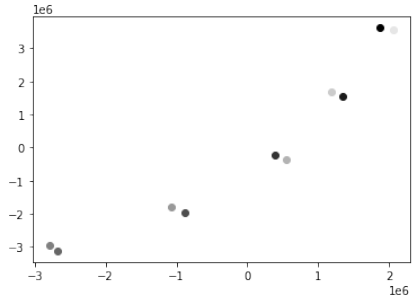
(b) The merge trees (T_{f_0}, h_{f_0}) and (T_{f_3}, h_{f_3}) associated to the functions in Figure 5(a).



(c) The persistence diagram representing the functions in Figure 5(a). The point $(0, 1)$ has multiplicity equal to the number of local minima minus 1.



(d) Matrix of pairwise distances of the merge trees obtained from $\{f_i\}_{i=0}^{10}$, with d_E.



(e) Multidimensional scaling embedding in $\mathbb{R}^2$ of the matrix of pairwise distances shown in Figure 5(d). The shades of gray describe, from white to black, the ordering of the trees.

Figure 5. Plots related to the simulated scenario presented in Section 7.1.

For $i = 0, \ldots, 9$, let $h_i : [0, 11] \to \mathbb{R}$ be such that $h_i \equiv 0$ on $[0, 11] - [i + 1/3, i + 2/3]$, while, on $[i + 1/3, i + 2/3]$, h_i is the linear interpolation of $(i + 1/3, 0)$, $(i + 1/2, 1)$, and $(i + 2/3, 0)$. Then, for $i = 0, \ldots, 9$, define H_i as $H_i \equiv 0$ on $[0, 11] - [i + 2/3, i + 1]$, while, on $[i + 2/3, i]$, H_i is the linear interpolation of $(i + 2/3, 0)$, $(i + 3/4, 5)$, and $(i + 1, 0)$.

Then, f_i, $i = 0, \ldots, 9$, is obtained as follows:

$$f_i = H_i + \sum_{j=0}^{9} h_j.$$

Refer to Figure 5(a) to better visualize this dataset: we have a constant set of lower peaks at height 1 and a higher peak with height 5, which is shifting from left to right as i increases. In this way, we are just changing the left-right distribution of the smaller peaks w.r.t. the highest one.

We obtain the associated merge trees and then compute the pairwise distances between the merge trees with d_E. The results are represented in Figure 5(d): the shortest edit path between the i-th merge tree and $i + 1$-th is given by the deletion of one leaf in each tree to make the disposition of leaves coincide between the two trees. The more the peaks' disposition is different between the two trees, the more one needs to delete leaves in both trees to find the path between them. Note that the first function (the one in which the highest peak is the second peak) and the last function (the one in which the highest peak is the second-last peak) can be obtained one from the other via a y-axis symmetry and translation. Similarly, the second function is equal, up to homeomorphisms of the domain, to the third-last one, etc. Thus, the merge trees are the same (see also [39]). To sum up the situation depicted in the first row of Figure 5(d), first we get (left-to-right) farther away from the first merge tree, and then we return closer to it. This intuition is confirmed by looking at the multidimentional scaling (MDS) embedding in $\mathbb{R}^2$ of the pairwise distance matrix (see Figure 5(e), and note that the shades of gray reflect, from white to black, the ordering of the merge trees). The discrepancies between the couple of points which should be identified are caused by numerical errors.

First, it is very easy to observe that all such functions can't be distinguished by PDs, since they all share the PD in Figure 5(c). Second, the interleaving distance between any two merge trees representing two functions f_i and f_j is $1/2$ if $i \neq j$ and 0 otherwise. Thus, the metric space obtained with d_I from the dataset $\{f_i\}_{i=0}^{9}$ is isometric to the discrete metric space on 5 elements, where each point is on the radius 1 sphere of any other point. Note that this results in a poorer "geometric structure" .

We point out that there are applications in which it would be important to separate f_0 and f_4 more than f_0 and f_1, because they differ by "a higher amount of edits": for instance, in [10], merge trees are used to represent tumors, with leaves being the lesions, and it is well-known in literature that the number of lesions is a non-negligible factor in assessing the severity of the illness [33], and, thus, a metric more sensible to the cardinality of the trees is more suitable than d_I.

7.2. Stability vs sensitivity

Now, we pick a standard generative model in functional data analysis and showcase how the universal properties of the interleaving distance do not make it necessarily the best data analysis tool to be used, even in presence of noise and absence of denoising procedures. Roughly speaking, we consider two smooth functions f and g, add pointwise noise to them and see which metric, between d_E and d_I, can better recognize if the observed function is a noisy observation of f or of g (without employing any smoothing procedure).

More formally, the data generating pipeline is the following. We generate two smooth functions f and g by interpolating, with cubic splines, two sets of randomly chosen couples in $[0, 1] \times \mathbb{R}$. The distribution that we use to sample those sets of couples is the following: we take $0 = x_1 < \ldots < x_N = 1$,

$N = 20$, forming a regular grid in $[0, 1]$. Then, y_i^j, $i = 1, \ldots, N$, $j = 1, 2$ are sampled independently from a Gaussian with mean 0 and standard deviation 50. Then, $\{(x_i, y_i^1)\}_{i=1,\ldots,N}$ are interpolated to obtain f and $\{(x_i, y_i^2)\}_{i=1,\ldots,N}$ are interpolated to obtain g. Other methods to sample a couple of random smooth functions could be employed as well.

We now report one of the standard generative models for functional data (see [42]), which we call model (M).

(M) We have a real random variable $\mathcal{X}$ distributed with density p, supported on a compact interval and bounded away from 0 on its support. We then consider a real random variable $\mathcal{Y}$ such that $\mathcal{Y} \mid \mathcal{X} = x \sim h(x) + \delta$, with δ being an independent noise variable with zero mean and finite variance, and h being a smooth function.

When the analyst observes a set of couples $\{(a_i, b_i)\}_{i=1,\ldots,n}$ which can be regarded as a partial observation of a function (i.e., functional data), such couples are usually modeled as i.i.d. samples taken from (M). We specify model (M), calling it (M_f^σ), assuming p equal to the uniform density on $[0, 1]$, h equal to f, and the noise variable δ being a Gaussian with mean zero and standard deviation σ. We also consider (M_g^σ), which is analogous to (M_f^σ), except the smooth function h is taken equal to g.

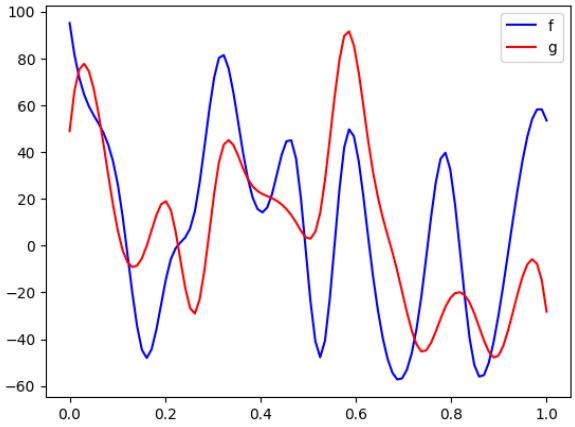
For each value of $\sigma \in \{0.1, 1, 10, 15, 25\}$, we sample a dataset D_f^σ where each of the 30 i.i.d. observations (i.e., each of the partially observed functions) is obtained by sampling independently $n = 50$ couples from (M_f^σ). Similarly, we obtain a dataset D_g^σ in which each of the 30 i.i.d. observations is obtained by sampling independently $n = 50$ couples from (M_g^σ). The dataset D^σ is then the union of D_f^σ and D_g^σ.

For each observation in D^σ, we compute the merge tree of the linear interpolation of the couples, and then take the matrices of pairwise distances between the trees using d_u and d_E. As σ, which controls the magnitude of the noise, increases, we are interested in two things:

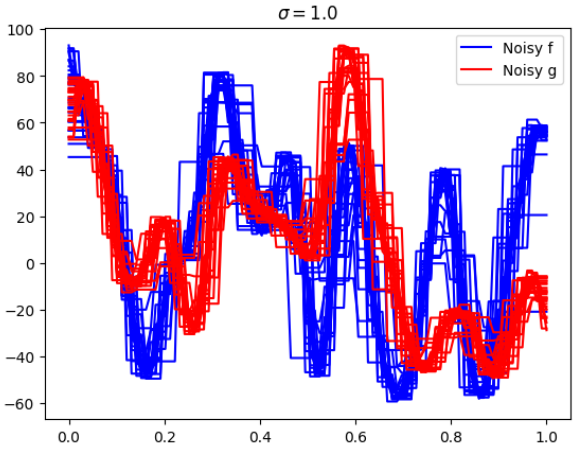
1) Observing the consequences of the stability properties on the distances;

2) Observing to what extent d_u and d_E can capture the clustering structure in the data, separating data sampled from D_f^σ versus data sampled from D_g^σ.

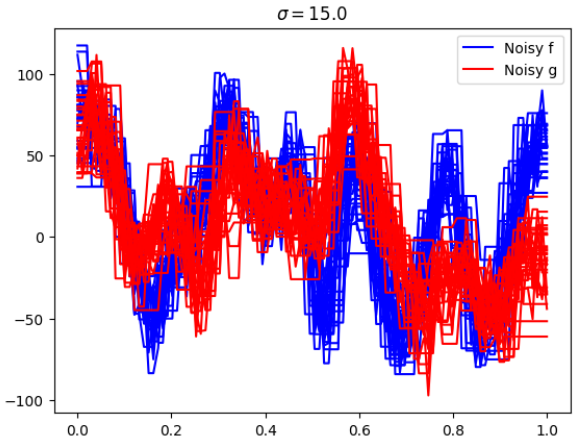
Figure 6 contains some plots related to this simulated scenario: in Figure 6(a), we see the smooth "baseline" functions f, g; while in Figures 6(b)–6(d), we report the datasets D^σ, for some increasing values of σ. The sampled partial observations of functions have been extended on $[0, 1]$ via a nearest value extension (NVE) technique, both for plotting purposes and to compute the sup norm between the extended functions, to check the stability of the upper bound d_u. As already mentioned, merge trees, instead, have been computed by considering the piecewise affine functions induced by each sample $\{(a_i, b_i)\}_{i=1,\ldots,n}$ - without extending the resulting functions to the whole interval $[0, 1]$.



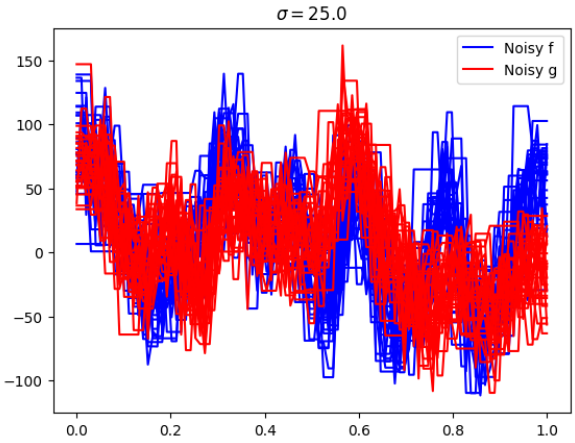
(a) The two random functions $f, g : [0, 1] \to \mathbb{R}$.



(b) The two datasets D_f^σ and D_g^σ, for $\sigma = 1$.



(c) The two datasets D_f^σ and D_g^σ, for $\sigma = 15$.



(d) The two datasets D_f^σ and D_g^σ, for $\sigma = 25$.

Figure 6. Plots related to the simulated scenario presented in Section 7.2. Each of the partially observed functions, generated according to the models (M_f^σ) or (M_g^σ), have been extended on $[0, 1]$ using NVE.

First, we comment on the practical comparison between the stability of d_u and d_E. Figure 7(a) summarizes the behavior of the distances (interleaving or edit) between the merge trees of the noisy functions and either T_f or T_g, the merge trees of the underlying smooth functions, depending on the generative model. More precisely,

1) The orange line interpolates, for each value of σ, the median of:

$$\{d_u(T_h, T_{h,\sigma}) \mid h \in \{f, g\}, T_{h,\sigma} \text{ is the merge tree of some } \{(a_i, b_i)\}_{i=1,\dots,50} \in D_h^\sigma\};$$

2) The green line interpolates, for each value of σ, the median of:

$$\{d_E(T_h, T_{h,\sigma}) \mid h \in \{f, g\}, T_{h,\sigma} \text{ is the merge tree of some } \{(a_i, b_i)\}_{i=1,\dots,50} \in D_h^\sigma\};$$

3) The blue line interpolates, for each value of σ, the median of:

$$\{\| h - h_\sigma \|_\infty \mid h \in \{f, g\}, h_\sigma \text{ is the NVE on } [0, 1] \text{ of some } \{(a_i, b_i)\}_{i=1,\dots,50} \in D_h^\sigma\}.$$

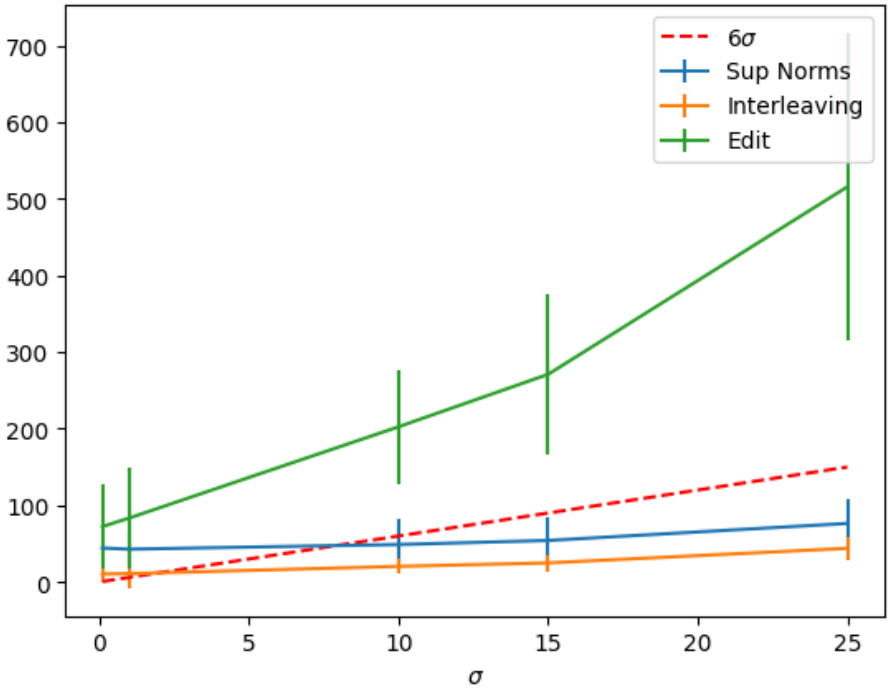
The vertical bars in Figure 7(a) represent the boxplots, i.e., 1.5 times the interquartile range (IQR) below and above the median. In other words, we are checking how far from the true merge trees the observed merge trees tend to fall, as noise increases, where far is either in terms of d_u or d_E.

This is precisely where stability properties become crucial, as they provide robustness to support the analyst's findings. For a reference, we also report the plot of the line $(\sigma, 6\sigma)$: we know that for every $(\mathcal{X}, \mathcal{Y})$ distributed according to model (M_f^σ), $P(\mathcal{Y} \in (f(\mathcal{X}) - 3\sigma, f(\mathcal{X}) + 3\sigma) \mid \mathcal{X} = x) \sim 1$, for every $x \in [0, 1]$. In particular, for $\{(\mathcal{X}_i, \mathcal{Y}_i)\}_{i=1,\dots,50}$ i.i.d., we have $P(\max_{i=1,\dots,50} \mid \mathcal{Y}_i - f(\mathcal{X}_i) \mid < 6\sigma) \sim 0.9$. Analogous computations hold for g and (M_g^σ). In other words, $[0, 6\sigma]$ is a 0.9 confidence interval for the max difference between the smooth functions and the noisy observations, on the grid given by the realizations of $\{\mathcal{X}_i\}_{i=1,\dots,50}$. Note that, despite this, outside such grid, i.e., outside the points where the functions are observed, the error between the smooth functions and the extended noisy observations can be higher. Still, the line $(\sigma, 6\sigma)$ can serve as a rough reference for the pointwise difference between the smooth functions and the noisy observations.

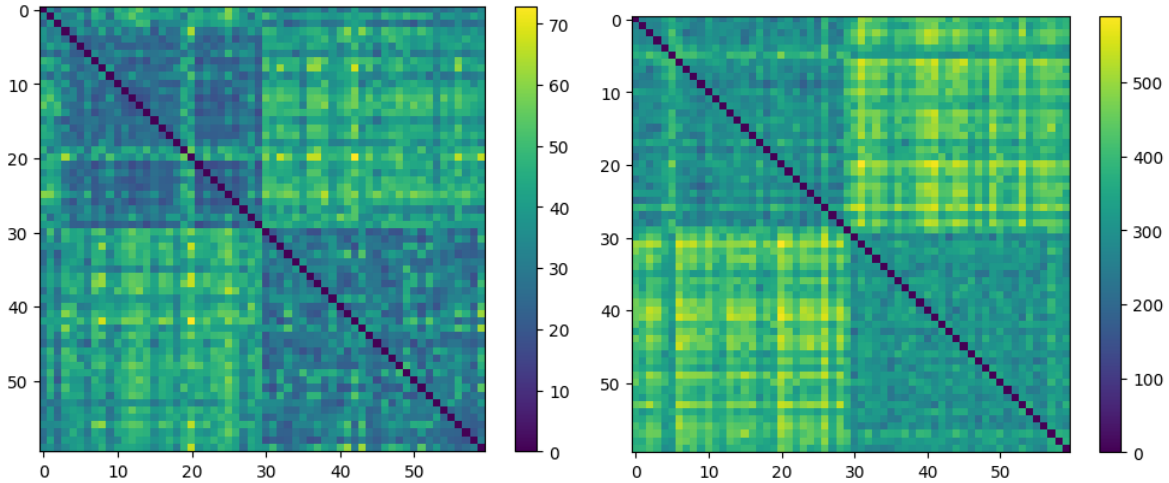
In Figure 7(a), we see that the deviation between the true and the observed merge trees behave as we would expect: the deviation in terms of interleaving distance increases with the magnitude of the noise σ, being controlled by the sup norm between the smooth functions and the extension on $[0, 1]$ of the noisy observations. The deviation in terms of d_E, instead, explodes as σ increases, as d_E depends on a combination of 1) the magnitude of the noise, and 2) the number of oscillations in the functions. In fact, when sampling and interpolating the partial and noisy observations of the functions, we are also potentially increasing the number of oscillations of the data w.r.t. the original f and g: we can have up to $50/2$ local minima in the observed functions, while in the smooth ones we can have at most $20/2$. This is coherent with the behavior we observe in Figure 7(a).

The line $(\sigma, 6\sigma)$ further supports the coherence of our findings: for small values of σ, the noise introduced by the variable δ has an almost negligible effect, and most of the errors are induced by having to extend the partially observed functions from a random grid to $[0, 1]$; meanwhile, for $\sigma > 10$, the variable δ has a much bigger effect, and, thus, we see that the whole boxplot of the (sup of the) pointwise differences between the functions is below the dashed line.

We now examine how the metrics d_E and d_u reflect the structure of D_f^σ and D_g^σ as σ increases. To this end, we consider the matrices of the pairwise distance matrices between merge trees for each D^σ, both in terms of d_E and d_u. For each value of σ, we then construct the corresponding agglomerative clustering dendrograms using average and complete linkage criteria, and cut each dendrogram to obtain two clusters. We then evaluate the results turning the clustering problem into a classification problem, where the true labels are given by the data generating models (i.e., D_f^σ or D_g^σ). In a cross validation (leave-one-out) fashion, each sample is first assigned to a cluster, and then predicted a label, depending on the majority of the true labels of the other elements in the cluster. We then compute the accuracy of these predicted labels. The results are shown in Table 3.



(a) Lines resulting from the interpolation of the medians of the distances between the merge trees of the smooth functions (i.e., f and g) and the merge trees obtained from their noisy observations (contained in, respectively, D_f^σ and D_g^σ), for different values of σ. For each value of σ, we report also the sup of the difference between the smooth functions and the noisy observations (extended on $[0, 1]$). The dashed line represents the line $(\sigma, 6\sigma)$, plotted for reference. For each σ, the vertical bars represent the boxplots of the data. For bigger values of σ, we clearly see that d_E ("Edit") explodes, while d_u ("Interleaving") is controlled by the difference between the functions.



(b) Matrices of pairwise distances between merge trees obtained from the functions in D^σ, for $\sigma = 15$, computed with d_I (left) and d_E (right). Elements in D^σ have been ordered to that D_f^σ comes before D_g^σ.

Figure 7. Additional plots related to the simulated scenario presented in Section 7.2.

Table 3. The leave-one-out accuracies of the simulation described in Section 7.2: as a function of the magnitude of the noise, σ, we have built a classification pipeline based on a hierarchical clustering algorithm, employing average and complete linkages. Even with abundant noise, the performance of d_E is comparable, if not superior, to the one of d_I. The only exception to that being $\sigma = 25$, with complete linkage.

	Average linkage		Complete linkage	
	d_E	d_u	d_E	d_u
$\sigma = 0.1$	0.93	1	1	0.93
$\sigma = 1$	1	1	1	0.93
$\sigma = 10$	0.92	0.97	0.97	0.92
$\sigma = 15$	0.93	0.92	0.92	0.73
$\sigma = 25$	0.52	0.52	0.53	0.61

While no unstable behaviors of d_u have been observed, the uncertainty regarding the estimates of d_I provided by d_u behaves as follows:

- For $\sigma = 0.1$: for the computations regarding the distances between the noisy and the smooth trees the relative discrepancy Δ (see Eq (7.1)) is less than $2.5 \cdot 10^{-15}$ for 95% of the computations. Meanwhile, when comparing different noisy trees, we obtain a Δ of less than $3 \cdot 10^{-8}$ for 95% of the computations;
- For $\sigma = 1$: for the computations regarding the distances between the noisy and the smooth trees, Δ is less than $2.9 \cdot 10^{-9}$ for 95% of the computations. Meanwhile, when comparing different noisy trees, we obtain a Δ of less than $1.3 \cdot 10^{-3}$ for 95% of the computations, whereas 90% of the computations exhibit a relative error of less than $1.8 \cdot 10^{-08}$;
- For $\sigma = 10$: for the distances between the noisy and the smooth trees, Δ is less than $4.1 \cdot 10^{-8}$ for 95% of the computations. Meanwhile, when comparing different noisy trees, we obtain a Δ of less than 0.05 for 90% of the computations and less than 0.008 for 85% of the computations;
- For $\sigma = 15$: for the distances between the noisy and the smooth trees, Δ is less than $6 \cdot 10^{-8}$ for 90% of the computations. Meanwhile, when comparing different noisy trees, we obtain a Δ of less than 0.1 for 90% of the computations and less than 0.02 for 80% of the computations;
- For $\sigma = 25$: for the distances between the noisy and the smooth trees, Δ is less than 0.09 for 90% of the computations. Meanwhile, when comparing different noisy trees, we obtain a Δ of less than 0.12 for 90% of the computations and less than 0.05 for 80% of the computations.

As a consequence of the above, in the considered simulated scenarios, we obtained solid estimates of d_I, strengthening the conclusions that can be drawn from these simulations.

In Table 3, it is evident that, despite d_E being more affected by noise (as shown by Figure 7(a)), it is still comparable or even better at distinguishing between D_f^σ and D_g^σ than d_u, also for high values of σ. The only exception to that being $\sigma = 25$, with complete linkage, where d_I outperforms d_E by 8% accuracy. However, with the same linkage and $\sigma = 15$, d_E heavily outperforms d_I.

This is a clear example of what discussed in Section 6: the sensitivity of d_E, i.e., how much d_E can separate the samples in D_f^σ from the ones in D_g^σ, is very often enough to overcome the corruption

introduced by noise when generating samples. On the other hand, d_u is predictably less corrupted by the presence of noise, but the underlying trade-off between stability and sensitivity makes it less powerful in discriminating between data. Thus, even in these hand-crafted circumstances, we see that considering universal metrics it not a "one-size-fits-all" solution to a data analysis problem.

7.3. Benchmark case studies

Now, we compare d_I and d_E on some benchmark functional datasets, tackling some classification and regression problems.

The datasets we considered are suitable for our purposes for three main reasons: 1) they are freely available and easily accessible, 2) for most datasets, the functions are defined on the same domain, and so we can check the stability of d_u, and 3) they provide a relevant testing ground, despite not always being specifically chosen for their suitability to TDA techniques.

Most of our pipelines, the only exception being the "Aneurisk65" dataset, follows these steps: we compute merge trees, derive pairwise distances with d_u and d_E, and embed the resulting metric spaces into Euclidean spaces using Isomap [2]. Depending on the data analysis task, we then apply either quadratic discriminant analysis (QDA) or linear regression to the embedded vectors. The "Aneurisk65" dataset differs in that each statistical unit consists of a bivariate vector of functions, resulting in two merge trees per unit. To incorporate this additional structure, we merge the information from the respective distance matrices into a single final distance matrix, which is then processed using Isomap and QDA. Further details on this procedure are provided in the following.

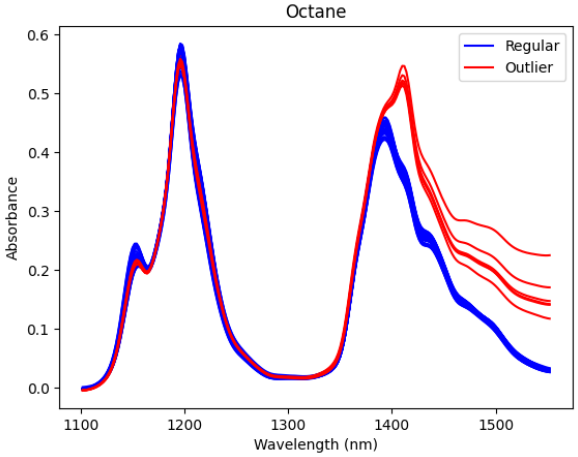
Since all functions in each dataset are evaluated on the same grid, we also implement baseline pipelines by reducing the dimension of the functions' vectors via principal component analysis (PCA) and applying QDA or linear regression directly to the resulting vectors. The baseline results for the "Aneurisk65" dataset, instead, are taken from [44].

The Isomap parameters, embedding dimension and neighborhood size — and the dimension of the PCA, for the baseline pipelines, are selected via leave-one-out cross-validation, optimizing for classification accuracy or mean squared error in regression.

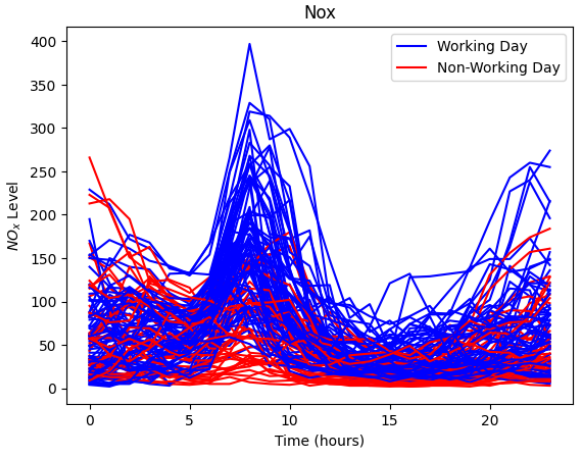
The datasets, and the corresponding data analysis problems, we consider are the following:

- The "Octane" dataset [20] consists of near-infrared spectra of gasoline samples, measured at wavelengths from 1102 nm to 1552 nm in 2 nm increments. It includes six outlier samples containing added ethanol, as required by certain regulations. Our goal is to develop a classifier for outlier detection. As shown in Figure 8(a), there is a clear separation between the outliers and the rest of the data. Consistently, Table 4 demonstrates that all the models we tested successfully accomplished this task;

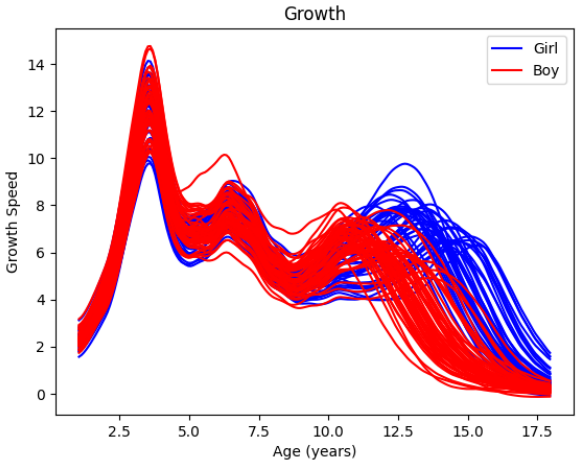
- The "NO_x" dataset [22] contains hourly measurements of daily nitrogen oxides (NO_x) emissions in the Barcelona area. Since NO_x contributes to ozone formation and global warming, identifying days with abnormally high emissions is crucial for implementing control measures, as these emissions primarily stem from motor vehicles and industrial combustion processes. The data is labeled based on whether the emission curve was recorded on a weekday or a weekend, and our goal is to reconstruct this labeling through supervised classification. As shown in Figure 8(b), this classification task is more challenging than the previous one. Nevertheless, Table 4 demonstrates that our pipelines achieve very high cross-validation accuracy, with d_E outperforming d_I;



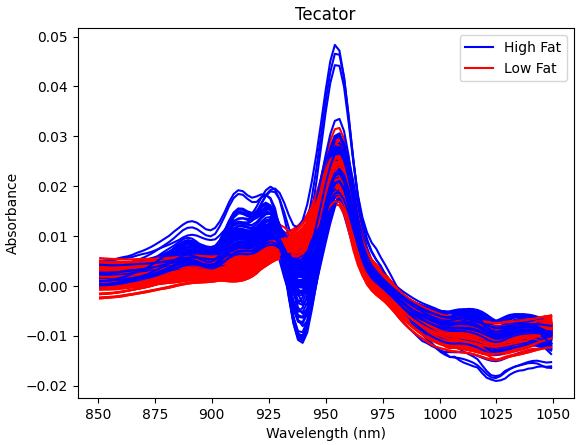
(a) The "Octane" dataset. The red functions represent the outliers with the addition of ethanol. By visual inspection we can see that the two classes are nicely clustered and distinguishable.



(b) The "NO_x" dataset. By visual inspection we see that working and non-working days follow different dynamics, especially regarding the morning peak. This also explains why, as shown by Table 4, merge trees work better than the vanilla model.



(c) The "Growth" dataset. Differences in the growth dynamics between boys and girls are visible, and partially related to the different biological clocks between males and females. Despite that, merge trees achieve good performances in distinguishing between the two classes.



(d) The derivatives of the functions in the "Tecator" dataset. The oscillatory patterns of the curves differ between the high fat and low fat classes, which is coherent with the merge trees models having very high accuracy in the classification task.

Figure 8. Plots related to the case studies presented in Section 7.3.

- The "Growth" dataset [50], also known as "The Berkeley Growth Study", is widely used in functional data analysis. It contains height measurements (in cm) for 54 girls and 39 boys, recorded between ages 1 and 18. This dataset has been extensively studied, including research focused on aligning and re-parameterizing growth curves to account for individual differences in biological timing, thereby enabling clearer identification of growth patterns (see [51] and

references therein). A common approach is to analyze the first derivative of the growth curves to distinguish growth dynamics between boys and girls. Following this, we smooth the measured curves using a kernel smoother with a bandwidth of 3 (and with the same kernel employed in [39]), compute numerical derivatives, and apply classification techniques to discriminate between the two groups. As shown in Table 4, all models achieve good performances on this task, with d_E outperforming d_I;

- The "Tecator" dataset (`https://lib.stat.cmu.edu/datasets/tecator`) consists of publicly available measurements collected using the "Tecator Infratec Food and Feed Analyzer", which operates in the 850–1050 nm wavelength range based on the near-infrared transmission principle. Each sample represents a curve extracted from meat with varying moisture, fat, and protein content, allowing for a range of data analysis applications. Building on the derivatives of these curves, we explore both a classification and a regression problem, as in [24]:

  1) Classifying meat samples into high fat (fat content > 20) and low fat (fat content ≤ 20);
  2) Predicting fat content using linear regression based on the absorbance curves.

  The dataset includes 215 observations, with 77 high-fat and 138 low-fat samples. Table 4 reports the leave-one-out accuracy for classification models and the mean squared error. The results show that the merge tree approach (both with d_I and d_E) slightly outperforms the baseline model in classification (with d_E outperforming d_I), while both d_E and the baseline models do better than d_I in terms of mean squared error (MSE) in the regression problem;

- The final case study addresses a classification problem using the "Aneurisk65" dataset [44], previously analyzed in [39]. This dataset, generated by the AneuRisk Project (`https://statistics.mox.polimi.it/aneurisk`), investigates the relationship between internal carotid artery (ICA) morphology and cerebral aneurysm occurrence. It consists of $3D$ angiographic images from 65 patients, from which ICA centerlines and radius values were extracted [43], see Figure 9. Given its complexity, the "Aneurisk65" dataset serves as a benchmark for functional data analysis methods requiring alignment and re-parametrization (see Electronic Journal of Statistics, 2014, Vol. 8). Following [39, 44], we address the classification problem of predicting aneurysm location, or its absence, based on ICA curvature and radius. Patients are grouped as Upper (U) (aneurysm at/after the ICA's terminal bifurcation), Lower (L) (before bifurcation), or None (N) (no aneurysm). We classify L & N vs. U. The data used for this task consists of a bivariate vector of functions: a function describing the radius of the ICA along the centerline, and a function describing the curvature of the centerline. Following [39], we modify the pipeline followed for the other datasets in order to merge the information given by the two functional covariates:

  1) As in [39], we denoise the functions using kernel smoothers;
  2) For each patient we extract a merge tree from the (smoothed) curvature and one from the (smoothed) radius function;
  3) Separately for radius and curvature, we compute the pairwise distance matrices using d_u (as the results for d_E are already contained in [39]);

4) We merge the information from the curvature and radius functions by producing a new matrix using the formula $d^2_{\text{mixed}} = d^2_{\text{curvature}} + d^2_{\text{radius}}$, where $d_{\text{curvature}}$ and d_{radius} are the pairwise distance matrices of the merge trees derived from curvature and radius functions, respectively;
5) We use d_{mixed} as input for the same pipeline employed in the previous scenarios: Isomap Embedding and QDA.

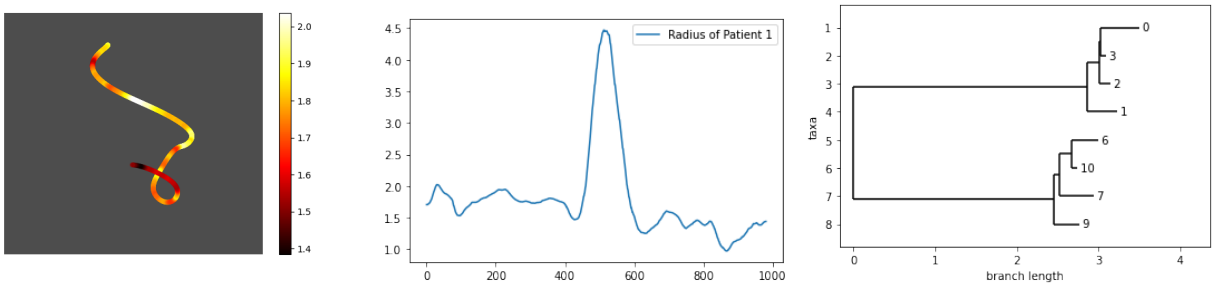


Figure 9. Patient 1 in the AneuRisk65 dataset; on the left, the ICA of the patient is colored according to the radius value, in the middle we see the associated radius function, and on the right we report the associated merge tree. Patient 1 belongs to the Lower group (L). The merge tree has been truncated at a height equal to the maximum of the function.

Table 4 shows that merge trees, with d_E, outperform the functional pipeline developed in [44], while the results with d_u don't improve on the baseline results.

Table 4. Results of the classification and regression problems considered in Section 7.3. Each column deals with a different task, with the first four columns representing the classification problems and the rightmost one being the regression problem. Each row contains the leave-one-out accuracy or leave-one-out MSE of the best performing model obtained with d_E, d_u, or the baseline models. For d_u and d_E, we also report the parameters of the Isomap embedding, $(dim, neigh)$, which is used in the respective models, while for the Baseline models we report the PCA dimension. Parameters were selected to maximize accuracy or to minimize MSE using leave-one-out cross-validation.

Results

	Octane	NOx	Growth	Tecator	Aneurisk65	Tecator (MSE)
d_E	1	0.89	0.88	> 0.99	0.85	6.22
d_u	1	0.86	0.87	0.98	0.8	11.41
Baseline	1	0.94	0.93	0.97	0.82	6.14

Optimal parameters

	Octane	NOx	Growth	Tecator	Aneurisk65	Tecator (MSE)
d_E	$(2, 11)$	$(6, 17)$	$(8, 53)$	$(10, 165)$	$(7, 14)$	$(36, 212)$
d_u	$(2, 8)$	$(4, 5)$	$(10, 8)$	$(10, 181)$	$(8, 13)$	$(56, 210)$
Baseline	2	6	2	4	–	32

In almost all the considered datasets, the computation of d_u gives a very solid estimate of d_I, with Δ being less than 0.008 for 95% of the computed distances in each dataset. The only exception is the "Aneurisk65" dataset, for which the Δs associated to the curvature functions do not exceed 0.002 for 95% of the distances, while for the radius functional covariate, 75% of the distances exhibits values of Δ lower than 0.01, and 70% has values of Δ lower than 10^{-7}. Of course, this does not imply that, in the remaining cases, the distances computed with d_u for the radius functions are necessarily far from d_I, as Δ only provides an upper bound for such error. Nevertheless, to strengthen our results, we also repeated the pipeline used for the "Aneurisk65" dataset replacing d_u first with d_l and then with $(d_u + d_l)/2$, obtaining, respectively, a leave-one-out accuracy of 0.77% and 0.78%. We did observe, for the "Tecator" dataset, 7 instances of d_u exceeding $\|\cdot\|_\infty$. Since the values by which d_u exceeded $\|\cdot\|_\infty$ were always below 10^{-8}, we believe it is due to numerical errors. Thus, we can conclude by saying that d_E systematically outperformed d_I in our case studies.

8. Conclusions

With the present manuscript, we propose a novel edit distance for merge trees, which provides both theoretical soundness and practical applicability. By discussing its stability properties and comparing it with existing metrics, we have highlighted its potential advantages in real-world data scenarios. One key observation is that while universal metrics like the interleaving distance offer strong theoretical guarantees, they may not always provide the best discriminatory power for data analysis tasks. In contrast, our edit distance effectively captures fine-grained structural variations, making it a valuable tool in applications. As for all metrics between unlabeled trees, computational efficiency remains an important consideration: while we prove that we can borrow polynomial time upper bounds and heuristic approaches to optimize the computation of our metric, we leave this investigation, and the resulting considerations about the scalability of our approach, to future works.

To enhance the utility of the metric data analysis scenarios, we also plan to investigate the following topics:

- Developing consistent estimators for merge trees, w.r.t. d_E, extending the work of [39], allowing more general data and models;
- Study the existence and approximation of Frechét means of sets of trees, which are a diffused and powerful statistics used for non-Euclidean data;
- Develop local linearizations of the tree-space to enable the use of more refined statistical and machine learning techniques.

Lastly, we also wish that the stability-sensitivity trade-off of metrics between topological summaries can become a more active topic of discussion in TDA. This would help in finally bridging the gap between practical considerations of data analysts and the abstract focus of methodological research, pushing forward the boundaries of TDA in a rigorous and effective way.

Use of Generative-AI tools declaration

AI language tools have been used to streamline the language of some paragraphs.

Acknowledgments

This work was carried out as part of my PhD Thesis, under the supervision of Professor Piercesare Secchi. I thank Professor Justin Curry for his helpful suggestions and comments. I also acknowledge the support of the Wallenberg AI, Autonomous Systems and Software Program (WASP), and of the SciLifeLab and Wallenberg National Program for Data-Driven Life Science (DDLS), which fund the project: Topological Data Anlysis of Functional Genome to Find Covariation Signatures.

Conflict of interest

The author declares that no conflicts of interest.

Data availability

All the considered benchmark datasets are freely available and easily accessible via the `scikit-fda` python package [41] or at the AneuRisk Project website `https://statistics.mox.polimi.it/aneurisk`. The generative process for the simulated datasets is described in the manuscript.

Code

The code is publicly available at `https://github.com/pego91/edit_finitely`.

References

1. H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, et al., Persistence images: A stable vector representation of persistent homology, *J. Mach. Learn. Res.*, **18** (2017), 1–35.
2. M. Balasubramanian, E. L. Schwartz, The isomap algorithm and topological stability, *Science*, **295** (2002), 7. https://doi.org/10.1126/science.295.5552.7a
3. D. Beers, J. Leygonie, The fiber of persistent homology for trees, arXiv: 2303.16176, (2023). https://doi.org/10.48550/arXiv.2303.16176
4. K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, B. Hamann, Measuring the distance between merge trees, In: *Topological methods in data analysis and visualization III. Mathematics and visualization*, Cham: Springer, 2014, 151–165. https://doi.org/10.1007/978-3-319-04099-8_10
5. S. Biasotti, D. Giorgi, M. Spagnuolo, B. Falcidieno, Reeb graphs for shape analysis and applications, *Theor. Comput. Sci.*, **392** (2008), 5–22. https://doi.org/10.1016/j.tcs.2007.10.018
6. O. Bobrowski, S. Mukherjee, J. E. Taylor, Topological consistency via kernel estimation, *Bernoulli*, **23** (2017),288–328.
7. P. Bubenik, Statistical topological data analysis using persistence landscapes, *J. Mach. Learn. Res.*, **16** (2015), 77–102.
8. R. Cardona, J. Curry, T. Lam, M. Lesnick, The universal ℓ^p-metric on merge trees, **224** (2022), 24:1–24:20. https://doi.org/10.4230/LIPIcs.SoCG.2022.24

9. G. Carlsson, F. Mémoli, Classifying clustering schemes, *Found. Comput. Math.*, **13** (2013), 221–252. https://doi.org/10.1007/s10208-012-9141-9

10. L. Cavinato, M. Pegoraro, A. Ragni, M. Sollini, P. A. Erba, F. Ieva, Author correction: Imaging-based representation and stratification of intra-tumor heterogeneity via tree-edit distance, *Sci. Rep.*, **12** (2022), 19607. https://doi.org/10.1038/s41598-022-23752-2

11. F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, S. Y. Oudot, Proximity of persistence modules and their diagrams, In: *Proceedings of the twenty-fifth annual symposium on Computational geometry*, New York: Association for Computing Machinery, 2009, 237–246. https://doi.org/10.1145/1542362.1542407

12. F. Chazal, B. T. Fasy, F. Lecci, A. Rinaldo, L. Wasserman, Stochastic convergence of persistence landscapes and silhouettes, *J. Comput. Geom.*, **6** (2015), 140–161. https://doi.org/10.20382/jocg.v6i2a8

13. D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of persistence diagrams, *Discrete Comput. Geom.*, **37** (2007), 103–120. https://doi.org/10.1007/s00454-006-1276-5

14. J. Curry, The fiber of the persistence map for functions on the interval, *J. Appl. Comput. Topology*, **2** (2018), 301–321. https://doi.org/10.1007/s41468-019-00024-z

15. J. Curry, J. DeSha, A. Garin, K. Hess, L. Kanari, B. Mallery, From trees to barcodes and back again ii: Combinatorial and probabilistic aspects of a topological inverse problem, *Computational Geometry*, **116** (2024), 102031. https://doi.org/10.1016/j.comgeo.2023.102031

16. J. Curry, H. B. Hang, W. Mio, T. Needham, O. B. Okutan, Decorated merge trees for persistent topology, *J. Appl. Comput. Topology*, **6** (2022), 371–428. https://doi.org/10.1007/s41468-022-00089-3

17. V. de Silva, E. Munch, A. Patel, Categorified reeb graphs, *Discrete Comput. Geom.*, **55** (2016), 854–906. https://doi.org/10.1007/s00454-016-9763-9

18. H. Edelsbrunner, D. Letscher, A. Zomorodian, Topological persistence and simplification, *Discrete Comput. Geom.*, **28** (2002), 511–533. https://doi.org/10.1007/s00454-002-2885-2

19. H. Edelsbrunner, J. Harer, Persistent homology—a survey, In: *Surveys on discrete and computational geometry*, Providence: American Mathematical Society, 2008, 257–282,

20. K. H. Esbensen, D. Guyot, F. Westad, L. P. Houmoller, *Multivariate data analysis: in practice: An introduction to multivariate data analysis and experimental design*, Norway: CAMO Process AS, 2002.

21. E. F. Touli, Y. S. Wang, Fpt-Algorithms for computing gromov-hausdorff and interleaving distances between trees, *J. Comput. Geom.*, **13** (2022), 89–124, https://doi.org/10.20382/jocg.v13i1a4

22. M. Febrero, P. Galeano, W. González-Manteiga, Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels, *Environmetrics*, **19** (2008), 331–345. https://doi.org/10.1002/env.878

23. J. Felsenstein, *Inferring phylogenies*, Sunderland: Sinauer Associates, 2003.

24. F. Ferraty, P. Vieu, *Nonparametric functional data analysis: theory and practice*, New Youk: Springer, 2006, https://doi.org/10.1007/0-387-36620-2 .

25. E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, Y. S. Wang, Intrinsic interleaving distance for merge trees, *La Matematica*, **4** (2025), 40–65, https://doi.org/10.1007/s44007-024-00143-9 .
26. A. Hatcher, *Algebraic topology*, Cambrige: Cambridge University Press, 2001.
27. E. Hong, Y. Kobayashi, A. Yamamoto, Improved methods for computing distances between unordered trees using integer programming, In: *Combinatorial optimization and applications. COCOA 2017*, Cham: Springer, 2017, 45–60. https://doi.org/10.1007/978-3-319-71147-8_4
28. T. Jiang, L. S. Wang, K. Z. Zhang, Alignment of trees—an alternative to tree edit, *Theor. Comput. Sci.*, **143** (1995), 137–148. https://doi.org/10.1016/0304-3975(95)80029-9
29. L. Kanari, A. Garin, K. Hess, From trees to barcodes and back again: theoretical and statistical perspectives, *Algorithms*, **13** (2020), 335. https://doi.org/10.3390/a13120335
30. M. Lesnick, The theory of the interleaving distance on multidimensional persistence modules, *Found. Comput. Math.*, **15** (2015), 613–650. https://doi.org/10.1007/s10208-015-9255-y
31. J. Milnor, *Morse theory*, Princeton: Princeton University Press, 1963. https://doi.org/10.1515/9781400881802
32. F. Murtagh, P. Contreras, Algorithms for hierarchical clustering: An overview, ii, *WIREs Data Min. Knowl.*, **7** (2017), e1219. https://doi.org/10.1002/widm.1219
33. P. Ost, K. Decaestecker, B. Lambert, V. Fonteyne, L. Delrue, N. Lumen, et al., Prognostic factors influencing prostate cancer-specific survival in non-castrate patients with metastatic prostate cancer, *The Prostate*, **74** (2014), 297–305. https://doi.org/10.1002/pros.22750
34. A. Patel, Generalized persistence diagrams, *J. Appl. Comput. Topology*, **1** (2018), 397–419. https://doi.org/10.1007/s41468-018-0012-6
35. M. Pegoraro, A metric for tree-like topological summaries, arXiv:2108.13108v3, (2021).
36. M. Pegoraro, A persistence-driven edit distance for trees with abstract weights, arXiv:2304.12088v4, (2025). https://doi.org/10.48550/arXiv.2304.12088
37. M. Pegoraro, A finitely stable edit distance for functions defined on merge trees, arXiv:2108.13108v9, (2025). https://doi.org/10.48550/arXiv.2108.13108
38. M. Pegoraro, A graph-matching formulation of the interleaving distance between merge trees, *AIMS Mathematics*, **10** (2025), 13025–13081. https://doi.org/10.3934/math.2025586
39. M. Pegoraro, P. Secchi, Functional data representation with merge trees, arXiv:2108.13147v6, (2024). https://doi.org/10.48550/arXiv.2108.13147
40. M. Pont, J. Vidal, J. Delon, J. Tierny, Wasserstein distances, geodesics and barycenters of merge trees, *IEEE T. Vis. Comput. Gr.*, **28** (2021), 291–301. https://doi.org/10.1109/TVCG.2021.3114839
41. C. Ramos-Carreño, J. L. Torrecilla, M. Carbajo-Berrocal, P. Marcos, A. Suárez, scikit-fda: A Python package for functional data analysis, *J. Stat. Softw.*, **109** (2024), 1–37. https://doi.org/10.18637/jss.v109.i02
42. J. O. Ramsay, B. W. Silverman, *Functional data analysis*, 2 Eds., New York: Springer, 2005, https://doi.org/10.1007/b98888
43. L. M. Sangalli, P. Secchi, S. Vantini, Analysis of AneuRisk65 data: K-mean alignment, *Electron. J. Statist.*, **8** (2014), 1891–1904. https://doi.org/10.1214/14-EJS938A

44. L. M. Sangalli, P. Secchi, S. Vantini, A. Veneziani, A case study in exploratory functional data analysis: Geometrical features of the internal carotid artery, *J. Am. Stat. Assoc.*, **104** (2009), 37–48. https://doi.org/10.1198/jasa.2009.0002

45. S. M. Selkow, The tree-to-tree editing problem, *Inform. Process. Lett.*, **6** (1977), 184–186. https://doi.org/10.1016/0020-0190(77)90064-3

46. Y. Shinagawa, T. L. Kunii, Y. L. Kergosien, Surface coding based on Morse theory, *IEEE Comput. Graph.*, **11** (1991), 66–78. https://doi.org/10.1109/38.90568

47. R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, V. Natarajan, Edit distance between merge trees, *IEEE T. Vis. Comput. Gr.*, **26** (2020), 1518–1531. https://doi.org/10.1109/TVCG.2018.2873612

48. R. Sridharamurthy, V. Natarajan, Comparative analysis of merge trees using local tree edit distance, *IEEE T. Vis. Comput. Gr.*, **29** (2021), 1518–1530. https://doi.org/10.1109/TVCG.2021.3122176

49. E. F. Touli, Frechet-like distances between two merge trees, arXiv:2004.10747v1, (2020).

50. R. D. Tuddenham, M. M. Snyder, Physical growth of California boys and girls from birth to eighteen years, *Publ. Child. Dev. Univ. Calif.*, **1** (1954), 183–364.

51. L. M. Sangalli, P. Secchi, S. Vantini, V. Vitelli, Functional clustering and alignment methods with applications, *Communications in Applied and Industrial Mathematics*, **1** (2010), 205–224.

52. F. Wetzels, M. Anders, C. Garth, Taming horizontal instability in merge trees: On the computation of a comprehensive deformation-based edit distance, *2023 Topological Data Analysis and Visualization (TopoInVis)*, Melbourne, Australia, 2023, 82–92. https://doi.org/10.1109/TopoInVis60193.2023.00015

53. F. Wetzels, C. Garth, A deformation-based edit distance for merge trees, In: *2022 Topological Data Analysis and Visualization (TopoInVis)*, 2022, 29–38. https://doi.org/10.1109/TopoInVis57755.2022.00010

54. F. Wetzels, H. Leitte, C. Garth, Branch decomposition-independent edit distances for merge trees, *Comput. Graph. Forum*, **41** (2022), 367–378. https://doi.org/10.1111/cgf.14547

55. F. Wetzels, H. Leitte, C. Garth, Accelerating computation of stable merge tree edit distances using parameterized heuristics, *IEEE T. Vis. Comput. Gr.*, **31** (2025), 3706–3718. https://doi.org/10.1109/TVCG.2025.3567120

56. K. Q. Wu, S. Zhang, A contour tree based visualization for exploring data with uncertainty, *Int. J. Uncertain. Quan.*, **3** (2013), 203–223. https://doi.org/10.1615/Int.J.UncertaintyQuantification.2012003956

57. K. Z. Zhang, A constrained edit distance between unordered labeled trees, *Algorithmica*, **15** (1996), 205–222. https://doi.org/10.1007/BF01975866

A. Comparison with other distances for merge trees

In this section, we illustrate some behaviors of the metric d_E, extensively comparing it with other definitions of edit distances between merge trees which appeared in literature. In this way, we can better portrait how the variability between merge trees is captured by the proposed edit distance. We

consider separately the different metrics we compare d_E against; plus, at the end, we consider some topics which are common to all/most of the different comparisons. We avoid a comparison with [48] as the goal of such metric is comparing subtrees of merge trees, and so it behaves very differently from all other distances.

A.1. Editing a merge tree

Before commencing the comparisons, we devote this subsection to exploring, with some easy examples, the definitions and results given in Sections 4.1 and 4.2. This should help the reader also in following the upcoming discussion.

First note that, by construction, $\mathrm{Tr}_K((T, h_T))$, for K big enough, is a representation of the merge tree (T, h_T) coherent with the metric d_E and, thus, can be used also to visually compare two merge trees. We can then consider a merge tree (T, h_T) and edit (T, w_T) according to the rules in Section 3.1: via shrinking, deletions, and ghosting of vertices and the inverse operations. We look at the results of the edits in light of the merge tree $(\mathrm{Tr}_K)^{-1}((T, w_T))$.

Let $(T, h_T) = \mathcal{M}(\pi_0(X_\bullet))$ and consider an edge $e = (v, v') \in E_T$, with $t = h_T(v)$ and $t' = h_T(v')$. Since the height of the root is fixed and equal to K, shrinking e reducing its weight by some value $\varepsilon > 0$ (with $\varepsilon < w_T(e)$) amounts to "moving upward" $sub_T(v')$ by ε, that is, changing $h_T(v'') \mapsto h_T(v'') + \varepsilon$, as in Figure 1(c)→Figure 1(d) or in Figure 12(a) and Figure 12(c) (left). Having $\varepsilon > w_T(e)$ means deleting e. Similarly, increasing $w_T(e)$ by ε amounts to lowering $sub_T(v')$ by ε, as in Figure 1(f), with the insertion of the red internal vertex. Consider now the splitting of the edge e into $e_1 = (v, v'')$ and $e_2 = (v'', v')$, with T' being the novel tree structure and $w_{T'}(e_1) = \varepsilon_1$ and $w_{T'}(e_2) = \varepsilon_2$ - as for any of the yellow vertices in Figure 1(e). We must have $\varepsilon_i > 0$ and $w_T(e) = \varepsilon_1 + \varepsilon_2$. This clearly induces a well defined height function $h_{T'}(v'')$. The merge tree $(T', w_{T'})$ differs from (T, w_T) by the degree two vertex v'', while the height function on $V_{T'} - \{v''\}$ is still the same. Also, accordingly, the associated AMTs are the same $\pi_0(X_\bullet) = \mathcal{F}((T, h_T)) \cong \mathcal{F}((T', h_{T'}))$ (with $\mathcal{F}$ being as in Proposition 4). Thus, we have changed the graph structure of T without changing the topological information it represents.

Remark 5. In light of this paragraph, it would be natural to try to define a family of metrics indexed by integers $p \geq 1$ by saying that the costs of an edit path the p-th root of sum of the costs of the edit operations to the p-th power. However, now we can easily see that for any $p > 1$ this has no hope of being a meaningful pseudo metric for weighted trees. In fact, consider the case of a weighted tree made by two vertices and one edge with weight 1. The cost of shrinking the p-metric would be $\| 1 \|_p = 1$. At the same time, one can split it in half with 0 cost, and the cost of shrinking this other tree would be $\| (1/2, 1/2) \|_p < 1$. Splitting the segment again and again will make its shrinking cost go to 0. In other words, all weighted trees, if considered up to degree 2 vertices, would be at distance zero from the tree with no branches.

A.2. A deformation-based edit distance for merge trees [53]

As mentioned in the main body of the manuscript, the approach of [53] shares some similarities with ours. We have also highlighted that the metric between weighted trees which we employ to define d_E appeared in even earlier preprints [35].

The metric in [53] is an edit distance built with deletions and insertions, which also requires that any time we obtain a degree 2 vertex via some deletions, that vertex is removed from the tree with

a ghosting-like procedure. Despite these similarities, there are also important differences between the metrics: there are subtle differences between the edit operations, and there are more profound differences in the definition of merge trees. More in details:

1) [53] doesn't explicitly model mathematically the ghosting of a degree 2 vertex or the splitting of an edge at any point during the edit sequence. This, in particular, implies that an edit sequence according to [53] is also an edit sequence according to our definition, but the vice versa in general does not hold. However, Lemma 1 also implies that for every edit path with our edit operations, there is an edit path according to [53] with the same cost;

2) The metric defined in [53] is applied on weighted trees which are obtained by truncating the edge at infinity of (our) merge trees at the maximum of the associated function and then attaching weights to edges by taking differences between the heights of the extremes. These objects are therein called abstract merge trees; see Definition 1 in [53]. Our definition of merge trees, with the edge at infinity, is more in line with the TDA perspective of indexing sublevel sets filtrations on $\mathbb{R}$ (which, for instance, causes points in persistence diagrams to have death time equal to $+\infty$), while the definition in [53] is more in line with the computer vision community.

We bridge between the two definitions of merge trees with the following proposition, in which we put ourselves in the set of hypotheses considered in [53] and consider merge trees according to the definition therein contained, which will be called G_1 and G_2, and our merge trees, which will be called T_1 and T_2.

Proposition 7. Consider $f_1, f_2 : X \to \mathbb{R}$ to be Morse functions defined on X, a compact manifold with boundary. Let $(T_1, h_{T_1}), (T_2, h_{T_2})$ be the merge trees obtained according to our definition (see Section 5), and let $(G_1, w_{G_1}), (G_2, w_{G_2})$ be the weighted trees obtained truncating T_i at $\max f_i$, for $i = 1, 2$, and considering the difference in heights between the extremes of each edge. The following holds:

$$| d_E(T_1, T_2) - d_E(G_1, G_2) | = | \max f_1 - \max f_2 | .$$

Moreover, we have:

$$d_E(G_1, G_2) \leq 2(\text{size}(T_1) + \text{size}(T_2)) \, \| f_1 - f_2 \|_\infty . \tag{A.1}$$

Proof. We have already seen that for any couple of weighted trees we can always choose an edit path suitable for the edit operations in [53]. Moreover, by [53], we know that, in the considered setting, the roots of G_1 and the root of G_2 have only one child.

Without loss of generality, suppose $\max f_1 \leq \max f_2$. Let G_1' be the weighted tree obtained truncating T_1 at height $\max f_2$. Note that $d_E(T_1, T_2) = d_E(G_1', G_2)$.

Set $K = \max f_2 - \max f_1$. We know that the tree structures of G_1 and G_1' are isomorphic. Moreover, we have that the weight functions of G_1 and G_1' coincide on all edges apart from the edge that goes into the root. Te difference between the weights of such edge in G_1 and G_1' is exactly K.

Thus, we can take any edit path from G_1 to G_2 and, via the tree structure isomorphism of G_1 and G_1', it will induce an edit path from G_1' to G_2. Vice versa, using the inverse of such isomorphism, we can take any edit path from G_1' to G_2 and induce an edit path from G_1 to G_2.

Since all the weights of the edges stay the same, a part from the one that goes into the root, the cost of any edit path changes exactly either by $+K$ or $-K$, when replacing G_1 with G_1', and when replacing G_1' with G_1. Since $d_E(G_1', G_2) = d_E(T_1, T_2)$, we have the thesis.

For the second part of the proof, we consider a mapping M between G_1' and G_2 obtained as in the proof of Theorem 2 (see [39]). Then, we induce, via the isomorphism of G_1' and G_1, a mapping between G_1 and G_2. Via the sections "Results on Internal Vertices" and "Internal Vertices" in the proof of Theorem 2, we know that every couple $(v, w) \in M$ satisfies the following. Let v' be the parent of v after the deletions to be done on G_1 and let w' be the parent of w after the deletions to be done on G_2. We always have:

$$| h_{T_1}(v) - h_{T_2}(w) | \leq \| f_1 - f_2 \|_\infty;$$

and, unless v' and w' are the roots of the trees, we have:

$$| h_{T_1}(v') - h_{T_2}(w') | \leq \| f_1 - f_2 \|_\infty .$$

In particular, this implies that if v' and w' are the roots of the trees, the cost of shrinking (v, v') onto (w, w'), when editing G_1' to obtain G_2, is at most $\| f_1 - f_2 \|_\infty$, as v' and w' are both set at a height equal to $\max f_2$ via the truncation process.

In the induced edit path from G_1 to G_2, this shrinking will cost:

$$| (\max f_1 - h_{T_1}(v)) - (\max f_2 - h_{T_2}(w)) | \leq 2 \| f_1 - f_2 \|_\infty .$$

Since the cost of the other edits is unchanged, we obtain the thesis. □

These facts have the following consequences, which are also exemplified in Figures 10 and 11:

1) The metric defined in [53] is completely insensitive to vertical translation, that is, it is not able to distinguish between the merge tree of f and the merge tree of $f + h$, for any $h \in \mathbb{R}$. In particular, Theorem 3 does not hold for this metric;

2) We believe that explicitly modeling degree 2 vertices removal and addition is fundamental to study the geometric and topological properties of the space of merge trees, and to design data analysis tools, like means and embeddings in linear spaces. See Figure 11.

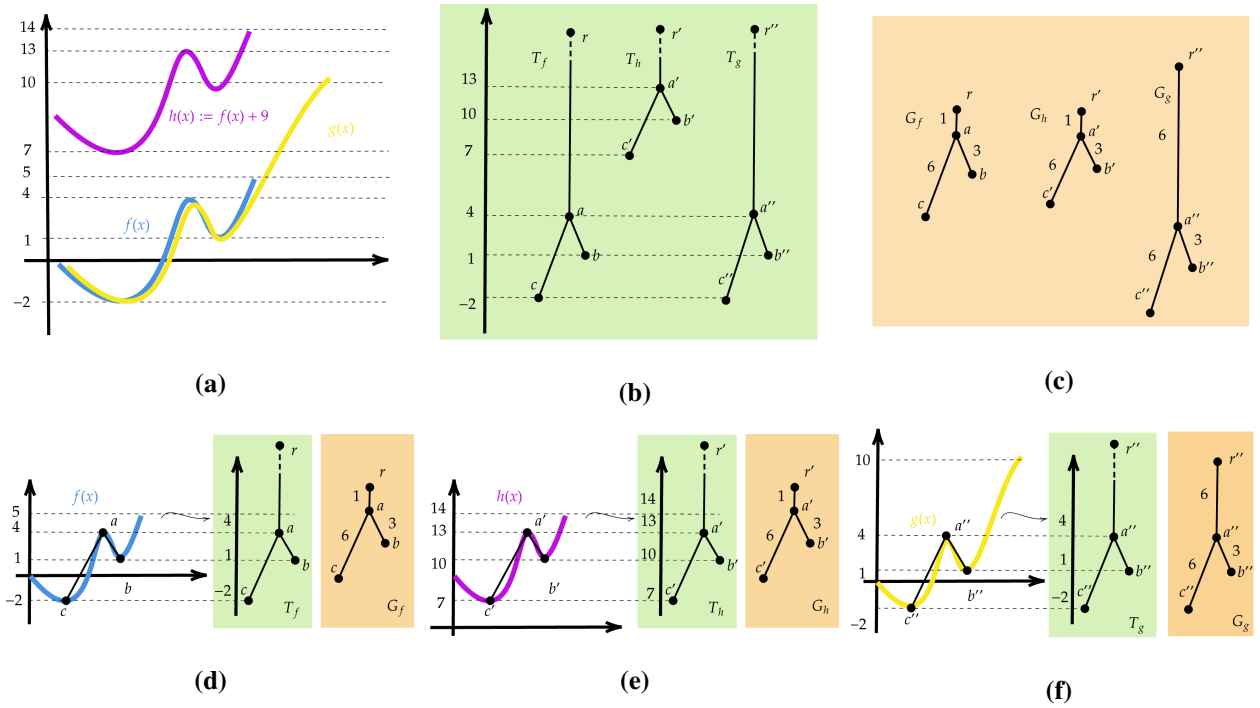


Figure 10. In this figure, we showcase the differences between our metric and the one defined in [53]. In Figure 10(a), we see tree functions f, g, h defined on a compact interval. In Figure 10(b), we see their associated merge trees according to our definition, while in Figure 10(c), we see the weighted trees used by [53] to compute the metric therein defined. Figures 10(d)–(f) compare the two approaches for extracting merge trees from each function separately. Figure 10(b) highlights that d_E cannot distinguish between T_g and T_g as, in fact, those two functions have the shame "shape" topologically (their abstract merge trees as defined in Section 5 are isomorphic, i.e., the interleaving distance between them is zero; this also implies that their persistence diagrams are the same). Meanwhile, [53] separates those functions as their maxima are very different. On the other hand, the approach in [53] cannot distinguish between G_f and G_h, as the translation factor of 9 - $h(x)$ = $f(x)$ + 9 - is completely lost when the weights associated to the edges are obtained. Instead, d_E and d_I are able to separate between such merge trees. In particular, Theorem 3 does not hold for the metric in [53].

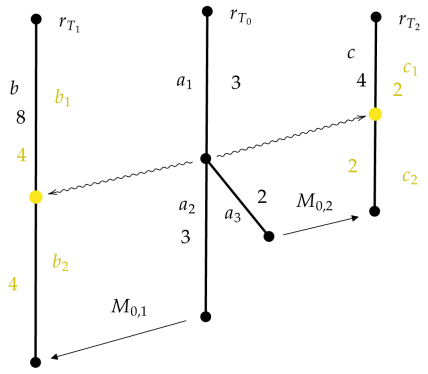


Figure 11. In this figure, we show the potential of explicitly modeling the addition and removal of degree 2 vertices. Consider the weighted trees T_0, T_1, and T_2 represented in the figure. Let $M_{0,i}$, with $i = 1, 2$ be two mappings $M_{0,i}$ defined, respectively, by the matchings given by the straight arrows (via the usual edge-vertex identification). $M_{0,1}$ is then completed by the deletion of a_3 and the ghosting of its parent. Similarly, $M_{0,2}$ is then completed by the deletion of a_2 and the ghosting of its parent. Suppose now that we split T_1 and T_2 by inserting the yellow vertices, resulting in the edges with yellow labels and the weighted trees T_1' and T_2'. At this point, we would be able to match b_1 with a_1 and b_2 with a_2 without raising the cost of $M_{0,1}$, and, similarly, to match c_1 with a_1 and c_2 with a_3 without raising the cost of $M_{0,2}$. Thus, we "refined" our weighted trees and obtained mappings as expensive as the starting ones, but which involve only deletions and shrinkings, which are the operations giving the classical edit distance. Such distance is much more regular compared to d_E (and to the one defined in [53]). For instance, with the identifications $a_1 \sim b_1 \sim c_1$, $a_2 \sim b_2$, and $a_3 \sim c_2$, we can represent T_0 with the vector $v_0 = (3, 3, 2)$, T_1' with the vector $v_1 = (4, 4, 0)$, and T_2' with the vector $v_0 = (2, 0, 2)$, and the mappings we described between those trees are straight lines in $\mathbb{R}^3$ with $\|\cdot\|_1$. This cannot be done with the edits in [53].

To conclude this section we point out again that the relationship between the deformation based distance and d_E proven in Proposition 7 enables the use of computational methods developed by the same authors [55] to speed up the computations for d_E .

A.3. Edit distance between merge trees [47] and Wasserstein distance [40]

The edit distance in [47] is similar to classical edit distances, with the edit operations being restricted to insertion and deletion of vertices and with a *relabeling* operation which is equivalent to our shrinking operation. There is, however, the caveat that vertices are in fact understood as persistence pairs (m, s), with m being the leaf representing the local minimum giving birth to the component, and s the internal vertex representing the saddle point where the components merge with an earlier born component, and, thus, dies according to the elder rule. There is, therefore, a one-to-one correspondence between persistence pairs in the merge tree and in the associated persistence diagram.

Editing a vertex m implies editing also its saddle point s: deleting (m, s) means deleting all vertices m' such that their persistence pair (m', s') satisfies $s' < s$. If then s becomes of degree 2, it is removed. In particular, the authors highlight the impossibility to make any deletion, with the word "deletion" to

be understood according to our notation, on internal vertices, without deleting a portion of the subtree of the vertex. So, they cannot delete and then insert edges to swap parent-children relationships. To mitigate the effects of such an issue, they remove in a bottom-up fashion, as a preprocessing step, all saddles $s \in V_T$ such that $w_T((s, s')) < \varepsilon$ for a certain positive threshold ε. All persistence pairs of the form (m, s) are turned into (m, s'). Such issue is further discussed in Appendix A.6.

Two merge trees are then matched via mappings representing these edit operations. To speed up the computations, the set of mappings between the trees is constrained so that disjoint subtrees are matched with disjoint subtrees. The cost of the edit operation on an edit pair (m, s) is equal to the edit operation being applied on the corresponding points in the associated PDs with the 1-Wasserstein distance: deleting a persistence pair has the cost of matching the corresponding point to the diagonal, and relabeling a persistence pair with another in the second tree has the cost of matching the two points across the two diagrams; see [47, Section 4.3.1].

A closely related metric between merge trees is the Wasserstein distance defined in [40], which extends the metric by [47] also producing further analyses on the resulting metric space of merge trees by addressing the problem of barycenters and geodesics. In this work, the authors rely on a particular branch decomposition of a merge tree, as defined in [54], from which they induce the branch decomposition tree (BDT [40], Section 2.3) used to encode the hierarchical relationships between persistence pairs. A branch decomposition is roughly a partition of the graph T of a merge tree (T, h_T) via ordered sequences of adjacent vertices [54]. The chosen branch decomposition is the one induced by the elder rule and persistence pairs. Edit operations on such BDTs entail improved matchings and deletions between persistence pairs. To obtain the (squared) 2-Wasserstein distance, the vertices of two BDTs are matched and the resulting costs are squared and then added. However, the authors then explain that, with this first definition, geodesics cannot by found via linear interpolation of persistence pairs for the hierarchical structure of the merge tree can be broken. To mitigate that, they employ a normalization which shrinks all the branches on $[0, 1]$, irrespective of their original persistence [40, Section 4.2], leading to simple geodesics obtained with linear interpolation between persistence pairs. To mitigate for this invasive procedure, they introduce yet another preprocessing step, artificially modifying small persistence features to reduce the normalization effects.

Some of the limitations of this approach are listed in [40, Section 7.3]. Also, [54, Section 3.3] adds on that with further details and examples. Namely, the restricted space of possible matchings between trees, which is key to obtain the computational performances of the metrics, forces unstable behaviors: issues with saddle swaps (see [40, Section 4.4 and Figure 10]) and instability of persistence pairs, so that elder ruled-based matchings may force very high persistence features to be matched with other very high persistence features, even in situations where this implies making many unreasonable changes in the tree structures as in Figure 13(f), (see also [54, Figure 1, Figure 2(b), and Section 3.3], and [40, Section 7.3]). Moreover, [40] does not address the interactions between the normalization and the two preprocessing steps.

A.4. Branch decomposition-independent edit distances for merge trees [54]

The work [54] starts from the shortcomings of [40, 47] trying to overcome them. Namely, it defines branch decompositions (the persistence pairs of [47], induced one such branch decomposition), and in order to avoid issues related with the instability of the persistence pairs, [54] introduces also the possibility to optimize the chosen branch decomposition. The only big issue with such approach is

that it does not define a metric on the space of merge trees, for the triangle inequality is not satisfied; see [54, Theorem 2 and Figure 3].

A.5. Heights vs weights

In this section, we try to better understand the different behavior of d_E when compared to the persistence-based metrics presented in the previous sections. The basic idea is that w_T encodes the reciprocal positions of the merging points, instead of having the persistence pairs being free to move independently, at least locally, inside a constrained space.

Using $d_B(PD(T), PD(G)) \leq d_I(T, G)$ [4] and Theorem 3, one obtains:

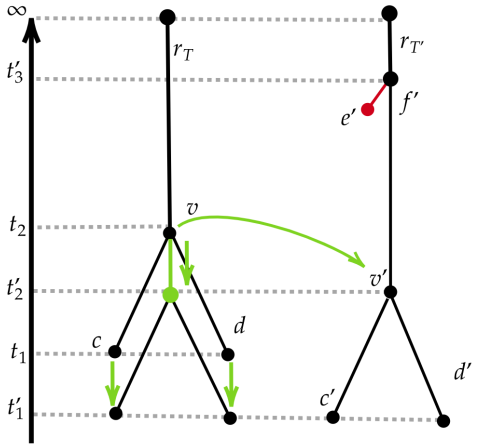
$$W_1(PD(T), PD(G)) \leq (\dim(T) + \dim(G))d_E(T, G),$$

with $PD(T)$ being the persistence diagram associated to the merge tree T. As we will see shortly, this bound cannot be improved. It is thus evident that working with weights $w_T(e)$ instead of persistence pairs, as PDs do, creates differences in how the variability between trees is captured by these two metrics, despite the similarity in their stability properties.

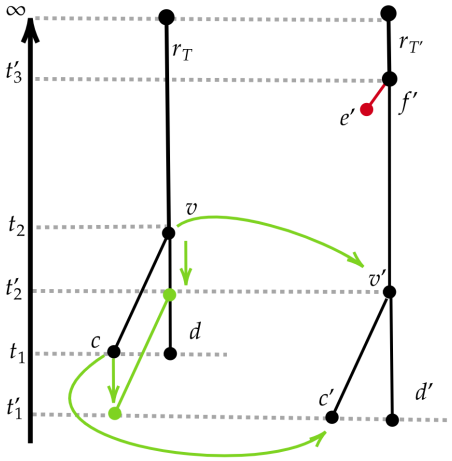
Take, for instance, the merge trees in Figure 3(b): the persistence pairs are (c, v) and (d, r_T). The pairs of the rightmost tree instead are $(c', v'), (e', f')$, and $(d', r_{T'})$. Deleting (e', f'), according to [40, 47, 54], amounts to deleting e' according to our framework. Instead, shrinking (v, r_T) on $(v', r_{T'})$, after the deletion of (e', f'), for us means lengthening (v, r_T) by $t_2 - t_2'$, and so lowering v and all the vertices below v by the same amount, as in Figure 12(a). On the other hand, matching the persistence pair (c, v) with (c', v') for [40, 47] is equivalent to shifting the edge (c, v) downward toward (c', v'), leaving all other vertices fixed, as in Figure 12(b). We can compare the results of such edits in Figure 12(c).

Thus, given a persistence pair (m, s) inside a merge tree, corresponding to the point (b, d) inside the persistence diagram, moving s upward by some $\varepsilon > 0$ such that $h_T(s) + \varepsilon < h_T(s')$ costs ε in terms of the other edit distances, of the interleaving distance and in terms of the 1-Wasserstein distance between PDs and leaves all other points with the same height. In terms of the edit distance d_E, instead, moving s upward by $\varepsilon > 0$ shortens the edge above s, keeping all other edges of a fixed length and, thus, moves the vertices upward by ε.

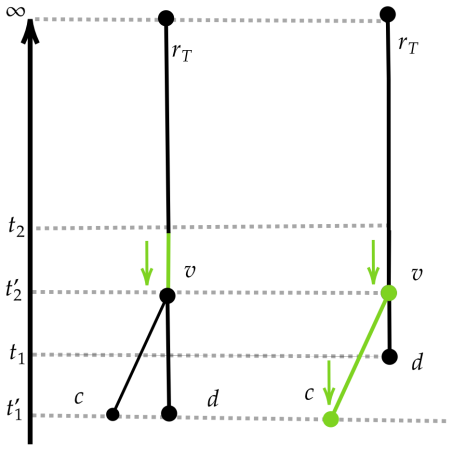
We can interpret this fact as the metrics in [4, 40, 47] better capturing similarities between trees via the location of their vertices in terms of heights; while d_E better captures the "shape" of the tree, i.e., relative positions of its vertices, being less sensible on the height variability since you can move more vertices at one. This is exactly what happens in the example $f : I \to \mathbb{R}$ and $g(x) = f(x) + h$, $h \in \mathbb{R}$, as in Figure 12, we have $W_1(PD(T), PD(G)) = d_E(T, T')(\dim(T) + \dim(G))$.



(a) Two merge trees T (left) and T' (right). We first delete (e', f') and then edit T with the shrinking defined by (v, v').



(b) The same merge trees as in Figure 12(a), with some edges drawn at different angles for visualization purposes. We edit T matching the persistence pair (c, v) with (c', v') according to [40, 47].



(c) The results of the edits applied on T in Figure 12(a) (left) and in Figure 12(a) (right).

Figure 12. A comparison between the weight based edits on which is based d_E and the persistence based edits in [40, 47].

A.6. Stability vs preprocessing

As already mentioned, the metrics in [40, 47] lack stability properties, which means that there are certain situations in which such metrics may perceive as very far trees which are in fact very close in terms of interleaving distance. In particular, they are unable to model "saddle swaps", that is, with our terminology, deleting and inserting internal vertices to change parent-children relationships. As noted also by the authors themselves, this issue needs to be addressed in some way. As already mentioned, to do so, the authors resort to a computational solution implemented as a preprocessing step: they fix a threshold $\varepsilon > 0$ for any couple of persistence pairs (m, s) and (m', s') with $s' > s$ and $s' - s < \varepsilon$, and

they change (m, s) into (m, s'), merging the two saddle points in a bottom-up recursive fashion.

In this section, we produce a brief investigation of such procedure, which is absent in the aforementioned works. We represent the possible outcomes of this preprocessing in Figure 13. All merge trees in Figure 13 are drawn with colors representing persistence pairs, and similar colors yields persistence pairs with the same persistence throughout the whole figure. They ideally should be matched by the metrics to achieve optimal distances, as the differences between edges of different colors could be arbitrary big.

In Figure 13(a) (left), we suppose that the edges marked with a red cross represent distances between saddles smaller or equal than $\varepsilon > 0$. Thus, we recursively merge each saddle point with the higher one, starting from the bottom and going upward. In this way, we obtain the merge tree T' in Figure 13(a) (right) which is then used as input for the metric.

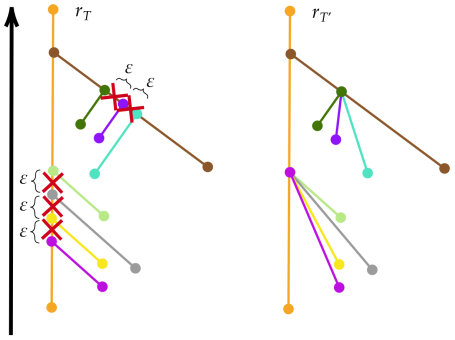
In Figure 13(b), we see two merge trees T and G such that their interleaving distance is $\varepsilon + \varepsilon'$, for we need to move points of T upward by $\varepsilon + \varepsilon'$ to match persistence pairs of the same color in G. Their edit distance d_E would be $3(\varepsilon + \varepsilon')$, for we need to delete and then insert back three small edges in T to swap parent-children relationships in a suitable way. Note that one can replicate analogous situations to make the interleaving distance between the two merge trees arbitrarily big: informally speaking, it is enough to add other persistence pairs as needed and force matchings between pairs in very "different positions".

In Figure 13(c), we see a possible output of the preprocessing routine. If the preprocessing threshold is bigger than ε and ε', then the pre-processed trees T' and G' are equal. This, in some sense, is the desired output of the authors of [40, 47] as, now, their metric can match persistence pairs according to their colors. They suggest that such loss of variability, $d(T', G') = 0$ even if $T \nsim_2 G$, could be mitigated by adding to the distance between the preprocessed trees the fixed threshold as many times as there are saddles merging in the procedure. Note that, even if this artificial addition approximately matches the variability removed from the pairs attached to the orange vertical pair, it introduces artificial variability at the level of the branches attached to the brown edge, as they do not need any modification to be matched correctly.

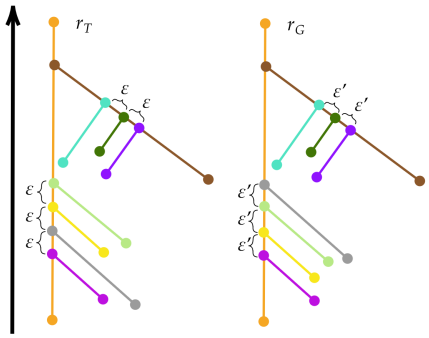
In Figure 13(d), instead, we suppose that the preprocessing threshold is smaller than ε and ε'. The metrics then are forced to match persistence pairs with different colors, causing an excess of variability which is pictured with red edges in the figure. We point out that, depending on the weights of the persistence pairs involved, these edges could be of arbitrary length.

In Figure 13(e), we represent the last possible output of the preprocessing procedure, which is a situation in which the threshold we fix is greater than ε' but smaller then ε. This is possibly the worst scenario: pairs are matched in an optimal way, but we have introduced a lot of artificial variability, represented with red edges, in G. Again, because of additive phenomena caused by recursive saddle merging, these edges can be arbitrarily long.

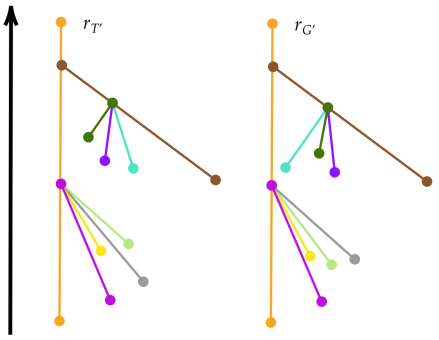
We stress that the preprocessing does not fix the issues presented in Figure 13(f), [54, Figure 1, Figure 2(b), and Section 3.3], and [40, Section 7.3].



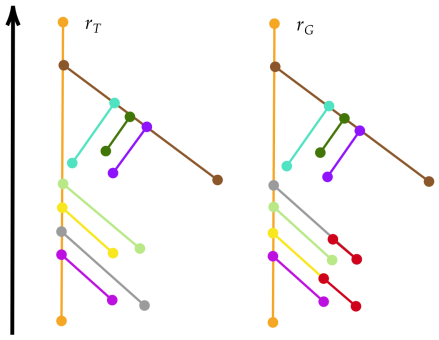
(a) A merge tree T (left) with its preprocessed version (right) according to the procedure in [40, 47].



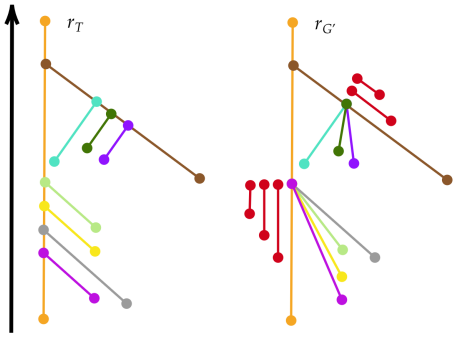
(b) Two merge trees T and G which are going to be preprocessed as in Appendix A.6.



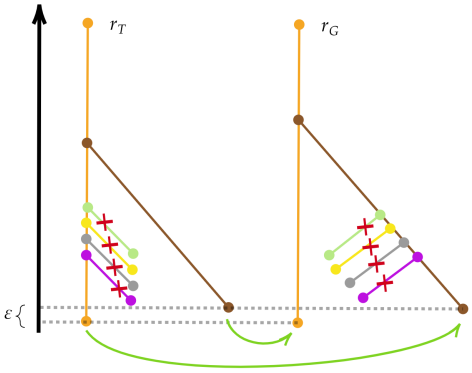
(c) A possible output of the preprocessing: the preprocessing threshold is bigger than ε and ε'.



(d) Another possible output of the preprocessing: the preprocessing threshold is smaller than ε and ε'.



(e) The last possible output of the preprocessing: the preprocessing threshold is smaller than ε but bigger than ε'.



(f) Metrics based on persistence pairs may delete the pairs with the red cross and points according to the remaining colors instead of following the green arrows.

Figure 13. Plots related to Appendix A.6.

To conclude, we point out the following fact. We deem fundamental to have metrics being

able to measure different kinds of variability in a dataset, especially if the information captured is interpretable. However, any attempt to adapt our edit distance d_E to work with heights instead of weights would face the problem of coherently defining deletion edits. While it seems reasonable to have $w_T((v, v'))$ as a cost to delete and edge, for instance, this cost amounts to the persistence of the feature if (v, v') is a persistence pair, any change in the height of v' changes the weight of (v, v') and, thus, the cost of its deletion, invalidating all the results about mappings in [37] and the algorithm therein. Similarly, adapting [40, 47] to handle saddle swaps would mean at least removing from their mappings the property that would be (M3) in our notation, creating many issues from the theoretical and computational point of view. In fact, internal edges are not really available in the representation used by such metrics, where all points are instead understood as part of a persistence pair.

B. Proofs

Proof of Proposition 2:

Consider a minimizing mapping $M \in M_2(G, G')$.

Apply the deletions and ghostings described by M both on G and on G' obtaining, respectively, the merge trees G_M and G'_M. Note that, since $M \in M_2(G, G')$, if v' is coupled with another edge by M, then v must be deleted, otherwise, v' would be of degree 2 after the deletions, and, thus, it should be ghosted. Similarly, if v' is deleted then v must be deleted as well as: otherwise, v' would be of degree 2 after all the other deletions, and, thus, it cannot be deleted by M. The same, of course, applies to w' and w.

Let $w_{T'}(v') = n'$, $w_{T'}(v) = n$, $w_{G'}(w') = m'$, and $w_{G'}(w) = m$. By construction, $n' = m'$ and, thus, the cost of the couple (v', w'), if it can be added a mapping, is zero. Since $M \in M_2(G, G')$, we know $(v', w') \notin M$. Our goal is to build a mapping containing (v', w') with cost equal to the cost of M.

- First, suppose that v is not deleted and w is not deleted.

  Then, there exist $a = v_1 \leq \ldots \leq v_k \leq v'$ and $b = w_1 \leq \ldots \leq w_r \leq w'$ vertices in T and T', respectively, such that:

  - $(a, b) \in M$;
  - v_i and w_i are ghosted for all i (as $M \in M_2(G, G')$ they cannot be deleted).

  Let $n_i = w_T(v_i)$ and $m_i = w_{T'}(w_i)$ for all i.

  We have seen that if $a = v'$, then v must be deleted, which is absurd. Similarly, we cannot have $b = w'$. Thus, $a = v_1 \leq \ldots \leq v_k = v < v'$ and $b = w_1 \leq \ldots \leq w_r = w < w'$. We have:

  $$| n' + \sum_{i=1}^{k} n_i - (m' + \sum_{i=1}^{r} m_i) | = | n' - m' | + | \sum_{i=1}^{k} n_i - \sum_{i=1}^{r} m_i | .$$

  This implies that we can add the couple (v', w') to M without increasing the cost of M.

- Suppose now v (and so v') is deleted, but w is not. For what we have said before, it means that w' is ghosted and w is either coupled or deleted. This implies that the root of G'_M is of order 1. However, then, also the root of G_M must be of order 1. In other words, there exist $a = v_1 \leq \ldots \leq v_k < v$ and $b = w_1 \leq \ldots \leq w_r \leq w$ vertices in T and T', respectively, such that:

– $(a, b) \in M$;

– v_i and w_i are ghosted for all i (as $M \in M_2(G, G')$ they cannot be deleted).

However, this also implies that, after all the other deletions v is of degree 2, and the same for v', but this cannot happen.

- Suppose, lastly, v and w are all deleted, which implies that also v' and w' are deleted. In this case, we can add (v, w) and (v', w') to M decreasing its cost.

Thus, we can always add (v', w') to M, and since the cost of such couple is zero, we have $d_E(T, T') = d_E(G, G') = d_E(sub_G(v'), sub_{G'}(w'))$.

■

Proof of Theorem 1:

Consider (T, h_T) and $(T', h_{T'})$ such that $\max h_T < K'$ and $\max h_{T'} < K'$. Consider $(G, w_G) = \mathrm{Tr}_{K'}((T, h_T))$ and $(G', w_{G'}) = \mathrm{Tr}_{K'}((T', h_{T'}))$. For any K such that $\max h_T < K < K'$ and $\max h_{T'} < K < K'$, there is a splitting of $(v, r_G) \in E_G$ with a vertex w and a splitting of $(v', r_{G'})$ with a vertex w' such that the obtained weighted trees (R, w_R) and $(R', w_{R'})$ satisfy: $w_R((w, r_G)) = w_{R'}((w', r_{G'}))$. Thus, by Proposition 2, we obtain $d_E(G, G') = d_E(R, R') = d_E(sub_R(w), sub_{R'}(w'))$. Moreover, $sub_R(w) = \mathrm{Tr}_K((T, h_T))$ and $sub_{R'}(w') = \mathrm{Tr}_K((T', h_{T'}))$.

Being K, K' arbitrary, we have that $d_K((T, h_T), (T', h_{T'}))$ does not depend on K, for $K > \max\{\max h_T, \max h_{T'}\}$. We need to prove the case $K = \max\{\max h_T, \max h_{T'}\}$.

Let $(G, w_G) = \mathrm{Tr}_K((T, h_T))$ and $(G', w_{G'}) = \mathrm{Tr}_{K'}((T', h_{T'}))$, for $K' > K$, and consider now the weighted tree $\star = (\{\star\}, w(\emptyset) = 0)$,

We have that $d_E(G, \star) = \sum_{e \in E_G} w_G(e)$ and $d_E(G', \star) = \sum_{e \in E_{G'}} w_{G'}(e)$. Thus, by the reversed triangle inequality:

$$| d_E(G, \star) - d_E(G', \star) | = K' - K \leq d_E(G, G').$$

So, we have:

$$d_E(\mathrm{Tr}_{K'}((T, h_T)), \mathrm{Tr}_K((T, h_T))) = K' - K,$$

and we can set $K = \max\{\max h_T, \max h_{T'}\}$ and take K' arbitrarily close to K to finish the proof.

■

Proof of Theorem 3:

Let $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$ be two AMTs, which are represented by the merge trees (T, h_T) and $(T', h_{T'})$, respectively. Then, we obtain the weighted trees (T, w_T) and $(T', w_{T'})$ via the some Tr_K. The proof is quite long and requires some technical definitions. We split it into different parts to make it more easily readable.

Introduction display posets: For ease of notation, we introduce the following objects.

Definition 15. [16] Given a persistent set $S : \mathbb{R} \to \mathbf{Sets}$, we define its display poset as:

$$D_S := \bigcup_{t \in \mathbb{R}} S(t) \times \{t\}.$$

The set D_S can be given a partial order with $(a, t) \leq (b, t')$ if $S(t \leq t')(a) = b$.

Let $D_{\pi_0(X_\bullet)}$ and $D_{\pi_0(Y_\bullet)}$ be the display posets induced by $\pi_0(X_\bullet)$ and $\pi_0(Y_\bullet)$. We call $h_{\pi_0(X_\bullet)}$ the height function of $D_{\pi_0(X_\bullet)}$ and $h_{\pi_0(Y_\bullet)}$ the height function of $D_{\pi_0(Y_\bullet)}$. This construction is natural, in the sense that natural transformation between persistent sets become order preserving maps between the display posets. Given $\alpha : S \to S'$ natural transformation between persistent sets, we have: $D_\alpha : D_S \to D_{S'}$ defined by:

$$D_\alpha((a,t)) = (\alpha_t(a), t).$$

Given $(a,t) \leq (b,t')$, we have that:

$$(\alpha_t(a), t) \leq (\alpha_{t'}(b), t'),$$

because α is a natural transformation. For more details, see [16].

Introduction couplings: Here, we leverage on the notion of couplings between merge trees defined in [38]. Before recalling such definition, we highlight a subtle difference in merge trees as defined in [38] compared to the definition we give here. In [38], the edge going to infinity which we require in our merge trees, (v, r_T) with $h_T(r_T) = \infty$, is not needed and, thus, such edge is removed. In other words, a merge tree (T, h_T) is the sense of [38] is such that $\deg_T(r_T) > 1$ and $h_T(r_T) \in \mathbb{R}$. We state the results in [38] with the notation of the present manuscript.

Consider $C \subset V_T \times V_{T'}$. Since V_T and $V_{T'}$ are posets, we can introduce a partial order relationship on any $C \subset V_T \times V_{T'}$, having $(a,b) < (c,d)$ if, and only if, $a < c$ and $b < d$. Thus, for each of the sets V_T, $V_{T'}$, and C (or for any subset of those sets), we can consider the set of the maximal elements and the set of the minimal elements, which we indicate with $\max C$, $\min C$ etc..

We define a "multivalued" function $\Lambda_C^T : V_T \to \mathcal{P}(V_T)$ with $\mathcal{P}(V_T)$ being the power set of V_T.

$$\Lambda_C^T(v) = \begin{cases} \max_{v'<v} \pi_T(C), & \text{if there exist } v' \in V_T \text{ such that } v' < v \text{ and } v' \in \pi_T(C); \\ \emptyset, & \text{otherwise.} \end{cases}$$

Given finite sets A, B, we indicate $A - B := \{a \in A \mid a \notin B\}$.

Definition 16. A coupling between two merge trees T and T' is a set $C \subset V_T \times V_{T'}$ such that:

(C1) $\#\max C = 1$;

(C2) the projection $\pi_T : C \to V_T$ is injective (the same for T');

(C3) given (a,b) and (c,d) in C, then $a < c$ if, and only if, $b < d$;

(C4) $a \in \pi_T(C)$ implies $\#\Lambda_C^T(a) \neq 1$ (the same for T').

The set of couplings between T and T' is called $C(T,T')$. Note that, thanks to (C3), (C1) can be replaced by $\#\max \pi_T(C) = \#\max \pi_{T'}(C) = 1$.

Similarly to mappings, each couple in C is associated to a cost, and $\| C \|_\infty$ is defined as the highest of such costs. In [38] it is proven that $d_I(T,T') \leq \| C \|_\infty$ for all couplings C.

Define the following functions as in [38]:

- Define $\varphi_C^T : V_T \to V_T$ so that $\varphi_C^T(x) = \min\{v \in V_T \mid v > x \text{ and } \#\Lambda(v) \neq 0\}$. Note that since the set $\{v \in V_T \mid v > x\}$ is totally ordered, $\varphi_C^T(x)$ is well-defined;
- Similarly, define $\delta_C^T : V_T \to V_T$, defined as $\delta_T(x) = \min\{v \in V_T \mid v \geq x \text{ and } v \in \pi_T(C)\}$;

- Set $\gamma_T^C : V_T - D_C^T \to V_{T'}$ to be:

$$\gamma_T^C(x) = \begin{cases} \arg\min\{h_{T'}(w) \mid (v,w) \in C, v < x\}, & \text{if } \#\{h_{T'}(w) \mid (v,w) \in C, v < x\} > 0 \\ \emptyset, & \text{otherwise.} \end{cases}$$

  If $\#\{h_{T'}(w) \mid (v,w) \in C, v < x\} > 0$, by *(G)*, $\gamma_T^C(x)$ is uniquely defined. Moreover, $\gamma_T^C(\varphi_T^C(x))$ is well-defined for any $v \in V_T$;

- Lastly, set $\eta_C^T(x) := \gamma_C^T(\varphi_C^T(x))$.

To lighten the notation, when clear from the context, we may omit subscripts and superscripts. The costs of the elements a coupling are defined in [38] as follows:

- If $(x,y) \in C$, $cost_C(x) =\mid h_T(x) - h_{T'}(y) \mid$;
- If $x \notin \pi_T(C)$, we have two different scenarios:
  - If $\#\Lambda(x) = 0$, then $cost_C(x) = \max\{(h_T(\varphi(x)) - h_T(x))/2, h_{T'}(\eta(x)) - h_T(x)\}$;
  - If $\#\Lambda(x) > 1$, we have $cost_C(x) =\mid h_T(x) - h_{T'}(w) \mid$ with $(\delta(x), w) \in C$;
  - zero otherwise.

Lastly, Remark 3, Theorems 1 and 3 [38] show that C induces

$$\alpha_C : D_{\pi_0(X_\bullet)} \to D_{\pi_0(Y_\bullet)} \text{ and } \beta_C : D_{\pi_0(Y_\bullet)} \to D_{\pi_0(X_\bullet)}$$

such that

$$cost_C(v) =\mid h_{\pi_0(Y_\bullet)}(\alpha_C(v)) - h_{\pi_0(X_\bullet)}(v) \mid$$

for any $v \in V_T$, and

$$cost_C(w) =\mid h_{\pi_0(X_\bullet)}(\beta_C(w)) - h_{\pi_0(Y_\bullet)}(w) \mid$$

for any $w \in V_{T'}$.

Main body of the proof: We now want to establish relations between mappings between $\mathrm{Tr}_K((T,h_T))$ and $\mathrm{Tr}_K((T',h_{T'}))$, with $K > \max h_T, \max h_{T'}$, and couplings between (T,h_T) and $(T',h_{T'})$. When working with $\mathrm{Tr}_K((T,h_T))$ and $\mathrm{Tr}_K((T',h_{T'}))$, we can apply the following corollary of Proposition 2.

Corollary 3. Given (T,w_T) and $(T',w_{T'})$ weighted trees, if r_T and $r_{T'}$ are of order 1, there exist a minimizing mapping M with $\#\max M = 1$.

Consider a minimizing mapping $M \in M_2(T,T')$ with $\#\max M = 1$. Thanks to Lemma 1, we have that $a \in \pi_T(M)$ implies $\#\Lambda_M^T(a) \neq 1$ (the same for T'). In fact, (C4) is equivalent to having no vertices of degree two after deletions and ghostings. This means that $C_M := \{(a,b) \in M \mid a \in V_T \text{ and } b \in V_{T'}\}$ is a coupling.

Conversely, given a minimizing coupling C, the set

$$\begin{aligned} M_C := C &\cup \{(a,\mathfrak{D}) \mid a \notin \pi_T(C) \text{ and } \#\Lambda_C^T(a) \neq 1\} \\ &\cup \{(\mathfrak{D},b) \mid b \notin \pi_{T'}(C) \text{ and } \#\Lambda_C^{T'}(b) \neq 1\} \end{aligned}$$

$$\cup \{(a, \mathfrak{G}) \mid a \notin \pi_T(C) \text{ and } \#\Lambda_C^T(a) = 1\}$$
$$\cup \{(\mathfrak{G}, b) \mid b \notin \pi_{T'}(C) \text{ and } \#\Lambda_C^{T'}(b) = 1\}$$

is a mapping. Clearly, $M_{C_M} = M$ and $C_{M_C} = C$.

Now, we prove the following lemma.

Lemma 2. Let $f : [a, K] \to [b, K]$ be a monotone function such that $f(K) = K$, $K \in \mathbb{R}$. For every $\{x_1 < \ldots < x_{n+1} = K\} \subset [a, K]$:

$$\max_{i=1,\ldots,n+1} d(f(x_i), x_i) \leq \sum_{i=1}^{n} \mid d(x_i, x_{i+1}) - d(f(x_i), f(x_{i+1})) \mid \tag{B.1}$$

Proof. For every $i = 1, \ldots, n+1$, set $v_i = f(x_i) - x_i$, and consider $m \in \{1, \ldots, n+1\}$ such that $\mid f(x_m) - x_m \mid = \max \mid f(x_i) - x_i \mid$.

Then, we can find ε_i, $i = 1, \ldots, n+1$ such that:

$$\begin{aligned} v_1 &= v_m + \varepsilon_1 \\ v_2 &= v_m + \varepsilon_1 + \varepsilon_2 \\ &\ldots \\ v_i &= v_m + \sum_{j=1}^{i} \varepsilon_j \\ &\ldots \\ v_{n+1} &= v_m + \sum_{j=1}^{n+1} \varepsilon_j. \end{aligned}$$

Using ε_i, we can rewrite Equation (B.1), which we need to prove, as:

$$\mid v_m \mid \leq \sum_{i=1}^{n} \mid x_i - f(x_i) - (x_{i+1} - f(x_{i+1})) \mid = \sum_{i=1}^{n} \mid v_i - v_{i+1} \mid = \sum_{i=1}^{n} \mid \varepsilon_{i+1} \mid .$$

Clearly, by construction, we have:

$$\varepsilon_1 \leq 0.$$

Moreover, since $f(x_{n+1}) = x_{n+1}$, we have $v_{n+1} = 0$. Thus, $v_m + \sum_{j=1}^{n+1} \varepsilon_j = 0$, which means $-v_m = \sum_{j=1}^{n+1} \varepsilon_j$. This implies:

$$\mid v_m \mid = \mid \sum_{j=1}^{n+1} \varepsilon_j \mid \leq \mid \sum_{j=2}^{n+1} \varepsilon_j \mid \leq \sum_{i=1}^{n} \mid \varepsilon_{i+1} \mid .$$

□

Consider now a leaf $l \in L_T$ and take $\zeta_l = \{p \in V_T \mid v \geq l\}$. Consider the interval $[h_T(l), K]$. The map $v \in \zeta_l \mapsto h_T(v)$ gives a $1:1$ correspondence between ζ_l and a finite collection of points in $[h_T(l), K]$. We define $f(h_T(v)) = h_{\pi_0(Y_\bullet)}(\alpha_C(v))$ for $v \in \zeta_l - \{r_T\}$, and $f(K) = K$. Note that α_C is constructed such that $\alpha_C(v) = w$ if $(v, w) \in C$. Thus, $f(h_T(v)) \leq K$ for $v \in \zeta_l$.

We extend f on $[h_T(l), K]$ via linear interpolation. Since α_C is monotone wrt the partial order on $D_{\pi_0(X_\bullet)}$, then f is monotone on $[h_T(l), K]$.

Now, consider T and apply on it all the deletions and ghostings which involve points which are not in ζ_l. We call $\{v_1 = l < \ldots < v_{n+1} = r_T\}$ the coupled points in ζ_l. Then:

$$| h_T(v_i) - h_T(v_{i+1}) - f(h_T(v_i)) - f(h_T(v_{i+1})) | \leq cost_{M_C}([v_i, v_{i+1}] \mapsto [\alpha_C(v_i), \alpha_C(v_{i+1})]), \tag{B.2}$$

with $cost_{M_C}([v_i, v_{i+1}] \mapsto [\alpha_C(v_i), \alpha_C(v_{i+1})])$ being the cost of the edits associated to points $v_i \leq p \leq v_{i+1}$ and $\alpha_C(v_i) \leq q \leq \alpha_C(v_{i+1})$. In fact, we have equality in Eq (B.2) if there are no deletions for any $p \in V_T$ such that $v_i \leq p \leq v_{i+1}$ and any $q \in V_{T'}$ such that $\alpha_C(v_i) \leq q \leq \alpha_C(v_{i+1})$. Otherwise, the total cost of the deletions and shrinking exceeds $| h_T(v_i) - h_T(v_{i+1}) - f(h_T(v_i)) - f(h_T(v_{i+1})) |$ since:

$$| n_1 + n_2 - (n_3 + n_4) | \leq n_1 + n_3 + | n_2 - n_4 |$$

for any $n_1, \ldots, n_4 \in \mathbb{R}_{\geq 0}$. Lastly, note that $| h_T(v_i) - f(h_T(v_i)) | = cost_C(v)$.

By Lemma 2, we have:

$$\begin{aligned} \max cost_C(v_i) &= \max h_T(v_i) - f(h_T(v_i)) \\ &\leq \sum_{i=1}^{n} | h_T(v_i) - f(h_T(v_i)) + f(h_T(v_{i+1}) - h_T(v_{i+1}) | \\ &\leq cost_{M_C}([l, r_T] \mapsto [w, r_{T'}]) \end{aligned} \tag{B.3}$$

with $(v, w) \in M_C$ and $v = \min \zeta_l \cap \pi_T(C)$.

Conclusion: Let $\| C \|_\infty = cost_C(x)$. WLOG $x \in V_T$. Then, $x \in \zeta_l$ for some $l \in L_T$. By applying Eq (B.3), we obtain the result.

Proof of Proposition 6:

In the following proof, we employ display posets, which are introduced in the proof of Theorem 3.

Consider $\mathcal{M}(\pi_0(X_\bullet))$ and $\pi_0(X^n_\bullet)$ such that $T = \mathcal{M}(\pi_0(X_\bullet))$ and $T_n = \mathcal{M}(\pi_0(X^n_\bullet))$. Note that we may often deliberately confuse merge trees and AMTs to lighten the notation.

Take any $\varepsilon > 0$, and then take N such that $d_I(T, T_n) \leq \varepsilon$ for every $n > N$. Let $T'_n := \mathcal{S}_{2\varepsilon}(T_n)$. Clearly, we always have $d_I(T_n, T'_n) = 2\varepsilon$. Suppose that $\sup_{n \in N} \text{size}(T'_n) = +\infty$. If this is the case, for every $K > 0$, there is a merge tree T_n, with more than K leaves v such that $i^{2\varepsilon}_{\pi_0(X^n_\bullet)}(v)$ (more precisely, $(i^{2\varepsilon}_{\pi_0(X^n_\bullet)})_{h_{T_n}(v)}(v)$) is a leaf for T'_n and, thus, $h_{T_n}(parent(v)) - h_{T_n}(v) > 2\varepsilon$. We now show that if K is bigger than the number of leaves of T, we obtain a contradiction as, in order to have $d_I(T, T_n) \leq \varepsilon$, we cannot contract edges which are longer than 2ε.

More formally, let α, β be ε-compatible maps between T and T_n. We know they exist as $d_I(T, T_n) \leq \varepsilon$. These natural transformations, as mentioned in the proof of Theorem 3, induce order preserving maps between display posets. In particular:

$$D_\alpha : D_{\pi_0(X_\bullet)} \to D_{\mathcal{S}_\varepsilon(\pi_0(X^n_\bullet))},$$
$$D_\beta : D_{\pi_0(X^n_\bullet)} \to D_{\mathcal{S}_\varepsilon(\pi_0(X_\bullet))}.$$

Note that $T'_n = \mathcal{S}_{2\varepsilon}(\pi_0(X^n_\bullet)) = \mathcal{S}_\varepsilon(\mathcal{S}_\varepsilon(\pi_0(X^n_\bullet)))$ and, in particular, $\mathcal{S}_\varepsilon(\pi_0(X^n_\bullet))$ cannot have less leaves than T'_n.

Since T has less leaves than T'_n and of $\mathcal{S}_\varepsilon(\pi_0(X^n_\bullet))$, we claim that there exist v, w leaves of T_n, with $t = h_{T_n}(w)$ and $t' = h_{T_n}(v)$, such that:

1) $(i^\varepsilon_{\pi_0(X^n_\bullet)})_t(w)$ and $(i^\varepsilon_{\pi_0(X^n_\bullet)})_{t'}(v)$ are leaves of $\mathcal{S}_\varepsilon(\pi_0(X^n_\bullet))$, and $(i^{2\varepsilon}_{\pi_0(X^n_\bullet)})_t(w)$ and $(i^{2\varepsilon}_{\pi_0(X^n_\bullet)})_{t'}(v)$ are leaves of T'_n;

2) $h_{T_n}(parent(v)) - t' > 2\varepsilon$ and $h_{T_n}(parent(w)) - t > 2\varepsilon$;

3) $D_\beta((w,t)) \leq D_\beta((v,t'))$.

We already know that 1) and 2) need to be satisfied for some leaves. Not being able to find two such leaves which also satisfy 3) would imply T having at least K leaves.

Since $D_\beta((w,t)) \leq D_\beta((v,t'))$, we have $D_\alpha(D_\beta((w,t)) \leq D_\alpha(D_\beta((v,t'))$. By definition, we know that $D_\alpha(D_\beta((v,t')) = (X_{t'\leq t'+2\varepsilon}(v), t' + 2\varepsilon)$. In particular, this means that $(v,t'), (w,t') \leq (X_{t'\leq t'+2\varepsilon}(v), t' + 2\varepsilon)$. This is absurd because we have found a point on the abstract merge tree (associated to) T_n, which is a common ancestor of both v and w and whose height differs from the ones of v and w by exactly 2ε. However, we know that the edges $(v, parent(v))$ and $(w, parent(w))$ are both longer than 2ε.

Thus, $\sup_{n>N} \text{size}(T'_n) = K$ for some $K \in \mathbb{R}$ such that $K \leq \text{size}(T)$. WLOG we can suppose $K = \text{size}(T)$. Since d is finitely stable, for some $C > 0$, we have:

$$\begin{aligned} d(T, T'_n) \leq C \cdot 2K d_I(T, T'_n) &\leq C \cdot 2K(d_I(T, T_n) + d_I(T_n, T'_n)) \\ &= C \cdot 2K(d_I(T, T_n) + \varepsilon). \end{aligned}$$

Thus, by choosing ε such that:

$$\max\{2\varepsilon, 2KC\varepsilon\} \leq \delta,$$

we obtain the thesis.