A Metric for Tree-Like Topological Summaries

Matteo Pegoraro∗

August 31, 2021

Abstract

In this work we define a novel metric structure for a family of tree-like topological summaries. This family of objects is a natural combinatoric generalization of merge trees of scalar fields and hierarchical dendrograms. The metric introduced can be computed with a dynamical integer linear programming approach and we showcase its feasibility and the effectiveness of the whole framework with simulated data sets. In particular we stress the versatility of these topological summaries, which prove to be very effective in situation where other topological data analysis tools, like persistence diagrams, can not be meaningfully employed.

Keywords: Topological Data Analysis, Merge Trees, Reeb Graphs, Dendrograms, Tree Edit Distance

1. Introduction

Topological Data Analysis (TDA) is the name given to an ensemble of techniques which are mainly focused on retrieving topological information from different kinds of data Lum et al. (2013). Consider for instance the case of point clouds: the topology of a point cloud itself is quite poor and it would be much more interesting if, using the point cloud, one could gather information about the topological space data was sampled from. Since, in practice, this is often not possible, one can still try to capture the “shape” of the point cloud. The idea of persistent homology (PH) Edelsbrunner and Harer (2008) is an attempt to do so: using the initial point cloud, a nested sequence of topological spaces is built, which are heavily dependent on the initial point cloud, and PH tracks along this sequence the persistence of the different topological features which appear and disappear. As the name persistent homology suggests, the topological features are understood in terms of generators of the homology groups Hatcher (2000) taken along the sequence of spaces. One of the fundational results in TDA is that this information can be represented by a set of points on the plane Edelsbrunner et al. (2002); Zomorodian and Carlsson (2005), with a point of coordinates \((x, y)\) representing a topological feature being born at time \(x\) along the sequence, and disappearing at time \(y\). Such representation is called persistence diagram (PD). Persistence diagrams can be given a metric structure through the Bottleneck and Wasserstein metrics, which, despite having good properties in terms of continuity with respect to perturbation of the original data Cohen-Steiner et al. (2007, 2010), provide badly behaved metric spaces. Various attempts to define tools to work in such spaces have been made Mileyko et al. (2011); Turner et al. (2012); Lacombe et al. (2018); Fasy et al. (2014), but still it proves to be an hard problem. In order to obtain spaces with better properties and information which is more easily represented in terms of fixed length vectors (needed for many Machine Learning techniques) a number of topological summaries, alternative

∗. MOX – Department of Mathematics, Politecnico di Milano
to PDs, have been proposed, such as: persistence landscapes Bubenik (2015), persistence images Adams et al. (2017) and persistence silhouettes Chazal et al. (2015).

All the aforementioned machinery has been successfully applied to a great number of problems in a very diverse set of scientific fields: complex shape analysis MacPherson and Schweinhart (2010), sensor network coverage Silva and Ghrist (2007), protein structures Kovacev-Nikolic et al. (2016); Gameiro et al. (2014), DNA and RNA structures Emmett et al. (2015); Rizvi et al. (2017), robotics Bhattacharya et al. (2015); Pokorny et al. (2015), signal analysis and dynamical systems Perea and Harer (2013); Perea et al. (2015); Maletić et al. (2015), materials science Xia et al. (2015); Kramár et al. (2013), neuroscience Giusti et al. (2016); Curto (2016), network analysis Sizemore et al. (2015); Pal et al. (2017), and even deep learning theory Hofer et al. (2017); Naitzat et al. (2020).

Related Works

Close to the definition of persistent homology for 0 dimensional homology groups, lie the ideas of merge trees of functions, phylogenetic trees and hierarchical clustering dendrograms. Merge trees of functions Morozov and Weber (2013) are a particular case of Reeb Graphs Shinagawa et al. (1991); Biasotti et al. (2008), occurring when using the sub-level sets of a bounded Morse function Audin et al. (2014) defined on a simply connected domain. Phylogenetic trees and clustering dendrograms are very similar objects which describe the evolution of a set of labels under some similarity measure or agglomerative criterion. Both objects are widely used respectively in phylogenetic and statistics and many complete overviews can be found, for instance see Felsenstein and Felenstein (2004); Garba et al. (2021) for phylogenetic trees and Murtagh and Contreras (2017); Xu and Tian (2015) for clustering dendrograms. Informally speaking, while persistence diagrams record only that, at certain level along a sequence of topological spaces, some path connected components merge, merge trees, phylogenetic trees and clustering dendrograms encode also the information about which components merge with which. Usually tools like phylogenetic trees and clustering dendrograms are used to infer something about a fixed set of labels, for instance an appropriate clustering structure, however, we are more interested in looking at the information they carry as unlabeled objects obtained with different sets of labels. For this reason most of the metrics available for phylogenetic trees and clustering dendrograms are not valuable for our purposes.

In the last years a lot of research sparkled on such topics, starting from the more general case of Reeb graphs, to some more specific works on merge trees. Different but related metrics have been proposed to compare Reeb graphs Di Fabio and Landi (2012); Bauer et al. (2016); Di Fabio and Landi (2016); Bauer et al. (2020, 2014a,b); Carrière and Oudot (2017), which have been shown to possess very interesting properties in terms of Morse functions on manifolds, connecting the combinatorial nature of Reeb Graphs with deformation-invariant characterizations of manifolds which are smooth, compact, orientable and without boundary. On the specific case of merge trees, there has been some research on their computation Pascucci and Cole-McLaughlin (2003); Morozov and Weber (2013) and on using them as visualization tools wu and Zhang (2013); Bock et al. (2017), while other works Beketayev et al. (2014); Morozov et al. (2013) started to build frameworks to analyze sets of merge trees, mainly proposing a suitable metric structure to compare them, as do some recent preprints Gasparovic et al. (2019); Touli (2020). The main issue with all the proposed metrics is their computational cost, causing a lack for examples and applications also when approximation algorithms are available Touli and Wang (2018). When applications and analysis are carried out Sridharamurthy et al. (2020), the employed metric does not have suitable properties and thus the authors must resort to
a “computational solution to handle instabilities” (Sridharamurthy et al. (2020), Section 1.2) to use their framework. Along with that, such metrics are difficult to be extended to more general objects than merge trees. Lastly, there is a recent preprint investigating structures lying in between merge trees and persistence diagrams, to avoid computational complexity while retaining some of the additional information provided by such objects Elkin and Kurlin (2020).

**Main Contributions**

The success of PDs highlighted before, strongly motivates the development of more refined and computable techniques to work with merge trees, phylogenetic trees and clustering dendrograms. Our contribution to such topic is three folded: first we introduce a novel use of tree-like structures as topological summaries with objects called *generalized dendrograms*, second we propose a metric structure for the space of *generalized dendrograms* in the form of a novel edit distance between weighted (in a very broad sense), unlabeled, unordered trees; lastly we develop a dynamical integer programming algorithm to make this metric viable for a good range of applications.

If we consider our framework restricted just to the case of merge trees, the works Di Fabio and Landi (2012); Bauer et al. (2016); Di Fabio and Landi (2016) contain a perspective which is very similar to ours, since they define edit distances for Reeb Graphs, and thus, for merge trees. However, at a closer look, the two approaches diverge immediately: the approach in Di Fabio and Landi (2012); Bauer et al. (2016); Di Fabio and Landi (2016) is focused of interpreting modifications of the graphs in terms of deformations of the initial topological space, while our is more concerned on the computational advantages offered by accurately defined edit distances and the possibility to extend the metric to more general kinds of trees. Thus, we end up with very different definitions and properties (see Section 4 and Remark 6 for more details). The edit distance we propose starts from usual tree edit distances Tai (1979); Bille (2005) but adds fundamental modifications in order to obtain the properties needed to compare topological information. A simplified but similar definition has already been considered in Koperwas and Walczak (2011), but it is just cited in few lines as a possibility without a real motivation, which lacks any kind of investigation (even whether or not it defines a proper metric). As already stated, we follow the path of edit distances instead of looking for modifications of other metrics for trees Billera et al. (2001); Feragen et al. (2012); Wang and Marron (2007) because of the computational properties which they often possess, making them suited for dealing with unordered and unlabelled trees Hong et al. (2017). The computational issues raised by those kind of trees are in fact a primary obstacle to designing feasible algorithms Hein et al. (1995). Nevertheless, we are able to obtain an Integer Linear Programming (ILP) algorithm which can compute the distance between two binary trees with $N$ and $M$ leaves respectively, by solving $\mathcal{O}(N \cdot M)$ ILP problems with $\mathcal{O}(N \cdot \log(N) \cdot M \cdot \log(M))$ variables and $\mathcal{O}(N + M)$ constraints.

**Outline**

The paper is organized as follows. In Section 2 we describe the main facts that motivate our work. In Section 3 we give formal definitions of generalized dendrograms. In high generality, with Section 4 we tackle the problem of finding a suitable metric structure to compare those objects. In Section 5 we detail how generalized dendrograms can be employed to build new topological summaries. In Section 6 we prove some properties of the metric previously defined, which lead to the algorithm presented in Section 7. In
Section 8 we present some simulations and examples to showcase the effectiveness of the proposed framework and we end up with some conclusions in Section 9. Proofs of results are found in Appendix A.

2. Main Ideas and Driving Examples

In TDA the main sources of information are sequences of homology groups with field coefficients: using different pipelines a single datum is turned into a sequence of topological spaces, which, in turn, induces - via some homology functor with coefficients in the field $\mathbb{K}$ - a sequence of vector spaces with linear maps which are usually all isomorphisms but for a finite set of points in the sequence. Any such sequence $A_i \xrightarrow{\psi_{i+1}} A_{i+1}$ is then turned into a topological summary, for instance a persistence diagram, which completely classifies such objects up to sequence isomorphisms. That is, if for two vector spaces sequences $A_i \xrightarrow{\psi_{i+1}} A_{i+1}$ and $B_i \xrightarrow{\eta_{i+1}} B_{i+1}$ exists a family of linear isomorphisms $g_i : A_i \to B_i$ such that for all $i$ holds $\eta_{i+1} \circ g_i = g_{i+1} \circ \psi_{i+1}$, then they are represented by the same persistence diagram. As already highlighted in the introduction, PDs have proven to be useful in a wide variety of tasks. However there might be cases where a more discriminative topological summary is needed, or a summary to which additional information can be meaningfully attached. In Elkin and Kurlin (2020) this topic is discussed and some motivational case studies are carried out, but we want to go further in this direction. To do so we present two simple examples and then try to give some informal intuition of the ideas which are going to be formalized in the following sections.

2.1 Hierarchical clustering

Consider the single linkage dendrograms and the zero dimensional PDs obtained from point clouds as in Figure 1 (for a quick introduction to persistence diagrams see Appendix C). The persistence diagrams (in Figure 1c and Figure 1f) are very similar, in fact they simply record that there are four major clusters which merge at similar times across the Vietoris-Rips filtrations Edelsbrunner and Harer (2008) of the two point clouds. The hierarchical dendrograms, instead, are clearly very different since they show that in the first case (Figure 1a, Figure 1b, Figure 1c) the cluster with most points is the one which is more separated from the others in the point cloud; while in the second case (Figure 1d, Figure 1e, Figure 1f) the two bigger clusters are the first that get merged and the farthest cluster of points on the right could be considered as made by outliers. In many applications it would be important to distinguish between these two scenarios, since the two main clusters get merged at very different heights on the respective dendrograms. We use this example to point out another fact: while both dendrograms have as many leaves as there are points in the point clouds, if one attaches to a vertex of the dendrogram the cardinality of the cluster obtained by cutting the edge above the vertex itself, then most of the information contained in the dendrogram could be summarized using a much smaller tree (in terms of number of leaves). For instance one could decide to remove all the vertices associated to clusters whose cardinality is smaller than a certain threshold.

2.2 Merge Trees of functions

Given a continuous function $f : [a, b] \to \mathbb{R}$ we can extract its merge tree. For a detailed definition of the procedure refer to Morozov and Weber (2013) or [Paper FDA]. Roughly
Figure 1: Data clouds, hierarchical clustering dendrograms and PDs involved in the first example.

Figure 2: Merge Trees of Functions

speaking the merge tree tracks the evolution of the path connected components of the sublevel sets $f^{-1}((-\infty, t])$, for an example see Figure 2b.

Again, we point out two facts. First PDs may not be able to distinguish functions one may wish to distinguish, as made clear by Figure 3. Second, Proposition 1 in [Paper FDA] states that if one changes the parametrization of a function by means of homeomorphisms, then, both the associated merge tree and persistence diagram do not change. A consequence of such result is that one can linearly shrink or spread the domain of the function $f : [a, b] \to \mathbb{R}$ at will, without changing its merge tree (and PD). There are cases in which such property may be useful but surely there are times when one may want to distinguish if an oscillation lasted for a time interval of $10^{-5}$ or $10^5$. 
Figure 3: We see a comparison of four functions with the same PD in dimension 0 but different merge trees. The function is displayed on the first row of each subplot, while on the second we have on the left the PD and on the right the merge tree.
2.3 Intuitions

Behind the examples in Section 2.1 and Section 2.2 there is the following phenomenon. Especially in the case of 0 dimensional homology, one can naturally fix a basis in the homology vector spaces (for instance the one induced by the connected components) such that the maps in the sequence coherently map the basis fixed in one space into the basis fixed into the following one. Both dendrograms and merge trees capture the information given by such maps and bases along their respective sequences. In addition to that, having fixed a basis inside every homology group, one can gather information about that basis and follow it at every step along the vector spaces sequence. For instance, in the Vietoris-Rips filtration of a finite point cloud, one could count the number of points in each connected component along the sequence of homology groups.

Remark 1  This idea of gathering additional information along the homology groups, a priori can be applied also to PD and the basis which is used to find the diagram representation. But we argue that this cannot be done meaningfully because minimal changes in the sequence of vector spaces, for instance exchanging the birth order of two classes, yields big changes in terms of basis representation, due to the elder rule. For more details see the second example in Section 5.3.

3. Tree-Like Summaries

In this section we start to formalize our ideas: we start from merge trees, built in a combinatorial fashion, then we define more general objects, called generalized dendrograms, on which we focus for the remaining of the manuscript. To begin with, we state which are the vector spaces sequences we want to work with.

Definition 1  A fixed basis vector spaces filtration is a family of couples $\{(A_t,a_t)\}_{t \in \mathbb{R}}$ where $a_t$ is a finite dimensional vector space of dimension $n_t$ and $a_t = \{a_{t1}^t, \ldots, a_{tn_t}^t\}$ is a basis for $A_t$; there are also maps for every $t < t' \in \mathbb{R}$: $\psi_{t'}^t : A_t \rightarrow A_{t'}$ which must satisfy the following conditions:

1. given $t < t' < t''$, then $\psi_{t'}^t \circ \psi_{t''}^{t'} = \psi_{t''}^t$, this is called the cocycle condition;
2. $\psi_{t'}^t(a_t) \subset a_{t'}$;
3. for any $t$, $\{\psi_{t'}^t\}_{t'}$ are all isomorphisms but for a finite set of $t' \in \mathbb{R}$; such $t'$ are called critical points;
4. there exists a value $t^-$ such that for any $t < t^-$, $(v_t, V_t) = (\emptyset, \{0\})$;
5. there exists a value $t^+$ such that, for any $t^+ \geq t$, $\dim(V_t) = 1$.

Remark 2  In Topology and more in general in Category Theory, filtrations (and filtered objects) are obtained with sequences of objects and morphisms (usually monomorphisms). For this reason, with a slight abuse of notation, we use the terms filtration and sequence to refer to ordered sets of objects indexed on the real line, but where “relevant changes” happen only in a finite set of values.

The the cocycle condition can be exploited to observe some facts about the critical points.
Remark 3 If \( t' \) is the first critical point for \( t \), that is, it is the smallest value \( t' \) bigger than \( t \) such that \( \psi^t_{t'} \) is not an isomorphism, then for all \( t'' \in [t, t') \) the value \( t' \) is critical and there are no other critical points for \( t'' \) in the interval \((t'', t')\). This holds thanks to the cocycle condition. In other words the critical points split the interval \([t^-, t^+]\) in a finite set of intervals \([t_i, t_{i+1})\) where \( t_i \) are the critical points. Note that for any \( t, t' \in [t_i, t_{i+1}) \), if \( t < t' \), \( \psi^t_{t'} \) is an isomorphism.

Now we define which equivalence classes of sequences we want to work with.

**Definition 2** Consider two fixed basis vector spaces filtrations \( V = \{(A_t, a_t)\}_{t \in \mathbb{R}} \) and \( W = \{(B_t, b_t)\}_{t \in \mathbb{R}} \), with maps \( \psi^t_t : A_t \to A_{t'} \) and \( \eta^t_t : B_t \to B_{t'} \) respectively. A basis preserving isomorphism of sequences \( \{g_t\}_{t \in \mathbb{R}} \) is family of linear isomorphisms \( g_t : A_t \to B_t \) such that \( g_t \) induces a bijection between \( a_t \) and \( b_t \), and, for all \( t \), the following square commutes:

\[
\begin{array}{ccc}
A_t & \xrightarrow{\psi^t_t} & A_{t'} \\
\downarrow{g_t} & & \downarrow{g_{t'}} \\
B_t & \xrightarrow{\eta^t_t} & B_{t'}
\end{array}
\]

### 3.1 Merge Trees

The definition of merge trees is not novel but usually is obtained in a more topological fashion and starting from functions Morozov and Weber (2013). Instead we use the more combinatorial approach found in [FDA] which we report in the following lines. Such definition relies on graph-based representations of unordered, unlabeled trees, which are called tree structures throughout the dissertation.

**Definition 3** A tree structure \( T \) is given by a set of vertices \( V_T \) and set of edges \( E_T \subset V_T \times V_T \) which form a connected rooted acyclic graph. The order of a vertex is the number of edges which have that vertex as one of the extremes. Any vertex with an edge connecting it to the root is its child and the root is its father. In this way we recursively define father and children (possibly none) relationships for any vertex on the tree. The vertices with no children are called leaves or taxa. The relationship father > child induces a partial order on \( V_T \). Similarly, the edges \( E_T \) are given in the form of ordered couples \((a, b)\) with \( a < b \). For any vertex \( v \in V_T \), sub\(_T\)(\( v \)) is the subtree of \( T \) rooted in \( v \), that is the tree structure given by the set of vertices \( v' \leq v \). If clear from the context we might omit the subscript \( T \).

A finite tree structure is a tree structure with \( V_T \) being a finite set.

Note that, identifying an edge \((v, v')\) with its lower vertex \( v \), gives a bijection between \( V_T - \{r_T\} \) and \( E_T \), that is \( E_T \simeq V_T \) as sets, and thus one can interpret the information associated to vertices as information associated to edges. Given this bijection, we often use \( E_T \) to indicate the vertices \( v \in V_T - \{r_T\} \), to simplify the notation.

Moreover, we do not want the vertex set of tree structures to carry any relevant structure, since we are going to consistently add pieces of information to a tree structure in a different way. For this reason we treat such objects up to the following isomorphism classes.

**Definition 4** Two tree structures \( T \) and \( T' \) are isomorphic if exists a bijection \( g : V_T \to V_{T'} \) that induces a bijection between the edges sets \( E_T \) and \( E_{T'} : (a, b) \mapsto (g(a), g(b)) \). Such \( g \) is an isomorphism of tree structures.
Now we can borrow from [Paper FDA] the following definition.

**Definition 5** A merge tree is a finite tree structure $T$ with a monotone increasing height function $h : V_T \rightarrow \mathbb{R}$. Two merge trees $(T, h)$ and $(T', h')$ are isomorphic if $T$ and $T'$ are isomorphic as tree structures and the isomorphism $g : V_T \rightarrow V_{T'}$ is such that $h = h' \circ g$. Such $g$ is an isomorphism of merge trees.

Section 2 of [Paper FDA] details how, given a function $f : [a, b] \rightarrow \mathbb{R}$, with $a, b \in \mathbb{R}$, a merge tree can be used to represent the fixed basis vector spaces filtration given by $A_t = H_0((f^{-1}((\neg\infty, t]))$ and with $a_t$ being the basis induced by path connected components. The same procedure can be used to represent any fixed basis vector spaces filtration $\{ (A_t, a_t) \}_{t \in \mathbb{R}}$ up to basis preserving isomorphism. The merge tree obtain is unique up to the choice of the vertex set, that is, up to isomorphism of merge trees. The general idea is the following. We consider only the maps $\psi_{t_i}^{t_{i-1}} : A_{t_{i-1}} \rightarrow A_{t_i}$, with $\{ t_i \}_{i=1, \ldots, n}$ being the critical points of the filtration: any time $(\psi_{t_i}^{t_{i-1}})^{-1}(a_{t_k}^{t_{i-1}}) = \emptyset$, we have a leaf $v$ with $h_T(v) = t_i$, and when we have $\psi_{t_i}^{t_{i-1}}(a_{t_k}^{t_{i-1}}) = \psi_{t_{i-1}}^{t_{i-1}}(a_{t_k}^{t_{i-1}})$ we have a vertex $v'$ whose children are the vertices of the tree structure associated to the path connected components which merge, with $h_T(v') = t_i$. For more details see Appendix B. Note that the image of the height function $h_T$ is the set of critical values of the filtration.

### 3.2 Generalized Dendrograms

Merge trees are the most natural starting point, since they can be used to represent fixed basis vector spaces filtration up to basis preserving isomorphism. But know we want to take a step forward, collecting and representing other kind of information about those sequences of vector spaces, generalizing merge trees.

Since deciding what kind of information we want to track along a fixed basis vector spaces filtration and deciding how to attach it to the tree-structure may end up overloading the notation and making too restrictive hypotheses, we take a more general approach, developing the theory forgetting about homology groups and fixed bases, but recovering such more specific point of view in Section 5.

**Definition 6** Given two sets $X$ and $Y$, consider their disjoint union $X \coprod Y$, and a tree structure $T$. A multiplicity function $\varphi$ is a function $\varphi : V_T \rightarrow X \coprod Y$, such that $\varphi(E_T) \subset X$ and $\varphi(r_T) \in Y$.

**Definition 7** A generalized dendrogram is a tree structure with a multiplicity function. Two generalized dendrograms $(T, \varphi)$ and $(T', \varphi')$ are isomorphic if there is a bijection $g : V_T \rightarrow V_{T'}$ which makes them isomorphic as tree structures and is such that $\varphi(v) = \varphi'(g(v))$.

### 4. Edit Distance for Generalized Dendrograms

The main goal of the following Sections is to propose a computable (pseudo) metric between generalized dendrograms. We want this metric to be suitable to compare topological information, in the sense explained by Section 4.2.

### 4.1 Edits of Dendrograms

The approach we follow is to define a distance which is inspired by the Tree Edit Distances Tai (1979), but with substantial differences in the edit operations. The philosophy of
these distances is to allow certain modifications of the base object, called edits, each being associated to a cost, and to define the distance between two objects as the minimal cost that is needed to transform the first object into the second with a finite sequence of edits. Edit distances in fact frequently enjoy some decomposition properties which simplify the calculations Hong et al. (2017), which are notoriously very heavy Hein et al. (1995).

First of all, let us make some hypotheses on the multiplicity functions and their codomains.

**Definition 8** A set $X$ is called editable if the following conditions are satisfied:

(P1) $(X, d)$ is a metric space

(P2) $(X, \oplus, 0)$ is a monoid (that is $X$ has an associative operation $\oplus$ with zero element 0)

(P3) the map $d(\cdot, 0) : X \to \mathbb{R}$ is a map of monoids between $(X, \oplus)$ and $(\mathbb{R}, +)$: $d(x \oplus y, 0) = d(0, x) + d(0, y)$.

(P4) $d$ is $\oplus$ invariant, that is: $d(x, y) = d(z \oplus x, z \oplus y) = d(x \oplus z, y \oplus z)$

Note that in property (P3), $d(x \oplus y, 0) = d(x, 0) + d(y, 0)$, implies that $x \oplus y \neq 0$. Moreover (P3)-(P4) imply that the points 0, $x$, $y$ and $x \oplus y$ form a rectangle which can be isometrically embedded in an Euclidean plane with the Manhattan geometry (that is, with the norm $\| \cdot \|_1$): $d(x, x \oplus y) = d(0, y)$, $d(y, x \oplus y) = d(0, x)$ and $d(x \oplus y, 0) = d(0, x) + d(0, y)$.

With these additional pieces of structure there are situations which we want to avoid, because they represent “degenerate” dendrograms which introduce formal complications.

**Definition 9** Given an editable space $X$ and a tree-structure $T$, a proper multiplicity function is a multiplicity function $\varphi$ such that $\varphi : E_T \to X$ and $0 \notin \varphi(E_T)$.

From now on we only work with editable spaces, and we want to consider exclusively proper multiplicity functions. To lighten the notation, however, we omit to write “proper” explicitly.

**Definition 10** Given an editable space $X$ and a metric space $Y$, the editable dendrogram space $(\mathcal{D}, Y \sqcup X)$ is given by the set of generalized dendrograms with (proper) multiplicity functions with values in $Y \sqcup X$.

Given an editable dendrogram space $(\mathcal{D}, Y \sqcup X)$, with $(X, \oplus, 0)$ editable space, we can define our edits.

- We call **shrinking** of a vertex/edge a change of the multiplicity function. The new multiplicity function must be equal to the previous one on all vertices, apart from the “shrunk” one. In other words, for an edge $e$, this means changing the value $\varphi(e)$ with another non zero value in $X$. For the root, this means changing arbitrarily its multiplicity value inside $Y$.

- A **deletion** is an edit with which a vertex/edge is deleted from the dendrogram. Consider an edge $(v_1, v_2)$. The result of deleting $v_1$ is a new tree structure, with the same vertices a part from $v_1$ (the smaller one), and with the father 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 multiplicity function on the new edge. This edit cannot be done on the root.
Lastly we define a transformation which eliminates an order two vertex, connecting the two adjacent edges which arrive and depart from it. 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 order two. Then, we can remove \( v_2 \) and merge \( e \) and \( e' \) into a new edge \( e'' = (v_1, v_3) \), with \( \varphi(e'') = \varphi(e) \oplus \varphi(e') \). This transformation is called the \textit{ghosting} of the vertex. This edit cannot be done on the root. Its inverse transformation is called the \textit{splitting} of an edge.

A generalized dendrogram \( T \) can be edited to obtain another dendrogram, on which one can apply a new edit to obtain a third dendrogram and so on. One can think of this as composing two edits \( e_0, e_1 \) which are not defined on the same dendrogram, since the second edit is defined on the already edited dendrogram. This is what we mean by composition of edits. Any finite composition of edits is called \textit{edit path}. The notations we use are functional notations, even if the edits are not operators, since an edit is not defined on the whole space of dendrograms but on a single dendrogram; for example \( e_1 \circ e_0(T) \) means that \( T \) is edited with \( e_0 \), and then \( e_0(T) \) with \( e_1 \).

4.1.1 Merge Trees

An example of editable space is \((\mathbb{R}_{\geq 0}, +, 0)\) with the metric given by \( d(x, y) = |x - y| \). Given a merge tree \( T \), with its height function \( h_T \), upon replacing \( h_T \) with a weight function \( w_T \), such that for each edge \( e = (v, v') \in E_T \), \( w_T(v) = h_T(v') - h_T(v) \), and \( w_T(r_T) = h_T(r_T) \), we retrieve a framework to work with merge trees. In fact, by the monotonicity of \( h_T \), \( w_T \) is a proper multiplicity function. Note that the “map” \( h_T \mapsto w_T \) can be naturally inverted, so that from any weight function \( w_T : V_T \rightarrow \mathbb{R}_{\geq 0} \) we can recover a unique height function \( h_T \). In Figure 4 we can see examples of edit operations for such dendrograms.

4.1.2 Curves in Editable Spaces

Consider an editable space \( X \). Then the space of functions \( \{ f : \mathbb{R} \rightarrow X \mid \int_{\mathbb{R}} d(f(t), 0)dt < \infty \} \) induces an editable space. The monoid operation is defined pointwise: \( (f \oplus g)(t) := f(t) \oplus g(t) \) and a pseudo-metric is given by \( d(f, g) := \int_{\mathbb{R}} d(f(t), g(t))dt \). If then all functions which differ on zero measure sets (with respect to the Lebesgue measure on \( \mathbb{R} \)) are identified with an equivalence relationship, this becomes a metric space. The function \( d \) is always non-negative, so if properties (P3) and (P4) hold pointwise, then they hold also for integrals.

Figure 4: (a)→(e) form an edit path made by one deletion, one ghosting and a final shrinking, between merge trees.
For instance we verify (P3) as follows:

\[ d(f \oplus g, 0) = \int_{\mathbb{R}} d(f(t) \oplus g(t), 0) dt = \int_{\mathbb{R}} d(f(t), 0) + d(g(t), 0) dt = d(f, 0) + d(g, 0) \]

We name such editable space \( L_1(\mathbb{R}, X) := \left\{ f : \mathbb{R} \to X \mid \int_{\mathbb{R}} d(f(t), 0) dt < \infty \right\} / \sim. \)

### 4.1.3 Finite Products of Spaces

Consider two editable spaces \( X \) and \( X' \), that is \((X, \odot, 0_X)\) and \((X', \odot, 0_{X'})\) satisfying properties (P1)-(P4). Then \((X \times X', \oplus, (0_X, 0_{X'}))\) is an editable space, with \( \oplus \) being the component-wise operations \( \odot \) and \( \odot \), and the metric \( d \) on \( X \times X' \) being the sum of the component-wise metrics of \( X \) and \( X' \).

### 4.2 Order 2 vertices

When deleting an edge in a merge tree, the father of the deleted vertex becomes an order two vertex. Such vertex carries no topological information, since the merging that the point was representing, is no more happening (was indeed deleted). This fact gives the intuition that order 2 vertices (apart from the root) are completely irrelevant and must not be taken into account when comparing dendrograms: they appear when nothing interesting happens topologically. Thus, informally speaking, dendrograms “equal” up to order two vertices, should be considered equal. This means that the isomorphism classes considered in Definition 5 and Definition 7 might be “too small” in the sense that one would like to regard as equivalent bigger sets of merge trees or dendrograms. Thanks to the definitions in Section 4.1 we can formalize the meaning of “equal up to order 2 vertices”.

**Definition 11** Generalized dendrograms are equal up to order 2 vertices if they become isomorphic after applying a finite number of ghostings or splittings.

Definition 11 induces an equivalence relationship. The set of generalized dendrograms inside \((D, X \coprod Y)\) that we want to treat as equal are exactly the equivalence classes given by Definition 11. We call \((D_2, X \coprod Y)\) the space of equivalence classes of dendrograms in \((D_2, X \coprod Y)\), equal up to order 2 vertices.

**Definition 12** A pseudo-metric on \((D, X \coprod Y)\) which induces a non trivial pseudo-metric on \((D_2, X \coprod Y)\) is called topologically stable.

In other words a topologically stable pseudo-metric for dendrograms is a (non trivial) pseudo-metric which identifies dendrograms which are equivalent up to order 2 vertices.

### 4.3 Edits and Costs

Now we associate to every edit a cost, that is a length measure in the space \((D, X \coprod Y)\). The costs of the edit operations are defined as follows:

- if, via shrinking, an edge goes from multiplicity \( x \) to multiplicity \( y \), then the cost of such operation is \( d(x, y) \). This holds both for shrinkages happening in \( X \) and for shrinkages done on the root, which take place in \( Y \);
- for any deletion/insertion of an edge with multiplicity \( x \), the cost is equal to \( d(x, 0) \);
- the cost of ghosting operations is \( |d(x \oplus y, 0) - d(x, 0) - d(y, 0)| = 0 \).
Remark 4 With such costs, it would be natural to try to define a family of metrics indexed by integers \( p \geq 1 \) by saying that the costs of compositions are the \( p \)-th root of sum of the costs of the edit operations to the \( p \)-th power. But one immediately sees that for any \( p > 1 \) this has no hope of being a meaningful pseudo metric. In fact consider the case of merge trees (with multiplicity given by the weight function \( w_T \)) and in particular a tree made by a segment of length 1. The cost of shrinking it 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 trees would be in the same equivalence class of the tree with no branches.

Definition 13 Given two dendrograms \( T \) and \( T' \) in \((D, X \coprod Y)\), define:

- \( \Gamma(T, T') \) as the set of all finite edit paths between \( T \) and \( T' \);
- \( \text{cost}(\gamma) \) as the sum of the costs of the edits for any \( \gamma \in \Gamma(T, T') \);
- the dendrogram edit distance as:
  \[
  d_E(T, T') = \inf_{\gamma \in \Gamma(T, T')} \text{cost}(\gamma)
  \]

By definition the triangle inequality and symmetry must hold, but, up to now, this edit distance is intractable; one would have to search for all the possible finite edit paths which connect two dendrograms in order to find the minimal ones. And from Remark 4 we see that is not even obvious that \( d_E(T, T') > 0 \) for some dendrograms. However, since the cost of ghostings is zero, it is clear that \( d_E \) induces a pseudo-metric on classes of dendrograms up to order two vertices.

Remark 5 From the definition of the edit operations and their costs, it is clear that the roots play little to no role in editing a dendrogram: if one wants to turn a dendrogram \( T \) in a dendrogram \( T' \), he has no choice but shrinking the root \( r_T \) to match the multiplicity of \( r_{T'} \). So there are no degrees of freedom involved. For this reason, from now on, to lighten the notation, we simply forget \( Y \) and the multiplicity of the root and just focus on the weight space \( X \). Moreover we always assume to be working in an editable dendrogram space.

4.4 Mappings

Now we introduce a fundamental tool, called mapping, that, by parametrizing certain sets of edit paths, makes \( d_E \) computable and its properties more readily available. The idea of mappings is not novel Tai (1979) and often it is a the key ingredient both for proofs and calculations in Tree Edit Distances Hong et al. (2017). From now on we suppose that in the set of vertices of any dendrogram there are not the letters “D” and “G” (which are used to indicate “deletion” and “ghosting”). Recall that \( E_T \) identifies the vertices \( V_T - \{r_T\} \).

A mapping between \( T \) and \( T' \) is a set \( M \subset (E_T \cup \{D, G\}) \times (E_{T'} \cup \{D, G\}) \) with the following properties:

(M1) consider the projection of the Cartesian product \((E_T \cup \{D, G\}) \times (E_{T'} \cup \{D, G\}) \to (E_T \cup \{D, G\}) \); we can restrict this map to \( M \) obtaining \( \pi_T : M \to (E_T \cup \{D, G\}) \). The maps \( \pi_T \) and \( \pi_{T'} \) are surjective on \( E_T \subset (E_T \cup \{D, G\}) \) and \( E_{T'} \subset (E_{T'} \cup \{D, G\}) \) respectively;
(M2) $\pi_T$ and $\pi_{T'}$ are injective;

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

(M4) if $(a, G)$ (or $(G, a)$) is in $M$, let $b_1, \ldots, b_n$ be the children of $a$. Then there is one and only one $i$ such that for all $j \neq i$, for all $x \in V_{\text{sub}(b_j)}$, we have $(x, D) \in M$ (respectively $(D, x)$); and there is one and only one $c$ such that $c = \max\{x' \in \text{sub}(b_i)|(x', y) \in M \text{ for any } y \in V_{T'}\}$.

Conditions (M1)-(M2) are asking that every point is assigned to one and only one “transformation”; (M3) ensures that the associations induced by $M \cap (V_T \times V_{T'})$ respect the tree structures of $T$ and $T'$; lastly (M4) means that, once all vertices $v$ appearing in the couples $(v, D)$ or $(D, v)$ in $M$ are deleted, the points which are coupled with $G$ (that is $(a, G)$ or $(G, a)$) are all vertices of order two and therefore they can be ghosted.

Using $M$, we can parametrize a set of edit paths in the dendrogram space, starting from $T$ and ending in $T'$, which are collected under the name $\gamma_M$. The properties of $M$ allow the definition the following edit paths:

- $\gamma_d^T$ is a path made by the deletions to be done in $T$, that is, the couples $(v, D)$, executed in any order. So we obtain $T_d^M = \gamma_d^T(T)$, which, instead, is well defined and not depending on the order of the deletions.

- One then proceeds with ghosting all the vertices $(v, G)$ in $M$, in any order, getting a path $\gamma_g^T$ and the dendrogram $T_M := \gamma_g^T \circ \gamma_d^T(T)$.

- Since all the remaining points in $M$ are couples, the two dendrograms $T'_M$ (defined in the same way as $T_M$, but starting from $T'$) 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 $\gamma_s^T$.

By definition:

$$\gamma_s^T \circ \gamma_g^T \circ \gamma_d^T(T) = T'_M,$$

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 $\gamma_M$ the set of all possible edit paths:

$$(\gamma_d^T)^{-1} \circ (\gamma_g^T)^{-1} \circ \gamma_s^T \circ \gamma_g^T \circ \gamma_d^T.$$

obtained by changing the order in which the edit operations are executed inside $\gamma_d$, $\gamma_g$ and $\gamma_s$. Observe that, even if $\gamma_M$ is a set of paths, its cost is well defined:

$$\text{cost}(M) := \text{cost}(\gamma_M) = \text{cost}(\gamma_d^T) + \text{cost}(\gamma_g^T) + \text{cost}(\gamma_s^T).$$

See Figure 5 for an example of a mapping between merge trees.

Before moving on, we fix some notation and call $\text{Mapp}(T, T')$ the set of all mappings between $T$ and $T'$. This set is never empty, in fact $M = \{(v, D) : v \in E_T\} \cup \{(D, v') : v' \in E_{T'}\}$ is always a mapping between $T$ and $T'$. In other words one can always delete all the edges of a generalized dendrogram, and then insert all the edges of the other.
Figure 5: The edit path in Figure 4 is represented by the mapping $M = \{(B,D),(C,G),(A,A'),(R,R')\}$ between $T$ and $T'$. Figure 4b represents $T_dM$, Figure 4d presents $T_M$; in this case $T'_M = T'$.

**Theorem 1 (Main Theorem)** Given two generalized dendrograms $T$ and $T'$, for every finite edit path $\gamma$, exists a mapping $M \in \text{Mapp}(T,T')$ such that $\text{cost}(M) \leq \text{cost}(\gamma)$.

A first Corollary immediately follows.

**Corollary 1** Since $\text{Mapp}(T,T')$ is a finite set we have the following well defined pseudo-metric:

$$d_E(T,T') = \inf\{\text{cost}(\gamma) | \gamma \in \Gamma(T,T')\} = \min\{\text{cost}(M) | M \in \text{Mapp}(T,T')\}$$

which we will refer to as the edit distance between $T$ and $T'$.

A second Corollary is obtained observing that, if a mapping has cost equal to zero, then it must contain only ghostings.

**Corollary 2** Given $T$ and $T'$ dendrograms, $d_E(T,T') = 0$ if and only if $T$ and $T'$ are equal up to order 2 vertices. In other words $d_E$ is a metric for generalized dendrograms considered up to order 2 vertices.

**Remark 6** If we compare the definitions and the results carried out in this section, with the ones in Di Fabio and Landi (2012, 2016); Bauer et al. (2016), we can recognize the different perspectives with which the different edit distances have been developed: in the cited works, the authors are more focused on transformations of the base topological space, while we are focused on local modifications of dendrograms. In fact, the available edit operations are different: in Di Fabio and Landi (2012, 2016); Bauer et al. (2016) there are six kinds of edits to be done on edges, along with their inverses, which avoid having to deal with the removal of single vertices, situation which, instead, we treat with ghostings. Moreover, even in the case of merge trees, there are some edits in Di Fabio and Landi (2012, 2016); Bauer et al. (2016) which cannot be seen as modifications involving just a single vertex, and this makes difficult to employ something like the mappings as we define, since a mapping is solely based on the fact that we can completely encode any edit with a couple of elements: being it two vertices (of different dendrograms) or a vertex and a letter (either “D” or “G”).

We can appreciate the different behaviours of the metrics looking at the stability results with respect to uniform convergence of functions: comparing for instance Theorem 28 in Di Fabio and Landi (2016) and Theorem 1 in [Paper FDA] we see that the behaviour of the two metrics is very different, with the metric in Di Fabio and Landi (2012, 2016); Bauer et al. (2016) being more stable with respect to sup norm between functions.
5. Back to Vector Spaces Filtrations

At this point we go back to fixed basis vector spaces filtrations and we employ the machinery defined in Section 3.2 and Section 4 to extract information from such families of vector spaces. We want to define a pipeline to build proper multiplicity functions with values in an editable space, obtaining dendrograms which in some sense generalize merge trees. The precise meaning of “generalizing” merge trees is the following: starting from two fixed basis vector spaces filtrations, if we obtain generalized dendrograms which are isomorphic, we ask that also their merge trees are isomorphic.

Consider \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \) fixed basis vector spaces filtration with maps \( \psi_t' : A_t \rightarrow A_{t'} \). An information function for \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \) is a function \( \Theta : B \rightarrow X \) such that \( a_t \subset B \) for all \( t \in \mathbb{R} \), with \( X \) being an editable space. One should think at \( \Theta \) as a kind of “sufficient statistic” to be extracted from the fixed basis \( a_t \): it is the information one wants to extract from the elements of the basis at time \( t \) along the chosen filtration and it must be carefully designed depending on the aim of the analysis.

Consider the merge tree obtained from \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \) with its tree structure \( T \) and its height function \( h_T \). Moreover consider an edge \( e = (v, v') \), with \( t_i = h_T(v) \) and \( t_j = h_T(v') \). We know by construction that there is a basis element, which we call \( a_{t_i} \), associated to \( v \), which is such that \( (\psi_{t_i}')^{-1}(\psi_{t_i}'(a_{t_i}')) = \{a_{t_i}'\} \) for all \( t' \in [t_i, t_j] \). We define the multiplicity function \( \varphi_T^\Theta(e) \) so that \( \varphi_T^\Theta(e)(t') = \Theta(\psi_{t_i}'(a_{t_i}')) \) for all \( t' \in [t_i, t_j] \), and \( \varphi_T^\Theta(e)(t') = 0 \) otherwise.

**Definition 14** Given \( S \) a set of fixed basis vector spaces filtrations and \( X \) editable space, an \( S \)-proper information function \( \Theta : B \rightarrow X \), with \( X \) editable space, is a function such that:

- for every \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \in S \), \( a_t \subset B \)
- \( \varphi_T^\Theta \) is a proper multiplicity function with values in \( L_1(\mathbb{R}, X) \) for every \( T \in S \)
- if \( (T, \varphi_T^\Theta) \) and \( (T', \varphi_{T'}^\Theta) \) obtained from two elements of \( S \) are isomorphic as generalized dendrograms, then the merge trees \( (T, h_T) \) and \( (T', h_{T'}) \) associated to the same filtrations are isomorphic as merge trees.

Since \( \varphi_T^\Theta(e) \) is by construction zero outside a compact interval, there are many natural conditions to be required for \( \Theta \) so that \( \varphi_T^\Theta \) is a multiplicity function with values in the editable space \( L_1(\mathbb{R}, X) \). For instance we could ask that \( d(\Theta(\cdot), 0) \) is bounded by some positive constant. Similarly, if we want \( \varphi_T^\Theta \) to be a proper multiplicity function, it is enough that \( \Theta(\psi_{t_i}'(a_{t_i}')) = 0 \) only for \( t' \) belonging to measure zero subsets (wrt Lebesgue measure) of \([t_i, t_j] \). Both conditions, as well as the last one requested by Definition 14, which, again, has to do with the zeros of the function \( \Theta \), can be attained without much effort in many interesting situations, as shown in the upcoming examples.

Note that the operation of ghosting a vertex, with this structure, assumes a quite natural form. Suppose we have two edges \( e = (v_1, v_2) \) and \( e' = (v_2, v_3) \), with \( v_1 < v_2 < v_3 \) and \( v_2 \) of order two. We have \( \varphi_T^\Theta(e) \) with support on \([t_{v_1}, t_{v_2}]\) and \( \varphi_T^\Theta(e') \) with support on \([t_{v_2}, t_{v_3}]\), with \( t_{v_2} \) being \( h_T(v) \). If we ghost \( v_2 \) obtaining \( e'' = (v_1, v_3) \), then \( \varphi(e'') = \varphi_T^\Theta(e) + \varphi_T^\Theta(e') \) is supported on \([t_{v_1}, t_{v_3}]\) and is such that \( \varphi_T^\Theta(e'')(t) = \varphi_T^\Theta(e)(t) \) on \([t_{v_1}, t_{v_2}]\) and \( \varphi_T^\Theta(e'')(t) = \varphi_T^\Theta(e')(t) \) on \([t_{v_2}, t_{v_3}]\). Which means that we track down the information collected by \( \Theta \) as if \( v_2 \) did not exist.
Remark 7 From Definition 1, it is clear that the values $t^-$ and $t^+$ are not unique for a fixed basis vector spaces filtration. When building merge trees this is not a concern, because all relevant topological changes happen between the minimum and the maximum critical values. When tracking down some kind of information with $\Theta$, it may happen, for instance, that it is valuable to collect values of $\Theta(a_i^1)$ for $t > \max_{i=1,...,n}t_i$, and thus one can fix $t^-$ and $t^+$ according to the aim of the analysis. In particular $t^- = \min_{i=1,...,n}t_i$ is a sensible choice in most cases, since if $t^- < \min_{i=1,...,n}t_i$, then $a_i^1 = \emptyset$. Instead, fixing $t^+ > \max_{i=1,...,n}t_i$ can be useful for instance when considering sublevel set filtrations of functions, as in Section 5.3. In this case, one should consider $t^+$ as being another critical value for the filtration, so that information is tracked with $\Theta$ up to $t^+$.

Remark 8 The one presented in this Section is not the only way to design multiplicity functions which track some kind of information along a fixed basis vector spaces filtration, but for the aforementioned reasons and, as motivated by the following examples, it is a quite natural and flexible framework to consider.

5.1 Merge Trees
Consider the special case of all information functions for any filtration being the constant function $\Theta : \text{Sets} \rightarrow \mathbb{R}_{\geq 0}$, such that $\Theta(s) = 1$ for all sets s, that is $\varphi^\Theta_T(e) = \chi_{(t_i,t_j)}$ for an edge $e$ spanning from height $t_i$ to height $t_j$, with $\chi_{I}$ being the characteristic function over the set $J \subseteq \mathbb{R}$. If two dendrograms $(T, \varphi^\Theta_T)$ and $(T', \varphi^\Theta_{T'})$ obtained with such information function are isomorphic, then they must have isomorphic tree structures and the associated fixed basis vector spaces filtrations must share all critical values. Otherwise, for any isomorphism of dendrograms $\eta : E_T \rightarrow E_{T'}$, the functions $\varphi^\Theta_T(e)$ and $\varphi^\Theta_{T'}(e')$, for at least one couple of edges such that $\eta(e) = e'$, are characteristic functions over different intervals of the form $[a, b]$ and thus their distance in $L_1(\mathbb{R}, \mathbb{R}_{\geq 0})$ is positive. This immediately implies that $\Theta = 1$ is a proper family of information functions with values in $\mathbb{R}_{\geq 0}$ (for all fixed basis vector spaces filtrations) with which one can recover the information contained in merge trees: isomorphism between dendrograms implies isomorphism of merge trees and viceversa. These bijections between merge trees and dendrograms induced by $\Theta = 1$, however, are not isometries wrt the distance $d_E$.

Example Consider for instance the functions $f = ||x| - 1|$ and $g = ||x| - 1| + 1$, both defined on the interval $[-2, 2]$. See Figure 6a. Let $A_t = H_0(f^{-1}((-\infty,t)))$ and $B_t = H_0(g^{-1}((\infty, t)))$, both with the bases given by path connected components.

For the sequence $\{(A_t,a_t)\}_{t \in \mathbb{R}}$ we can fix $t^- = 0$ and $t^+ = 1$. For any $t \in [t^-, t^+]$, $a_t = \{a_1^t, a_1^{t-}\}$, with $a_1^t = [1 - t, 1 + t]$ and $a_1^{t-} = [-1 - t, -1 + t]$. The critical points are $t_0 = t^-$ and $t_1 = t^+$. Thus the merge tree $(T, h_T)$ associated to $\{(A_t,a_t)\}_{t \in \mathbb{R}}$ has a tree structure given by a root with two children being the leaves. We represent this with the vertex set $\{v_1, v_{-1}, r_T\}$ and edges $e_1 = (v_1, r_T)$ and $e_2 = (v_{-1}, r_T)$. The height function has values $h_T(v_1) = h_T(v_{-1}) = t^- = 0$ and $h_T(r_T) = t^+ = 1$. See Figure 6b. The multiplicity function $\varphi^\Theta_T$, instead, is defined as follows: $\varphi^\Theta_T(e_1) = \varphi^\Theta_T(e_2) = \chi_{[0,1]}$ and $\varphi^\Theta_T(r_T) = \chi_{(1)}$.

In a similar fashion the sequence $\{(B_t,b_t)\}_{t \in \mathbb{R}}$ has $t^- = 1$ and $t^+ = 2$. For any $t \in [t^-, t^+]$, $b_t = \{b_1^t, b_1^{t-}\}$, with $b_1^t = [1 - t, 1 + t]$ and $b_1^{t-} = [-1 - t, -1 + t]$. The critical points are $t_0 = t^-$ and $t_1 = t^+$. Again the merge tree $(T', h_{T'})$ associated to $\{(B_t,b_t)\}_{t \in \mathbb{R}}$ has a tree structure given by a root with two children being the leaves. We fix the vertex set $\{v_1, v_{-1}, r_{T'}\}$ and the edges $e_1' = (v_1, r_{T'})$ and $e_2' = (v_{-1}, r_{T'})$. The height function has values $h_{T'}(v_1) = h_{T'}(v_{-1}) = t^- = 1$ and $h_{T'}(r_{T'}) = t^+ = 2$. The multiplicity function $\varphi^\Theta_{T'}$, instead, is defined as follows: $\varphi^\Theta_{T'}(e_1') = \varphi^\Theta_{T'}(e_2') = \chi_{[1,2]}$ and $\varphi^\Theta_{T'}(r_{T'}) = \chi_{(2)}$. 

17
Consider the mapping $M = \{(v_1, w_1), (v_{-1}, w_{-1}), (r_T, r_{T'})\}$. Its easy to check, because of the small size of the sets $V_T$ and $V_{T'}$, that this is a minimizing mapping both for the merge trees $(T, w_T)$ and $(T', w_{T'})$, and for the dendrograms $(T, \varphi^\Theta_T)$ and $(T', \varphi^\Theta_{T'})$.

The cost of $M$ for the merge trees is:

$$\sum_{i=-1,1} |w_T(v_i) - w_{T'}(v_i)| + |w_T(r_T) - w_{T'}(r_{T'})| = 0 + 0 + 1$$

while for the dendrograms is

$$\sum_{i=-1,1} ||\varphi^\Theta_T(v_i) - \varphi^\Theta_T(w_i)|| + ||\varphi^\Theta_T(r_T) - \varphi^\Theta_{T'}(r_{T'})|| = ||\chi_{[0,2]}|| + ||\chi_{[0,2]}|| + 0 = 4$$

5.2 Clustering Dendrograms

Consider now the case of an agglomerative clustering dendrogram $(T, h_T)$ built on a finite set $\{x_1, \ldots, x_n\}$ with some linkage rule. We can look at clustering dendrograms as the merge trees associated to the filtration given by $t^- = 0$ and for $t \geq 0$, $A_t$ is the vector space generated by the clusters obtained by cutting the dendrogram at height $t$. The basis is the one given by $a_t = \{\{x_1, \ldots, x_{1,n_1}\}, \ldots, \{x_{k,1}, \ldots, x_{k,n_k}\}\}$ where $\{x_{j,1}, \ldots, x_{j,n}\}$ is the $j$-th cluster, which has cardinality $n_j$, obtained by cutting the dendrogram at height $t$.

We call clustering filtrations all the fixed basis vector spaces filtrations obtained from agglomerative clustering dendrograms following the procedure just outlined.

A sensible information that one may want to track down along $\{(A_t, a_t)\}_{t \in \mathbb{R}}$ is the cardinality of the clusters. Thus we can take $\Theta : FSets \rightarrow \mathbb{R}_{\geq 0}$, defined on all finite sets $(FSets)$, such that $\Theta(\{x_{j,1}, \ldots, x_{j,n}\}) = n_j$. Clearly, for a clustering filtration on $n$ elements, $1 \leq \Theta \leq n$ and so $\varphi^\Theta_T(e) \in L_1(\mathbb{R}, \mathbb{R}_{\geq 0})$. Note that $\varphi^\Theta_T(e) = m\chi_{[t_i,t_j]}$, for some positive cardinality $m$ and some critical points $t_i, t_j$. Thus, $\Theta(\{x_{j,1}, \ldots, x_{j,n}\}) = n_j$ is a proper family of information functions for all clustering filtrations.

**Example** Consider the finite set $\{v_{-1} = -1, v_0 = 0, v_2 = 2\}$ and build the single linkage hierarchical dendrogram of such set using the euclidean metric. The filtration obtained from such hierarchical dendrogram is $A_t \simeq \mathbb{R}^3$ for $t \in [0,1)$, $A_t \simeq \mathbb{R}^2$ for $t \in [1,2]$ and $A_t \simeq \mathbb{K}$ for $t \geq t^+ = 2$. The fixed bases are $a_t = \{\{v_{-1}\}, \{v_0\}, \{v_2\}\}$ for $t \in [0,1)$, $a_t = \{\{v_{-1}, v_0\}, \{v_2\}\}$ for $t \in [1,2]$ and $a_t = \{\{v_{-1}, v_0, v_2\}\}$ for $t \geq t^+ = 2$. The associated merge tree $(T, h_T)$ - see Figure 6c - can be represented with the vertex set $V_T = \{\{v_{-1}\}, \{v_0\}, \{v_2\}, \{v_{-1}, v_0\}, r_T\}$. The leaves are $\{v_{-1}\}, \{v_0\}$ and $\{v_2\}$; the children of $\{v_{-1}, v_0\}$ are $\{v_{-1}\}$ and $\{v_0\}$, and the ones of $r_T$ are $\{v_{-1}, v_0\}$ and $\{v_2\}$. The height function $h_T$ is given by $h_T(\{v_i\}) = 0$ for $i = -1, 0, 2$, $h_T(\{v_{-1}, v_0\}) = 1$ and $h_T(r_T) = 2$. The multiplicity function $\varphi^\Theta_T$ is thus the following: $\varphi^\Theta_T(\{v_i\}) = \chi_{[0,1]}$ for $i = -1, 0$, $\varphi^\Theta_T(\{v_2\}) = \chi_{[0,2]}$, $\varphi^\Theta_T(\{v_{-1}, v_0\}) = 2\chi_{[1,2]}$ and $\varphi^\Theta_T(r_T) = 3\chi_{[2]}$. See Figure 6d.

5.3 Dendrograms of Functions

Now turn to the situation showcased in Section 2.2. Consider $U \subset \mathbb{R}^m$ convex bounded open set, with $\overline{U}$ being its topological closure, and let $L$ be the Lebesgue measure in $\mathbb{R}^m$. Let $f : \overline{U} \rightarrow \mathbb{R}$ be a tame Chazal et al. (2016) continuous function. Consider the sublevel set filtration $A_t = H_0(f^{-1}((-\infty, t]))$ with $a_t = \{U^t_1, \ldots, U^t_n\}$ being the path connected components of $f^{-1}((-\infty, t])$. Here the tameness condition is simply asking that $a_t$ is a finite set for every $t$. Call $\psi^\Theta_t$ the functions $\psi^\Theta_t : A_t \rightarrow A_{t'}$. We choose as information function $\Theta = L$, that is: $\Theta(U^t_i) = L(U^t_i)$. We can set $t^- = \inf_t f$ and $t^+ = \sup_t f$. Let $(T, h_T)$ being the merge tree representing $\{(A_t, a_t)\}_{t \in \mathbb{R}}$. and $\varphi^\Theta_T$ the
associated multiplicity function. Being $f$ continuous, for and edge $e = (v, v') \in E_T$ spanning from height $h_T(v) = t_i$ to $h_T(v') = t_j$, now we prove that $\varphi_T^\Theta(e) > 0$ on $(t_i, t_j)$. We now that $v$ is associated to a connected component $U^{i_1}_{k_1}$ for some $k$. If $v$ represents the merging of two or more path connected components $U_{k_1}^{i_1-\varepsilon}$ and $U_{k_2}^{i_2-\varepsilon}$, for some small $\varepsilon > 0$, with $\mathcal{L}(U_{k_1}^{i_1-\varepsilon}), \mathcal{L}(U_{k_2}^{i_2-\varepsilon}) > 0$, then, since $U_{k_1}^{i_1-\varepsilon} \cup U_{k_2}^{i_2-\varepsilon} \subset U_{k}^{i}$, we have $\mathcal{L}(U_k^{i}) > 0$. Thus if we prove the statement for $v$ leaf, we are done.

So, suppose $v$ is a leaf and consider $x_0 \in U^{i_1}_{k_1}$. We know $f(x_0) = t_i$. By the continuity of $f$, there exists $\delta > 0$ such that if $|x - x_0| < \delta$, then $f(x_0) \leq f(x) < f(x_0) + \varepsilon$. Since $\{x \in U^{i}||x - x_0|| < \delta\}$ is convex, then it is contained in $\psi^{i_1+\varepsilon}(U_{k_1}^{i_1})$. Moreover, since it contains the non-empty open set $\{x \in U^{i}||x - x_0|| < \delta\}$, we have $\mathcal{L}(\psi^{i_1+\varepsilon}(U_{k}^{i})) > 0$ for every $\varepsilon > 0$. As a consequence, supp$(\varphi_T^\Theta(e)) = [t_i, t_j]$. Putting the pieces together this means that $\Theta = \mathcal{L}$ is a proper information function for sublevel set filtrations obtained from real valued, bounded, tame functions defined over the closure of convex, bounded, open subsets of $\mathbb{R}^n$.

**Example** Consider again the function $f = ||x| - 1|$ defined on the interval $[-2, 2]$. Let $A_t = H_0(f^{-1}((t, t)])$ with the bases given by path connected components. The Example in Section 5.1 shows how to obtain the merge tree $(T, h_T)$ associated to the sequence $\{(A_t, a_t)\}_{t \in \mathbb{R}}$. Using the same notation of Section 5.1, now we obtain the multiplicity functions $\varphi_T^\Theta(e_1)$, with $\Theta$ being the Lebesgue measure as just discussed.

The function $\varphi_T^\Theta(e_1) = |1 + t - 1 + t| = 2t$ for $t \in [0, 1]$, and 0 otherwise. Clearly $\varphi_T^\Theta(e_1) = \varphi_T^\Theta(e_2)$. Lastly $\varphi_T^\Theta(r_T) = 4\chi(2)$.

**Example** Lastly we consider the following functions defined on $[-1, 2]$:

- $f(x) = |x - 1| + \varepsilon$ if $x \geq 0$ and $f(x) = |2x - 1|$ if $x < 0$; while $g(x) = |x - 1|$ if $x \geq 0$ and $g(x) = |2x - 1| + \varepsilon$ if $x < 0$ for a fixed $\varepsilon > 0$; as in Figure 6e. Let $(T, h_T)$ and $(T', h_{T'})$ be the merge trees associated to the sublevel set filtrations of $f$ and $g$; moreover let $\varphi_T^\Theta$ and $\varphi_T'^\Theta$ the two respective multiplicity functions with $\Theta$ being the Lebesgue measure on $\mathbb{R}$. Note that $||f - g|| = 2\varepsilon$. The local minima of the functions are the points $\{-0.5, 1\}$, with $f(-0.5) = 0$, $f(1) = \varepsilon$, $g(-0.5) = \varepsilon$ and $g(1) = 0$. Thus the merge trees have isomorphic tree structures: we represent $T$ with the vertex set $\{v_{-0.5}, v_1, r_T\}$ and edges $\{(v_{-0.5}, r_T), (v_1, r_T)\}$; and $T'$ with vertices $\{v_{-0.5}, v_1, r_{T'}\}$ and edges $\{(v_{-0.5}, r_{T'}), (v_1, r_{T'})\}$. The height functions are the following: $h_T(v_{-0.5}) = 0$, $h_{T'}(v_{-0.5}) = \varepsilon$, $h_T(v_1) = \varepsilon$, $h_{T'}(v_1) = 0$ and $h_T(r_T) = h_{T'}(r_T) = 1 + \varepsilon$.

Lastly, the multiplicity functions (see Figure 6f) are given by: $\varphi_T^\Theta(v_{-0.5})(t) = t\chi_{[0, 1]} + \chi_{[1, 1+\varepsilon]}$ and $\varphi_T^\Theta(v_1)(t) = 2(t - \varepsilon)\chi_{[\varepsilon, 1+\varepsilon]}$, and $\varphi_T'^\Theta(v_{-0.5})(t) = (t - \varepsilon)\chi_{[\varepsilon, 1+\varepsilon]}$ and $\varphi_T'^\Theta(v_1)(t) = 2t \cdot \chi_{[0, 1]} + 2\chi_{[1, 1+\varepsilon]}$.

The zero-dimensional persistence diagram associated to $f$ (we name it $PD_0(f)$) is given by a point with coordinates $(0, +\infty)$, associated to the connected component $[-t/2 - 0.5, t/2 - 0.5]$ which is born at $t = 0$, and the point $(\varepsilon, 1 + \varepsilon)$, associated to the component $[1 - (t - \varepsilon), 1 + (t - \varepsilon)]$, born at level $t = \varepsilon$ and “dying” at level $t = 1 + \varepsilon$, due to the elder rule, since it merges an older component, being the other component born at a lower level.

For the function $g$, the persistence diagram $PD_0(g)$ is made by the same points, but the situation is in some sense “reversed”. In fact, the point $(0, +\infty)$ is associated to the connected component “centered” in 1, which is $[1 - t, 1 + t]$, and the point $(\varepsilon, 1 + \varepsilon)$, is associated to the component “centered” in 0.5, that is $[-(t - \varepsilon)/2 - 0.5, (t + \varepsilon)/2 - 0.5]$.

The consequence of this change in the associations between points and the components originating the points of the diagrams is that the information regarding the two components, end up being associated to very different spatial locations in the two diagrams: $(0, +\infty)$ and $(\varepsilon, 1 + \varepsilon)$. And this holds for every $\varepsilon > 0$. Thus it seems very hard
to design a way to “enrich” $PD_0(f)$ and $PD_0(g)$ with additional information, originating the “enriched diagrams” $D_f$ and $D_g$, respectively, and design a suitable metric $d$, so that $d(D_f, D_g) \to 0$ as $\varepsilon \to 0$.

Instead, if we consider the mapping $M = \{(v_{-0.5}, v_{-0.5}), (v_1, v_1), (r_T, r_{T'})\}$ we have $d_{\mathcal{E}}((T, \varphi^T_T), (T', \varphi^T_{T'})) \leq cost(M) = 3\varepsilon$. Thus it is very likely that some kinds of continuity/“stability” results, depending on the application, can be proven with our framework, while it seems much harder to do the same for persistence diagrams.

**Remark 9** In the previous Sections we presented three frameworks dealing with merge trees, clustering dendrograms and sublevel sets filtrations of functions. More general and complex frameworks can be defined, for instance we could consider suitable functions defined on Riemannian manifolds, with $\Theta$ being the Riemannian volume. Similarly, instead of taking information functions with values in $X = \mathbb{R}_{\geq 0}$, we could design functions with values in other editable spaces, such as $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$.

6. Decomposition Properties and Optimization Problems

In this section we develop some results and formulations needed to obtain the algorithm presented in Section 7. In Section 6.1 we prove the theoretical result that allows to recursively split up the calculations (following ideas found in Hong et al. (2017)) and in Section 6.2 such result is translated in terms of integer optimization problems.

6.1 Decomposition Result

Since $d_{\mathcal{E}}$ is topologically stable one can always suppose that a generalized dendrogram is given without order 2 vertices. Name $T_2$ the only representative without order 2 vertices inside the equivalence class of $T$. For notational convenience, from now on we suppose $T = T_2$ and $T' = T_2'$. To help us in the calculations define the following set of mappings: $\mathcal{M}(T, T') \subset \text{Mapp}(T, T')$ made by mappings $M$ such that $(v, G)$ or $(G, w)$ is in $M$ if and only if, respectively, $v \in V_T$ or $w \in V_{T'}$ is of order 2 after the deletions. The following Lemma then applies.

**Lemma 1**

$$\min\{\text{cost}(M) | M \in \text{Mapp}(T, T')\} = \min\{\text{cost}(M) | M \in \mathcal{M}(T, T')\}$$

In addition to that, we consider some particular subsets of $E_T \times E_{T'}$ which play a fundamental role from now on. Recall that, using $E_T \simeq V_T - \{r_T\}$, we can induce $\pi_T : E_T \times E_{T'} \to V_T$.

A set $M^* \subset E_T \times E_{T'}$ is in $\mathcal{C}^*(T, T')$ if:

(A1) the points in $\pi_T(M^*)$ form antichains in $V_T$ (and the same for $\pi_{T'}(M^*)$ in $V_{T'}$), with respect to the partial order given by father $> son$. This means that any two distinct vertices of $T$ (respectively of $T'$) which appear in $M^*$ are incomparable with respect to “>”;

(A2) the projections $\pi_T : M^* \to V_T$ and $\pi_{T'} : M^* \to V_{T'}$ are injective.

Consider now $M^* \in \mathcal{C}^*(T, T')$. Starting from such set of couples we build a set of edits which form a “partial” mapping between $T$ and $T'$. The meaning of “partial” will be made clear by Proposition 1. The main idea is that $M^*$ is used as a “dimensionality reduction tool”: instead of considering the problem of finding directly the optimal mapping between
(a) The functions \( f \) and \( g \) in the Example in Section 5.1.

(b) A representation of \((T, h_T)\), see the Example in Section 5.1.

(c) Single linkage clustering dendrogram referring to the Example in Section 5.2.

(d) In the context of the Example in Section 5.2, we see the sum of the multiplicity functions of vertices going from \( v_0 \) to the root \( r_T \): \( \varphi_T^\Theta(\{v_0\}) + \varphi_T^\Theta(\{v_0, v_{-1}\}) + \varphi_T^\Theta(r_T) \). The dotted lines represent critical points.

(e) The functions \( f \) and \( g \) in the Example in Section 5.3 with \( \varepsilon = 0.3 \).

(f) In the context of the Example in Section 5.3, we report \( \varphi_T^\Theta(v_1) \) and \( \varphi_T^\Theta(v_{-0.5}) \). The dotted lines represent critical points.

Figure 6: Plots referring to the examples in Section 5.
T and T’, we split up the problem in smaller subproblems, and put the pieces together using M*. To do that, some other pieces of notation are needed.

Let v ∈ E_T. One can walk on the (unweighted) graph of the tree-structure of T going towards any other vertex. For any v ∈ E_T, ζv is the shortest (in terms of vertices touched) graph-path connecting v to r_T. Note that this is the ordered set ζ_v = {v’ ∈ V_T | v’ > v}. Similarly, denote with ζ_y the shortest path on the graph of T connecting x and x’. By Property (A1), given x ∈ V_T ∩ π_T(M*), there exist x ≠ x such that:

\[ \tilde{x} = \min\{\cup_{(x’,y’)} \in M^*, x’ \neq x\}(\zeta_x \cap \zeta_{x’}) \]

And the same holds for y ∈ V_T ∩ π_T(M*).

With these bits of notation, given M* ∈ C*(T, T’), we now build the partial mapping α(M*) with the following rules. Consider v ∈ V_T:

1. if (v, w) ∈ M*, then (v, w) ∈ α(M*);
2. if there is not x ∈ V_T such that v < x or v > x, then (v, D) ∈ α(M*);
3. if there is x ∈ V_T such that v > x then (v, D) ∈ α(M*);
4. if there is x ∈ V_T such that v < x:
   a) if v ∈ ζ_x then (v, G) ∈ α(M*)
   b) if v < vi for some vi ∈ ζ_x = \{v_0 < v_1 < ... < v_n\} then (v, D) ∈ α(M*);
   c) if v < x no edit is associated to v.

**Remark 10** By Properties (A1) and (A2), the conditions used to build α(M*) are mutually exclusive. This means that each v ∈ V_T satisfies one and only one of the above conditions and so α(M*) is well defined.

The idea behind α(M*) is that, for all couples (x, y) ∈ M*, we want to turn ζ_x and ζ_y into single edges of the form (x, x) and (y, y) respectively, and then shrink one in the other. Informally speaking, α(M*) is a mapping that takes care of all the vertices in T and T’, a part from the vertices ∪_{(x,y) ∈ M*} {x’ ∈ E_T | x’ < x} and ∪_{(x,y) ∈ M*} {y’ ∈ E_T | y’ < y}, that is the vertices which are below x, for x ∈ V_T ∩ π_T(M*). For this reason we say that α(M*) is a partial mapping.

Consider T and T’ and M* ∈ C*(T, T’). We obtain from such dendrograms, respectively, the dendrograms T_M* and T’_M* by deleting all the points v < x and w < y for each couple (x, y) ∈ M*. With the following Proposition we give a formal description of the set of edits α(M*).

**Proposition 1** The set α(M*) is a mapping in M(T_M*, T’_M*).

Now we have all the pieces to obtain the following result.

**Theorem 2 (Decomposition)** Given T, T’ dendrograms:

\[ d_E(T, T’) = \min_{M* ∈ C*(T, T’)} \sum_{(x,y) ∈ M*} d_E(\text{sub}_T(x), \text{sub}_{T’}(y)) + \text{cost}(\alpha(M*)) \]  

(1)
6.2 Dynamical Integer Linear Programming problems

We want to use the Decomposition Theorem to write a dynamical, integer linear optimization algorithm to calculate \( d_E \): by translating Theorem 2 into a Integer Linear Programming (ILP) problem, one obtains a single step in a bottom-up procedure. Here we give a concise description of the optimization problems involved in calculating \( d_E \); a more detailed and comprehensive explanation can be found in the proof of Proposition 2.

6.2.1 Notation

Consider \( x \in V_T \) and \( y \in V_T \). Along with keeping the notation defined in Section 6.1, define \( T_x := \text{sub}_T(x) \) and \( T_y := \text{sub}_T(y) \), \( N_x := \text{dim}(T_x) = \#E_T \) and \( N_y := \text{dim}(T_y) = \#E_T \). In particular, given \( v \in V_{T_x} \), the sequence \( v_0 = v < v_1 < \ldots < r_T \) indicates the points in \( \zeta_v \). The same with \( w \in V_{T_y} \).

6.2.2 Setup and Variables

Suppose we already have \( W_{xy} \) which is a \( N_x \times N_y \) matrix such that \( (W_{xy})_{v,w} = d_E(T_v, T_w) \) for all \( v \in E_{T_x} \) and \( w \in E_{T_y} \). Note that:

- if \( x \) and \( y \) are leaves, \( W_{xy} = 0 \).
- if \( v, w \) are vertices of \( T_x, T_y \), then \( W_{vw} \) is a submatrix of \( W_{xy} \).

The function to be optimized is defined on the following set of binary variables: for every \( v \in E_{T_x} \) and \( w \in E_{T_y} \), for \( v_i \in \zeta_v, v_i < r_{T_x} \), and \( w_j \in \zeta_w, w_j < r_{T_y} \), take a binary variable \( \delta_{i,j}^{v,w} \). We write a constrained optimization problem such that having \( \delta_{i,j}^{v,w} = 1 \) means pairing the segments \( \zeta_v^{i+1} \) (that is, the sequence of edges which starts with \( (v, v_1) \) and ends with \( (v_i, v_{i+1}) \)) and \( \zeta_w^{j+1} \), and shrinking one in the other in the induced mapping.

6.2.3 Objective Function

Consider \( v \in E_{T_x} \) and interpret \( \delta_{i,j}^{v,w} = 1 \) as coupling the segments \( \zeta_v^{i+1} \) and \( \zeta_w^{j+1} \); then \( v \) is coupled with some \( w \in E_{T_y} \) if \( C(v) := \sum_{i,w,j} \delta_{i,j}^{v,w} = 1 \) and is ghosted if \( G(v) := \sum_{v' < v < v_{i+1}, w,j} \delta_{i,j}^{v',w} = 1 \). The vertex \( v \) is instead deleted if \( D(v) := 1 - C(v) - G(v) = 1 \). We introduce also the following quantities, which correspond to the cost of shrinking \( \zeta_v^{i+1} \) on \( \zeta_w^{j+1} \):

\[
\Delta_{i,j}^{v,w} = d(\oplus_{v' \in \zeta_v} \varphi_{T_x}(v'), \oplus_{w' \in \zeta_w} \varphi_{T_y}(w'))
\]

Use \( \delta \) to indicate the matrix of variables \( (\delta_{i,j}^{v,w})_{v,w,i,j} \). The function which computes the cost given by coupled points is therefore:

\[
F^C(\delta) := \sum_{v,w,i,j} \Delta_{i,j}^{v,w} \cdot \delta_{i,j}^{v,w}
\]

The contribution of deleted points is: \( F^D(\delta) - F^-(\delta) \), where

\[
F^D(\delta) := \sum_{v \in T_x} D(v) \cdot d(\varphi_{T_x}(v), 0) + \sum_{w \in T_y} D(w) \cdot d(\varphi_{T_y}(w), 0)
\]

and

\[
F^-(\delta) := \sum_{v \in T_x} C(v) \cdot ||\text{sub}_{T_x}(v)|| + \sum_{w \in T_y} C(w) \cdot ||\text{sub}_{T_y}(w)||
\]
where the “norm” of a tree $T$ is $||T|| = \sum_{e \in E_T} d(\varphi(e), 0)$.

Finally, one must take into account the values of $d_E(T_v, T_w)$, whenever $v$ and $w$ are coupled; this information is contained in $(W_{xy})_{v,w}$:

$$F^S(\delta) := \sum_{v,w}(W_{xy})_{v,w} \cdot (\sum_{i,j} \delta_{i,j}^{v,w})$$

6.2.4 Constraints

The last piece of the equation is given by the constraints which must be satisfied by the variables $\delta_{i,j}^{v,w}$. For each $v' \in V_{T_x}$ we call $\Phi(v') := \{ \delta_{i,j}^{v',w} | v' = v'' \in \zeta_{v,w}^{v'' + 1} \}$. In an analogous way we define $\delta(w')$ for $w' \in V_{T_y}$. Call $K$ the set of values of $\delta$ such that for each leaf $l$ in $V_{T_x}$:

$$\sum_{v' \in K} \sum \delta_{i,j}^{v',w} \leq 1 \quad (2)$$

and for each leaf $l'$ in $V_{T_y}$:

$$\sum_{w' \in K} \sum \delta_{i,j}^{v',w'} \leq 1 \quad (3)$$

Proposition 2 With the notation previously introduced:

$$d_E(T_x, T_y) = \min_{\delta \in K} F^C(\delta) + F^D(\delta) - F^-(\delta) + F^S(\delta) \quad (4)$$

Remark 11 A solution to Problem Equation (4) exists because the minimization domain is finite; it is not unique in general.

7. Bottom-Up Algorithm

In this section the results obtained in Section 6 are used to obtain the algorithm implemented to compute the metric $d_E$ between generalized dendrograms. Some last pieces of notation are introduced in order to describe the “bottom-up” nature of the algorithm.

Given $x \in V_T$, define $\text{len}(x)$ to be the number of vertices in $\zeta_x$ and $\text{len}(T) = \max_{v \in V_T} \text{len}(v)$.

Therefore, $\text{lvl}(x) = \text{len}(T) - \text{len}(x)$. Lastly, $\text{lvl}(n) = \{ v \in V_T | \text{lvl}(v) = n \}$

The key property is that: $\text{lvl}(x) > \text{lvl}(v)$ for any $v \in \text{sub}(x)$. Thus, if $W_{xy}$ is known for any $x \in \text{lvl}(n)$ and $y \in \text{lvl}(m)$, then for any $v, w$ in $V_T$ such that $\text{lvl}(v) < n$ and $\text{lvl}(w) < m$, $W_{vw}$ is known as well. With this notation we can write down Algorithm 1.

Algorithm 1: Bottom-Up Algorithm.

Result: $d_E(T, T')$

initialization: $N = \text{len}(T)$, $M = \text{len}(T')$, $n = m = 0$;

while $n \leq N$ or $m \leq M$ do

for $(x, y) \in V_T \times V_{T'}$ such that $\text{lvl}(x) \leq n$ and $\text{lvl}(y) \leq m$ do

Calculate $(W_{T_xT_y})_{x,y}$ solving Problem (4);

end

$n = n + 1$; $m = m + 1$;

end

return $(W_{T_xT_y})_{T_x, T_y}$

We end up with a result to analyze the performances of Algorithm 1 in the case of dendrograms with binary tree structures.
Proposition 3 Let $T$ and $T'$ be two generalized dendrograms with full binary tree structures with $\dim(T) := #E_T = N$ and $\dim(T') = M$.

Then $d_E(T, T')$ can be computed by solving $O(N \cdot M)$ ILP problems with $O(N \cdot \log(N) \cdot M \cdot \log(M))$ variables and $O(N + M)$ constraints.

Note that binary dendrograms are dense (with respect to $d_E$) in any generalized dendrogram space as long as for any $\varepsilon > 0$, there is $x \in (X, \oplus, 0)$ such that $d(x, 0) < \varepsilon$. So this is indeed a quite general result.

7.1 Example

Here we present in details the first steps of the Algorithm 1, used to calculate the distance between two merge trees.

We consider the following couple of merge trees. Let $(T, h_T)$ be the merge tree given by: $V_T = \{a, b, c, d, r_T\}$, $E_T = \{(a, d), (b, d), (a, r_T), (c, r_T)\}$ and $w_T(a) = w_T(b) = w_T(d) = 1$, $w_T(c) = 5$; the merge tree $(T', h_T')$ instead, is defined by: $V_{T'} = \{a', b', c', d', r_T\}$, $E_{T'} = \{(a', d'), (b', d'), (d', r_T), (c', r_T)\}$ and $w_{T'}(a) = 1$, $w_{T'}(b) = w_{T'}(c) = 2$ and $w_{T'}(d) = 3$.

Step: $n = m = 0$

This step is trivial since we only have couples between leaves, like $(a, a')$, which have trivial subtrees and thus $d_E(sub_T(a), sub_T(a')) = 0$.

Step: $n = m = 1$

The points $x \in V_T$ with $lvl_T(x) \leq 1$ are $\{a, b, c, d\}$ and the points $y \in V_{T'}$ with $lvl_{T'}(y) \leq 1$ are $\{a', b', c', d'\}$. Thus the couples $(x, y)$ which are considered are: $(d, d')$, $(d, a')$, $(d, b')$, $(d, c')$ and $(a, d')$, $(b, d')$, $(c, d')$. The couples between leaves, like $(a, a')$ have already been considered.

Couple: $(d, d')$ Let $T_d = sub_T(d)$ and $T_{d'} = sub_{T'}(d')$. The set of internal vertices are respectively $E_{T_d} = \{a, b\}$ and $E_{T_{d'}} = \{a', b'\}$. For each vertex $v < root$ in each subtree, where “root” stands for $d$ or $d'$, roots of $T_d$ and $T_{d'}$ respectively, we have $\zeta_v = \{v_0 = v, v_1 = root\}$. Thus, the binary variables we need to consider, are the following: $\delta_{0,0}^{a,a'}, \delta_{b,0}^{b,b'}, \delta_{b,0}^{b,a'}$ and $\delta_{b,0}^{b,b'}$. The quantities $\Delta_{v_i, v_{i'}}$ are given by: $\Delta_{0,0}^{a,a'} = 0$, $\Delta_{0,0}^{b,b'} = 1$, $\Delta_{0,0}^{b,a'} = 0$ and $\Delta_{0,0}^{b,b'} = 1$. Thus:

$$F_C(\delta) = 0 \cdot \delta_{0,0}^{a,a'} + \delta_{0,0}^{a,b'} + 0 \cdot \delta_{0,0}^{b,a'} + \delta_{0,0}^{b,b'}$$

While:

$$F_D(\delta) = (1 - \delta_{0,0}^{a,a'} - \delta_{0,0}^{a,b'}) \cdot 1 + (1 - \delta_{0,0}^{b,b'} - \delta_{0,0}^{b,a'}) \cdot 1 + (1 - \delta_{0,0}^{a,b'} - \delta_{0,0}^{b,a'}) \cdot 1 + (1 - \delta_{0,0}^{a,a'} - \delta_{0,0}^{a,b'}) \cdot 2$$

and:

$$F_{-}(\delta) = (\delta_{0,0}^{a,a'} + \delta_{0,0}^{a,b'}) \cdot 0 + (\delta_{0,0}^{b,a'} + \delta_{0,0}^{b,b'}) \cdot 0 + (\delta_{0,0}^{a,a'} + \delta_{0,0}^{a,b'}) \cdot 0 + (\delta_{0,0}^{b,a'} + \delta_{0,0}^{b,b'}) \cdot 0$$

and:

$$F_S(\delta) = \delta_{0,0}^{a,a'} \cdot 0 + \delta_{0,0}^{a,b'} \cdot 0 + \delta_{0,0}^{b,a'} \cdot 0 + \delta_{0,0}^{b,b'} \cdot 0$$

Lastly the constraints are:

$$\delta_{0,0}^{a,a'} + \delta_{0,0}^{a,b'} \leq 1; \delta_{0,0}^{b,a'} + \delta_{0,0}^{b,b'} \leq 1; \delta_{0,0}^{a,a'} + \delta_{0,0}^{b,a'} \leq 1; \delta_{0,0}^{a,b'} + \delta_{0,0}^{b,b'} \leq 1$$

A solution is given by $\delta_{0,0}^{a,a'} = \delta_{0,0}^{b,b'} = 1$ and $\delta_{0,0}^{b,a'} = \delta_{0,0}^{a,b'} = 0$, which entails $F_C(\delta) = 1$, $F_D(\delta) = 0$, $F_{-}(\delta) = 0$ and $F_S(\delta) = 0$ and $d_E(T_d, T_{d'}) = 1$. 25
**Couple:** $(d, a')$  Obviously: $d_E(sub_T(d), sub_T(a')) = ||sub_T(d)||$. All the couples featuring a leaf and an internal vertex (that is, a vertex which is not a leaf), such as $(d, b')$, $(a, d')$ etc. behave similarly.

**Step:** $n = m = 2$

The points $x \in V_T$ with $l(v_T(x)) \leq 2$ are $\{a, b, c, d, r_T\}$ and the points $y \in V_{T'}$ with $l(v_{T'}(y)) \leq 2$ are $\{a', b', c', d', r_{T'}\}$. Thus the couples $(x, y)$ which are considered are $(d, r_T)$, $(r_T, d')$, $(r_T, r_{T'})$ and then the trivial ones: $(r_T, a')$, $(r_T, b')$, $(r_T, c')$ and $(a, r_{T'})$, $(b, r_{T'})$, $(c, r_{T'})$. Some couples have already been considered and thus are not repeated.

**Couple:** $(d, r_T)$ Let $T_d = sub_T(d)$ and $T' = sub_T(r_T)$. The set of internal vertices are respectively $E_{T_d} = \{a, b\}$ and $E_{T'} = \{a', b', c', d'\}$. Thus, the binary variables we need to consider, are the following: $\delta_{a,a'}$, $\delta_{a,a'}$, $\delta_{a,b'}$, $\delta_{a,c'}$, $\delta_{a,d'}$, $\delta_{b,a'}$, $\delta_{b,b'}$, $\delta_{b,b'}$, $\delta_{b,c'}$, $\delta_{b,d'}$ and $\delta_{c,c'}$.

The quantities $\Delta_{i,j}^{v,w}$ are given by: $\Delta_{0,0}^{a,a'} = 0$, $\Delta_{0,1}^{a,a'} = 3$, $\Delta_{0,0}^{a,b'} = 1$, $\Delta_{0,1}^{a,b'} = 4$, $\Delta_{0,0}^{a,c'} = 1$, $\Delta_{0,1}^{a,c'} = 2$, $\Delta_{0,0}^{b,a'} = 0$, $\Delta_{0,1}^{b,a'} = 3$, $\Delta_{0,0}^{b,b'} = 1$, $\Delta_{0,1}^{b,b'} = 4$, $\Delta_{0,0}^{b,c'} = 1$, $\Delta_{0,1}^{b,c'} = 2$. The function $F_C(\delta)$ is easily obtained by summing over $\delta_{i,j}^{v,w} \cdot \Delta_{i,j}^{v,w}$.

While:

$$F_D(\delta) = (1 - \delta_{0,0}^{a,a'} - \delta_{0,1}^{a,a'} - \delta_{0,0}^{a,b'} - \delta_{0,1}^{a,b'} - \delta_{0,0}^{a,c'} - \delta_{0,1}^{a,c'}) \cdot 1 + \ldots + (1 - \delta_{0,0}^{a,d'} - \delta_{0,0}^{b,d'}) \cdot 3$$

and:

$$F^-(\delta) = (\delta_{0,0}^{a,a'} + \delta_{0,1}^{a,a'} + \delta_{0,0}^{a,b'} + \delta_{0,1}^{a,b'} + \delta_{0,0}^{a,c'} + \delta_{0,1}^{a,c'}) \cdot 0 + \ldots + (\delta_{0,0}^{a,d'} + \delta_{0,0}^{b,d'}) \cdot 3$$

and:

$$F^S(\delta) = (\delta_{0,0}^{a,a'} + \delta_{0,1}^{a,a'}) \cdot 0 + (\delta_{0,0}^{a,b'} + \delta_{0,1}^{a,b'}) \cdot 0 + \ldots + (\delta_{0,0}^{a,d'} + \delta_{0,0}^{b,d'}) \cdot 3$$

Lastly the constraints are:

$$\delta_{0,0}^{a,a'} + \delta_{0,1}^{a,a'} + \delta_{0,0}^{a,b'} + \delta_{0,1}^{a,b'} + \delta_{0,0}^{a,c'} + \delta_{0,1}^{a,c'} \leq 1$$

$$\delta_{0,0}^{b,a'} + \delta_{0,1}^{b,a'} + \delta_{0,0}^{b,b'} + \delta_{0,1}^{b,b'} + \delta_{0,0}^{b,c'} + \delta_{0,1}^{b,c'} \leq 1$$

$$\delta_{0,0}^{c,a'} + \delta_{0,1}^{c,a'} + \delta_{0,0}^{c,b'} + \delta_{0,1}^{c,b'} + \delta_{0,0}^{c,d'} + \delta_{0,1}^{c,d'} \leq 1$$

$$\delta_{0,0}^{a,b'} + \delta_{0,1}^{a,b'} + \delta_{0,0}^{b,b'} + \delta_{0,1}^{b,b'} + \delta_{0,0}^{a,c'} + \delta_{0,1}^{a,c'} \leq 1$$

$$\delta_{0,0}^{a,c'} + \delta_{0,1}^{a,c'} \leq 1$$

In this case there are many minimizing solutions. One is given by: $\delta_{0,0}^{a,a'} = \delta_{0,0}^{b,b'} = 1$ and all other variables equal to 0. This value of $\delta$ is feasible since the variables $\delta_{0,0}^{a,a'}$ and $\delta_{0,0}^{b,b'}$ never appear in the same constraint. This value of $\delta$ entails $F_C(\delta) = 3 + 1$, $F_D(\delta) = 2$, $F^-(\delta) = 0$ and $F^S(\delta) = 0$, and thus $d_E(T_d, T') = 6$.

Another solution can be obtained with: $\delta_{0,0}^{a,d'} = \delta_{0,0}^{b,c'} = 1$ and all other variables equal to 0. Also this value of $\delta$ is feasible since the variables $\delta_{0,0}^{a,d'}$ and $\delta_{0,0}^{b,c'}$ never appear in the same constraint. This value of $\delta$ entails $F_C(\delta) = 2 + 1$, $F_D(\delta) = w_T(a') + w_T(b') = 1 + 2$, $F^-(\delta) = ||sub_T(d')|| = 3$ and $F^S(\delta) = d_E(sub_T(a), sub_T(d')) = ||sub_T(d')|| = 3$, and thus $d_E(T_d, T') = 3 + 3 + 3 + 3 = 6$.

**Couple:** $(r_T, d')$  This and the other couples are left to the reader.
8. Numerical Simulations

In this section, the feasibility of the algorithm presented in Section 6 is assessed by means of some numerical simulations. We also deal with the problem of approximating the metric $d_E$ when the number of leaves in the tree structures in the data set is too big to be handled. The effectiveness of such approximations is showcased using some simple case studies, which also give some practical examples of the issues raised in Section 2.

In the implementations, dendrograms are always considered with a binary tree structure, obtained by adding negligible edges, that is edges $e$ with arbitrary small $d(\varphi(e), 0)$, when the number of children of a vertex exceeds 2.

8.1 Edit Distance Simulations

To get some concrete ideas of proper runtimes needed to calculate distances, we fix the number of leaves $n$ and for 100 times the following procedure is repeated: generate two random samples of $n$ points from the uniform distribution on a compact, real interval, take their single linkage hierarchical dendrograms (with multiplicity function equal to the weight function $w_T$) and compare them with $d_E$. This whole pipeline is repeated for any integer $n$ in the interval $[5, 20]$. In Figure 7 there are the average runtimes as a function of the number of leaves of the involved binary trees. The standard deviations over the repetitions are also reported, which show a quite large band around the mean. The different curves in Figure 7 concern the portion of time effectively spent by the solver to compute the solution of the ILP problems, and the amount of time employed to setup such problems. All code is written in Python and thus this second part of the runtimes can likely be greatly reduced by using more performing programming languages. The green line of total time is computed parallelizing the for loop in Algorithm 1. Note that dendrograms with the same number of leaves may end up having different tree-structures and so different dimensions. This is the main reason for the big shaded regions around the mean. If the trees were aggregated by dimension, the standard deviation of runtimes would decrease. Nevertheless, in applications, the only thing one can reasonably control is the number of leaves (which is given by the number of minima in the function, the number of clusters in a dendrogram, etc.) and for this reason the trees are aggregated as in Figure 7.

The computations are carried out on a 2016 laptop with Intel(R) processor Core(TM) i7-6700HQ CPU @ 2.60GHz, 4 cores (8 logical) and 16 GB of RAM. The employed ILP solver is the freely available IBM CPLEX Optimization Studio 12.9.0.

8.2 Pruning

In Section 2.1 we point out that the merging information carried by the dendrograms in Figure 1 is mostly contained in a number of vertices which is much lower than the actual number of leaves in the dendrogram. If one defines a proper multiplicity function, with values in an editable space $X$, coherently with the aim of the analysis, then the value $d(\varphi(e), 0)$ can be thought as the amount of information carried by the edge $e$. The bigger such value is, the more important that edge will be for the dendrogram. In fact such edges are the ones most relevant in terms of $d_E$.

A sensible way to reduce the computational complexity of the metric $d_E$, losing as little information as possible, is therefore the following: consider any couple of leaves, if the “amount of information” $d(\varphi(e), 0)$ of one of the two leaves is below a certain threshold, that leaf is deleted and its father ghosted. If both are below the threshold, only the leaf with smaller $d(\varphi(e), 0)$ is deleted (if $d(\varphi(e), 0)$ is equal across two siblings, one of them is
randomly deleted). This operation is repeated recursively until no leaf with multiplicity under the fixed threshold is left. This operation is called pruning; the operator which assigns to a dendrogram the pruned dendrogram with $\varepsilon \geq 0$ threshold is called $P_\varepsilon$. Note that pruning a dendrogram removes leaves, but keeps unchanged the distance from the root to the leaves which are not deleted. For instance, in the case of merge trees, this means that the range of the height function $h_T$ does not change upon pruning the tree.

We can quantify the (normalized) lost information with what we call pruning error ($PE$): $(||T|| - ||P_\varepsilon(T)||)/||T||$.

### 8.3 Examples

Now we use two simulated data sets to put to work the frameworks defined in Section 5. The examples are basic, but suited to assert that generalized dendrograms and the metric $d_E$ capture the information we designed them to grasp. In particular, since examples in Section 2.1 and Section 2.2 already give insights into the role of the tree-structured information, we want to isolate and emphasize the key role of multiplicity functions. The examples presented concern hierarchical clustering dendrograms and dendrograms representing scalar fields.

#### 8.3.1 Hierarchical Clustering Dendrograms

We consider a data set of 30 points clouds in $\mathbb{R}^2$, each with 150 or 151 points. Point clouds are generated according to three different processes and are accordingly divided into three classes. Each of the first 10 point clouds is obtained by sampling independently two clusters of 75 points respectively from normal distributions centered in $(5,0)$ and $(-5,0)$, both with $0.5 \cdot Id_{2 \times 2}$ covariance. Each of the subsequent 10 point clouds is obtained by sampling independently 50 points from each of the following Gaussian distributions: one centered in $(5,0)$, one in $(-5,0)$ and one in $(-10,0)$. All with covariance $0.5 \cdot Id_{2 \times 2}$. Lastly, to obtain each of the last 10 point clouds, we sample independently 150 points as done for the first 10 clouds, that is 75 independent samples from a Gaussian centered $(5,0)$ and
75 from one centered in \((-5, 0)\), and then, to such samples, we add an outlier placed in \((-10, 0)\).

Some clouds belonging to the second class and third classes are plotted respectively in Figure 8a and Figure 8b. We resort to pruning because of the high number of leaves, but we still expect to be able to easily separate point clouds belonging to the first and third classes (that is, with two major clusters) from clouds belonging to the second class, which feature three clusters, thanks to the multiplicity function defined in Section 5.2. All dendrograms have been pruned with the same threshold, giving an average pruning error of 0.15.

We can see in Figure 8c that this indeed the case. It is also no surprise that persistence diagrams do not perform equally good in this classification task, as displayed in Figure 8d. In fact PDs have no information about the importance of the cluster, making it impossible to properly recognize the similarity between data from the first and third class. They are, however, able to distinguish clouds belonging to class two from clouds belonging to class three since the persistence of the homology class associated to the leftmost cluster in clouds belonging to class two is smaller compared to what happens in clouds from the third class. The cluster centered in \(-10\) and the one in \(-5\) are in fact closer when the first one is a proper cloud, than when it is a cluster made by a single point.

### 8.3.2 Dendrograms of Functions

This time our aim is to work with merge trees of functions, adding the multiplicity function induced by the Lebesgue measure of the sublevel sets, as defined in Section 5.3, and using them to discriminate between two classes in a functional data set.

We simulate the data set so that the discriminative information is contained in the size of the sublevel sets and not in the structure of the critical points. To do so a situation which is very similar to the one shown by Sangalli et al. (2010) for the Berkeley Growth Study data is reproduced, where all the variability between groups in a classification task is explained by warping functions. We fix a sine function defined over a compact 1D real interval (with the Lebesgue measure) and we apply to its domain 100 random non linear warping functions belonging to two different, but balanced, groups. Warpings from the first group are more likely to obtain smaller sublevel sets, while in the second groups we should see larger sublevel sets and so “bigger” multiplicity functions defined on the edges. Note that, being the Lebesgue measure invariant with the translation of sets, any horizontal shifting of the functions would not change the distances between dendrograms.

The base interval is \(I = [0, 30]\) and the base function is \(f(x) = \sin(x)\). The warping functions are drawn in the following way. Pick \(N\) equispaced control points in \(I\) and then...
we draw $N$ samples from a Gaussian distribution truncated to obtain only positive values. We thus have $x_1, \ldots, x_N$ control points and $v_1, \ldots, v_N$ random positive numbers. Define $y_i := \sum_{j=1}^N v_j$. The warping is then obtained interpolating with monotone cubic splines the couples $(x_i, y_i)$. Being the analysis invariant to horizontal shifts in the functions, we fix $x_0 = y_0 = 0$ for visualization purposes.

The groups are discriminated by the parameters of the Gaussian distribution from which we sample the positive values $v_i$ to set up the warping. For the first class we sample $N = 10$ positive numbers from a truncated Gaussian with mean 3 and standard deviation 2; for the second the mean of the Gaussian is 5 and the standard deviation is 2. Thus we obtain each of the first 50 functions sampling 10 values $v_i$ from the truncated Gaussian centered in 3, building the warping function as explained in the previous lines, and then reparametrizing the sine function accordingly. The following 50 functions are obtained with the same pipeline but employing a Gaussian centered in 5. Note that, by construction, all the functions in the data set share the same merge tree.

Examples of the warping functions can be seen in Figure 9c; the resulting functions can be seen in Figure 9a. The key point here is that we want to see if the dendrograms can retrieve the information contained in the warping functions. For this reason we compare the $L_2$ pairwise distances between such functions (see Figure 9f) and the pairwise distances
obtained with dendrograms (see Figure 9d). The visual inspection confirms the close relationships between the two sources of information. Moreover, if we vectorize the arrays given by the two matrices (considering only entries above the diagonal) and compute the Fisher correlation, we get a score of 0.85 (see Figure 9g). Instead, a naive approach with the $L_2$ metric applied directly to the data set would capture no information at all, as we can observe from Figure 9e and the Fisher correlation with the matrix obtained from warping functions is 0.15 (see Figure 9h).

Note that, in general, the problem of finding warping functions to align functional data is deeply studied and with no easy solution (see, for instance, the special issue of the Electronic Journal of Statistics dedicated to phase and amplitude variability - year 2014, volume 8 or Srivastava et al. (2011)) especially for non-linear warping of multidimensional or non-euclidean domains. Instead, generalized dendrograms less sensitive to such dimensionality issues, in the sense that they only arise in calculating the connected components and measure of the sublevel sets.

9. Conclusions

Starting from the problems outlined in Section 2, we develop a framework to work with topological information which can be represented with tree-like structures. As motivated throughout the manuscript, we argue that these kinds of topological summaries can succeed in many situations where persistence diagrams are not effective. They also provide a great level of versatility because of the wide range of additional information that can be extracted from data. Possibly the greatest drawback in these representations is the computational complexity involved in comparing them. We define a novel metric structure which works locally on the trees and can be calculated by solving a set of smaller and easier subproblems. This metric proves to be feasible and we carry out some examples to showcase its effectiveness in situations which are of interest in different branches of data analysis.

Along with the more general perspective of finding other ways to enrich the information extracted by TDA from data, this work leaves many paths that can be followed. We think that the hypotheses on the set of weights which can be added to dendrograms can be relaxed; however, the algorithm presented in this manuscript may need to be adapted to the properties of the chosen weight space. Even without increasing the range of available weights, case-specific pipelines can be designed and studied, as done in the case of merge trees of functions in [Paper FDA]. Moreover interaction with the more general case of Reeb Graphs can be investigated, possibly following the decomposition presented in Stefanou (2020). Lastly other families of metrics can be defined, starting from $d_E$, aiming at emphasizing or overlooking on certain kinds of variability in the dendrograms, providing more “stable” metrics.

Acknowledgments

This work was carried out as part of my PhD Thesis, under the supervision of Professor Piercesare Secchi.
Appendix A. Proofs

Proof of Theorem 1
To lighten the notation we use the following symbols:

- the edit induced by \((v, D)\) is called \(v_d\) and \(v_d^{-1}\) stands for \((D, v)\).
- the edit induced by \((v, G)\) is called \(v_g\) and \(v_g^{-1}\) stands for \((G, v)\).
- the edit induced by \((v, v')\) is called \(v_{\phi, \phi'}\) with \(\phi\) being the original multiplicity function, and \(\phi'\) the multiplicity function after the shrinking.

We know that the set of finite edit paths between two dendrograms is nonempty. Suppose that \(\gamma\) is a finite edit path. This means that \(\gamma\) is the composition of a finite set of edits. We indicate such ordered composition with \(\gamma = \prod_{i=0}^{N} e_i\) with \(e_i\) edit operation. We would like to change the order of the edit operations without raising the cost and changing the extremes of the edit path. This is not always possible. However we can work it around in the useful cases using properties (P1)-(P4). In particular, we would like to know when we can commute a generic edit \(e_i\) in the following situations:

- \(v_d \circ e_i\) and \(e_i \circ v_d^{-1}\)
- \(v_g \circ e_i\) and \(e_i \circ v_g^{-1}\).

Moreover we want to reduce the edit path to max one edit for any vertex of \(T\) and \(T'\).

We divide the upcoming part of the proof in subsections, each devoted to different combinations of edits.

\(v_d\) AND \(v_d^{-1}\)
When we delete or insert one vertex, we are modifying the tree structure at the level of its father and its children. Therefore, we are only taking into account operations on the father, on the vertex itself or on the children of the deleted/inserted vertex.

- \(v_d \circ v_g'\) with \(v\) son of \(v'\), can be safely replaced with \(v_d \circ v_{g'}\). Instead of ghosting the father and then deleting the whole edge, we can delete both edges one by one; conserving the length of the path (P3). If \(v\) is father of \(v'\) then we can safely commute the operations.

- \(v_d \circ v_g^{-1}\) can be replaced with \(v'_{\phi, \phi'}\) with \(v'\) father of \(v\) (after the insertion) and \(\phi'\) properly defined not to raise the cost of the path. In fact we are inserting \(v\) on an edge and then deleting it. This can obviously be achieved by shrinking the original edge (without changing the path length - (P4)).

- similarly, \(v_d \circ v_{g'}^{-1}\) with \(v'\) to be father of \(v\) can again be replaced safely by a proper shrinking: instead of inserting a point in an edge, and deleting then the edge below, we can directly shrink the original edge (P4). If \(v'\) is to be inserted below \(v\) this is the same situation, but seen from the point of view of the son of \(v\).

- \(v_d \circ v_{\phi, \phi'}\) can be replaced by \(v_d\) potentially diminishing the length of the path, but surely not raising it (P1).
• $v' \circ v_d^{-1}$. If $v'$ is the father of $v$, this edit can be replaced with just $v'_\phi \circ v_d^{-1}$ with appropriate weights: we are inserting an edge under a vertex which (in this case) becomes of order two and is ghosted. We can directly modify the edge without changing the length of the path (P4). If $v'$ is the vertex which would become son of $v$, we can simply shrink $v$ to obtain the same result without raising the cost (P4).

• $v_g^{-1} \circ v_d^{-1}$, with $v'$ to appear on the edge inserted with $v_d^{-1}$ cannot commute (otherwise can always commute), but can be replaced by two insertions: instead of inserting an edge and then splitting it, we can directly insert two smaller edges; without changing the cost of the path (P3).

• $v'_\phi \circ v_d^{-1}$ can be replaced with an insertion directly with multiplicity $\phi'$, possibly shortening the path (P1).

• consider $v_d^{-1} \circ v_d$ with $v'$ to be inserted with, as father, the father of $v$; if the children of $v'$ are different from the children of $v$, this operation cannot commute. If the children are the same, it can be changed with a shrinking of $v$, reducing the length of the path by at most $\text{cost}(v'_d) + \text{cost}(v_d)$ (P1).

$v_g$ AND $v_g^{-1}$

Like in the previous case, we only take into account transformations concerning the father and the son of the added/ghosted order two vertex.

• $v_g \circ v'_d$, with $v$ and $v'$ being on adjacent edges, can commute (P2).

• $v_g \circ v'_d^{-1}$, with $v$ and $v'$ being on adjacent edges, can commute provided we define carefully the splitting $v'_d^{-1}$ (P2).

• $v_g \circ v'_\phi \circ v'_d$ means that we are shrinking a vertex before ghosting it. However, we can achieve the same result, without increasing the path length, by ghosting the vertex at first, and then shrinking its son (P1)-(P4).

• $v'_\phi \circ v_g^{-1}$ either with $v' = v$, or with $v$ father of $v'$, can be replaced with an appropriate shrinking of the (future) son of $v$, and then an appropriate insertion of $v'$ without changing the length of the path (P3)-(P4).

• $v_g \circ v'_d$ with $v$ father of $v'$ cannot be commuted and cannot be replaced by a similar operation which inverts ghosting and deletion.

$v_w, v'_w$

• $v'_w \circ v_w \circ v'_w$ can be replaced by $v'_w \circ v'_w$, which is either conserving or shortening the path (P1).

• $v'_w^{-1} = v_w \circ v'_w$.

Thanks to these properties we can take a given path $\gamma = \prod_{i=0}^{N} e_i$ and modify the edit operations in order to obtain the following situation:

• the first operations are all in the form $v_d$; this can be achieved because $v_d \circ -$ can be always rearranged, potentially by changing the path as shown before and shortening it. Of course there can be only one deletion for each vertex of $T$.
• then we have all the edits in the form $v_g$; since $v_g \circ -$ is exchangeable any time but when we have $v_g \circ v'_g$, this is not a problem. Observe that all order two vertices which were not deleted can be ghosted (at most one time);

• in the same way we can put last all the paths in the form $v_d^{-1}$ and before them $v_g^{-1}$. All the new vertices appearing with the insertion of edges and the splitting of edges with order two vertices are all nodes which remain in $T'$ and which are not further edited;

• in the middle we are left with the shrinking paths. Since we can substitute $v_{\varphi,\varphi'} \circ v_{\varphi'',\varphi'''}$ with $v_{\varphi,\varphi'''}$, we can obtain just one single transformation on a vertex.

Thus
\[
\gamma = (\gamma_T')^{-1} \circ (\gamma_T')^{-1} \circ \gamma_T \circ \gamma_T.
\]

with:

• $\gamma_T^d = \prod v_d$

• $\gamma_T^g = \prod v_g$

• $\gamma_s^T = \prod v_{\varphi,\varphi'}$

• $\gamma_T'^{-1} = \prod v_g^{-1}$

• $\gamma_T^d'^{-1} = \prod v_d^{-1}$

is such that $\gamma(T) = \gamma(T') = T'$ and $\text{cost}(\gamma) \leq \text{cost}(\gamma)$. The key point is that $\gamma$ can be easily realized as a mapping in the following way:

• $(v, D) \forall v_d \in \gamma_T^d$

• $(v, G) \forall v_g \in \gamma_T^g$

• $(v, v') \forall v_{\varphi,\varphi'} \in \gamma_s^T$, where $v'$ is the renaming of $v$, with multiplicity given by $\varphi'$.

• $(G, v) \forall v_g^{-1} \in (\gamma_T')^{-1}$

• $(D, v) \forall v_d^{-1} \in (\gamma_T^d')^{-1}$

\[\blacksquare\]

**Proof of Theorem 1.**
Any order 2 vertex which is not ghosted is paired with another order 2 vertex. Ghosting both of them does not increase the cost of the mapping. 

\[\blacksquare\]

**Proof of Proposition 1.**
Condition (M2) coincide with condition (A2). Condition (M3) is clearly satisfied because of the antichain condition (A1). Consider a vertex $v \in E_T$. The only case in which $v$ is not edited is when $v < x$ with $x \in v_T \cap \pi_T(M^*)$. However, in this case $v$ does not appear in $\sim T_M$, and thus (M1) is satisfied. Moreover, all and only order 2 vertices, after the deletions, are ghosted, and (M4) follows . 

\[\blacksquare\]
Proof of Theorem 2.
Let $M \in \mathcal{M}(T, T')$ such that $d_E(T, T') = \text{cost}(M)$.

We note that father > son induces a partial order relationship also on the pairs given by coupled points in $M$: $(x, y) > (v, w)$ if $x > v$ and $y > w$. In fact, by property (M3), $x > v$ if and only if $y > w$. So we can select $(x_i, y_i)$, the maxima with respect to this partial order relationship. Thus, we obtain $(x_0, y_0), \ldots, (x_n, y_n)$ which form an antichain (both in $V_T$ and $V_{T'}$).

Clearly $M^* = \{(x_0, y_0), \ldots, (x_n, y_n)\} \in \mathcal{C}^*(T, T')$. Now we build $\alpha(M^*)$ and compare the cost of its edits with the ones in $M$. Let $\bar{x}$ be the minimal common parent between $x_i$ and $x_j$. Since $\bar{x} > x_i, x_j$, it is not coupled in $M$. Since $x_i$ and $x_j$ are coupled, $\bar{x}$ cannot be ghosted, so it is deleted in $M$. Any point $x$ above $\bar{x}$ is deleted for the same reasons. So the edits above $\bar{x}$ are shared between $\alpha(M^*)$ and $M$.

In $\alpha(M^*)$ we ghost any point between $\bar{x}$ and $x_i$ (and the same for $x_j$) and this is not certain to happen in $M$ (some points could be deleted). Nevertheless, even in the worst case, these ghostings are guaranteed not to increase the distance. For instance, suppose $x_i < x < \bar{x}$ is deleted in $M$ and ghosted by $\alpha(M^*)$, then:

$$d(x_i \oplus x, y_i) \leq d(x_i \oplus x, y_i \oplus x) + d(y_i \oplus x, y_i) = d(x_i, y_i) + d(x, 0)$$

by properties (P1)-(P4). Since $\alpha(M^*) \in \mathcal{M}(\tilde{T}_{M^*}, \tilde{T}_{M^*})$ by Proposition 1, we have:

$$\sum_{(x, y) \in M^*} d_E(\text{sub}_T(x), \text{sub}_T(y)) + \text{cost}(\alpha(M^*)) \leq \text{cost}(M)$$

Now we prove the other inequality.

Consider $M^*$ which realizes the minimum of the right side of Equation (1), and $M_i$ which realizes $d_E(\text{sub}(x_i), \text{sub}(y_i))$ with $(x_i, y_i) \in M^*$. We build a mapping $M$ collecting edits in the following way: for every $x' \in E_T$ if $x' \in \text{sub}(x_i)$, we take the edit associated to it from $M_i$, otherwise we know that it is edited by $\alpha(M^*)$, and we take it from there; the set of these assignments gives $M \in \mathcal{M}(T, T')$ whose cost is exactly $\sum_{(x, y) \in M^*} \text{cost}(M_i) + \text{cost}(\alpha(M^*))$. This gives the second inequality.

Proof of Proposition 2.

We use all pieces of notation defined in Section 6.1. Consider $x \in E_T$ and $y \in E_{T'}$. Recall that $T_x := \text{sub}_T(x)$ and $T_y := \text{sub}_T(y)$, and $N_x := \text{dim}(T_x)$ and $N_y := \text{dim}(T_y)$. Moreover, given $v \in V_{T_x}$, we use the sequence $v_0 = v < v_1 < \ldots < r_T$ to indicate the points in $\zeta_v$. The same with $w \in V_{T_y}$.

**SETUP**

We have $W_{xy}$ which is a $(N_x - 1) \times (N_y - 1)$ matrix such that $(W_{xy})_{v,w} = d_E(T_v, T_w)$ for all $v \neq x \in T_x$ and $w \neq y \in T_y$.

We would like to find $M^* \in \mathcal{C}^*(T_x, T'_y)$ minimizing Equation (1) for $T_x$ and $T_y$, but this is a difficult task. In fact, as evident in the construction of $\alpha(M^*)$, a set $M^* \in \mathcal{C}^*(T_x, T_y)$ has the role of pairing segments of dendrograms: if $(v, w) \in M^*$, then the paths $\zeta^v$ and $\zeta^w$ are paired and then shrunk one on the other by $\alpha(M^*)$. But the points $\tilde{v}$ and $\tilde{w}$ depend on the whole set $M^*$, and not simply on the couple $(v, w)$. Modeling such global dependence gives rise to non-linear relationships between coupled points, and so leading to a non linear cost function, in terms of points interactions, to be minimized. For this
reason we “weaken” the last term in Equation (1), allowing also mappings different from \(\alpha(M^*)\) to be built from \(M^*\). In other words we minimize the following equation:

\[
\sum_{(x,y)\in M^*} d_E(sub_T(x), sub_T(y)) + \text{cost}(\beta(M^*))
\]

(5)

where \(M^*\) is the set of coupled points in \(\beta(M^*)\) and lies in \(C * (T, T^*)\), and \(\beta(M^*)\) is a mapping in \(\mathcal{M}(\tilde{T}_{M^*}, \tilde{T}'_{M^*})\). Since \(\alpha(M^*)\) fits these conditions, minimizing Equation (1) or Equation (5) gives the same result.

**Variables**

As already stated, we are considering the following set of binary variables: for every \(v \in E_T\) and \(w \in E_{T_y}\), for \(v_i \in \zeta_v, v_i < r_{T_x}\), and \(w_j \in \zeta_w, w_j < r_{T_y}\), we have a binary variable \(\delta_{i,j}^{v,w}\). We want to write a constrained optimization problem such that having \(\delta_{i,j}^{v,w} = 1\) means that we pair the segments \(\zeta_v^{i+1}\) and \(\zeta_w^{j+1}\), and shrink one in the other in the induced mapping. This, for instance implies that \((v, w) \in M^*\) and, when designing the constraints, \(v_{i+1}\) (and the same for \(w_{j+1}\)) is the first point going from \(v\) towards \(r_{T_x}\), such that there can be another point in \(sub_{T_x}(v_{i+1})\) paired in \(\beta(M^*)\).

Now we state how \(\delta_{i,j}^{v,w} = 1\) contributes to define edits in \(\beta(M^*)\), which is then given by the edits induced by all the variables equal to 1. In order to pair and shrink the segments \(\zeta_v^{i+1} = \{v = v_0, v_1, \ldots, v_{i+1}\}\) and \(\zeta_w^{j+1} = \{w = w_0, w_1, \ldots, w_{j+1}\}\) we need to induce the following edits on \(T_x\):

- all the points \(v_k \in \zeta_v^{i+1}\) with \(0 < k < i + 1\) are ghosted, that is \((v_k, G) \in \beta(M^*)\);
- if \(v' < v_k\) for some \(0 < k < i + 1\), then \((v', D) \notin \beta(M^*)\);
- if \(v' \geq v_{i+1}\) and \(v' \neq r_T\), then \((v', D) \in \beta(M^*)\)
- \((v, w) \in \beta(M^*)\).

Of course analogous edits must be induced on points in \(T_y\). Thus, the edit \((v, w) \in \beta(M^*)\), in the edit paths given by the mapping \(\beta(M^*)\), means shrinking the edge \((v, v_{i+1})\) onto \((w, w_{j+1})\). Recall that, if \(\delta_{i,j}^{v,w} = 1\), we do not need to define edits for \(sub_{T_x}(v)\) and \(sub_{T_y}(w)\), by assumption, we already know \(d_E(T_v, T_w)\).

**Constraints**

Clearly, not all combinations of \(\delta_{i,j}^{v,w}\) are acceptable, in that they do not induce a proper partial mapping \(\beta(M^*)\), with \(M^* \in C^*(T_x, T_y)\). Segments could even be paired multiple times. To avoid such issues, we build a set of constraints for the variable \(\delta\). Recall that the set of acceptable values \(K\) is defined by Equation (2) and Equation (3).

The following Proposition clarifies the properties of any value of \(\delta \in K\).

**Proposition 4** If \(\delta \in K\):

- the couples \((v, w)\) such that \(\delta_{i,j}^{v,w} = 1\) define a set \(M^* \in C^*(T_x, T_y)\);
- the edits induced by all \(\delta_{i,j}^{v,w} = 1\) give a mapping \(\beta(M^*)\) in \(\mathcal{M}(\tilde{T}_{M^*}, \tilde{T}_{M^*})\).

**Remark 12** If for every \(\delta_{i,j}^{v,w} = 1\), \(v_{i+1} = \tilde{v}\), then \(\beta(M^*) = \alpha(M^*)\).
**Objective Function**

Now we build a linear cost functions which calculates the results of Equation (5) for $\delta \in K$.

The key point we need to address is how to calculate the cost of $\beta(M^*)$.

Consider $v \in E_{T_x}$; it is clear that $v$ is coupled if $C(v) = 1$ and ghosted if $G(v) = 1$. If $C(v) = G(v) = 0$, then $v$ can be either deleted or be in the subtree of some coupled point and it is not edited by $\beta(M^*)$ since it does not appear in $\tilde{T}_{xM^*}$. One simple way to take care of this difference is to simply consider deleted all points that are not paired nor ghosted, and then subtract the cost of the points which are not supposed to be deleted, that is, the ones in $\text{sub}(v)$ with $C(v) = 1$. In other words, the vertex $v$ is considered deleted if $D(v) = 1$, but, if $C(v) = 1$, we must correct the total cost by:

$$||\text{sub}_{T_x}(v)|| = \sum_{v' < v} d(\varphi_{T_x}(v'), 0)$$

Now we calculate the costs associated to these three possibilities. If $G(v) = 1$, $v$ is ghosted then the cost of such edit is zero. If $D(v) = 1$ the cost of this edit is $d(\varphi_{T_x}(v), 0)$. If $v$ is coupled, and so $\delta_{v,w}^{i,j} = 1$ for some unique $i, w, j$, the cost is:

$$\Delta_{i,j,w}^{v,w} = d(\oplus_{v \in \zeta_v} \varphi_{T_x}(v'), \oplus_{w' \in \zeta_w} \varphi_{T_y}(w'))$$

Thus, the contribution of coupled points is $F^C(\delta)$ and the contribution of deleted points is $F^D(\delta) - F^-(\delta)$.

Now it is straightforward to write down the linear function that calculates the cost of $\beta(M^*)$: $F^3(\delta) := F^C(\delta) + F^D(\delta) - F^-(\delta)$. Lastly, $F^S(\delta)$ takes into account the value of $d_E(T_v, T_w)$, if $v$ and $w$ are coupled.

By Theorem 2, combined with Proposition 4, the solution of the following optimization problem:

$$\min_{\delta \in K} F^S(\delta) + F^\beta(\delta)$$

is equal to $d_E(T_x, T_y)$.

\[\blacksquare\]

**Proof of Proposition 4.**

Having fixed a leaf $l$, the constraint in Equation (2) allows for at most one path $\zeta_{v}^{v_{i+1}} \subset \zeta$ which is kept after the editing induced by the variables equal to 1. Moreover if $\delta_{v,j}^{i,w} \in \Phi(v) \cap \Phi(v')$, then $v = v'' = v'$. Thus, variables are added at most one time in the constraint. Which means that for any $v' \in V_{T_x}$, we are forcing that $v'$ can be an internal vertex of at most one of the kept segments $\zeta_v^{v_{i+1}}$. In other words if two “kept” segments $\zeta_v^{v_{i+1}}$ and $\zeta_{v'}^{v'_{j+1}}$ intersect each other, it means that they just share the upper extreme $v_{i+1} = v'_{j+1}$. These facts together imply that (if the constraints are satisfied) the edits induced on $T_x$ by $\delta_{v,j}^{i,w} = 1$ and $\delta_{v',w'}^{v',j'} = 1$ are always compatible. Lastly, by noticing that if $\delta_{v,j}^{i,w} \in \Phi(v')$ then $\delta_{v',w'}^{v',j'} \in \Phi(v')$ for all other possible $w'$ and $j'$, we see that every segment $\zeta_v^{v_i}$ is paired with at most one segment $\zeta_{v'}^{v_j}$, and vice versa.

As a consequence, at most one point on the path $\zeta_{v'}$ is coupled in $M^*$, for any vertex $v'$ in any of the tree structures, guaranteeing the antichain condition. Moreover any point of $T_x$ which is in $M^*$ is assigned to one and only point of $T_y$ and vice versa. The edits induced by $\delta_{v,j}^{i,w} = 1$ clearly satisfy properties (M2)-(M4). Passing to $\tilde{T}_{M^*}$ and $\tilde{T}'_{M^*}$, also (M1) is satisfied.

\[\blacksquare\]
Proof of Proposition 3.
In a full binary tree structure, at each level \( l \) we have \( 2^l \) vertices. Let \( L = \text{len}(T) \) and \( L' = \text{len}(T') \). We have that, for any vertex \( v \in V_T \) at level \( l \), the cardinality of the path from \( v \) to any of the leaves in \( \text{sub}_T(v) \) is \( L - l \) and the number of leaves in \( \text{sub}_T(v) \) is \( 2^{L-l} \).

So, given \( v \in V_T \) at level \( l \) and \( w \in V_{T'} \) at level \( l' \), to calculate \( d_E(\text{sub}_T(v), \text{sub}_{T'}(w)) \) (having already \( W_{vw} \)) we need to solve an integer linear problem with \( 2^{L-l} \cdot (L-l) \cdot 2^{L-l'} \cdot (L'-l') \) variables and \( 2^{L-l} + 2^{L-l'} \) linear constraints.

Thus, to calculate \( d_E(T,T') \), we need to solve \( (2^{L+1} - 1) \cdot (2^{L'+1} - 1) \) linear integer optimization problems, each with equal or less than \( 2^L \cdot L \cdot 2^{L'} \cdot L' \) variables and equal or less that \( 2^L + 2^{L'} \) constraints. Substituting \( L = \log_2(N) \) and \( L' = \log_2(M) \) in these equations gives the result.

Appendix B. Merge Trees

Denote \( \#C \) the cardinality of a finite set \( C \). Given a fixed basis vector spaces filtration \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \), with maps \( \psi_t: A_t \to A_{t'} \), we build a merge tree \( (T, h_T) \) which represents it up to isomorphisms. The tree structure \( T \) and the height function \( h_T \) are built along the following rules:

- set a leaf with height \( t_0 \) for every element in \( a_{t_0} \);
- for every \( a_k^{i+1} \in a_{t_i+1} \) such that \( a_k^{i+1} \notin \text{Im}(\psi_{t_i}^{i+1}) \), set a leaf with height \( t_{i+1} \);
- for every \( t_i \) such that \( \psi_{t_i}^{i+1}(a_k^i) = \psi_{t_i}^{i+1}(a_s^i) \), with \( a_k^i \) and \( a_s^i \in a_{t_i} \), set a vertex with height \( t_{i+1} \), where the vertices associated to respectively \( (\psi_{t_i}^{i+1})^{-1}(a_k^i) \) and \( (\psi_{t_i}^{i+1})^{-1}(a_s^i) \) merge. Where \( t_k = \min\{t_j | \#(\psi_{t_j}^{i+1})^{-1}(a_k^i) = 1 \} \) and \( t_s = \min\{t_j | \#(\psi_{t_j}^{i+1})^{-1}(a_s^i) = 1 \} \).

The last merging happens at height \( t_n \), which is the root of the tree structure. In this way, from \( \{(A_t, a_t)\}_{t \in \mathbb{R}} \), we obtain a merge tree which is unique up merge tree isomorphism. Viceversa, through the merge tree \( (T, h_T) \) we can build a fixed basis vector spaces filtration by cutting \( (T, h_T) \) at every height \( t \) and taking as \( a_t = \{e_1, \ldots, e_k\} \) the edges in \( E_T \) which are met at height \( t \). The vector spaces are generated over \( \mathbb{K} \) by \( a_t \). The maps are then given by the tree structure: if two edges merge at height \( t \), then, at height \( t \), the two corresponding basis elements are sent into the edge in which they merge. Otherwise the edges are just sent into themselves. The root \( r_T \) gives the basis element at height \( h_T(r_T) \).

Appendix C. Persistence Diagrams

Persistence diagrams are arguably among the most well known tools of TDA; for a detailed survey see, for instance, Edelsbrunner and Harer (2008).

A zero-dimensional persistence diagram extracted from a filtration of topological spaces \( \{X_t\}_{t \in \mathbb{R}} \), with \( X_t \subset X_{t'} \) if \( t \leq t' \), represents, up to isomorphism of sequence the vector spaces filtration \( \{H_0(X_t)\}_{t \in \mathbb{R}} \). Loosely speaking it is a collection of points \( (c_x, c_y) \) in the first quadrant of \( \mathbb{R}^2 \), with \( c_y > c_x \) and such that: \( c_x \) is the value of \( t \) corresponding to the first appearance of a path connected component in \( X_t \) (birth), while \( c_y \) is the “time” \( t \) where the same class merges with a different class appeared before \( c_x \) (death). A similar definition holds for homology classes in higher dimensions. For details about homology see Hatcher (2000). The convention that states that the “older” components survives and the “younger” dies, is called elder rule.
References

H. Adams, Tegan Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, Sofya Chepushtanova, E. Hanson, F. Motta, and Lori Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *J. Mach. Learn. Res.*, 18:8:1–8:35, 2017.

Michele Audin, Mihaï Damian, and Reinie Erné. *Morse theory and Floer homology*. Springer, 2014.

Ulrich Bauer, Xiaoyin Ge, and Yusu Wang. Measuring distance between reeb graphs. In *Proceedings of the thirtieth annual symposium on Computational geometry*, pages 464–473, 2014a.

Ulrich Bauer, Elizabeth Munch, and Yusu Wang. Strong equivalence of the interleaving and functional distortion metrics for reeb graphs. *arXiv preprint arXiv:1412.6646*, 2014b.

Ulrich Bauer, Barbara Di Fabio, and Claudia Landi. An edit distance for reeb graphs. 2016.

Ulrich Bauer, Claudia Landi, and Facundo Mémoli. The reeb graph edit distance is universal. *Foundations of Computational Mathematics*, pages 1–24, 2020.

Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H. Weber, and Bernd Hamann. Measuring the distance between merge trees. In *Topological Methods in Data Analysis and Visualization*, 2014.

Subhrajit Bhattacharya, Robert Ghrist, and Vijay Kumar. Persistent homology for path planning in uncertain environments. *IEEE Transactions on Robotics*, 31:1–13, 06 2015. doi: 10.1109/TRO.2015.2412051.

Silvia Biasotti, Daniela Giorgi, Michela Spagnuolo, and Bianca Falcidieno. Reeb graphs for shape analysis and applications. *Theoretical Computer Science*, 392:5–22, 02 2008. doi: 10.1016/j.tcs.2007.10.018.

Philip Bille. A survey on tree edit distance and related problems. *Theoretical computer science*, 337(1-3):217–239, 2005.

Louis J Billera, Susan P Holmes, and Karen Vogtmann. Geometry of the space of phylogenetic trees. *Advances in Applied Mathematics*, 27(4):733–767, 2001.

Alexander Bock, Harish Doraiswamy, Adam Summers, and Claudio Silva. Topoangler: Interactive topology-based extraction of fishes. *IEEE Transactions on Visualization and Computer Graphics*, PP:1–1, 08 2017. doi: 10.1109/TVCG.2017.2743980.

Peter Bubenik. Statistical topological data analysis using persistence landscapes. *Journal of Machine Learning Research*, 16(3):77–102, 2015.

Mathieu Carrière and Steve Oudot. Local equivalence and intrinsic metrics between reeb graphs. *arXiv preprint arXiv:1703.02901*, 2017.

F. Chazal, Brittany Terese Fasy, F. Lecci, A. Rinaldo, and L. Wasserman. Stochastic convergence of persistence landscapes and silhouettes. *J. Comput. Geom.*, 6:140–161, 2015.

Frédéric Chazal, Vin De Silva, Marc Glisse, and Steve Oudot. *The structure and stability of persistence modules*. Springer, 2016.
D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discrete & Computational Geometry*, 37:103–120, 2007.

David Cohen-Steiner, Herbert Edelsbrunner, John Harer, and Yuriy Mileyko. Lipschitz functions have lp-stable persistence. *Foundations of Computational Mathematics*, 10: 127–139, 02 2010. doi: 10.1007/s10208-010-9060-6.

Carina Curto. What can topology tell us about the neural code? *Bulletin of the American Mathematical Society*, 54, 05 2016. doi: 10.1090/bull/1554.

Barbara Di Fabio and Claudia Landi. Reeb graphs of curves are stable under function perturbations. *Mathematical Methods in the Applied Sciences*, 35(12):1456–1471, 2012.

Barbara Di Fabio and Claudia Landi. The edit distance for reeb graphs of surfaces. *Discrete & Computational Geometry*, 55(2):423–461, 2016.

H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. *Discrete & Computational Geometry*, 28:511–533, 2002.

Herbert Edelsbrunner and John Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

Y. Elkin and V. Kurlin. The mergegram of a dendrogram and its stability. *ArXiv*, abs/2007.11278, 2020.

K. Emmett, Benjamin Schweinhart, and R. Rabadan. Multiscale topology of chromatin folding. In *BICT*, 2015.

Brittany Fasy, Fabrizio Lecci, Alessandro Rinaldo, Larry Wasserman, Sivaraman Balakrishnan, and Aarti Singh. Confidence sets for persistence diagrams. *The Annals of Statistics*, 42:2301–2339, 03 2014.

Joseph Felsenstein and Joseph Felenstein. *Inferring phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.

Aasa Feragen, Pechin Lo, Marleen de Bruijne, Mads Nielsen, and Francois Lauze. Toward a theory of statistical tree-shape analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12 2012. doi: 10.1109/TPAMI.2012.265.

Marcio Gameiro, Yasuuki Hiraoka, Shunsuke Izumi, Miroslav Kramár, Konstantin Mischaikow, and Vidit Nanda. A topological measurement of protein compressibility. *Japan Journal of Industrial and Applied Mathematics*, 32:1–17, 03 2014. doi: 10.1007/s13160-014-0153-5.

Maryam Kashia Garba, Tom MW Nye, Jonas Lueg, and Stephan F Huckemann. Information geometry for phylogenetic trees. *Journal of Mathematical Biology*, 82(3):1–39, 2021.

Ellen Gasparovic, E. Munch, S. Oudot, Katharine Turner, B. Wang, and Yusu Wang. Intrinsic interleaving distance for merge trees. *ArXiv*, abs/1908.00063, 2019.

Chad Giusti, Robert Ghrist, and Danielle Bassett. Two’s company, three (or more) is a simplex: Algebraic-topological tools for understanding higher-order structure in neural data. *Journal of Computational Neuroscience*, 41, 01 2016. doi: 10.1007/s10827-016-0608-6.
Allen Hatcher. *Algebraic topology*. Cambridge Univ. Press, Cambridge, 2000.

Jotun Hein, Tao Jiang, Lusheng Wang, and Kaizhong Zhang. On the complexity of comparing evolutionary trees. In Zvi Galil and Esko Ukkonen, editors, *Combinatorial Pattern Matching*, pages 177–190, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. ISBN 978-3-540-49412-6.

C. Hofer, R. Kwitt, M. Niethammer, and A. Uhl. Deep learning with topological signatures. In *NIPS*, 2017.

Eunpyeong Hong, Yasuaki Kobayashi, and Akihiro Yamamoto. Improved methods for computing distances between unordered trees using integer programming. In *International Conference on Combinatorial Optimization and Applications*, pages 45–60. Springer, 2017.

Jakub Koperwas and Krzysztof Walczak. Tree edit distance for leaf-labelled trees on free leafset and its comparison with frequent subsplit dissimilarity and popular distance measures. *BMC bioinformatics*, 12:204, 05 2011. doi: 10.1186/1471-2105-12-204.

Violeta Kovacev-Nikolic, Peter Bubenik, Dragan Nikolic, and Giseon Heo. Using persistent homology and dynamical distances to analyze protein binding. *Statistical applications in genetics and molecular biology*, 15:19–38, 03 2016. doi: 10.1515/sagmb-2015-0057.

Miroslav Kramár, Arnaud Goullet, Lou Kondic, and K Mishaikow. Persistence of force networks in compressed granular media. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 87:042207, 04 2013. doi: 10.1103/PhysRevE.87.042207.

T. Lacombe, Marco Cuturi, and S. Oudot. Large scale computation of means and clusters for persistence diagrams using optimal transport. In *NeurIPS*, 2018.

P. Y. Lum, G. Singh, A. Lehman, T. Ishkanov, M. Alagappan, J. Carlsson, G. Carlsson, and Mikael Vilhelm Vejdemo Johansson. Extracting insights from the shape of complex data using topology. *Scientific Reports*, 3, February 2013. ISSN 2045-2322. doi: 10.1038/srep01236.

Robert MacPherson and Benjamin Schweinhart. Measuring shape with topology. *Journal of Mathematical Physics*, 53, 11 2010. doi: 10.1063/1.4737391.

Slobodan Maletić, Yi Zhao, and Milan Rajkovic. Persistent topological features of dynamical systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 26, 10 2015. doi: 10.1063/1.4949472.

Yuriy Mileyko, Sayan Mukherjee, and John Harer. Probability measures on the space of persistence diagrams. *Inverse Problems - INVERSE PROBL*, 27, 12 2011. doi: 10.1088/0266-5611/27/12/124007.

D. Morozov and G. Weber. Distributed merge trees. In *PPoPP ’13*, 2013.

Dmitriy Morozov, Kenes Beketayev, and Gunther Weber. Interleaving distance between merge trees. *Discrete and Computational Geometry*, 49:22–45, 01 2013.

Fionn Murtagh and Pedro Contreras. Algorithms for hierarchical clustering: An overview. ii. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7, 09 2017. doi: 10.1002/widm.1219.
Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. Topology of deep neural networks. *Journal of Machine Learning Research*, 21(184):1–40, 2020.

Siddharth Pal, Terrence J. Moore, Ram Ramanathan, and Ananthram Swami. Comparative topological signatures of growing collaboration networks. In Bruno Gonçalves, Ronaldo Menezes, Roberta Sinatra, and Vinko Zlatic, editors, *Complex Networks VIII*, pages 201–209, Cham, 2017. Springer International Publishing. ISBN 978-3-319-54241-6.

Valerio Pascucci and Kree Cole-McLaughlin. Parallel computation of the topology of level sets. *Algorithmica*, 38:249–268, 10 2003. doi: 10.1007/s00453-003-1052-3.

Jose Perea and John Harer. Sliding windows and persistence: An application of topological methods to signal analysis. *Foundations of Computational Mathematics*, 15, 07 2013. doi: 10.1007/s10208-014-9206-z.

Jose Perea, Anastasia Deckard, Steven Haase, and John Harer. Sw1pers: Sliding windows and 1-persistence scoring: discovering periodicity in gene expression time series data. *BMC bioinformatics*, 16:257, 08 2015. doi: 10.1186/s12859-015-0645-6.

Florian Pokorny, Majd Hawasly, and Subramanian Ramamoorthy. Topological trajectory classification with filtrations of simplicial complexes and persistent homology. *The International Journal of Robotics Research*, 35, 08 2015. doi: 10.1177/0278364915586713.

Abbas Rizvi, Pablo Camara, Elena Kandror, Thomas Roberts, Ira Schieren, Tom Maniatis, and Raul Rabadam. Single-cell topological rna-seq analysis reveals insights into cellular differentiation and development. *Nature Biotechnology*, 35, 05 2017. doi: 10.1038/nbt.3854.

Laura Sangalli, Piercesare Secchi, Simone Vantini, and Valeria Vitelli. K-mean alignment for curve clustering. *Computational Statistics & Data Analysis*, 54:1219–1233, 05 2010. doi: 10.1016/j.csda.2009.12.008.

Y. Shinagawa, T. L. Kunii, and Y. L. Kergosien. Surface coding based on morse theory. *IEEE Computer Graphics and Applications*, 11(5):66–78, 1991.

Vin Silva and Robert Ghrist. Coverage in sensor networks via persistent homology. *Algebraic & Geometric Topology*, 7, 04 2007. doi: 10.2140/agt.2007.7.339.

Ann Sizemore, Chad Giusti, and Danielle Bassett. Classification of weighted networks through mesoscale homological features. *Journal of Complex Networks*, 5, 12 2015. doi: 10.1093/comnet/cnw013.

R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26 (3):1518–1531, 2020.

A. Srivastava, W. Wu, S. Kurtek, E. Klassen, and J. S. Marron. Registration of functional data using fisher-rao metric. *arXiv: Statistics Theory*, 2011.

Anastasios Stefanou. Tree decomposition of reeb graphs, parametrized complexity, and applications to phylogenetics. *Journal of Applied and Computational Topology*, 4(2): 281–308, 2020.

Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26:422–433, July 1979. ISSN 0004-5411.
Elena Farahbakhsh Touli. Frechet-like distances between two merge trees. *ArXiv*, abs/2004.10747, 2020.

Elena Farahbakhsh Touli and Yusu Wang. Fpt-algorithms for computing gromov-hausdorff and interleaving distances between trees. In *ESA*, 2018.

Katharine Turner, Yuriy Mileyko, Sayan Mukherjee, and John Harer. Frechet means for distributions of persistence diagrams. *Discrete & Computational Geometry*, 52, 06 2012. doi: 10.1007/s00454-014-9604-7.

Haonan Wang and J. S. Marron. Object oriented data analysis: Sets of trees. *Ann. Statist.*, 35(5):1849–1873, 10 2007. doi: 10.1214/009053607000000217.

Keqin Wu and Song Zhang. A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification*, 3:203–223, 01 2013. doi: 10.1615/Int.J.UncertaintyQuantification.2012003956.

Kelin Xia, Xin Feng, Yiying Tong, and Guo We. Persistent homology for the quantitative prediction of fullerene stability. *Journal of computational chemistry*, 36, 03 2015. doi: 10.1002/jcc.23816.

Dongkuan Xu and Yingjie Tian. A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2, 08 2015. doi: 10.1007/s40745-015-0040-1.

Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete and Computational Geometry*, 33:249–274, 02 2005.