AIMS Mathematics, 10(6): 13025–13081.
DOI:10.3934/math.2025586
Received: 07 March 2025
Revised: 28 April 2025
Accepted: 06 May 2025
Published: 06 June 2025

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

Research article

A graph-matching formulation of the interleaving distance between merge trees

Matteo Pegoraro*

Department of Mathematics, KTH, Stockholm, 10044, Sweden

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

Abstract: In this work, we studied the interleaving distance between merge trees from a combinatorial point of view. In the first part of the paper, we used a particular type of matching between trees to obtain a novel formulation of the distance. This formulation unveiled a link connecting the interleaving distance and edit distances between merge trees, which was of great interest due to the recursive decomposition properties and constrained formulations with polynomial time algorithms that these distances often enjoyed. In the second part of the paper, we built on this connection by applying tools from edit distances to obtain a constrained formulation of the interleaving distance and a recursive procedure which allowed us to find algorithms for upper and lower bounds of the interleaving distance. We implemented those algorithms and used them to test another upper bound presented by other authors, and tackled some simulations and case studies. Motivated by the literature on edit distances, we believe that our novel formulation could lead to novel heuristics to compute the interleaving distance and that applying our recursive scheme to the constrained interleaving distance would produce polynomial time upper bounds of practical relevance.

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

1. Introduction

Topological data analysis (TDA) is a scientific field lying at the crossroads of applied topology and data analysis: objects like functions and point clouds are typically studied by means of possibly multidimensional complexes of homology groups [29] obtained with different pipelines. Homology groups considered with coefficients in a field $\mathbb{K}$ are vector spaces, whose dimension is determined by different kinds of "holes" which can be found in a space and, thus, provide a rich characterization of the shape of the space the analyst is considering. More precisely, each statistical unit induces a whole

family of topological spaces indexed on $\mathbb{R}^n$, which then, via homology, produces a family of vector spaces indexed on the same set more precisely, a functor [35] $(\mathbb{R}^n, \leq) \to \mathrm{Vect}_{\mathbb{K}}$. Such collections of vector spaces are called persistence modules [14], with multidimensional persistence modules being a special reference to the cases in which $n > 1$, and describe the shape of the data by considering interpretable topological information at different resolutions. A topological signature/summary or an invariant of a persistence module is a representation which usually maps persistence modules into a metric space or even a vector space so that some kind of analysis can be carried out. The most used topological summaries for 1-D persistence (i.e., $n = 1$) include persistence diagrams [21], persistence landscapes [12], persistence images [2], and persistence silhouettes [15]. When dealing with 0-dimensional homology and 1-D persistence, if the space X is path connected, another topological summary called merge tree [9] can be employed. A merge tree is a tree-shaped topological summary which captures the evolution of the path connected components $\pi_0(X_t)$ of a filtration of topological spaces $\{X_t\}_{t\in\mathbb{R}}$, with $X_t \hookrightarrow X_{t'}$ if $t \leq t'$.

One of the central ideas in TDA is the one of *stability* properties, i.e., the properties related to the continuity of the operator mapping data into topological signatures. One definition which has gained a lot of success is the one of ε-interleavings between topological representations: when adding some kind of ε-noise to the data, the associated summary "moves" by at most ε. If this happens, then the operator mapping functions to topological representations is a non-expansive Lipschitz operator. Starting from the bottleneck distance for persistence diagrams [22], this idea has been applied to 1-D persistence modules [14], multidimensional persistence modules [34], merge trees [9], sheaves [17, 38], and more general situations [10, 19]. In this paper, we focus on the problem of studying interleavings for merge trees.

The main reason to pursue such research direction is that merge trees and persistence diagrams are not equivalent as topological summaries, that is, they do not represent the same information with merge trees being able to distinguish between scenarios which persistence diagrams and all the other equivalent summaries cannot [16, 23, 33, 47]. As a consequence, in recent years, a lot of research sparked on merge trees, mainly driven by the need of having a stable metric structure to compare such objects. Stable and unstable edit distances between Reeb graphs or merge trees have been proposed by [7, 20, 41, 48], while other works focus on (unstable) Wasserstein distances [43], l_p distances [13] and interleaving distances between merge trees [3, 9, 11, 25, 28] or Reeb graphs [8, 17, 18, 38].

1.1. Related works

The problem of computing the interleaving distance between merge trees has already been approached by [3] and [25] in an attempt to approximate the Gromov-Hausdorff (GH) distance when both metric spaces are metric trees. In this situation, in fact, up to carefully choosing the root of the metric trees, the two metrics are equivalent. Both problems approximating the GH distance and computing the interleaving distance have been shown to be NP-hard [3, 11], and, thus, even obtaining feasible approximation algorithms for small trees is a daunting task.

In [3], the authors provide a polynomial time algorithm to approximate the interleaving distance between merge trees via binary search. They build a set which contains the solution of the optimization problem and then obtain a criterion to assess if certain values of the aforementioned set can be excluded from the set of solutions. If the length of the branches of both trees is big enough, one can carry out the binary search over all possible couples of vertices, with one vertex from the first tree and one from the

second. If trees have branches that are too small, the decision procedure is coupled with a trimming step to deal with these smaller edges. The algorithm returns an approximation of the interleaving distance, with the approximation factor depending on the ratio between the longest and the shortest edge in the tree and the number of vertices.

The authors of [25], instead, propose the first algorithm for the exact computation of the interleaving distance. Starting from a novel definition of the interleaving distance, they develop a faster alternative to exclude values from the same set of solutions considered by [3]. By filtering a finite subset of points on the metric trees (in general, bigger than the set of vertices of the "combinatorial" trees) according to a) their heights and b) a candidate optimal value δ, in an up-bottom fashion, points of the second tree are matched to subsets of points of the first tree. Such matching means that all points of the first tree inside in the chosen collection can be potentially mapped in the point of the second tree via some function viable for the interleaving distance. Each such matching (S, w) is then used to establish similar couples (set, point) between the children of the points involved: the set of points of the first tree is to be chosen among the compatible subsets of the children of S, and the point in the second tree among the children of w. If no such couples are found, δ is discarded. The algorithm they propose to compute the interleaving distance has complexity $O(n^2 \log^3(n) 2^{2\tau} \tau^{\tau+2})$, where n is the sum of the vertices in the two merge trees, and τ is a parameter depending on the input trees, which is defined by the authors. The key issue is that τ can be very big also for small-sized trees: In [25], in Figure 2, the authors showcase a tree with 8 leaves and $\tau = 13$ (thus, the complexity of the algorithm is at least $O(n^2 \log^3(n) 2^{26} 13^{15}) \sim O(n^2 \log^3(n) 10^{24})$.

Due to its computational complexity, the algorithm by [25] has not been implemented by other authors working with merge trees [16], which instead have exploited another formulation of the same metric, provided by [28]. This last formulation relies on a definition of the interleaving distance between merge trees which are endowed with a fixed set of labels on their vertices. The usual interleaving distance is then shown to be equivalent to choosing an appropriate set of labels potentially upon adding some vertices to the trees for the two given merge trees. This formulation, per se, does not provide computational advantages over the classical one, since evaluating all the possible labelings of a tree would still be unfeasible. However, [16] proposes a labeling strategy which should provide "good" labels. Despite being computationally accessible even for very big trees, this approach has the downside of not providing methods for assessing the error of the produced estimate.

1.2. Contributions

In this work, we pursue two main objectives. First, we aim to establish the theoretical foundations necessary to derive reliable polynomial upper and lower bounds for the interleaving distance. Second, we seek to develop practical techniques that complement those proposed in [3] and [25]. As demonstrated in Appendix B, our methods offer computational procedures that are more efficient than those in [25] and more reliable than those in [3].

To do so, we take a perspective which differs from all the aforementioned works: instead of obtaining progressively sharper bounds for the distance by locally looking at differences between trees or instead of looking for optimal labelings, we aim at describing jointly a global matching between two merge trees whose cost returns the exact interleaving distance between them. In this way, we obtain a novel combinatoric formulation of the interleaving distance between merge trees, which we also conjecture to be equivalent to yet another formulation, supported by the already established link

between the interleaving and the GH distances between trees [25].

Expressing distances between trees by matching their vertices is a very common approach for edit distances. Notably, there are very strong relationships between the couplings we use in this paper and the mappings which appear ubiquitously in edit distance literature. This allows us to leverage ideas and techniques developed in such research areas, advancing the theoretic and computational tools at our disposal. In particular, as summarized by Figure 14, in Appendix A, we specifically adopt two key ideas from edit distance frameworks:

1) Defining constrained variations of computationally expensive metrics, which maintain strong practical performance while improving computational efficiency. Most works on edit distances for merge trees [43,48,49,52,53] follow this exact path: they define "unconstrained" distances whose computational cost is NP-Hard [30], and then exploit the polynomial algorithms in [32, 46, 54] to obtain some "constrained" upper bounds, which can also be regarded as a metric on their own;

2) Developing recursive methods for computing both constrained and unconstrained formulations of distances, thereby reducing the computational burden and enabling recursive mixed/integer linear programming approaches, as in [31, 41, 52].

We exploit our novel formulation precisely to produce 1) a constrained version of the interleaving distance and 2) a recursive procedure to obtain upper and lower bounds for such distance via mixed linear programming. As previously stated, the bounds we derive in this way complement the procedures in [3] and [25], thereby extending the applicability of the interleaving distance, as discussed in Appendix B. It is important to note, however, that this comparison is rather technical and considers some particular scenarios, due to the profound different natures of the considered algorithms.

Given that our recursive procedure returns bounds whose computational costs are very close to the ones of [43, 48, 49, 52, 53], we believe that studying the combination of the recursive procedure on the constrained interleaving distance may similarly lead to polynomial-time bounds for the interleaving distance mirroring what has been achieved for the edit distances in the cited works. See Proposition 6 and Section 9.2.9.

In the last part of the paper, we test those bounds by comparing them with the method proposed by [16] to estimate the interleaving distance and by tackling a number of simulated and real-data case studies, showcasing also how to empirically assess the validity of our estimates.

1.3. Outline

The paper is organized as follows: In Section 2, we introduce merge trees first in a combinatorial fashion and then as "continuous" posets and metric spaces, recalling the definition of interleaving distance. In Section 3, we introduce a partial matching between trees called *coupling*. Relationships between these matchings and maps between trees are then presented in Section 4. Section 5 and Section 6 prove the equivalence between the usual definition of interleaving distance and the problem of finding optimal couplings.

In the second part of the manuscript, we leverage our novel formulation to introduce a constrained variant of the interleaving distance in Section 7, followed by the development of decomposition procedures for the interleaving distance in Section 8. We then present estimation methods for the interleaving distance in Section 9, and evaluate their performance through experiments described in

Section 10. Finally, Section 11 concludes the main body of the paper with a brief discussion of our findings.

Appendix A contains two flowcharts, which can help the reader navigating the structure of the manuscript and Table 3, which collects most of the main pieces of notation introduced in the manuscript, and can help the reader navigating the most technical sections. Appendix B features a comparison of our algorithms with the ones of [3] and [25]. Appendix C presents a further simulation dealing with the computational runtimes of the current implementation of our methods. Appendix D contains some of the proofs of the results in the manuscript, while the other ones are reported right after the statement they relate to, as they can help the reader in following the discussion.

2. Merge trees and interleaving distance

We start by introducing merge trees as combinatorial objects along with their "continuous" counterpart.

2.1. Merge trees as finite graphs

In accordance with other authors, we call a merge tree a rooted tree with a suitable height function defined on its vertices. We now state the formal definition, introducing some pieces of notation which we will use to work with those objects throughout the paper.

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

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

Given finite sets A, B, we indicate with $\#A$ the cardinality of one set and with $A - B := \{a \in A \mid a \notin B\}$. Moreover, given a finite poset P, we indicate with $\max P$ the set of all its maxima, and $\min P$ the set of all its minima. Given $A \subset V_T$ and $v \in V_T$, we may also write $\max_{v'<v} A$ to indicate the maximal elements in the set $\{v' \in A \mid v' < v\}$ according to the poset structure inherited from V_T.

Identifying an edge (v, v') with its lower vertex v gives a bijection between $V_T - \{r_T\}$ and E_T, that is, $E_T \simeq V_T - \{r_T\}$ as sets. Given this bijection, we may use E_T to indicate the vertices $v \in V_T - \{r_T\}$ to simplify the notation.

Now, we give the notion of least common ancestor between vertices.

Definition 2. Given a set of vertices $A = \{a_1, \ldots, a_n\} \subset V_T$, we define $\mathrm{LCA}(a_1, \ldots, a_n) = \min \bigcap_{i=1}^{n} \{v \in V_T \mid v \geq a_i\}$.

Now, to obtain a merge tree, we add an increasing height function to a tree structure.

Definition 3. A merge tree (T, f) is a finite tree structure T such that the root is of degree > 1, coupled with a monotone increasing function $f : V_T \to \mathbb{R}$.

Remark 1. Definition 3 is slightly different from the definition of merge trees found in [28], [40], and other works. In particular, in the present work, we do not need to have a root at infinity and thus, for us, the root is the highest vertex with finite height. Similarly, the function coupled with the tree structure in literature is usually referred to as h_T being a "height" function. To avoid overloading the notation, since we need to introduce many subscripts and superscripts, we call these functions with more usual functional notations like f or g.

Assumption 1. To avoid formal complications, we make the following standard genericity assumption for any merge tree (T, f):

(G) $f : V_T \to \mathbb{R}$ is injective.

Example 1. Given a real valued Morse [37] function $f : X \to \mathbb{R}$ defined on a path connected compact space X, the *sublevel set* filtration is given by $X_t = f^{-1}((-\infty, t])$, together with the maps $X_{t<t'} = i : f^{-1}((-\infty, t]) \hookrightarrow f^{-1}((-\infty, t'])$. We can describe the family of sets and maps $\{\pi_0(X_t)\}_{t\in\mathbb{R}}$ via a merge tree. See, for instance, [16, 40]. Similarly, we can consider a finite set $C \subset \mathbb{R}^n$ and take its *Céch* filtration: $X_t = \bigcup_{c\in C} B_t(c)$, with $B_t(c) = \{x \in \mathbb{R}^n \mid\mid\mid c - x \mid\mid< t\}$. As before: $X_{t<t'} = i : \bigcup_{c\in C} B_t(c) \hookrightarrow \bigcup_{c\in C} B_{t'}(c)$ and $\{\pi_0(X_t)\}_{t\in\mathbb{R}}$ can be represented via a merge tree.

Before proceeding, we need one last graph-related definition which we use to denote some particular kinds of paths on the tree structure of a merge tree.

Definition 4. Given a merge tree T, a sequence of edges is an ordered sequence of adjacent edges $\{e_1, \ldots, e_n\}$. This means that we have $e_1 < \ldots < e_n$, according to the order induced by the bijection $E_T \leftrightarrow V_T - \{r_T\}$, and that e_i and e_{i+1} share a vertex. We will use the notation $[v, v']$ to indicate a sequence of edges which starts in the vertex v and ends before v', with $v < v'$. Thus, the edge associated to v (via $E_T \simeq V_T - \{r_T\}$) is included in the sequence, while the one associated to v' is the first one excluded.

In the left column of Figure 1, the reader can find an example of a merge tree, which can be used to get familiar with the definitions just given. For instance, one can check that, using the notation of the figure, $e = \mathrm{LCA}(a, b, d)$ and $[a, c, r]$ is an ordered sequence of edges, while $[c, r, d]$ is not.

2.2. Metric merge trees

Now we consider the continuous version of a merge tree, intuitively obtained by considering all the points in the edges of a merge tree as points of the tree itself. For a visual intuition of the following ideas, the reader may refer to Figure 1.

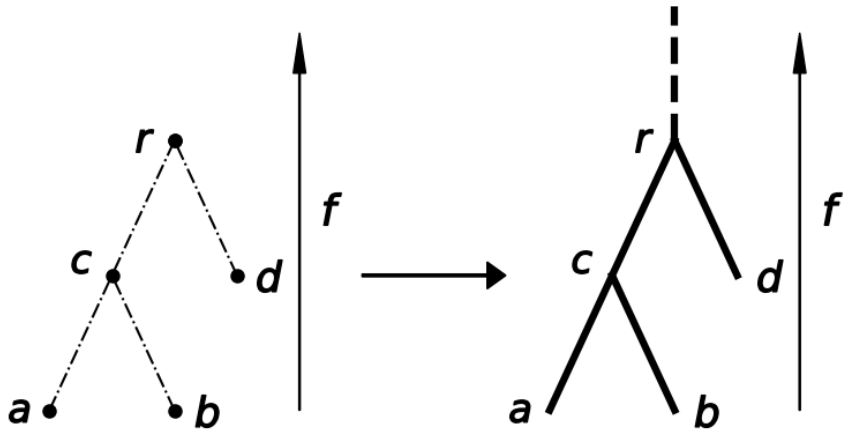


Figure 1. A merge tree (left) with its associated metric merge tree (right).

Definition 5. Given a merge tree T, we obtain the associated metric merge tree as follows:

$$\mathbf{T} = \left([f(r_T), +\infty) \sqcup \left(\bigsqcup_{(x,x') \in E_T} [f(x), f(x')] \right) \right) / \sim,$$

where, for every $v \in V_T$, $f(v) \sim f(v')$ if $v = v'$. We refer to the points in $\mathbf{T}$ with the same notation we use for vertices in v_T: $x \in \mathbf{T}$; with $f(x)$, we indicate the height values of x.

For every point $x \in V_T$, we can identify x with the point $f(x) \in [f(x), f(x')]$, with $(x, x') \in E_T$, or, equivalently, with $f(x) \in [f(x'), f(x)]$, if $(x', x) \in E_T$. This induces a well-defined map $V_T \hookrightarrow \mathbf{T}$. Thus, given $v \in V_T$, with an abuse of notation, we can consider $v \in \mathbf{T}$. Note that the height function $f : \mathbf{T} \to \mathbb{R}$ extends the function defined on the vertices of the merge tree. Every point in $\mathbf{T} - V_T$ belongs to one, and only one, interval of the form $[f(x), f(x')]$. Thus, we can induce a partial order relationship on $\mathbf{T}$ by ordering points first with the partial order in E_T and then with the increasing internal order of $[f(x), f(x')]$. Thus, we can explicitly write down the shortest path metric in $\mathbf{T}$: $d(x, x') = f(\mathrm{LCA}(x, x')) - f(x) + f(\mathrm{LCA}(x, x')) - f(x')$.

Remark 2. (Notation and previous works) If we start from a morse function $f : D \to \mathbb{R}$ defined on a compact and path connected topological space D, what we call "metric merge tree" is essentially the "merge tree" of f, according to the topological definition given in [9], also called classical merge tree in [16]. Moreover, display posets of persistent sets, as defined in [16], are up to some additional hypotheses equivalent to metric merge trees. However, 1) we are interested at working on merge trees in a combinatorial fashion, so we build merge trees as graphs and then bridge to this continuous construction; and 2) in order to use display posets, we would need to give other technical definitions, which we do not need in this work; thus, we avoid working with such objects. To sum up, all these procedures/definitions are different ways to build an $\mathbb{R}$-space $(X, f_X : X \to \mathbb{R})$ with f_X continuous function [18] with the underlying topological space X being the CW-complex associated to a tree. In particular, the display poset highlights the poset structure of such topological space while the approach in [9] describes a geometric construction, which gives the display poset a natural topology.

For each metric tree $\mathbf{T}$, we have a family of continuous maps, $s_T^k : \mathbf{T} \to \mathbf{T}$, with $k \geq 0$, called *structural maps*, defined as follows: $s_T^k(x) = x'$ with x' being the only point in $\mathbf{T}$ such that $x' \geq x$ and $f(x') = f(x) + k$.

2.3. Interleaving distance between merge trees

Now, we recall the main facts about the interleaving distance between merge trees, relying on [9].

Definition 6. [9] Two continuous maps $\alpha : \mathbf{T} \to \mathbf{G}$ and $\beta : \mathbf{G} \to \mathbf{T}$ between two metric merge trees, $(\mathbf{T}, f)$ and $(\mathbf{G}, g)$, are ε-compatible, with $\varepsilon \geq 0$, if the following hold:

(I1) $g(\alpha(x)) = f(x) + \varepsilon$ for all $x \in \mathbf{T}$ and $f(\beta(y)) = g(y) + \varepsilon$ for all $y \in \mathbf{G}$;

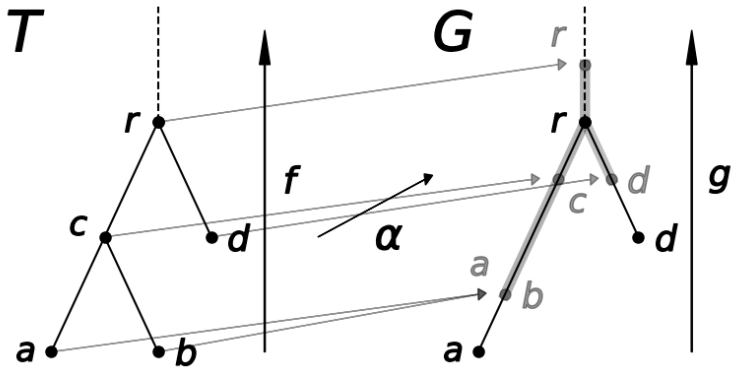
(I2) $\alpha\beta = s_G^{2\varepsilon}$ and $\beta\alpha = s_T^{2\varepsilon}$.

The interleaving distance between $\mathbf{T}$ and $\mathbf{G}$ is then:

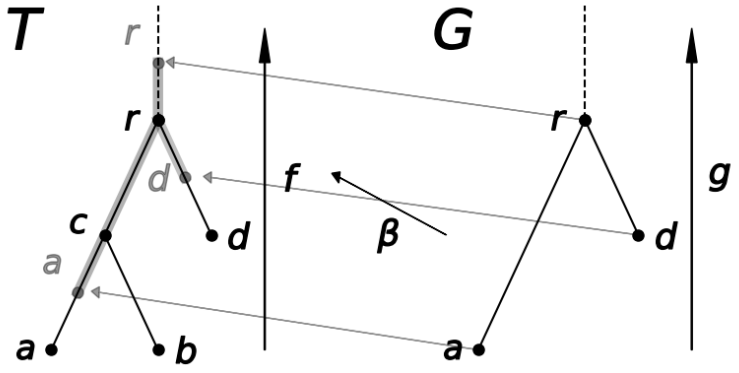
$$d_I(\mathbf{T}, \mathbf{G}) = \inf\{\varepsilon \mid \text{there are } \alpha, \beta, \ \varepsilon\text{-compatible maps}\}.$$

For an example of continuous maps α and β satisfying *(I1)*, see Figure 2. Such maps also satisfy *(I2)*, as shown by Figure 3.

The work of [25] shows that the existence of α and β is in fact equivalent to the existence of a single map α with some additional properties, which are stated in the next definition.



(a) A graphical representation of a continuous function $\alpha : \mathbf{T} \to \mathbf{G}$ between metric merge trees satisfying condition *(I1)* in Definition 6.



(b) A graphical representation of a continuous function $\beta : \mathbf{G} \to \mathbf{T}$ between metric merge trees satisfying condition *(I1)* in Definition 6.

Figure 2. An example of maps α and β giving the ε-interleaving of two metric merge trees.

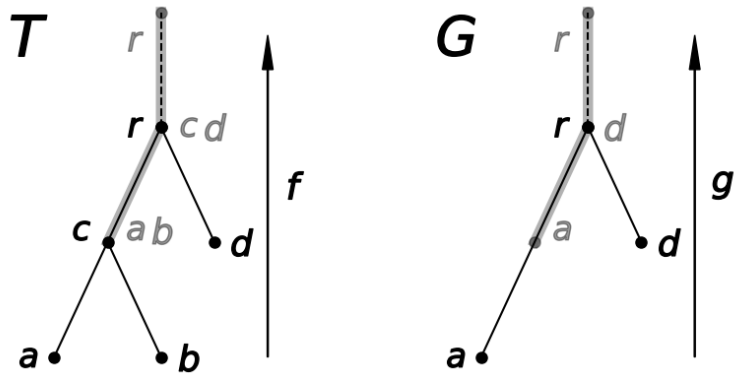


Figure 3. A representation of the images of the maps $\beta\alpha$ (left) and $\alpha\beta$ (right), with α and β being the maps in Figure 2.

Definition 7. [25] Given two metric merge trees $(\mathbf{T}, f)$ and $(\mathbf{G}, g)$, an ε-good map $\alpha : \mathbf{T} \to \mathbf{G}$ is a continuous map such that:

(P1) $g(\alpha(x)) = f(x) + \varepsilon$ for all $x \in \mathbf{T}$;

(P2) if $\alpha(x) < \alpha(x')$, then $s_T^{2\varepsilon}(x) < s_T^{2\varepsilon}(x')$;

(P3) if $y \in \mathbf{G} - \alpha(\mathbf{T})$, then, given $w = \min\{y' > y \mid y' \in \alpha(\mathbf{T})\}$, we have $g(w) - g(y) \leq 2\varepsilon$.

As anticipated, [25] proves that two merge trees are ε-interleaved if, and only if, there is an ε-good map between them.

3. Couplings

Now, we start our combinatorial investigation. Across the following sections, we need to introduce several pieces of notation, which are also summarized in Table 3 in the appendix.

Given two merge trees T and G, we would like to match their vertices and compute the interleaving distance relying only on such matches. To match the two graphs, we will use a set $C \subset V_T \times V_G$, which will tell us which vertex is coupled with which. For such C, we call $\pi_T : C \to V_T$ the restriction of the projection of the cartesian product. Since V_T and V_G are posets, we can introduce a partial order relationship on any $C \subset V_T \times V_G$, having $(a, b) < (c, d)$, if and only if, $a < c$ and $b < d$. Thus, as for each of the sets V_T, V_G, and any subset of those sets, we can consider the set of the maximal elements and the set of the minimal elements of C (and its subsets), which we indicate with $\max C, \min C$, etc.

Clearly, a coupling C must satisfy some constraints, which we now introduce.

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

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

Definition 8. A coupling between two merge trees (T, f) and (G, g) is a set $C \subset V_T \times V_G$ such that:

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

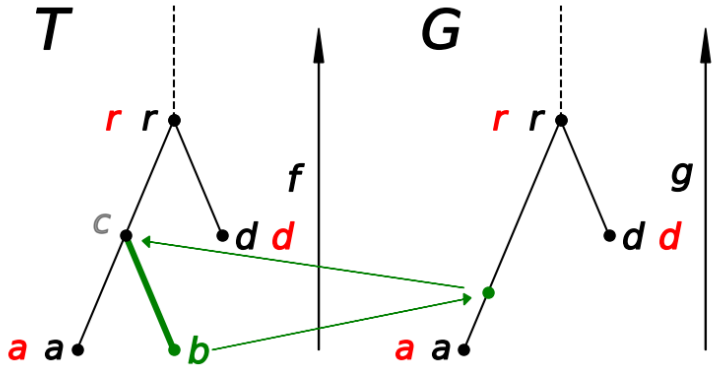
(C2) the projection $\pi_T : C \to V_T$ is injective (the same for G);

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

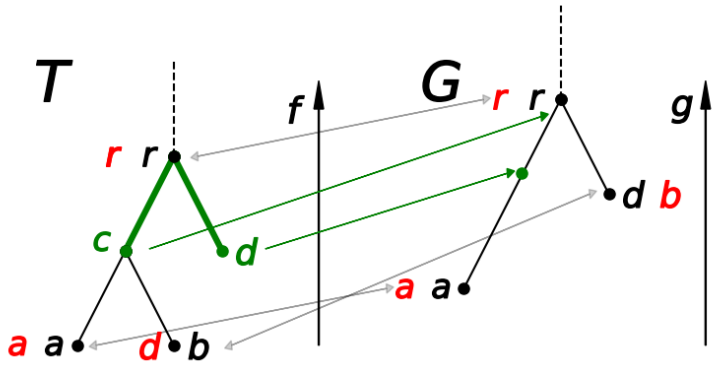
(C4) $a \in \pi_T(C)$ implies $\#\Lambda_C^T(a) \neq 1$ (the same for G).

The set of couplings between T and G is called $C(T, G)$. Note that, thanks to (C3), (C1) can be replaced by $\#\max \pi_T(C) = \#\max \pi_G(C) = 1$.

We invite the reader to follow the remaining of the section looking at Figure 4, which can help in understanding the comments we present on properties (C1)–(C4) and visualizing the pieces of notation we need to establish.



(a) The couples of adjacent letters red and black indicate a coupling between T and G. The vertex c in gray represents the only vertex with $\#\Lambda(c) = 1$, and belongs to U^T. Instead, the leaf b belongs to D^T and, clearly, $\#\Lambda(b) = 0$. The green arrows suggest the path of the two-step deletion of b; note that $\varphi^T(b) = c$, and so $\eta^T(b) = a$.



(b) The gray double-headed arrows, reinforced by the couples of adjacent letters (red and black), represent the coupling $C' = \{(a, a), (b, d), (r, r)\} \in C(T, G)$. The green edges are the deleted ones: vertex $d \in V_T$ is deleted, as in Figure 4(a), with $\#\Lambda(d) = 0$, while vertex $c \in V_T$ is deleted with $\#\Lambda(c) = 2$. The green arrows represent where such vertices are sent by the associated map $\alpha_{C'}$ introduced in Section 4. In particular, note that $\chi^T(c) = \mathrm{LCA}(\{a, b\}) = r$, $\varphi^T(d) = r$, and $\eta^T(d) = \gamma^T(r) = a \in V_G$.

Figure 4. Two examples of couplings, displaying all possible cases of unused vertices and deletions.

We collect the set of points $v \in E_T$ such that $\#\Lambda_C^T(v) = 1$ in a set, which we call U_C^T. Instead, the

points such that $v \notin \pi_T(C)$ and $\#\Lambda_C^T(v) \neq 1$ are collected in a set D_C^T. Note that the sets $\pi_T(C)$, U_C^T and D_C^T are a partition of V_T.

A couple $(v, w) \in C$ means that we match a vertex $v \in V_T$ with a vertex $w \in V_G$. It is clear from the definition that there can be vertices left out from this matching. We regard these vertices as unnecessary for the coupling C: when we will induce maps between metric merge trees starting from couplings, the position of these vertices inside the metric merge tree $\mathbf{G}$ will be completely induced by other vertices in V_T. Among the vertices not appearing in the couples of C, we distinguish between two situations: having a vertex v in D_C^T, informally, means that the edge $(v, father(v))$ except in certain special cases is not contained in the image of $\beta(\alpha(\mathbf{T}))$, with α and β as in Definition 6. For this reason, we say that the vertices in D_C^T and D_C^G are deleted by the coupling C. Instead, the vertices $v \in U_C^T$ are vertices which are unused, ignored by the coupling, and will be of no importance in the computation of the distance.

In this context, property (C1) is asking that there is a unique maximal couple in C, while (C2) is asking that each vertex is paired with at most one other vertex of the other merge tree. Note the maps $\alpha : \mathbf{T} \to \mathbf{G}$ in Definition 6 are not forced to be injective, but, as we will prove in later sections, for our purposes it is enough to couple points just one time. Condition (C3) is asking that the coupling respects the tree structures of T and G; in particular, this implies that $(\max \pi_T(C), \max \pi_G(C)) \in C$. Lastly, due to condition (C4), we avoid coupling vertices which have only one coupled element below them.

Given a coupling, we want to associate to a vertex a cost which indicates how much the coupling moves that vertex. In order to do so, we define the following functions:

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

$$\gamma_C^T(x) = \begin{cases} \arg\min\{g(w) \mid (v, w) \in C, v < x\}, & \text{if } \#\{g(w) \mid (v, w) \in C, v < x\} > 0 \\ \emptyset, & \text{otherwise.} \end{cases}$$

Note that if $\#\{g(w) \mid (v, w) \in C, v < x\} > 0$, by (G), $\gamma_C^T(x)$ is uniquely defined, and so γ_C^T is not a multivalued function. Moreover, $\gamma_C^T(\varphi_C^T(x))$ is well-defined for any $v \in V_T$.

Remark 3. When clear from the context, to lighten the notation, we might omit the subscripts and superscripts. For instance, if we fix $C \in \mathcal{C}(T, G)$ and $x \in V_T$, we refer to $\gamma_C^T(\varphi_C^T(x))$ as $\gamma(\varphi(x))$. Similarly, we use $x \in U$, $x \in D$, respectively, for $x \in U_C^T$ and $x \in D_C^T$.

Given $x \in V_T$, in what follows we will encounter the following objects (see Figure 4),

- $\eta(x) := \gamma(\varphi(x))$ is the vertex in V_G obtained in this way: starting from x, we go toward r_T until we meet the first point of V_T which has at least one vertex in its subtree which is not deleted; we consider the subtree of such a vertex and take all the vertices of V_G which are coupled with elements in this subtree (such a set is not empty); lastly, we take the vertex with the lowest height among this set;

- Given x such that $x \in D$ and $\#\Lambda(x) > 1$, we often consider $\delta(x)$; note that for every v such that $x \leq v < \delta(x)$, then $v \in D$ and $\#\Lambda(v) > 1$. Thus, $[x, \delta(x)]$ contains only deleted vertices;
- Similarly, for x such that $x \in D$ and $\#\Lambda(x) > 1$, we take $\chi(x)$: that is, the lowest point in G, where x can be sent compatibly with C and the tree structures of T and G. Thus, if $(x, y) \in C$, $y \geq \chi(x)$.

Remark 4. As anticipated in the introduction, the definition of couplings has strong connections with the definition of mappings given in [41] and which is used to compute the edit distance therein defined. Such definition generalizes more classical definitions of mappings, which are ubiquitous in the literature of edit distances for trees. In particular, one can show that given a coupling $C \in \mathcal{C}(T, G)$, if we add to C:

- Couples of the form $(a, \mathfrak{D})$ for every $a \in V_T$ such that a is deleted, i.e., $a \notin \pi_T(C)$ and $\#\Lambda_C^T(a) \neq 1$;
- Couples of the form $(\mathfrak{D}, b)$ for every $b \in V_G$ such that b is deleted, i.e., $b \notin \pi_G(C)$ and $\#\Lambda_C^G(b) \neq 1$;
- Couples of the form $(a, \mathfrak{G})$ for every $a \in V_T$ such that a is unused, i.e., $a \notin \pi_T(C)$ and $\#\Lambda_C^T(a) = 1$;
- Couples of the form $(\mathfrak{G}, b)$ for every $b \in V_G$ such that b is unused, i.e., $b \notin \pi_G(C)$ and $\#\Lambda_C^G(b) = 1$.

We obtain a set M_C, which is a mapping in the sense of [41].

4. Couplings, maps, and costs

We now start introducing our novel formulation of the interleaving distance. We do so in a 3 step procedure which is summarized in Figure 14(a) in the appendix.

In this section, we establish some correspondence between couplings and maps $\alpha : \mathbf{T} \to \mathbf{G}$, giving also the definition of the cost of a coupling. As a first step, given a coupling C, we induce two maps $\alpha_C : V_T - U_C^T \to \mathbf{G}$ and $\beta_C : V_G - U_C^G \to \mathbf{T}$, following these rules:

1) If $(x, y) \in C$, then $\alpha_C(x) := y$.
2) $x \in D_C^T$ with $\#\Lambda(x) = 0$, then $\alpha_C(x) := s_T^k(\eta(x))$ with k being:
   - $k = f(x) + \frac{1}{2}(f(\varphi(x)) - f(x)) - g(\eta(x))$ if $g(\eta(x)) \leq f(x) + \frac{1}{2}(f(\varphi(x)) - f(x))$;
   - $k = 0$, otherwise.

   Where the idea is that we want to send x above some coupled vertex changing its height "as little as possible". Bear in mind that the sequence $[x, \varphi(x)]$ should not appear in the image of $\beta\alpha$.
3) If instead $x \in D_C^T$ with $\#\Lambda(x) > 1$, then $\alpha_C(x) := \chi(x)$.

The map β_C is obtained in the same way, exchanging the roles of (T, f) and (G, g). Figure 5(b) shows an example in which β_C is obtained starting from a coupling.

As a first result, we obtain that the maps we induce respect the tree structures of the respective merge trees.

Proposition 1. Consider $x, x' \in V_T - U_T$: if $x < x'$, then $\alpha_C(x) \leq \alpha_C(x')$.

Now, we need to extend α_C to a continuous function between $\mathbf{T}$ and $\mathbf{G}$. To formally extend the map, we need the following lemma.

Lemma 1. Any $x \in \mathbf{T}$ such that $x \notin \pi_T(C) \cup D$, is contained in a sequence of edges $[l_C(x), u_C(x)]$ such that $l_C(x) \in \max\{v \leq x \mid v \in \pi_T(C) \cup D\}$, and $u_C(x)$ is either $u_C(x) = +\infty$ or $u_C(x) = \min\{v \geq x \mid v \in \pi_T(C) \cup D\}$. Moreover, $l_C(x)$ is unique if $\{v \in U \mid l_C(x) \leq v \leq x\} = \emptyset$, meanwhile, if $\{v \in U \mid l_C(x) \leq v \leq x\} \neq \emptyset$, there exists a unique $l_C(x)$ such that $l_C(x) \notin D$.

Now, we can obtain a continuous map $\alpha_C : \mathbf{T} \to \mathbf{G}$.

Proposition 2. We can extend α_C to a continuous map between $\mathbf{T}$ and $\mathbf{G}$. This map, with an abuse of notation, is still called α_C.

Proof. We define $\alpha_C(x)$ for $x \notin \pi_T(C) \cup D_C^T$. By Lemma 1, we have $x \in [l(x), u(x)]$, with α_C being defined for $l(x)$ and $u(x)$. Whenever $l(x)$ is not unique, we take the unique $l(x) \in \pi_T(C)$. Clearly, we have $f(l(x)) < f(x) < f(u(x))$. Thus, if $u(x) < +\infty$, there is a unique $\lambda \in [0, 1]$ such that $f(x) = \lambda f(l(x)) + (1 - \lambda) f(u(x))$. Having fixed such λ, we define $\alpha_C(x)$ to be the unique point in $[\alpha_C(l(x)), \alpha_C(u(x))]$, which is a sequence of edges thanks to Proposition 1 with height $\lambda g(\alpha_C(l(x))) + (1-\lambda) g(\alpha_C(u(x)))$. If $u(x) = +\infty$, then $x > v$ with $v = \max \pi_T(C)$. Then, we set $\alpha_C(x) = y$ such that $y \geq \alpha_C(v)$ and $g(y) = g(\alpha_C(v)) + f(x) - f(v)$. Note that, for x, x' such that $u(x), u(x') = +\infty$, the map α_C always preserves distances, and we have $g(\alpha_C(x)) - f(x) = g(\alpha_C(v)) - f(v)$. Thus, at such points it is a continuous function.

Consider now a converging sequence $x_n \to x$ in $\mathbf{T}$. We know that definitively $\{x_n\}_{n\in\mathbb{N}}$ is contained in one or more edges containing x. Thus, we can obtain a finite set of converging subsequences by intersecting $\{x_n\}$ with such edges. With an abuse of notation, from now on, we use $\{x_n\}_{n\in\mathbb{N}}$ to indicate any such sequence. Each of those edges is contained in a unique sequence of edges of the form $[l(x'), u(x')]$, for some x' (up to, eventually, taking $l(x') \notin D$). Thus, $\{x_n\} \subset [l(x'), u(x')]$ induces a unique sequence $\{\lambda_n\} \subset [0, 1]$ such that $f(x_n) = \lambda_n f(l(x')) + (1 - \lambda_n) f(u(x'))$ and $\lambda_n \to \lambda_x$, with $f(x) = \lambda_x f(l(x')) + (1 - \lambda_x) f(u(x'))$. By Lemma 1, $\alpha_C(x) \in [\alpha_C(l(x')), \alpha_C(u(x'))]$. Moreover, by construction, $g(\alpha_C(x_n)) \to \lambda_x g(\alpha_C(l(x'))) + (1 - \lambda_x) g(\alpha_C(u(x'))) = g(\alpha_C(x))$. Thus, α_C is continuous. □

In order to start relating maps induced by couplings and ε-good maps, we define the *cost* of the coupling C, which is given in terms of how much the (continuous) $\alpha_C : \mathbf{T} \to \mathbf{G}$ moves the points of $\mathbf{T}$. To highlight the domains of the maps involved in the following definition, we make use of the identity map $\mathrm{Id}_{\mathbf{T}} : \mathbf{T} \to \mathbf{T}$, for a metric tree $\mathbf{T}$.

Definition 9. Given $C \in \mathcal{C}(T, G)$ and $x \in \mathbf{T}$, we define $cost_C(x) = \mid g(\alpha_C(x)) - f(x) \mid$. Coherently, we define $\| C \|_\infty = \max\{\| g \circ \alpha_C - f \circ \mathrm{Id}_{\mathbf{T}} \|_\infty, \| f \circ \beta_C - g \circ \mathrm{Id}_{\mathbf{G}} \|_\infty\}$.

Note that, given $C \in \mathcal{C}(T, G)$ and $x \in \mathbf{T}$, we have one of the following possibilities:

- If $(x, y) \in C$, $cost_C(x) = \mid f(x) - g(y) \mid$;
- If $x \notin \pi_T(C) \cup D$, $cost_C(x) \leq \max\{cost_C(l(x)), cost_C(u(x))\}$; in fact,

$$\begin{aligned} \mid g(\alpha_C(x)) - f(x) \mid &= \mid \lambda g(\alpha_C(l(x))) - \lambda f(l(x)) + (1 - \lambda) g(\alpha_C(u(x))) - (1 - \lambda) f(u(x)) \mid \\ &\leq \lambda cost_C(l(x)) + (1 - \lambda) cost_C(u(x)); \end{aligned}$$

- We are left with the case $x \in D$. We have two different scenarios:

a) If $\#\Lambda(x) = 0$, then $cost_C(x) = \max\{(f(\varphi(x)) - f(x))/2, g(\eta(x)) - f(x)\}$. We point out that $(f(\varphi(x)) - f(x))/2$ is the cost of deleting the path $[x, \varphi(x)]$ in two steps: when applying α we halve the distance between x and $\varphi(x)$, and then with β we obtain $f(\beta\alpha(x)) = f(\varphi(x))$. This can happen if below w (with $(\delta(x), w) \in C$) there is "room" to send x at height $f(x) + (f(\varphi(x)) - f(x))/2$; if, instead, $\eta(x)$ is higher than $f(x) + (f(\varphi(x)) - f(x))/2$, we simply send x to $\eta(x)$;

b) If $\#\Lambda(x) > 1$, we have $cost_C(x) = | f(x) - g(\chi(x)) |$.

As a consequence, we point out two facts:

1) For every $\alpha : \mathbf{T} \to \mathbf{G}$ such that, for every $x \in \pi_T(C) \cup D$, $\alpha(x) = \alpha_C(x)$, we have:

$$\| g \circ \alpha_C - f \circ \mathrm{Id}_{\mathbf{T}} \|_\infty \leq \| g \circ \alpha - f \circ \mathrm{Id}_{\mathbf{T}} \|_\infty .$$

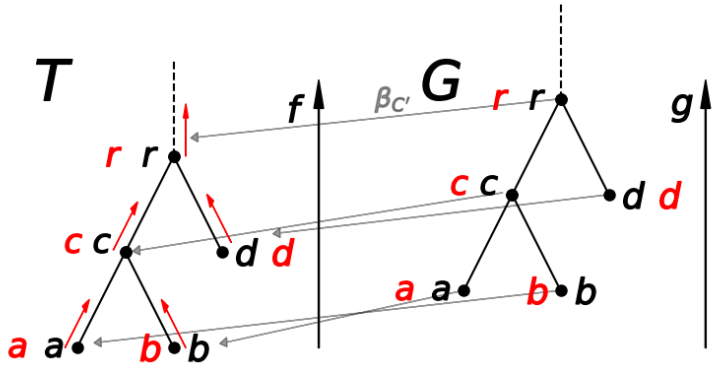
2) If we fix some total ordering of the elements in $V_T \coprod V_G$, so that we can write $V_T \coprod V_{T'} = (a_1, \ldots, a_n)$, then $\| C \|_\infty = \max(cost(a_1), \ldots, cost(a_n))$.

5. From couplings to ε-good maps

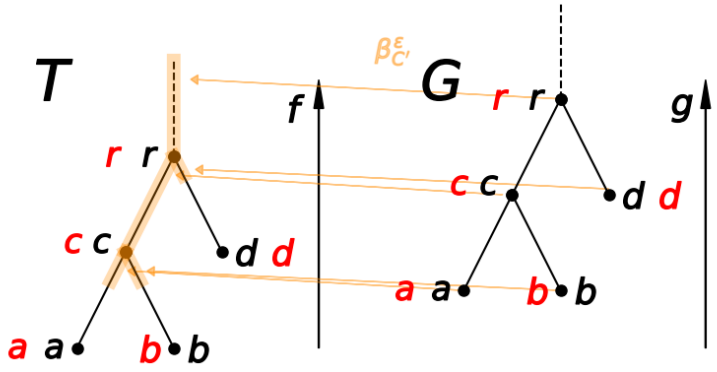
In this section, we prove the first fundamental interaction between couplings and the interleaving distance between merge trees, with our final goal being to replace the problem of finding optimal maps with the combinatorial problem of obtaining optimal couplings. Figure 5 summarizes all the steps of the procedure, which is formally addressed in this section. Throughout the section, we fix a coupling C and some $\varepsilon \geq \| C \|_\infty$.

We build a map $\alpha_C^\varepsilon : \mathbf{T} \to \mathbf{G}$, defined as $\alpha_C^\varepsilon(x) = s_G^{k_x}(\alpha_C(x))$ with $k_x = f(x) + \varepsilon - g(\alpha_C(x))$ depending on x. Note that, since $g(\alpha_C(x)) - f(x) \leq \epsilon$ for every $x \in \mathbf{T}$, we always have $k_x \geq 0$. Moreover, by construction, $g(\alpha_C^\varepsilon(x)) = g(\alpha_C(x)) + f(x) + \varepsilon - g(\alpha_C(x)) = f(x) + \varepsilon$.

We want to prove that α_C^ε is an ε-good map.



(a)



(b)

Figure 5. The red and black couples of adjacent letters indicate a coupling $C' = \{(a,b),(b,a),(c,c),(r,r)\} \in C(T,G)$. In Figure 5(a), the gray arrows represent the map $\beta_{C'} : V_G \to V_T$, which then is naturally extended to the metric trees. The image of such map is then shifted upward using the structure map s^k_T as in Figure 5(b), to obtain $\beta^{\varepsilon}_{C'}$, with the shift being indicated by the upward red arrows. The ε-good map $\beta^{\varepsilon}_{C'}$ is represented in Figure 5(b), via the orange arrows and the shaded orange portion of **T**.

Remark 5. The map α^{ε}_C is such that, given $x, x' \in \mathbf{T}$ with $x < x'$, we have $\alpha^{\varepsilon}_C(x) < \alpha^{\varepsilon}_C(x')$.

As a first step, we obtain that α^{ε}_C satisfies some necessary conditions in order to be ε-good, it is in fact continuous, and moves points "upward" by ε.

Proposition 3. The map α^{ε}_C is a continuous map between **T** and **G** such that for every $x \in \mathbf{T}$, we have $g(\alpha^{\varepsilon}_C(x)) = f(x) + \varepsilon$.

Before proving the main result of this section, we need a short lemma.

Lemma 2. Consider $(v,w),(v',w') \in C$, and take $x = \mathrm{LCA}(v,v')$, $y = \mathrm{LCA}(w,w')$. Then, $|\, f(x) - g(y)\, | \leq \varepsilon$.

At this point, we are ready to prove the remaining properties which make α^{ε}_C an ε-good map.

Theorem 1. The map α^{ε}_C defined in previous lines is an ε-good map.

6. From ε-good maps to couplings

This time, we start from an ε-good map $\alpha : \mathbf{T} \to \mathbf{G}$ and our aim is to induce a C with $\| C \|_\infty \leq \varepsilon$.

Given a metric merge tree $\mathbf{T}$, we define the following map: $\ell_T : \mathbf{T} \to V_T$ given by $\ell_T(x) = \max\{v \in V_T \mid v \leq x\}$. Note that, if $x \in V_T$, then $\ell_T(x) = x$; otherwise, x is an internal vertex of one edge (a, b) (possibly with $b = +\infty$) and $\ell_T(x) = a$. With this notation, we introduce the following maps:

$$\begin{aligned} \phi : V_T &\to V_G, & \psi : V_G &\to V_T, \\ v &\mapsto \ell_T(\alpha(v)), & w &\mapsto \ell_G(\beta(w)). \end{aligned}$$

Note that we always have $g(\phi(v)) \leq g(\alpha(v)) \leq f(v) + \varepsilon$. These maps will be pivotal in the proof of the upcoming theorem, since they will help us in closing the gap between the continuous formulation of the interleaving distance and the discrete matching of merge trees via couplings. The reader should refer to Figure 6 for a visual example of the above definitions.

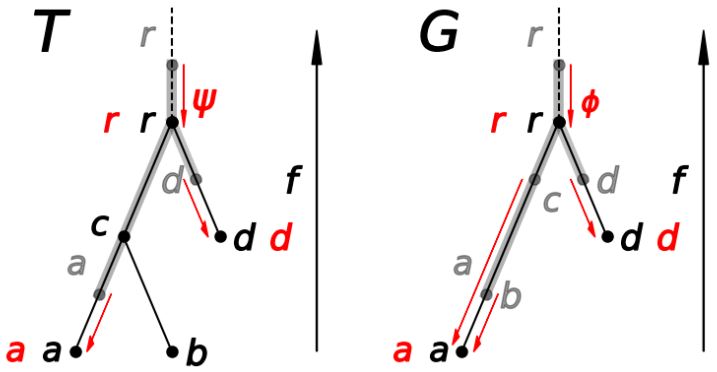


Figure 6. Given the two maps α, β of Figure 2, the shaded gray represents the image of those maps, the red arrows give the maps ψ (left) and ϕ (right), see Section 6, and the red letters next to the black ones indicate the coupling $\{(a, a), (d, d)\}$, satisfying Theorem 2. Such coupling is compatible with the procedure outlined in the proof of Theorem 2 (upon perturbing T to meet the generality condition (G)).

We prove a corollary which characterizes the maps we just defined.

Corollary 1. Let $v, v' \in V_T$; if $v < v'$, then $\phi(v) \leq \phi(v')$.

Clearly, an analogous result holds for ψ, which implies that, in the setting of the corollary, $\psi(\phi(v)) \leq \psi(\phi(v'))$.

Now, we prove the main result of this section.

Theorem 2. Given α ε-good map between $\mathbf{T}$ and $\mathbf{G}$, there is a coupling C between T and G such that $\| C \|_\infty \leq \varepsilon$.

Putting together Theorems 1 and 2, we see that two merge trees are ε-interleaved if, and only if, there is a coupling C between the merge trees such that $\| C \|_\infty = \varepsilon$. Thus, computing the interleaving distance amounts to finding a minimal-cost coupling between two merge trees. As a byproduct of this

result, we obtain another proof that the interleaving distance between metric merge tree can be found as a minimum and not just as an infimum as in Definition 6, for the set of available couplings is finite.

We close this section with a conjecture which we want to investigate in future works, which stems from one of the definitions of the GH distance [1] and the already exploited link between such distance in the case of metric merge trees and the interleaving distance [25].

Given any map $\alpha : \mathbf{T} \to \mathbf{G}$, we define $H(\alpha) := \max_{x\in\mathbf{T}} \mid g(\alpha(x)) - f(x) \mid$. Similarly, for any map $\mu : \mathbf{T} \to \mathbf{T}$, which thus rearranges the points of $\mathbf{T}$, we set $2 \cdot D(\mu) = \max_{x\in\mathbf{T}} d(x, \mu(x))$ (where d is the shortest path metric on $\mathbf{T}$).

Conjecture 1. We conjecture that:

$$d_I(\mathbf{T}, \mathbf{G}) = \min_{\alpha,\beta}\{H(\alpha), H(\beta), D(\beta\alpha), D(\alpha\beta)\},$$

where $\alpha : \mathbf{T} \to \mathbf{G}$ and $\beta : \mathbf{G} \to \mathbf{T}$ vary in the set of continuous monotone maps.

7. Constrained interleaving distance

In this section, exploiting couplings, we define a variation of the interleaving distance, which we call constrained interleaving distance, following the literature on tree edit distances. In particular, we borrow the constrained condition for couplings from [54].

Definition 10. A coupling C is constrained if the following hold:

(CONS) for every $a, b, c \in E_T$ and $a', b', c' \in E_{T'}$ such that $(a, a'), (b, b'), (c, c') \in C$, we have:

$$\mathrm{LCA}(\{a, b\}) \geq c \text{ if and only if } \mathrm{LCA}(\{a', b'\}) \geq c'.$$

Restricting the optimization domain to constrained mappings, we obtain the *constrained* interleaving distance.

Definition 11. We define the constrained interleaving distance between two merge trees T, T' as:

$$d_I^c(T, T') := \min\{\| C \|_\infty \mid C \in C(T, T') \text{ and } C \text{ is constrained}\}.$$

As already mentioned, constrained definitions of mappings have been successfully employed to bring the computational cost of edit distances for merge trees into the realm of polynomial time algorithms [43, 48, 49, 52, 53]. Since our couplings are closely related to the mappings employed in [39, 52] (see Remark 4), we intend to use this formulation as a pivotal building block to obtain practically accessible and reliable upper bounds also for the interleaving distance.

We point out that the definition of d_I^c depends heavily on the formulation of the interleaving distance via couplings, as condition (CONS) is much harder to characterize via the maps α_C or α_C^ε. For instance, even if C is a constrained coupling, we may have a vertex c with two children $\{a, b\}$ such that b is coupled, a is deleted, and $\alpha_C(a) \geq \alpha_C(b)$. In this case, we would have:

$$\alpha_C(a) = \mathrm{LCA}(\{\alpha_C(a), \alpha_C(b)\}) < \alpha_C(c) \text{ but } \mathrm{LCA}(\{a, b\}) = c.$$

The same scenario can be replicated also with α_C^ε.

Due to the challenges and technicalities arising in characterizing the maps α_C coming from a constrained coupling C, we leave a proper investigation of this constrained definition to future works. In particular, we conjecture that d_I^C is a metric between merge trees. Plus, together with the recursive procedure described in Section 8, we plan on using it to obtain a polynomial time algorithm viable for applications.

8. Recursive decomposition properties

Now, we exploit the formulation of the interleaving distance via couplings to obtain recursive decomposition properties inspired by the ones of edit distance between trees. In particular, we prove that, under some conditions, optimal couplings can be computed via the solution of smaller independent subproblems, which are then aggregated to solve the global one. If the aforementioned conditions do not apply, we still obtain lower and upper bounds for the interleaving distance.

The main result of the section involves couplings with some strong properties, which we call *special* and collect under the following notation:

$$C^o(T,G) := \{C \in C(T,G) \mid \arg\min_{v<\max \pi_T(C)} f(v) \in \pi_T(C) \text{ and } \arg\min_{w<\max \pi_G(C)} g(w) \in \pi_G(C)\}.$$

Before proceeding, we need a few pieces of notation used to lighten the dissertation and one last technical definition. Given $x \in V_T$ and $y \in V_G$, we define $T_x = sub_T(x)$ and $G_y = sub_G(y)$. Moreover, $C_R(T,G) := \{C \in C(T,G) \mid (r_T, r_G) \in C\}$. Similarly, we have $C_R^o(T,G) := C_R(T,G) \cap C^o(T,G)$.

Definition 12. Given a partially ordered set $(A, <)$, A is an anti-chain if for any $a, b \in A$, $a \neq b$, neither $b > a$ or $a > b$ hold. A subset of a poset is an anti-chain if it satisfies this condition with the restriction of the poset structure.

Now, we introduce the combinatorial objects we will use to decompose the following optimization problem: $\min_{C \in C(T,G)} \| C \|_\infty$. The reader may look at Figure 7 to find examples involving the following definition.

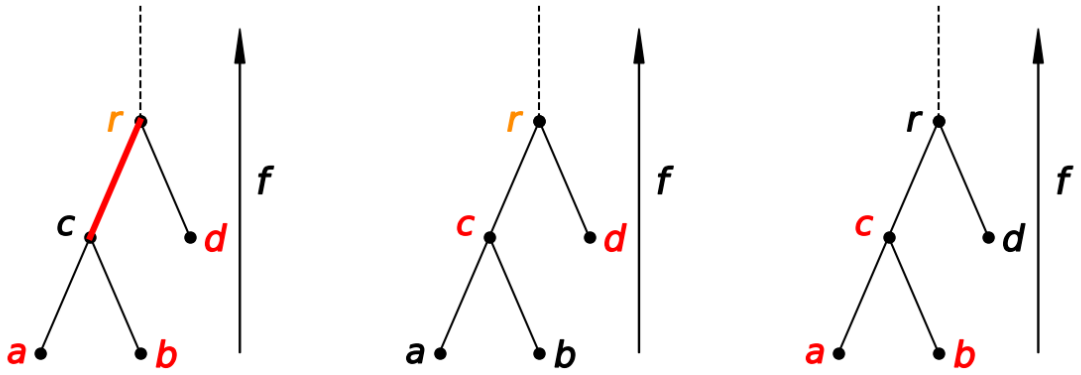


Figure 7. The letters in red show three examples of subsets of V_T such that: the two leftmost examples are anti-chains (see Definition 13), while the rightmost example is not. The red branch in the leftmost tree signifies that if we have some coupling C^* such that red vertices give the set $\pi_T(C^*)$, then $\#\Lambda_{C^*}(c) = 2$, and so c is deleted.

Definition 13. We define $C^*(T,G)$ as the set of all $C^* \subset V_T \times V_G$ such that:

(A0) $C^* \cup \{ (\mathrm{LCA}(\pi_T(C^*)), \mathrm{LCA}(\pi_G(C^*))) \} \in C(T,G)$;

(A1) C^* is an anti-chain.

The intuition behind the definition of $C^*(T,G)$ is that we are assuming that we already know $\arg\min\{\| C' \|_\infty | C' \in C_R(T_x, G_y)\}$ for every $x \in E_T$ and $y \in E_G$, and we use $C^* \in C^*(T,G)$ to optimally aggregate these results to find the optimal global coupling between T and G. We formalize such an idea in the following definition.

Definition 14. Let $C_{x,y} \in C_R(T_x, G_y)$ for every $x, y \in V_T \times V_G$. Given $C^* \in C^*(T,G)$, with $r = \mathrm{LCA}(\pi_T(C^*))$ and $r' = \mathrm{LCA}(\pi_G(C^*))$, we define the extension of C^* by means of $\{C_{x,y}\}_{(x,y)\in C^*}$ as:

$$e(C^*) = \{(r, r')\} \cup \left(\bigcup_{(x,y)\in C^*} C_{x,y} \right).$$

The set of all possible extensions of C^* is called $E(C^*)$. If $C_{x,y} \in \arg\min\{\| C' \|_\infty | C' \in C_R(T_x, G_y)\}$ for all x, y, then we call the extension minimal. We collect all minimal extensions of C^* in the set $E_m(C^*)$, which is never empty. If all $C_{x,y}$ and $e(C^*)$ are special couplings, then we call the extension special. We collect all special extensions of C^* in the set $E^o(C^*)$, which is never empty. We name $E^o_m(C^*) = E_m(C^*) \cap E^o(C^*)$ the set of minimal special extensions of C^*; this set could be empty.

Since extensions are couplings, it is obvious that:

$$d_I(T,G) \leq \min_{C^*\in C^*(T,G)} \min_{C\in E_m(C^*)} \| C \|_\infty . \tag{8.1}$$

$$d_I(T,G) \leq \min_{C^*\in C^*(T,G)} \min_{C\in E^o(C^*)} \| C \|_\infty . \tag{8.2}$$

See also Figure 8(a). Moreover, it is also clear that any coupling is an extension of some $C^* \in C^*(T,G)$ and, thus:

$$d_I(T,G) = \min_{C^*\in C^*(T,G)} \min_{C\in E(C^*)} \| C \|_\infty .$$

The upcoming theorem states that there are strong relationships between d_I and extensions obtained via a fixed family of $C_{x,y} \in C_R(T_x, G_y)$.

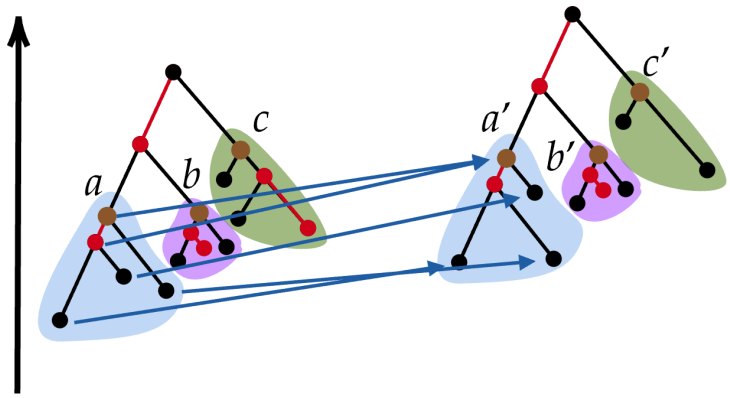
Theorem 3. (Decomposition) Consider two merge trees T and G and take a collection $\{C_{x,y}\}_{(x,y)\in V_T\times V_G}$ with $C_{x,y} \in \arg\min\{\| C' \|_\infty | C' \in C_R(T_x, G_y)\}$. Given $C^* \in C^*(T,G)$, we name $e(C^*)$ the extension obtained by means of $\{C_{x,y}\}$. We have:

$$\min_{C^*\in C^*(T,G)} \max\{cost_{e(C^*)}(v) \mid v \in \pi_T(e(C^*)) \text{ or } \#\Lambda_{e(C^*)}(v) > 0\} \leq d_I(T,G). \tag{8.3}$$

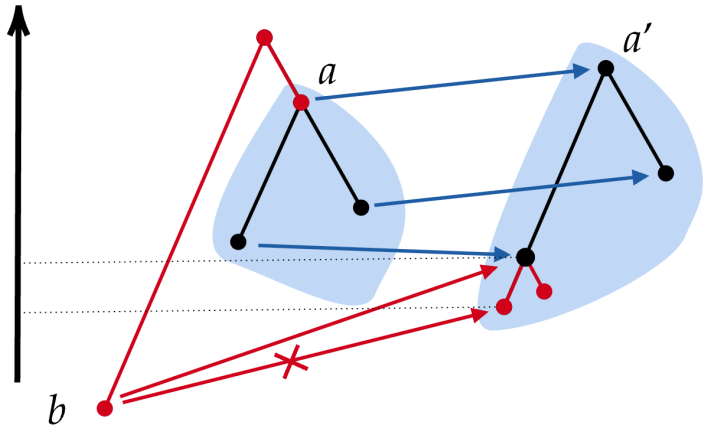
Moreover, if $C_{x,y}$ is also a special coupling for every x, y, we have:

$$d_I(T,G) = \min_{C^*\in C^*(T,G)} \| e(C^*) \|_\infty . \tag{8.4}$$

We remark that Theorem 3 is in some sense unexpected: if Eqs (8.1) and (8.2) are in some sense trivial, Eqs (8.3) and (8.4) are not. We highlight that the couplings $\{C_{x,y}\}$ are independently fixed at the beginning and not chosen with some joint optimization strategy. Moreover, the proof of Eq (8.4) depends strongly on the possibility to find optimal couplings which are also special.



(a) The couples $\{(a, a'), (b, b'), (c, c')\}$ form an anti-chain, both as couples and via π_T and π_G. The shaded regions indicate how the subtrees rooted in the vertices of the anti-chain are matched. Minimal couplings are displayed with deletions in red. The remaining vertices can be coupled by visual inspection. We have highlighted the couples of the blue subtree just as an example. Putting together all the couples, we obtain an extension of the anti-chain.



(b) Consider the anti-chain $C^* = \{(a, a')\}$: the coupling displayed by the blue arrows and the red deletions do not induce a special extension of C^*. As a result, the deletion of b is forced to follow a nonoptimal path, as the lowest vertex of the right tree is deleted.

Figure 8. Two figures related to decomposition and extensions of couplings.

9. Bounding the interleaving distance

The aim of this section is to exploit the coupling formulation of the interleaving distance and its properties to find ways to get upper and lower bounds for d_I. More in details: Section 9.1 resorts to pruning to reduce the complexity of the trees and finds upper bounds for d_I, while Section 9.2 takes a more complex approach: via a recursive decomposition of couplings obtains lower and upper bounds for d_I.

9.1. Pruning merge trees

We briefly introduce the pruning operator P_ε. Given a merge tree T and $\varepsilon > 0$, the merge tree $P_\varepsilon(T)$ is obtained through a recursive procedure: given a leaf x and its father x', if $f(x') - f(x) < \varepsilon$, we say that x is a small-weight leaf; we want to remove all small-weight leaves (unless two or more of them are siblings, i.e., children of the same father) and all degree 2 vertices from T. In this case, we want to remove all leaves but the one being the lowest leaf. To make this procedure well-defined and to make sure that, in the end, no small-weight leaves are left in the tree, we need to choose some ordering of the leaves and to resort to recursion.

(P) Take a leaf x such that $f(father(x)) - f(x)$ is minimal; if $f(father(x)) - f(x) < \varepsilon$, remove x and its father if it becomes a degree 2 vertex after removing x.

We take $T_0 = T$ and apply operation (P) to obtain T_1. On the result, we apply again (P) obtaining T_2, and we go on until we reach a fixed point $T_n = T_{n+1} = P_\varepsilon(T)$ of such sequence. To shed some light on this definition, we prove the following lemma; Figure 9 can be helpful in following the statements.

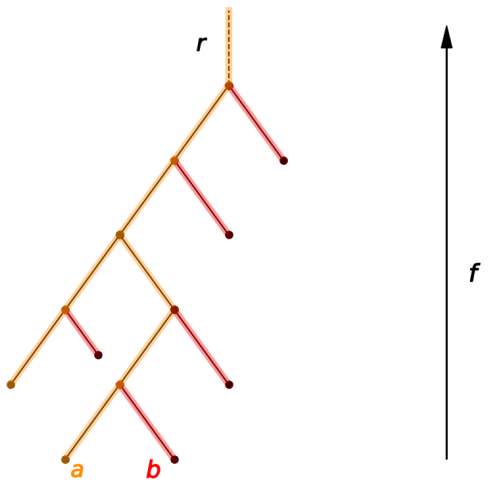


Figure 9. A pruning operator applied on a merge tree T. The red branches are removed from the tree, while the orange ones are kept and represent the metric merge tree $P_\varepsilon(T)$. We highlight that the root of the tree changes. Lastly, note that deleting a instead of b would give isomorphic merge trees.

Lemma 3. Given T, $\varepsilon > 0$, and $P_\varepsilon(T)$, we have a natural injective map $V_{P_\varepsilon(T)} \hookrightarrow V_T$ which identifies vertices in $P_\varepsilon(T)$ with vertices in T. Call $C_\varepsilon \subset V_{P_\varepsilon(T)} \times V_T$ the set of couples induced by $V_{P_\varepsilon(T)} \hookrightarrow V_T$.

The following hold:

1) C_ε is a coupling;

2) $L_{P_\varepsilon(T)} \subset L_T$, and for every v and v' such that $v' < v$ and $f(v) - f(v') \geq \varepsilon$, there is $x \in L_{P_\varepsilon(T)}$ such that $\mathrm{LCA}(x, v') < v$; in particular, $\arg\min f \in L_{P_\varepsilon(T)}$;

3) For every $v \in V_T - V_{P_\varepsilon(T)}$, we have $\#\Lambda^T_{C_\varepsilon}(v) \leq 1$; in particular, if $v \in D^T_{C_\varepsilon}$, we have $\#\Lambda^T_{C_\varepsilon}(v) = 0$ and $f(\varphi_{C_\varepsilon}(v)) - f(v) < \varepsilon$;

4) The map: $\eta_{C_\varepsilon} : D^T_{C_\varepsilon} \to V_T$ such that $f(\eta_{C_\varepsilon}(x)) < f(x)$ for all $x \in D^T_{C_\varepsilon}$;

5) $\| C_\varepsilon \|_\infty \leq \varepsilon/2$.

Having characterized the pruned trees with the above lemma, we can obtain the following proposition.

Proposition 4. Given two merge trees T and G, we have:

$$d_I(T, G) \leq \max\{d_I(P_\varepsilon(T), P_\varepsilon(G)), \varepsilon/2\}.$$

As a result, if the number of leaves of T and G is too high, we know that we can prune them, reducing the computational complexity of d_I, to obtain an estimate from above of $d_I(T, G)$.

We close this section with a conjecture we would like to investigate in the future.

Conjecture 2. The coupling C_ε is a minimizing coupling.

9.2. Recursive procedures for upper and lower bounds

In this section, we exploit Theorem 3 to obtain a dynamical approach to estimate, from above and below, an optimal coupling between two merge trees by solving recursively binary linear programming (BLP) problems.

9.2.1. Computing the cost of a coupling extension

As a first step, we separate the problem of finding $C^* \in C^*(T, G)$ with a cost-minimizing extension into two "independent" problems: the first one is to find a minimal-fixed root coupling, and the second one is to compute the cost of deleting the vertices which are not in the subtrees whose roots are fixed by C^*. To make the upcoming lemma more clear, we establish the following notation. Suppose $r = \mathrm{LCA}(\pi_T(C^*))$ and $r' = \mathrm{LCA}(\pi_G(C^*))$. Then for $v \not\leq r$, we set:

$$v_r = \min\{v' \geq r \text{ and } v' \geq v\},$$

and $f_r = \min_{v \leq r} f(v)$. Lastly, we define:

$$H_{r,r'} = \max\left\{\max_{v \not\leq r} \frac{f(v_r) - f(v)}{2}, \max_{v \not\leq r} g_{r'} - f(v), \max_{w \not\leq r'} \frac{g(w_{r'}) - g(w)}{2}, \max_{w \not\leq r'} f_r - g(w)\right\}. \tag{9.1}$$

Note that $H_{r,r'}$ does not depend on the chosen extension of C^*, and, in fact, it depends only on r, r'. The upcoming lemma states that $H_{r,r'}$ accounts for the deletions of all the vertices which are not below r or r'.

Lemma 4. Consider T, G merge trees and $C^* \in C^*(T,G)$, with $r = \mathrm{LCA}(\pi_T(C^*))$ and $r' = \mathrm{LCA}(\pi_G(C^*))$. Let $C_{r,r'}$ be any extension of C^*. With an abuse of notation, we consider $C'_{r,r'}$ as the coupling in $C(T_r, G_{r'})$ consisting of the same couples as $C_{r,r'}$. Then,

$$\| C_{r,r'} \|_\infty \geq \max\{\| C'_{r,r'} \|_\infty, H_{r,r'}\}. \tag{9.2}$$

If $C_{r,r'}$ is a special coupling, the inequality becomes an equality.

Since the computation of $H_{r,r'}$ can be easily accessed in a greedy fashion, from now on, we focus on estimating $C \in \arg\min\{\| C' \|_\infty | \, C' \in C_R(T,G)\}$.

9.2.2. Iterative approach

Now, we outline a procedure to estimate $C \in \arg\min\{\| C' \|_\infty | \, C' \in C_R(T,G)\}$. As in Theorem 3, we assume that we already have computed the couplings $C_{x,y} \in C_R(T_x, G_y)$ that we want to use to extend any $C^* \in C^*(T,G)$, as in Theorem 3. So for instance, if we want to work with special extensions, we assume that we have $C_{x,y} \in \arg\min\{\| C' \|_\infty | \, C' \in C^o_R(T_x, G_y)\}$ for all $x \in E_T$ and $y \in E_G$, and we exploit them to obtain a special extension of some $C^* \in C^*(T,G)$. If we want to work with minimal extensions, instead, we assume to have $C_{x,y} \in \arg\min\{\| C' \|_\infty | \, C' \in C_R(T_x, G_y)\}$. We anticipate that both kinds of extensions are important for our purposes: special extension will be used to produce upper bounds for the interleaving distance, while minimal extension will be used, exploiting Theorem 3, to obtain lower bounds.

As before, we set: $f_x = \min_{v \leq x} f(v)$ and $g_y = \min_{w \leq y} g(w)$. Lastly, we fix a constant $K > 0$ such that $K > \max_{x \in V_T, y \in V_G} | f(x) - g(y) |$. In Section 9.2.5, we point out which steps of the upcoming procedure may produce errors.

9.2.3. Variables and constraints

We consider the following set of binary variables: $a_{x,y}$ for $x \in E_T$ and $y \in E_G$; u_x for $x \in E_T$ and u_y for $y \in E_G$. The vector of all variables (upon choosing some arbitrary ordering) will be referred to as $\mathcal{V}$.

We also introduce the following auxiliary variables, which are linear functions of $a_{x,y}$, u_x, and u_y:

- $c_x = \sum_y a_{x,y}$ and $c_y = \sum_x a_{x,y}$;
- $\Lambda_x = \sum_{v \leq x} c_v$ and $\Lambda_y = \sum_{w \leq y} c_w$;
- $d_x = \sum_{v \leq x} c_v + \sum_{v > x} c_v$ and $d_y = \sum_{w \leq y} c_w + \sum_{w > y} c_w$.

Given such vatiable, we define the following linear constraints:

(1) For every $l \in L_T$: $\sum_{l \leq x < r_T} c_x \leq 1$ and for every $l' \in L_G$: $\sum_{l' \leq y < r_G} c_y \leq 1$;

(2) $u_x \leq 0.5\Lambda_x$ and $u_y \leq 0.5\Lambda_y$ for every x and y;

(3) $u_x \geq m_x\Lambda_x + q_x$ and $u_y \geq m_y\Lambda_y + q_y$ for every x and y. With m_x, q_x being any pair of (m,q) such that the following are satisfied: $q < 0$, $m + q < 0$, $2m + q > 0$, $m \cdot (\#L_T) < 1$ (analogously for m_y, q_y);

(4) (Only for special extensions) Let $\widetilde{x} = \arg\min_{v\in V_T} f(v)$ and $\widetilde{y} = \arg\min_{w\in V_G} g(w)$; then, we ask $\sum_{\widetilde{x}\le x<r_T} c_x \ge 1$ and $\sum_{\widetilde{y}\le y<r_G} c_y \ge 1$.

The set of vectors of variables which satisfy these constraints is called either $\mathcal{K}^m(T,G)$ or $\mathcal{K}^o(T,G)$, depending on whether constraints (4) are, respectively, included or not. Note that $\mathcal{K}^m(T,G) \supset \mathcal{K}^o(T,G)$. To lighten the notation, we often avoid explicit reference to T and G when talking about the set of possible solutions, and, when we do not wish to distinguish between $\mathcal{K}^m$ and $\mathcal{K}^o$, we simply write $\mathcal{K}$. Now, we briefly comment on the variables and constraints to try to give some intuition about their roles:

- The variables $a_{x,y}$ are used to match x with y, i.e., add the couple (x,y) to the coupling. In particular, constraints (1) ensure that, if $\mathcal{V} \in \mathcal{K}$, the set $C_{\mathcal{V}} := \{(x,y) \in E_T \times E_G \mid a_{x,y} = 1\}$ belongs to $C^*(T,G)$;
- The variables d_x and u_x are used to determine if x must be deleted. Start with d_x. The main idea, which our optimization procedure is based on, is that if $a_{v,w} = 1$ with $v > x$, it means that x is taken care of by the coupling $C_{v,w}$ and, thus, we want to "ignore" such x. In other words, having $d_x = 0$ implies that x is deleted with $\#\Lambda(x) = 0$;
- Now, we turn to u_x. Observe that, if $\Lambda_x \in \{0,1\}$, then $u_x = 0$ while if $\Lambda_x \ge 2$, $u_x = 1$. Note that $\Lambda_x \le \#L_T$ and $\Lambda_y \le \#L_G$. Constraints (2) and (3) are fundamental to fix the value of u_x, depending linearly on Λ_x: $u_x = 1$ if, and only if, $x \ge \mathrm{LCA}(v,v')$, with $a_{v,w} = 1$ and $a_{v',w'} = 1$, for some $w, w' \in V_G$. Thus, having $u_x = 1$ means that x is deleted with $\#\Lambda(x) > 1$;
- Lastly, we comment on constraints (4). These constraints are asking that there is one point above the lowest vertex in each tree which is coupled by $C_{\mathcal{V}}$; this in particular implies that, if we assume $C_{x,y} \in C^o(T_x, G_y)$, then $\widetilde{x}$ and $\widetilde{y}$ (with $\widetilde{x} = \arg\min_{v\in V_T} f(v)$ and $\widetilde{y} = \arg\min_{w\in V_G} g(w)$) are never deleted. Thus, $\{(r_T, r_G)\} \cup \left(\bigcup_{(x,y)\in C_{\mathcal{V}}} C_{x,y}\right)$ is a special extension, i.e., a coupling in $C^o_R(T,G)$.

As a consequence of these observations, any $C \in C^*(T,G)$ induces a unique vector of variables $\mathcal{V} \in \mathcal{K}$, such that $C_{\mathcal{V}} = C$. Moreover, if $C_{x,y} \in C^o(T_x, G_y)$ and $\mathcal{V} \in \mathcal{K}^o$, then $\{(r_T, r_G)\} \bigcup_{(x,y)\in C_{\mathcal{V}}} C_{x,y}$ is a special extension of $C_{\mathcal{V}}$.

9.2.4. Objective functions

A key fact that we highlight is that, by property (A1), if $x = \mathrm{LCA}(v,v')$, with $a_{v,w} = 1$, $a_{v',w'} = 1$ and $x < r_T$, then $\delta(x) = r_T$ due to the anti-chain condition. So, we know that any vertex x' such that $x \le x' < r_T$, is in $D^T_{C_{\mathcal{V}}}$. Thus, given $x \in V_T$, and with x_f being its father, the "local" objective function depends on the following quantities:

- $\sum_y a_{x,y} \parallel C_{x,y} \parallel_\infty$: it is the cost of matching T_x and a subtree $G_{y'}$, for some y', if (x,y') is added to C, i.e., if $a_{x,y'} = 1$. If $C_{x,y'}$ is not a minimal coupling, this is an upper bound;
- $\mid g(r_G) - f(x) \mid u_x$: it is an upper bound to the cost of deleting x, with $\#\Lambda(x) > 1$;
- $A_x = 0.5(f(x_f) - f_x)(1 - d_x)$: in case x is deleted with $\#\Lambda(x) = 0$, it gives a lower bound to the deletion of T_x, as it is equal to deleting the lowest point below x to the height of the father of x, in two step. It is an exact value if $x_f = \varphi(x)$ and $g(\eta(x)) - f_x < A_x$;

- $B_x = \{\sum_y a_{v,y}\left(g_y - f_x\right) - Kd_x\}_{v\in E_T, v<x_f}$: in case x is deleted with $\#\Lambda(x) = 0$, it helps giving an upper bound to the deletion of T_x, depending on what happens below x_f. If there is $v < x_f$ such that v is coupled with y and $C_{x,y}$ is a special coupling, then we know that $g_y \geq g(\eta(x))$, and so $g_y - f_x \geq g(\eta(x)) - f_x$. Thus, if $C_{x,y}$ is a special coupling, $\max\{\max B_x, A_x\} \geq cost(x)$.

For any $x \in E_T$, we define:

$$\Gamma^{\uparrow}(x) := \max\left\{\sum_y a_{x,y} \parallel C_{x,y} \parallel_{\infty}, (g(r_G) - f(x))u_x, A_x, \max B_x\right\},$$

similarly, for any $y \in E_G$:

$$\Gamma^{\uparrow}(y) := \max\left\{((f(r_T) - g(y))u_y, A_y, \max B_y\right\}.$$

In full analogy, we set:

$$\Gamma^{\downarrow}(x) := \max\left\{\sum_y a_{x,y} \parallel C_{x,y} \parallel_{\infty}, A_x\right\},$$

and

$$\Gamma^{\downarrow}(y) := A_y$$

for any $y \in E_G$.
Lastly, $\Gamma^{\uparrow}(r_T) := \Gamma^{\downarrow}(r_T) :=\mid r_T - r_G \mid$.

With an abuse of notation, we write:

$$\Gamma^{\uparrow}(\mathcal{V}) := \max_{x\in E_T \coprod E_G} \Gamma^{\uparrow}(x), \tag{9.3}$$

$$\Gamma^{\downarrow}(\mathcal{V}) := \max_{x\in V_T \coprod V_G} \Gamma^{\downarrow}(x). \tag{9.4}$$

Note that all the functions we defined in this section depend on T, G, and on the family $\{C_{x,y}\}_{(x,y)\in V_T\times V_G}$, but we avoid explicit reference to those dependencies for notational convenience.

We sum up the main properties of the definitions we have just stated with the following proposition.

Proposition 5. Given $C \in C^*(T, G)$ and $C_{x,y} \in C^o_R(T_x, G_y)$ for all $(x, y) \in C$, we call:

$$C^o := \{(r_T, r_G)\} \cup \left(\bigcup_{(x,y)\in C} C_{x,y}\right).$$

If C^o is a special extension, then:

$$\parallel C^o \parallel_{\infty} \leq \Gamma^{\uparrow}(\mathcal{V}) \tag{9.5}$$

with $\mathcal{V}$ being the unique vector in $\mathcal{K}^o$ such that $C_{\mathcal{V}} = C^o$.

Vice versa, given $C \in C^*(T, G)$, and $C_{x,y} \in C_R(T_x, G_y)$ with minimal cost for all $(x, y) \in C$, we call

$$C_m := \{(r_T, r_G)\} \cup \left(\bigcup_{(x,y)\in C} C_{x,y}\right).$$

Then,

$$\Gamma^{\downarrow}(\mathcal{V}) \leq \max\{cost_{C_m}(v) \mid v \in \pi_T(C_m) \text{ or } \#\Lambda_{C_m}(v) > 0\} \tag{9.6}$$

with $\mathcal{V}$ being the unique vector in $\mathcal{K}^m$ such that $C_{\mathcal{V}} = C_m$.

Putting together Theorem 3 and Proposition 5, we obtain as a corollary:

Corollary 2. Consider T, G merge trees, and take for every $(x, y) \in E_T \times E_G$:

1) $C_{x,y} \in \arg\min_{C' \in C_R(T_x, G_y)} \| C' \|_\infty$;

2) $C'_{x,y} \in \arg\min_{C' \in C^o_R(T_x, G_y)} \| C' \|_\infty$.

Then,

$$\min_{\mathcal{V} \in \mathcal{K}^m} \Gamma^{\downarrow}(\mathcal{V}) \leq \min_{C \in C_R(T,G)} \| C \|_\infty \leq \min_{\mathcal{V} \in \mathcal{K}^o} \Gamma^{\uparrow}(\mathcal{V}),$$

where $\Gamma^{\downarrow}$ is computed with the collection $\{C_{x,y}\}$ and $\Gamma^{\uparrow}$ is computed with the collection $\{C'_{x,y}\}$.

Now, we get rid of the fixed roots, obtaining an approximation for $d_I(T, G)$ by putting together Theorem 3, Lemma 4, and Proposition 5. To do so, we consider $(x, y) \in E_T \times E_G$, take the subtrees T_x and G_y, and apply the framework introduced in Section 9.2.3 and Section 9.2.4, replacing T and G with, respectively, T_x and G_y. We write $\mathcal{K}^m_{(x,y)}$, $\mathcal{K}^o_{(x,y)}$, $\Gamma^{\downarrow}_{(x,y)}$, and $\Gamma^{\downarrow}_{(x,y)}$ to imply that the constraints and the objective functions are defined using the subtrees T_x and G_y. To lighten the notation, we simply write $\mathcal{V}$ to identify the optimization variables. In other words, when we write $\Gamma^{\uparrow}_{(x,y)}(\mathcal{V})$, we are implying $\mathcal{V} \in \mathcal{K}^o_{(x,y)}$, etc.

Corollary 3. In the same setting as Corollary 2, for each $(x, y) \in V_T \times V_G$, we have:

$$W^{\downarrow}_{x,y} := \min_{\mathcal{V} \in \mathcal{K}^m_{(x,y)}} \Gamma^{\downarrow}_{(x,y)}(\mathcal{V}) \leq \min_{C \in C_R(T_x, G_y)} \| C \|_\infty \leq W^{\uparrow}_{x,y} := \min_{\mathcal{V} \in \mathcal{K}^o_{(x,y)}} \Gamma^{\uparrow}_{(x,y)}(\mathcal{V}),$$

where $\Gamma^{\downarrow}_{(x,y)}$ is computed with $\{C_{x,y}\}$ and $\Gamma^{\uparrow}_{(x,y)}$ is computed with $\{C'_{x,y}\}$.
Consequently,

$$\min_{(x,y) \in V_T \times V_G} \max\{H_{x,y}, W^{\downarrow}_{x,y}\} \leq d_I(T, G) \leq \min_{(x,y) \in V_T \times V_G} \max\{H_{x,y}, W^{\uparrow}_{x,y}\}. \tag{9.7}$$

9.2.5. Estimation errors

We take this section to briefly isolate which are the situations in which the procedure outlined in Section 9.2.3 and Section 9.2.4 may produce errors, w.r.t. the true interleaving distance.

1) $| g(r_G) - f(x) | u_x$: clearly, r_G (resp., r_T) is an upper bound for $\chi(x)$ (resp., $\chi(y)$) with x (resp., y) being deleted with $\#\Lambda(x) > 1$ (resp., $\#\Lambda(y) > 1$), and so $| g(r_G) - f(x) | u_x$ is an upper bound to the cost of the corresponding deletion.

2) $B_{x'}$: for $v \in D_T$ with $\#\Lambda(v) = 0$, $x = \varphi(v)$, and $x = father(x')$, we may have $| f(v) - g(\eta(v)) | \leq \max B_{x'}$.

We point out two facts. First, $\Gamma^{\downarrow}$ does not involve any of the aforementioned situations. Second, in the definition of $\Gamma^{\uparrow}$, the biggest potential source of error are the terms $\mid g(r_G) - f(x) \mid u_x$, which can overestimate the cost of deleting internal vertices in order to swap father-children relationships. We point out that metrics for merge trees like [43,48], even in their unconstrained formulation, do not even account for the possibility of deleting internal vertices in order to swap father-children relationships. Because of this, they are unstable and authors resort to preprocessing strategies, trying to make up for this source of instability. See also [41] for more details. Also, replacing $\mid g(r_G) - f(x) \mid u_x$ with the exact deletion cost would turn the linear optimization problem into a polynomial one.

9.2.6. Linearization

In Section 9.2.3, we have introduced a set of linear constraints needed to optimize a nonlinear function (either $\Gamma^{\uparrow}$ or $\Gamma^{\downarrow}$) of the form $\min_{\mathcal{V}\in\mathcal{K}} \max_i F_i(\mathcal{V})$ for some real-valued functions F_i which are linear in $\mathcal{V}$. See Section 9.2.4. We can turn this into a linear optimization problem by exploiting a standard trick, introducing auxiliary variables and additional constraints.

Suppose we need to find $\min_s \max(f(s), g(s))$, with f, g functions of s. We introduce the variable u, with the constraints $u \geq f(s)$ and $u \geq g(s)$, and then solve the linear problem $\min_{s\in\mathbb{R}^n, u\geq f(s), u\geq g(s)} u$, with n equal to the number of constrains. We want to use this procedure to compute $\min_{\mathcal{V}\in\mathcal{K}} \Gamma^{\uparrow}(\mathcal{V})$ and $\min_{\mathcal{V}\in\mathcal{K}} \Gamma^{\downarrow}(\mathcal{V})$. We write down the details only for $\min_{\mathcal{V}\in\mathcal{K}} \Gamma^{\uparrow}(\mathcal{V})$, and the case for $\Gamma^{\downarrow}(\mathcal{V})$ follows analogously.

Given $x \in E_T$, we define $F^1_x = \sum_y a_{x,y} \parallel C_{x,y} \parallel_\infty$, $F^2_x = (g(r_G) - f(x))u_x$, and $F^3_x = A_x$. Given $y \in E_G$, instead, we set $F^1_y = (f(r_T) - g(y))u_y$, and $F^2_y = A_y$. Analogously, we have $F^1_{r_T} = \mid f(r_T) - g(r_G) \mid$. Having fixed a total ordering on $V_T = \{a_0, \ldots, a_n\}$ and $V_G = \{b_0, \ldots, b_m\}$, respectively, we call $\mathcal{F}$ the vector containing all the components F^i_x and F^j_y of all the points taken in the chosen order:

$$\mathcal{F} := \left(F^1_{a_1}, F^2_{a_1}, F^3_{a_1}, \ldots, F^1_{a_i}, F^2_{a_i}, F^3_{a_i}, \ldots, F^1_{b_1}, F^2_{b_1}, \ldots, F^1_{b_m}, F^2_{b_m}\right) = (F_i)_{i\in I_1}\,, \tag{9.8}$$

for some set I_1 indexing the components of $\mathcal{F}$.

Similarly, for every x, we order the elements of the set B_x, so that we can write B_x as a vector $B_x = \left(B^1_x, \ldots, B^h_x, \ldots\right)$, then we define the vector:

$$\mathcal{B} := \left(B^1_{a_0}, \ldots, B^{h_0}_{a_0}, \ldots, B^1_{a_i}, \ldots, B^{h_i}_{a_i}, \ldots, B^1_{b_0}, \ldots, B^{t_0}_{b_0}, \ldots, B^1_{b_j}, \ldots, B^{t_j}_{b_j}\right) = (B_i)_{i\in I_2}\,, \tag{9.9}$$

for some set I_2 indexing the components of $\mathcal{B}$.

Lastly, we introduce the real valued auxiliary variable z and add the following linear constraints to the ones presented in Section 9.2.3:

(5) $z \geq F_i(\mathcal{V})$ for all $i \in I_1$;

(6) $z \geq B_i(\mathcal{V})$ for every $i \in I_2$,

and then solve $\min_{z,\mathcal{V}} z$ with all these constraints. We call $\mathcal{Z}(\mathcal{V})$ the set of admissible values defined by constraints (5) and (6), as a function of $\mathcal{V}$.

We stress again that F_i and B_i are linear in $\mathcal{V}$; so the final problem is linear with mixed binary valued variables. In case of $\Gamma^{\downarrow}$, we repeat the same operations, omitting the constraints in (6).

9.2.7. Bottom-Up algorithm and computational complexity

In this section, the results obtained in Section 8 and the formulation established in the previous parts of Section 9 are used to obtain the algorithm implemented to estimate the metric d_I between merge trees. We need to introduce some last pieces of notation in order to describe the "bottom-up" nature of the procedure.

Given $x \in V_T$, define $\mathrm{len}_T(x)$ to be the cardinality of $\{v \in V_T \mid x \leq v \leq r_T\}$ and $\mathrm{len}(T) = \max_{v \in V_T} \mathrm{len}_T(v)$; similarly, $\mathrm{lvl}_T(x) = \mathrm{len}(T) - \mathrm{len}_T(x)$, and so $\mathrm{lvl}_T^{-1}(n) = \{v \in V_T \mid \mathrm{lvl}_T(v) = n\}$.

The key property for our bottom-up procedure is that: $\mathrm{lvl}_T(x) > \mathrm{lvl}_T(v)$ for any $v \in sub_T(x)$. Thus, we will compute some quantity $\widetilde{W}^{\uparrow}_{x,y}$ for some $x \in \mathrm{lvl}_T^{-1}(n)$ and $y \in \mathrm{lvl}_G^{-1}(m)$, assuming that we will have already computed $\widetilde{W}^{\uparrow}_{v,w}$ for all v, w in V_T, V_G such that $\mathrm{lvl}_T(v) < n$ and $\mathrm{lvl}_G(w) < m$. We write down Algorithm 1, which refers to the computation of $\min \Gamma^{\uparrow}$. Note that thanks to constraints (4) this recursive procedure is always guaranteed to provide a special coupling. However, clearly, not all special couplings satisfy constraints (4).

For a couple $(x, y) \in V_T \times V_G$, we write $\widetilde{\mathcal{Z}}^{\uparrow}_{(x,y)}(\mathcal{V})$ to indicate the set of admissible values as a function of $\mathcal{V}$, described by constraints (5) and (6), where the functions F_i and B_i are given by the components of $\Gamma^{\uparrow}_{(x,y)}$, apart from $F_1^v = \sum_{w \in E_{G_y}} a_{v,w} \parallel C_{v,w} \parallel_\infty$, which is replaced by some quantity $\sum_{w \in E_{G_y}} a_{v,w} \widetilde{W}^{\uparrow}_{v,w}$ we will define in the upcoming lines. In particular, $\widetilde{W}^{\uparrow}_{v,w}$, which we will have computed at iterations prior to the one involving (x, y), is an estimate of $\parallel C_{v,w} \parallel_\infty$. See Algorithm 1.

Algorithm 1 Bottom-up algorithm for an upper bound to $d_I(T, G)$

$N \leftarrow \mathrm{len}(T)$
$M \leftarrow \mathrm{len}(G)$
$n, m \leftarrow 0$
while $n \leq N$ or $m \leq M$ **do**
  for $(x, y) \in V_T \times V_G$ such that $\mathrm{lvl}_T(x) \leq n$ and $\mathrm{lvl}_G(y) \leq m$ **do**
    $H_{x,y} \leftarrow$ Solution of Equation (9.1)
    $\widetilde{W}^{\uparrow}_{x,y} \leftarrow \min_{z \in \widetilde{\mathcal{Z}}^{\uparrow}_{(x,y)}(\mathcal{V}), \mathcal{V} \in \mathcal{K}^{o}_{(x,y)}} z$ ▷ We know $\widetilde{W}^{\uparrow}_{v,w}$ for all (v, w) needed
  end for
  $n \leftarrow n + 1$
  $m \leftarrow m + 1$
end while
return $\min_{(x,y) \in V_T \times V_G} \max\{H_{x,y}, \widetilde{W}^{\uparrow}_{x,y}\}$

The case for $\Gamma^{\downarrow}$ follows the same algorithm, but instead of $\widetilde{W}^{\uparrow}_{x,y}$, we have:

$$\widetilde{W}^{\downarrow}_{x,y} = \min_{z \in \widetilde{\mathcal{Z}}^{\downarrow}_{(x,y)}(\mathcal{V}), \mathcal{V} \in \mathcal{K}^{m}_{(x,y)}} z,$$

where $\widetilde{\mathcal{Z}}^{\downarrow}_{(x,y)}(\mathcal{V})$ stands for the set of admissible values described by constraints (5), with the functions F_i given by the components of $\Gamma^{\downarrow}_{(x,y)}$, apart from $F_1^v = \sum_{w \in E_{G_y}} a_{v,w} \parallel C_{v,w} \parallel_\infty$, which is replaced by $\sum_{w \in E_{G_y}} a_{v,w} \widetilde{W}^{\downarrow}_{v,w}$ for every $v \in E_{T_x}$. By Section 9.2.5, such a recursive procedure produces no error when replacing F_1^v with $\sum_{w \in E_{G_y}} a_{v,w} \widetilde{W}^{\downarrow}_{v,w}$.

In other words, for any collection $C_{x,y} \in \arg\min_{C \in C_R^m(T_x, G_y)} \| C \|_\infty$, with $(x, y) \in E_T \times E_G$, we have:

$$\min_{(x,y)\in V_T \times V_G} \max\{H_{x,y}, \widetilde{W}^{\downarrow}_{x,y}\} = \min_{(x,y)\in V_T \times V_G} \max\{H_{x,y}, W^{\downarrow}_{x,y}\} \leq d_I(T, G),$$

giving a valid lower bound for d_I. Similarly, since constraints (4) are not satisfied by all special couplings, given any collection $C'_{x,y} \in \arg\min_{C' \in C_R^o(T_x, G_y)} \| C' \|_\infty$, with $(x, y) \in E_T \times E_G$, we have:

$$d_I(T, G) \leq \min_{(x,y)\in V_T \times V_G} \max\{H_{x,y}, W^{\downarrow}_{x,y}\} \leq \min_{(x,y)\in V_T \times V_G} \max\{H_{x,y}, \widetilde{W}^{\downarrow}_{x,y}\},$$

giving a valid upper bound for d_I.

9.2.8. Error propagation

We make a brief observation to take care of the interactions arising between the estimation errors described in Section 9.2.5 and the bottom-up procedure proposed in Section 9.2.7.

Consider the setting of Section 9.2.4 and suppose that, instead of having computed the optimal couplings $C_{x,y}$, we have some nonoptimal $C'_{x,y}$ as it is the case for $W^{\uparrow}_{x,y}$ and $\widetilde{W}^{\uparrow}_{x,y}$ - with an error bound $e_{x,y}$ such that $|\| C_{x,y} \|_\infty - \| C'_{x,y} \|_\infty| < e_{x,y}$. We immediately see that the errors $e_{x,y}$ affect only the components of the form $\sum_y a_{x,y} \| C'_{x,y} \|_\infty$, as these are the only parts of the optimization procedure which depend on $\| C'_{x,y} \|_\infty$. Moreover, the potential errors occurring in $\Gamma^{\uparrow}$ and $\Gamma^{\downarrow}$, appear at the level of B_x and u_x (see Section 9.2.5), but u_x and B_x do not depend on $\| C'_{x,y} \|_\infty$ nor on $u_{x'}$ and $B_{x'}$, for some other x'. This implies that the only interaction between errors is of the form $\max\{a, b\}$, but they never aggregate in any way in the objective functions. Thus, the final error in the algorithm presented in Section 9.2.7 is the maximum of all the "independent" errors occurring at every iteration.

9.2.9. Computational complexity

Proposition 6. (Computational cost) Let T and T' be two merge trees with full binary tree structures with $\dim(T) = \#E_T = N$ and $\dim(T') = \#E_{T'} = M$.
Then,

- to compute $\widetilde{W}^{\uparrow}_{x,y}$ for every $(x, y) \in E_T \times E_{T'}$ with Algorithm 1, we need to solve $O(M \cdot N)$ mixed binary linear optimization problems, with $O(M \cdot N)$ variables and $O(M \cdot \log_2(M) + N \cdot \log_2(N))$ linear constraints;
- to compute $\widetilde{W}^{\downarrow}_{x,y}$ for every $(x, y) \in E_T \times E_{T'}$ with Algorithm 1, we need to solve $O(M \cdot N)$ mixed binary linear optimization problems, with $O(M \cdot N)$ variables and $O(\log_2(M) + \log_2(N))$ linear constraints.

Proof. In a full binary tree structure, at each level a, we have 2^a vertices. Let $A = \text{len}(T)$ and $A' = \text{len}(T')$. We have that, for any vertex $v \in V_T$ at level a, the cardinality of the path from v to any of the leaves in $sub_T(v)$ is $A - a$ and the number of leaves in $sub_T(v)$ is 2^{A-a}. The number of vertices in $sub_T(v)$ is instead $2^{A-a+1} - 1$ and the number of edges $2^{A-a+1} - 2$.

Consider $v \in V_T$ at level a and $w \in V_{T'}$ at level a'. According to Section 9.2.3 to compute $\Gamma^{\uparrow}$ with T_v and T'_w, we need $(2^{A-a+1} - 2) \cdot (2^{A'-a'+1} - 2) + 2^{A-a+1} - 2 + 2^{A'-a'+1} - 2$ binary variables and $2^{A-a} + 2^{A'-a'} + 2 \cdot (2^{A-a+1} - 2) + 2 \cdot (2^{A'-a'+1} - 2)$ linear constraints (if we consider constraints (1)–(3) in Section 9.2.3)

in the nonlinear problem. Constraint (4) adds just 1 further constraint, so we can ignore it. To linearize the problem, we need to add 1 real valued auxiliary variable and $3 \cdot (2^{A-a+1} - 2) + 2 \cdot (2^{A'-a'+1} - 2)$ constraints to take into account for the set $\mathcal{F}$ and additional:

$$\sum_{k=0}^{a} 2^k \cdot (2^{A-a-k+1} - 1) + \sum_{s=0}^{a'} 2^s \cdot (2^{A'-a'-s+1} - 1)$$

constraints for the set $\mathcal{B}$. Simplifying, we have: $(a+1)\cdot(2^{A-a+1}-1)+(a'+1)\cdot(2^{A'-a'+1}-1) \leq O(2^A+2^{A'})$ constraints for $\mathcal{B}$.

Putting the things together we have:

$$(2^{A-a+1} - 2) \cdot (2^{A'-a'+1} - 2) + 2^{A-a+1} - 2 + 2^{A'-a'+1} - 2 \tag{9.10}$$

binary variables and

$$(2^{A-a} + 2^{A'-a'} + 5 \cdot (2^{A-a+1} - 2) + 4 \cdot (2^{A'-a'+1} - 2) + a \cdot (2^{A-a+1} - 1) + a' \cdot (2^{A'-a'+1} - 1) \tag{9.11}$$

linear constraints.

Thus, to minimize $\Gamma^{\uparrow}$, we need to solve $(2^{L+1} - 1) \cdot (2^{A'+1} - 1)$ linear binary optimization problems, each with $2^A \cdot 2^{A'} + O(A + A')$ variables and $O(2^A + 2^{A'})$ constraints. Instead, for $\Gamma^{\downarrow}$, we need to solve $(2^{A+1} - 1) \cdot (2^{A'+1} - 1)$ linear binary optimization problems, each with $2^A \cdot 2^{A'} + O(A + A')$ variables and $O(A + A')$ constraints. Substituting $A = \log_2(N)$ and $A' = \log_2(M)$ in Eqs (10) and (11) gives the result. □

We highlight that the classical edit distance between unlabeled and unordered trees, obtained with BLP [31], can be computed by solving $O(N \cdot M)$ BLP problems with $O(N \cdot M)$ variables and $O(\log_2(M) + \log_2(N))$ constraints, $O(N + M)$ if we count also the constraints restricting the integer variables to $\{0, 1\}$, as the authors of [31] do. Thus, the approximating procedures we developed are not far from the computational complexity of the renowned tree edit distance: $\min \widetilde{W}^{\uparrow}_{x,y}$ differs by some linear factors while $\min \widetilde{W}^{\downarrow}_{x,y}$ has the same complexity.

As already mentioned, most of the edit distances for merge trees, in their unconstrained formulation, have equal or higher computational complexity w.r.t. the classical edit distance between unlabeled and unordered trees. Thus, the fact that the authors of those metrics obtain polynomial-time algorithms under constrained formulations supports our conjecture that a similarly constrained formulation of the interleaving distance could yield practically useful polynomial bounds.

In Appendix B, we provide a comparison between the computational complexity of our methods and those presented in [3] and [25].

10. Simulations and case studies

In this section, we extensively look at the practical behavior of the upper and lower bounds of d_I that we obtained in Section 9.2. We name $d_u(T, G)$ the upper bound obtained with Algorithm 1, and $d_l(T, G)$ the lower bound obtained with the analogous algorithm. Generally speaking, we aim at showcasing that we can use d_u in many practical scenarios, and we have a number of tools at our disposal to check the reliability of the obtained results. More in details:

- We use the upper bound d_u as reference estimate of d_I: we compare it with the upper bound developed in [16], use it to analyze datasets, etc.;
- We use the lower bound d_l and the bottleneck distance between persistence diagrams to find the potential error range of the computed upper bound, exploiting the fact that $d_B \leq d_I$ [9]. More precisely we check the relative discrepancy:

$$\Delta := (d_u - \max\{d_l, d_B\})/d_u. \tag{10.1}$$

  With this quantity, we can obtain an upper bound on the error between d_I and d_u via $d_u - d_I \leq \Delta \cdot d_u$;
- We use the sup norms between the considered functions to check if d_u shows instances of unstable behavior, i.e., $d_u(T_f, T_g) > \| f - g \|_\infty$.

We perform three kinds of simulations: in Section 10.1, we compare our estimates of d_I with the method presented by [16], which, to the best of our knowledge, is the only publicly available code to estimate d_I; in Section 10.2, we look at the empirical stability of d_u by checking how adding pointwise noise to a smooth function is reflected on the associated merge trees in terms of d_u; in Section 10.3, we tackle some benchmark classification case studies involving some functional datasets.

10.1. Comparison with [16]

The method proposed by [16] turns the unlabeled problem of the interleaving distance between merge trees into a labeled interleaving problem by proposing a suitable set of labels. The optimal labeling would give the exact value of the interleaving distance, but, in general, this procedure just returns an upper bound. We call d_{lab} the upper bound obtained with the labeled method proposed by [16].

For any fixed i, we generate a couple of point clouds $C_i^k = \{(x_j^k, y_j^k) \mid j = 1, \ldots, n_i\} \subset \mathbb{R}^2$, with $k = 1, 2$, according to the following process:

$$\begin{aligned}
x_j^k &\sim^{iid} \mathcal{N}(5, \sigma_{x,k}) \; j = 1, \ldots, n_i, \\
y_j^k &\sim^{iid} \mathcal{N}(5, \sigma_{y,k}) \; j = 1, \ldots, n_i, \\
\sigma_{x,k} &\sim \mathcal{N}(3, 1), \\
\sigma_{y,k} &\sim \mathcal{N}(3, 1),
\end{aligned}$$

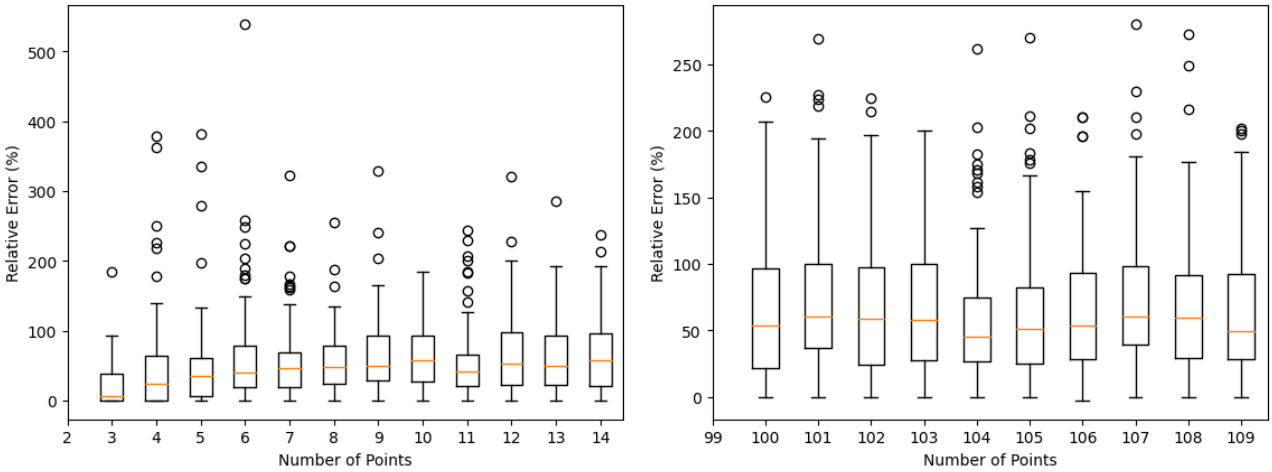
where $\mathcal{N}(\mu, \sigma)$ indicates a Gaussian distribution.

Note that n_i regulates the number of leaves in the trees (which we fix before sampling C_i^k). From C_i^k we obtain the single linkage hierarchical clustering dendrogram $T_{C_i^k}$ (that is, the merge tree representing the Vietoris Rips filtration of C_i^k), and then compute $d_u(T_{C_i^1}, T_{C_i^2})$, $d_l(T_{C_i^1}, T_{C_i^2})$, and $d_{lab}(T_{C_i^1}, T_{C_i^2})$. The distance d_{lab} is computed with the code available at `https://github.com/trneedham/Decorated-Merge-Trees`, while d_u is computed via the procedure described in Section 9.2.7. For each $n_i \in \{3, \ldots, 15\}$, we sample 100 pairs of point clouds and then, similarly to Eq (10.1), compute the relative difference between the two estimates: $(d_{lab} - d_u)/d_u$. Note that there are no absolute values.

We repeat the same experiment, this time with $n_i \in \{100, ..., 109\}$. Since in this case d_u requires too much time to be computed exactly, we exploit Proposition 4 and consider the smallest $\varepsilon > 0$ such that $P_\varepsilon(T_{C_i^1})$ and $P_\varepsilon(T_{C_i^2})$ have fewer or equal than 15 leaves. We then call $d_{opt}(T_{C_1}, T_{C_2}) = \max\{\varepsilon/2, d_u(P_\varepsilon(T_{C_1}), P_\varepsilon(T_{C_2}))\}$.

For $n_i \in \{100, ..., 109\}$, we compute the relative difference between d_{lab} and d_{opt} with the formula: $(d_{lab} - d_{opt})/d_{opt}$. Note, again, that there are no absolute values.

The results of the simulations can be seen in Figure 10. Looking at Figure 10, we see that, in the context of our data-generating process, the estimate given by d_{lab} is very unreliable, overestimating d_I by a median of at least 50–60% of its actual value; in particular, there are some outliers which are completely off, even w.r.t. the values obtained with d_u and d_{opt}, with errors of more than 3 times the actual value.



(a) Boxplots the relative difference $(d_{lab} - d_u)/d_u$, as a function of the number of leaves (between 3 and 14).

(b) Boxplots the relative difference $(d_{lab} - d_{opt})/d_{opt}$, as a function of the number of leaves (between 100 and 109).

Figure 10. Descriptive statistics of the relative differences between d_u, d_{opt}, and d_{lab}, with d_u being the upper bound of d_I we described in Section 9.2, d_{opt} being the pruning-assisted upper bound we obtained using the results in Section 9.1, and d_{lab} being the estimation procedure of [16]. Note that no absolute value appears in the formulas of the relative differences. For more details, see Section 10.1.

The estimates obtained with d_u in this simulation have a negligible difference with the actual values of d_I: across all $n_i \in \{3, \ldots, 15\}$, the maximum value of the relative difference Δ between d_u and d_l was 10^{-7}. Note that, for the other values of n_i, we cannot give a lower bound using d_l on the pruned trees.

As a conclusive remark, we say that the computational advantages of the labeled approach are immense and potentially adequate even for real-time applications, but from our simulation, we see that the results need to be taken with care, for the estimates produced are not always good. On the other hand, with the present implementation, the computational cost of our approach becomes prohibitive quite quickly as the number of leaves in the trees increases, even if there might be situations like the one in this simulation in which the scheme we used for $n_i \geq 100$ can produce good estimates. We think that interactions between the two approaches could lead to substantial speed-ups: being able to fix the

value of some variables in our algorithm exploiting the labeling scheme by [16] could greatly reduce the dimensionality of the problem and, thus, its computational cost.

10.2. Noisy functions

Now, we want to verify, experimentally, if d_u retains good stability properties w.r.t. the universal stability properties of the interleaving distance.

To do so, we pick a standard model in functional data analysis and look at how the distance between the merge trees associated to noisy observations and the original smooth function behave as we increase the noise in the model. Roughly speaking, the model considers a smooth function f, randomly sample a set of points in the domain where we evaluate f, and add pointwise noise to these values. The final set of couples is a single datum in the generated dataset.

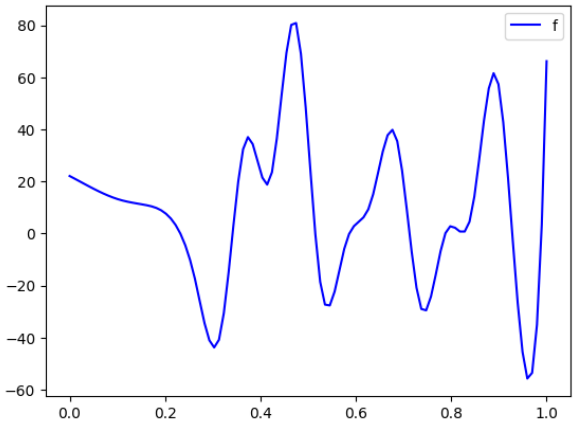
We generate f by interpolating, with cubic splines, a set of randomly chosen couples in $[0, 1] \times \mathbb{R}$. The distribution that we use to sample this set of couples is the following: we take $0 = x_1 < \ldots < x_N = 1$, $N = 20$, forming a regular grid in $[0, 1]$. Then, y_i, $i = 1, \ldots, N$, are sampled independently from a Gaussian with mean 0 and standard deviation 50. The couples $\{(x_i, y_i)\}_{i=1,\ldots,N}$ are interpolated to obtain f. Other methods to sample a smooth function could be employed as well.

We sample the noisy observations of f according to the following model, which belongs to a family of standard models for functional data (see [45]). We name such model (M_f^σ).

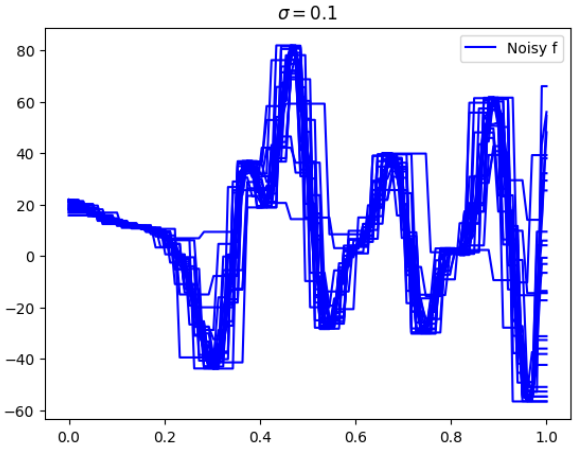
(M_f^σ) Let $\mathcal{X}$ be a random variable distributed with uniform density on $[0, 1]$, and consider the random variable $\mathcal{Y}$ such that $\mathcal{Y} \mid \mathcal{X} = x \sim f(x) + \epsilon$, with ϵ being an independent Gaussian with mean zero and standard deviation σ.

We obtain each of our partially observed noisy functions drawing i.i.d. samples from (M_f^σ). Specifically, for each value of $\sigma \in \{0.1, 1, 10, 15, 25\}$, we sample a dataset D^σ where each of the 100 i.i.d. observations (i.e., each partially observed function) is obtained by sampling independently $n = 50$ couples from (M_f^σ).

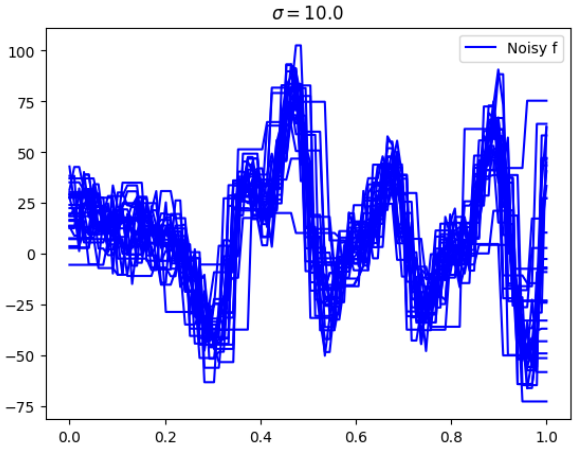
We compute the merge tree T_f of the smooth function f via linear interpolation of $\{(\tilde{x}_i, f(\tilde{x}_i))\}_{i=1,\ldots,300}$, with $\{\tilde{x}_i\}_{i=1,\ldots,300}$ being a uniform grid on $[0, 1]$. Then, for each $\{(a_i, b_i)\}_{i=1,\ldots,50}^j \in D^\sigma$, $j = 1 \ldots, 100$, we compute the merge tree of the linear interpolation of the sampled couples without extending it to the whole interval $[0, 1]$, and we estimate, via d_u and d_l, the interleaving distance between the merge tree obtained from the smooth f and from the noisy observation. As σ, which controls the magnitude of the noise, increases, we are interested in observing the behavior of the distance between the observed and the true merge trees. The sampled partial observations of functions have also been extended on $[0, 1]$ via a nearest value extension (NVE) technique, both for plotting purposes and to compute the sup norm between the extended functions, to check the stability of the upper bound d_u. See Figure 11.



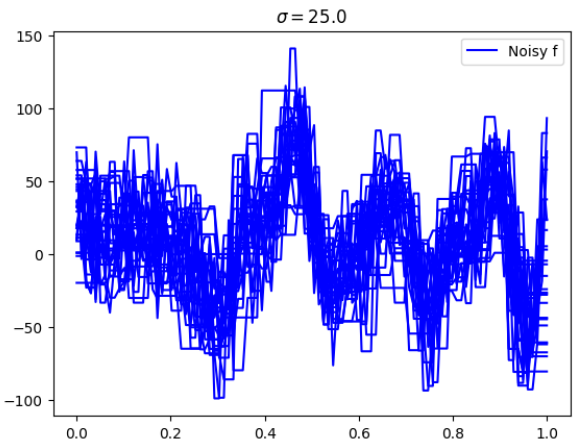
(a) The random smooth function $f : [0, 1] \to \mathbb{R}$.



(b) The dataset D^σ, for $\sigma = 1$.



(c) The dataset D^σ, for $\sigma = 10$.



(d) The dataset D^σ, for $\sigma = 25$.

Figure 11. Plots related to the simulated datasets featured in Section 10.2. Each of the partially observed functions, generated according to the model (M_f^σ), have been extended on $[0, 1]$ using nearest value extrapolation.

To better portrait the stability of d_u, we benchmark our upper bound against a metric for merge trees with very different stability properties: the edit distance d_E defined in [41]. Such edit distance can be computed exactly with a computational complexity similar of the ones of the upper and lower bounds we develop in this work, and satisfies the inequalities:

$$d_I(T_1, T_2) \leq d_E(T_1, T_2) \leq 2(\dim(T_1) + \dim(T_2))d_I(T_1, T_2).$$

This implies that d_E can grow linearly with the size of the trees.

Figure 11(d) summarizes the behavior of d_u and d_E in response to increasing levels of noise. More precisely:

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

$$\{d_u(T_f, T_\sigma^j) \mid T_\sigma^j \text{ is the merge tree of } \{(a_i, b_i)\}_{i=1,\dots,50}^j \in D^\sigma, j = 1 \dots, 30\};$$

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

$$\{d_E(T_f, T_\sigma^j) \mid T_\sigma^j \text{ is the merge tree of } \{(a_i, b_i)\}_{i=1,\dots,50}^j \in D^\sigma, j = 1\dots, 30\};$$

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

$$\{\| f - f_\sigma^j \|_\infty | \ f_\sigma^j \text{ is the NVE on } [0,1] \text{ of } \{(a_i, b_i)\}_{i=1,\dots,50}^j \in D^\sigma, j = 1\dots, 30\}.$$

The vertical bars in Figure 12 represent 1.5 times the interquartile range (IQR) below and above the median (i.e., the boxplots). In other words, we are checking how far from the true merge trees the observed merge trees tend to fall, as noise increases, where far is either in terms of d_u or d_E. This is precisely where stability properties become crucial, as they provide robustness to support the analyst's findings.

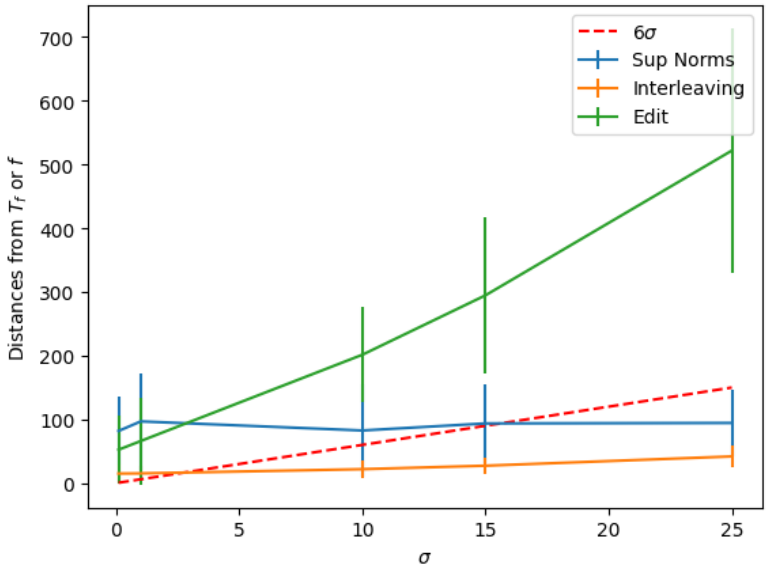


Figure 12. Lines resulting from the interpolation of the medians of the distances d_E and d_u, measured between the merge trees of f and the merge trees obtained from their noisy observations contained in D^σ, for different values of σ. For each value of σ, we report also the sup of the difference between the smooth function and the noisy observations (extended on $[0,1]$). The dashed line represents the line $(\sigma, 6\sigma)$, plotted for reference. For each σ, the vertical bars represent the boxplots of the data. We clearly see how the values of d_E "explode", while d_u behaves as we would expect from d_I: the universal properties of d_I guarantee that d_I is controlled by the difference between the functions. See Section 10.2 for more details.

For a reference, we also report the plot of the line $(\sigma, 6\sigma)$: we know that for every $(\mathcal{X}, \mathcal{Y})$ distributed according to model (M_f^σ), $P(\mathcal{Y} \in (f(\mathcal{X})-3\sigma, f(\mathcal{X})+3\sigma) \mid \mathcal{X} = x) \sim 1$, for every $x \in [0,1]$. In particular, for $\{(\mathcal{X}_i, \mathcal{Y}_i)\}_{i=1,\dots,50}$ i.i.d., we have $P(\max_{i=1,\dots,50} \mid \mathcal{Y}_i - f(\mathcal{X}_i) \mid < 6\sigma) \sim 0.9$. In other words, $[0, 6\sigma]$ is a 0.9 confidence interval for the max difference between the smooth functions and the noisy observations on the grid given by the realizations of $\{\mathcal{X}_i\}_{i=1,\dots,50}$. Note that, despite this, outside such grid, i.e., outside

the points were the functions are observed, the error between the smooth functions and the extended noisy observations can be higher. Still, the line $(\sigma, 6\sigma)$ can serve as a rough reference for the pointwise difference between the smooth functions and the noisy observations.

In Figure 12, we see that the deviation between the true and the observed merge trees behave as we would expect: the deviation in terms of d_u increases with the magnitude of the noise σ, being controlled by the sup norm between the smooth functions and the extension on $[0, 1]$ of the noisy observations. The deviation in terms of d_E, instead, explodes as σ increases, as d_E depends on a combination of 1) the magnitude of noise and 2) the number of oscillations in the functions. When sampling and interpolating the partial and noisy observations of f, we are also potentially increasing the number of oscillations of the data w.r.t. the original f: while T_f has 5 leaves, for $\sigma = 0.1$, the number of leaves ranges from 8 to 4, with a median of 6; for $\sigma = 1$, the number of leaves ranges from 12 to 4, with a median of 9; for $\sigma = 10$, the number of leaves ranges from 18 to 9, with a median of 14; for $\sigma = 15$, the number of leaves ranges from 19 to 11, with a median of 15; for $\sigma = 25$, the number of leaves ranges from 19 to 11, with a median of 15. Note that, at most, we can have 19 leaves, given that we sample on a grid of 50 points.

All of what we just said is coherent with the behavior we observe in Figure 12. The line $(\sigma, 6\sigma)$ further supports the coherence of our findings: for small values of σ, the error introduced by having to extend the observed values from a random grid to $[0, 1]$ overshadows the pointwise error; meanwhile, for $\sigma = 25$ we see that the whole boxplot of the sup norms is below the dashed line.

To conclude this simulation, we double-check the results we obtained with d_u by computing also the relative discrepancies with d_l, i.e., Δ, as in Eq (10.1). In Table 1, we report some percentiles - $95^{\text{th}}, 90^{\text{th}}, 85^{\text{th}}, 80^{\text{th}}$ of the values Δ as a function of σ, showing that in most cases we have negligible uncertainties in our computations. As σ increases, some non-negligible discrepancies start to appear in a limited number of calculations. However, even in the worst case scenario, 85% of the distances still have a relative discrepancy between d_u and d_I, which is guaranteed to be lower than 6%. No unstable behaviors of d_u, i.e., instances of $d_u(T_f, T_g) > \| f - g \|_\infty$, were registered. As a result, we can look at the results of this simulation with a very high level of confidence.

Table 1. A table reporting some percentiles $95^{\text{th}}, 90^{\text{th}}, 85^{\text{th}}, 80^{\text{th}}$ of the relative discrepancies between the upper and lower bounds $\Delta = (d_u - d_l)/d_u$ for the simulation presented in Section 10.2: the different values of σ represent the different magnitude of noise that we introduce when sampling our observations. The table shows that in the vast majority of the cases, we are guaranteed a negligible error in our computations. When noise levels are high, and so when the merge trees become more and more different, some discrepancy between d_u and d_l appears.

	$\sigma = 0.1$	$\sigma = 1$	$\sigma = 10$	$\sigma = 15$	$\sigma = 25$
95th Perc.	$< 10^{-13}$	$< 10^{-14}$	0.23	0.22	0.18
90th Perc.	$< 10^{-14}$	$< 10^{-14}$	0.09	0.06	0.14
85th Perc.	$< 10^{-14}$	$< 10^{-14}$	10^{-4}	$< 10^{-2}$	0.06
80th Perc.	$< 10^{-14}$	$< 10^{-15}$	$< 10^{-14}$	$< 10^{-7}$	0.02

10.3. Case studies

Having tested d_u (and d_l) in some controlled, simulated scenarios, now we test it on some benchmark functional datasets, tackling some classification problems.

The datasets we considered are suitable for our purposes for three reasons:
1) They are freely available and easily accessible via the `scikit-fda` python package [44];
2) Their complexity (in terms of number of statistical units, and number of oscillations of each function) allows the use of our estimation algorithms;
3) We can use the sup of the difference between the functions to check the stability of d_u.

To further validate our results, we build two kinds of alternative pipelines: a first one obtained by replacing d_u between merge trees, with d_B between the associated persistence diagrams; and a second one using directly the original functions. The general pipeline we follow for d_u and d_B is the following: we compute merge trees/diagrams, take pairwise distances, fit an embedding of the resulting metric space into an Euclidean space Isomap [6], and use quadratic discriminant analysis (QDA) on the embedded vectors. Since all the functions in each dataset are evaluated on the same grid, we also fit some "Baseline" pipelines, obtained by reducing the vectors' dimensionality with principal component analysis (PCA), to avoid overfitting, and using QDA directly on the resulting vectors. The parameters for the Isomap embeddings (the dimension and the number of points in a neighborhood) and the dimension of the PCA have been selected using leave-one-out cross validation, maximizing accuracy. We report the parameters and the results of the best performing models in Table 2.

Table 2. Results of the classification tasks considered in Section 10.3. Each column deals with a different task, while each row contains the leave-one-out accuracy, and the best performing parameters obtained with d_B, d_u (or d_{opt}), or the baseline models. For d_B and d_u, we also report the parameters of the Isomap embedding $(neigh, dim)$, while for the Baseline models we report the PCA dimension. Parameters were selected to maximize leave-one-out accuracy.

Results

	Octane	NOx	Growth	Tecator	Beef
d_B	1	0.88	0.84	0.98	0.58
$d_u(d_{opt})$	1	0.86	0.87	0.98	0.68
Baseline	1	0.94	0.93	0.97	0.7

Optimal parameters

	Octane	NOx	Growth	Tecator	Beef
d_B	$(2, 8)$	$(14, 44)$	$(6, 71)$	$(52, 209)$	$(12, 17)$
$d_u(d_{opt})$	$(2, 8)$	$(4, 5)$	$(10, 8)$	$(10, 181)$	$(12, 38)$
Baseline	2	6	2	4	8

We emphasize that the goal of these simulations is not to promote TDA tools over classical statistical methods, but rather to illustrate the practical behavior of our computational procedures and provide a blueprint for their application. In this regard, comparing d_u and d_l with more traditional and arguably more suitable approaches offers a grounded assessment of their effectiveness. Additionally,

we investigate whether the theoretical inequality $d_B \leq d_I$ [9], which suggests that d_I has greater discriminative power than d_B, also holds practical significance in these benchmark scenarios.

Accordingly, the selected datasets were not chosen specifically for their suitability to TDA techniques. This contrasts with applications such as those in [42], where functions must be analyzed up to warpings or re-parametrizations (see also [36]). In such cases, it has been shown that classical methods require complex and cumbersome pipelines to achieve reliable results, whereas the straightforward TDA approaches used here remain effective.

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

- "Octane" dataset [24]: contains near infrared spectra of gasoline samples, with wavelengths ranging from 1102nm to 1552nm with measurements every 2 nm; it contains six outliers to which ethanol was added, as sometimes required by some regulations. We want to build a classifier able to perform outlier detection. The dataset contains 39 observations, with 6 outliers, and the resulting merge trees have between 4 and 5 leaves, with a median of 4 leaves. In Figure 13(a), we see that there is clear separation between the outliers and the other data. Coherently, Table 2 shows that all the different models we considered successfully achieved this task;

- "NO_x" dataset [26]: contains hourly measured daily nitrogen oxides (NOx) emissions in the Barcelona area. Since NO_x causes ozone formation and contributes to global warning, it is of interest the identification of days with abnormally large emissions to allow the implementation of actions able to control their causes, which are primarily the combustion processes generated by motor vehicles and industries. The observations are labeled depending on if the emission curve was recorded during working days or during the weekend. We aim to reconstruct this labeling via a supervised classification problem. The dataset contains 115 observations, with 79 working days and 39 non-working days, and the resulting merge trees have between 2 and 9 leaves, with a median of 5 leaves. In Figure 13(b), we can visually appreciate that the problem is more challenging compared to the previous one. Despite that, Table 2 shows that we achieve very high accuracy, in cross-validation, with our pipelines, with the baseline approach outscoring the TDA one, and persistence diagrams achieving slightly better results than merge trees;

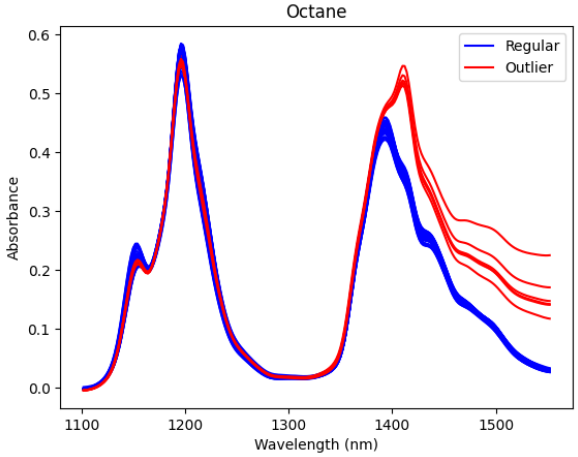
- "Growth" dataset [50]: is a popular dataset in functional data analysis, often referred to as "The Berkeley Growth Study", which contains the height measurements (in cm) of 54 girls and 39 boys between the ages of 1 and 18 years, see Figure 13(c). This dataset has been considered by a number of works, including works dealing with the problem of aligning/re-parametrizing the growth curves, factoring out of the analysis the "personal biological clock" [51] of each individual, to be able to more clearly identify any growth patterns in the data. See [51] and references therein. One interesting problem, often considered, is to compute the first derivative of the growth curves and try to recognize the different growth dynamics between boys and girls. This is also what we do: we smooth the functions via some kernel smoother (with a bandwidth of 3 and with the same kernel employed in [42]), consider the numerical derivatives of the smoothed curves, and try to discriminate boys from girls. The dataset contains 93 observations, 54 girls and 39 boys, and the merge trees of the derivatives have between 4 and 9 leaves, with a median of 5 leaves. In Table 2, we see that all the models achieve good results on this data, with the baseline being the best performing model and merge trees outscoring persistence diagrams;

- "Tecator" dataset (`https://lib.stat.cmu.edu/datasets/tecator`): this publicly available data is collected by a tool called "Tecator Infratec Food and Feed Analyzer", working in the wavelength range 850–1050 nm by the near infrared transmission principle, see Figure 13(d). Each sample contains a curve extracted from meat with different moisture, fat and protein contents. Many different data analysis questions and problems can be set up with this data. Relying on the derivatives of the curves, we consider the problem, as in [27], to discriminate between the curves coming from high fat meat (i.e., fat content > 20) and low fat meat (i.e., fat content ≤ 20). The dataset contains 215 observations, 77 high fat samples, and 138 low fat samples, and the merge trees of the derivatives have between 2 and 10 leaves, with a median of 5 leaves. Table 2 reports the cross-validation accuracy of the classification models. We see that the TDA approaches, both with d_u and d_B, outperform the baseline model, with merge trees again doing slightly better than persistence diagrams.

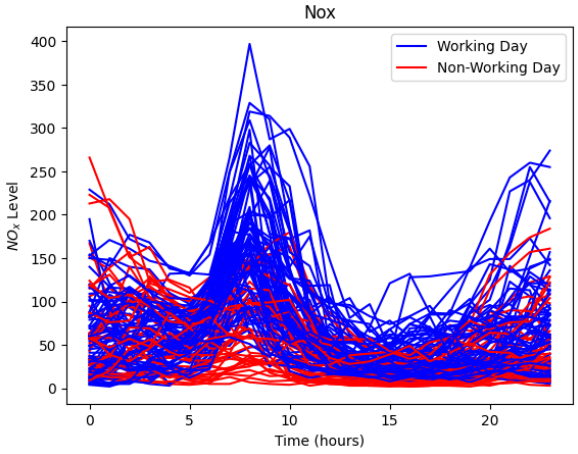
- "Beef" dataset [5]: contains five classes of mid-infrared spectrograms of beef, taken from a total of 60 cooked samples of minced silverside cuts with different kinds of adulterations, ranging from pure beef to beef adulterated with four different types of offal. This dataset is more complex than the previous ones, especially in terms of the number of oscillations of the spectrograms, with the corresponding trees ranging from 30 up to 64 leaves. As a consequence, we have resorted to the approach presented in Section 10.1, computing d_{opt} instead of d_u, pruning merge trees so that they are left with at most 14 leaves. Most of the pruned trees have 14, leaves, while approximately $1/3$ of the trees have 13 leaves, and two have 12 leaves. The leave-one-out results of this classification case study are reported in Table 2 and show that merge trees perform very similarly to the Baseline method, doing much better than persistence diagrams.

In the first four datasets, d_u gives very good estimates of d_I, with Δ being less than 0.008 for 95% of the computed distances in each dataset. We did observe, for the "Tecator" dataset, 7 instances of d_u exceeding $\|\cdot\|_\infty$. Since the values by which d_u exceeded $\|\cdot\|_\infty$ were always below 10^{-8}, we believe it is due to numerical errors. For the "Beef" dataset, we cannot give lower bounds, but we checked for unstable behaviors of d_{opt}, and none were recorded.

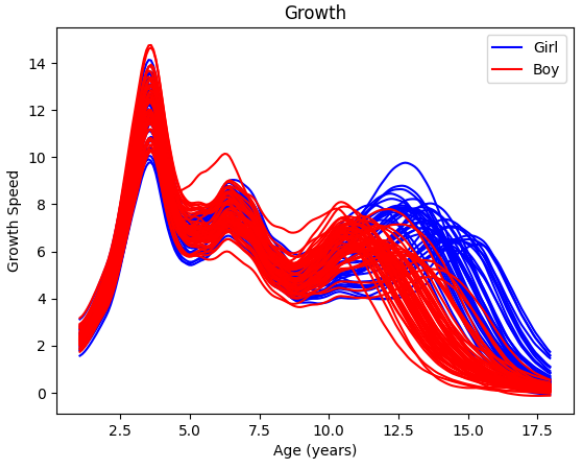
To conclude, we have shown several case studies in which d_u produced reliable estimates of d_I, validated by controlling the outputs with lower bounds (via d_l and d_B) and upper bounds (via $\|\cdot\|_\infty$). The results obtained via the implemented pipelines were bench-marked against established TDA and classical tools, showing equal or superior performances w.r.t. persistence diagrams in all but one situation.



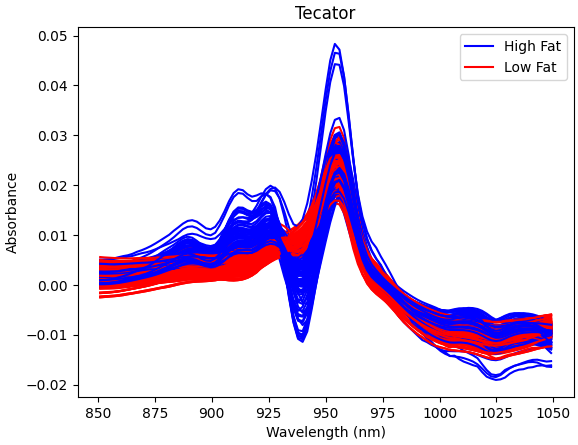
(a) The "Octane" dataset. The red functions represent the outliers with the addition of ethanol. By visual inspection we can see that the two classes are nicely clustered and distinguishable.



(b) The "NO_x" dataset. By visual inspection we see that working and non-working days follow different dynamics, especially regarding the morning peak. This also explains why, as shown by Table 2, merge trees work better than the vanilla model.



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



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

Figure 13. Plots related to the case studies presented in Section 10.3.

11. Conclusions

In this work, we introduced a novel graph-matching formulation for the interleaving distance between merge trees, leveraging global matchings rather than the traditional approach based on ε-good maps. This reformulation not only provides a new perspective on the metric but also facilitates computational advancements by enabling recursive decomposition techniques and constrained formulations inspired by the literature on edit distances. As a first practical result of our approach, we propose upper and lower bounds using a dynamic linear binary programming framework.

While we leave to future works the problem of adapting the dynamic linear binary programming algorithm to the constrained interleaving distance, conjecturing that it would result in a polynomial time upper bound, our empirical evaluation demonstrated that the presented approach reliably estimates the interleaving distance and provides more controlled error estimates compared to existing methods. Through a comprehensive set of case studies, we validated the practical efficacy of our bounds by benchmarking them against both traditional TDA methods and classical statistical approaches.

However, our method also comes with computational trade-offs: the reliance on integer programming comes with a high computational burden, making the approach suitable only for small to moderately sized datasets. Another drawback of the presented approach is that it only provides a-posteriori error bounds: we don't have methods to provide the reliability of our tools before actually computing the distance, or for calibrating their precision. In particular, it is not possible to set an arbitrary precision level by choosing the value of some parameter.

On top of the aforementioned research direction aiming at proving theoretical and computational properties of the constrained interleaving distance, our findings also suggest some additional directions. The relationship between optimal couplings and continuous monotone maps, as highlighted in Section 6, hints at yet another possible formulation of the interleaving distance purely in terms of metric tree mappings. Given that this is among the first works practically employing the interleaving distance, further comparisons between the interleaving distance and other merge tree distances could help elucidate their respective stability and sensitivity properties in different data analysis contexts.

In conclusion, our work provides both theoretical advancements and practical tools for computing and understanding the interleaving distance. Even if computational efficiency remains a challenge, we have provided effective tools to work in small-data regimes, and we believe that we have laid out the main steps for finally making the interleaving distance a disposable tool for analyzing data in topological data analysis.

Use of Generative-AI tools declaration

The author declares that he have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was carried out as part of my PhD Thesis, under the supervision of Professor Piercesare Secchi. The author also acknowledge the support of the Wallenberg AI, Autonomous Systems and Software Program (WASP), and of the SciLifeLab and Wallenberg National Program for Data-Driven

Life Science (DDLS), which fund the project: Topological Data Anlysis of Functional Genome to Find Covariation Signatures.

Conflict of interest

The author declares no conflict of interest.

References

1. H. Adams, J. Bush, N. Clause, F. Frick, M. Gómez, M. Harrison, et al., Gromov-hausdorff distances, borsuk-ulam theorems, and vietoris-rips complexes, (2022), arXiv: 2301.00246.
2. H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, et al., Persistence images: A stable vector representation of persistent homology, *J. Mach. Learn. Res.*, **18** (2017), 1–35.
3. P. K. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, Y. S. Wang, Computing the gromov-hausdorff distance for metric trees, *ACM T. Algorithms*, **14** (2018), 20. https://doi.org/10.1145/3185466
4. T. Akutsu, D. Fukagawa, A. Takasu, T. Tamura, Exact algorithms for computing the tree edit distance between unordered trees, *Theor. Comput. Sci.*, **412** (2011), 352–364. https://doi.org/10.1016/j.tcs.2010.10.002
5. O. Al-Jowder, E. K. Kemsley, R. H. Wilson, Detection of adulteration in cooked meat products by mid-infrared spectroscopy, *J. Agric. Food Chem.*, **50** (2002), 1325–1329. https://doi.org/10.1021/jf0108967
6. M. Balasubramanian, E. L. Schwartz, The isomap algorithm and topological stability, *Science*, **295** (2002), 7. https://doi.org/10.1126/science.295.5552.7a
7. U. Bauer, C. Landi, F. Mémoli, The reeb graph edit distance is universal, *Found. Comput. Math.*, **21** (2021), 1441–1464. https://doi.org/10.1007/s10208-020-09488-3
8. U. Bauer, E. Munch, Y. S. Wang, Strong Equivalence of the interleaving and functional distortion metrics for reeb graphs, *31st International Symposium on Computational Geometry (SoCG 2015)*, Schloss Dagstuhl, Leibniz-Zentrum für Informatik, 2015, 461–475. https://doi.org/10.4230/LIPIcs.SOCG.2015.461
9. K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, B. Hamann, Measuring the distance between merge trees, In: *Topological methods in data analysis and visualization III*, Cham: Springer, 2014, 151–165. https://doi.org/10.1007/978-3-319-04099-8_10
10. N. Berkouk, Algebraic homotopy interleaving distance, In: *Geometric science of information*, Cham: Springer, 2021, 656–664. https://doi.org/10.1007/978-3-030-80209-7_70
11. H. B. Bjerkevik, M. B. Botnan, M. Kerber, Computing the interleaving distance is NP-hard, *Found. Comput. Math.*, **20** (2020), 1237–1271. https://doi.org/10.1007/s10208-019-09442-y
12. P. Bubenik, Statistical topological data analysis using persistence landscapes, *J. Mach. Learn. Res.*, **16** (2015), 77–102.
13. R. Cardona, J. Curry, T. Lam, M. Lesnick, The universal ℓ^p-metric on merge trees, *38th International Symposium on Computational Geometry (SoCG 2022)*, Schloss Dagstuhl, Leibniz-Zentrum für Informatik, 2022, 1–24. https://doi.org/10.4230/LIPIcs.SoCG.2022.24 .

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

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

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

17. J. Curry, W. Mio, T. Needham, O. B. Okutan, F. Russold, Convergence of leray cosheaves for decorated mapper graphs, (2023), arXiv: 2303.00130.

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

19. V. de Silva, E. Munch, A. Stefanou, Theory of interleavings on categories with a flow, *Theor. Appl. Categ.*, **33** (2018), 583–607.

20. B. Di Fabio, C. Landi, The edit distance for reeb graphs of surfaces, *Discrete Comput. Geom.*, **55** (2016), 423–461. https://doi.org/10.1007/s00454-016-9758-6

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

22. H. Edelsbrunner, J. Harer, Persistent homology–a survey, In: *Surveys on discrete and computational geometry: twenty years later*, Providence: American Mathematical Society, 2008, 257–282. https://doi.org/10.1090/conm/453

23. Y. Elkin, V. Kurlin, The mergegram of a dendrogram and its stability, *45th International Symposium on Mathematical Foundations of Computer Science (MFCS 2020)*, Schloss Dagstuhl, Leibniz-Zentrum für Informatik, 2020, 1–13. https://doi.org/10.4230/LIPIcs.MFCS.2020.32

24. K. H. Esbensen, D. Guyot, F. Westad, L. P. Houmoller, *Multivariate data analysis: in practice: An introduction to multivariate data analysis and experimental design*, 5 Eds., AS: Camo Process, 2002.

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

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

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

28. E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, Y. S. Wang, Intrinsic interleaving distance for merge trees, *La Matematica*, **4** (2025), 40–65. https://doi.org/10.1007/s44007-024-00143-9

29. A. Hatcher, *Algebraic topology*, Cambrige: Cambridge University Press, 2001.

30. K. Hirata, Y. Yamamoto, T. Kuboyama, Improved max snp-hard results for finding an edit distance between unordered trees, In: *Combinatorial pattern matching*, Berlin: Springer, 2011, 402–415. https://doi.org/10.1007/978-3-642-21458-5_34
31. E. Hong, Y. Kobayashi, A. Yamamoto, Improved methods for computing distances between unordered trees using integer programming, In: *Combinatorial optimization and applications*, Cham: Springer, 2017, 45–60. https://doi.org/10.1007/978-3-319-71147-8_4
32. T. Jiang, L. S. Wang, K. Z. Zhang, Alignment of trees–an alternative to tree edit, *Theor. Comput. Sci.*, **143** (1995), 137–148. https://doi.org/10.1016/0304-3975(95)80029-9
33. L. Kanari, A. Garin, K. Hess, From trees to barcodes and back again: theoretical and statistical perspectives, *Algorithms*, **13** (2020), 335. https://doi.org/10.3390/a13120335
34. M. Lesnick, The theory of the interleaving distance on multidimensional persistence modules, *Found. Comput. Math.*, **15** (2015), 613–650. https://doi.org/10.1007/s10208-015-9255-y
35. S. M. Lane, *Categories for the working mathematician*, 2Eds., New York: Springer, 1998.
36. J. S. Marron, J. O. Ramsay, L. M. Sangalli, A. Srivastava, Functional data analysis of amplitude and phase variation, *Statist. Sci.*, **30** (2015), 468–484. https://doi.org/10.1214/15-STS524
37. J. W. Milnor, M. Spivak, R. Wells, *Morse theory*, Princeton: Princeton University Press, 1963. https://doi.org/10.1515/9781400881802
38. E. Munch, B. Wang, Convergence between categorical representations of Reeb space and mapper, In: *32nd International Symposium on Computational Geometry (SoCG 2016)*, Schloss Dagstuhl, Leibniz-Zentrum für Informatik, 2016, 1–16. https://doi.org/10.4230/LIPIcs.SoCG.2016.53
39. M. Pegoraro, A persistence-driven edit distance for graphs with abstract weights, (2024), arXiv: 2304.12088v3.
40. M. Pegoraro, A finitely stable edit distance for functions defined on merge trees, (2024), arXiv: 2108.13108.
41. M. Pegoraro, A finitely stable edit distance for merge trees, (2024), arXiv: 2111.02738.
42. M. Pegoraro, P. Secchi, Functional data representation with merge trees, (2024), arXiv: 2108.13147.
43. M. Pont, J. Vidal, J. Delon, J. Tierny, Wasserstein distances, geodesics and barycenters of merge trees, *IEEE T. Vis. Comput. Gr.*, **28** (2021), 291–301. https://doi.org/10.1109/TVCG.2021.3114839
44. C. R.-Carreño, J. L. Torrecilla, M. C. Berrocal, P. Manchón, A. Suárez, scikit-fda: A Python package for functional data analysis, *J. Stat. Softw.*, **109** (2024), 1–37. https://doi.org/10.18637/jss.v109.i02
45. J. O. Ramsay, B. W. Silverman, Functional data analysis, In: *International encyclopedia of the social and behavioral sciences*, 2 Eds., New York: Springer, 2005, 514–518. https://doi.org/10.1016/B978-0-08-097086-8.42046-5
46. S. M. Selkow, The tree-to-tree editing problem, *Inform. Process. Lett.*, **6** (1977), 184–186. https://doi.org/10.1016/0020-0190(77)90064-3
47. P. Smith, V. Kurlin, Generic families of finite metric spaces with identical or trivial 1-dimensional persistence, *J. Appl. Comput. Topology*, **8** (2024), 839–855. https://doi.org/10.1007/s41468-024-00177-6

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

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

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

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

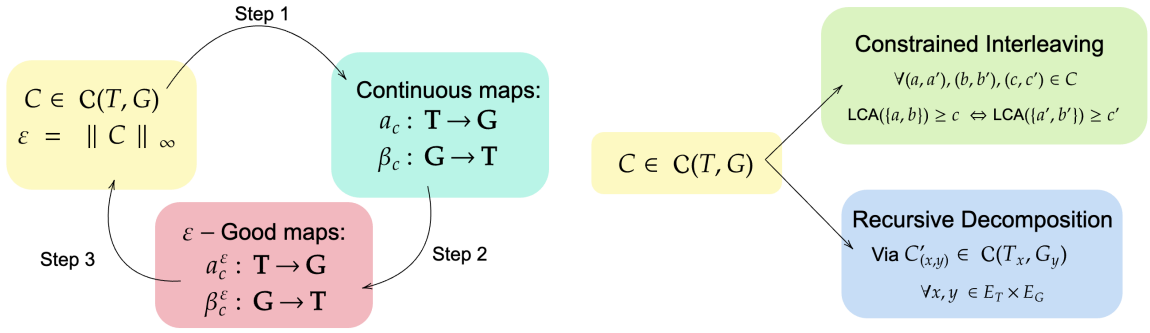
52. F. Wetzels, C. Garth, A deformation-based edit distance for merge trees, *2022 Topological Data Analysis and Visualization (TopoInVis)*, Oklahoma City, USA, 2022, 29–38. https://doi.org/10.1109/TopoInVis57755.2022.00010

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

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

A. Notation and flowcharts

To facilitate navigating through the manuscript, we prepared some flowcharts (Figure 14), and a table (Table 3), summarizing the structure of our theoretical investigation and the most important pieces of notation we introduce.



(a) A flowchart summarizing the structure of Section 4, Section 5, and Section 6, in which we reformulate the interleaving distance using couplings. **(b)** A flowchart summarizing the advantages of reformulating the interleaving distance via couplings, allowing us to borrow tools from edit distances.

Figure 14. Flowcharts.

Table 3. A table containing the most important functions which are defined throughout the manuscript. Such functions are fundamental, especially for the results contained in the first half of the manuscript.

Notation	Definition
$s_T^k : \mathbf{T} \to \mathbf{T}$	Structural map of a metric tree $\mathbf{T}$; shifts upwards the points by $k \in \mathbb{R}$
$\pi_T : C \to V_T$	Restirction of Cartesian Product Projection
$\Lambda_C^T : V_T \to \mathcal{P}(V_T)$	$\Lambda_C^T(v) = \max_{v'<v} \pi_T(C)$ or $\emptyset$
$\varphi_C^T : V_T \to V_T$	$\varphi_C^T(x) = \min\{v \in V_T \mid v > x \text{ and } \#\Lambda(v) \neq 0\}$
$\delta_C^T : V_T \to V_T$	$\delta_C^T(x) = \min\{v \in V_T \mid v \geq x \text{ and } v \in \pi_T(C)\}$
$\chi_C^T : V_T \to V_G$	$\chi_C^T(x) = \mathrm{LCA}(\{\pi_G((v,w)) \mid v \in \Lambda_T(x)\})$
$\gamma_C^T : V_T - D_C^T \to V_G$	$\arg\min\{g(w) \mid (v,w) \in C, v < x\}$ or $\emptyset$
$\eta_C^T : V_T \to V_G$	$\eta_C^T(x) = \gamma_C^T(\varphi_C^T(x))$
$\alpha_C : \mathbf{T} \to \mathbf{G}$	The continuous map induced by a coupling C
$\alpha_C^\varepsilon : \mathbf{T} \to \mathbf{G}$	The ε-good map induced by a coupling C: $\alpha_C^\varepsilon(x) = s_G^{k_x}(\alpha_C(x))$ with $k_x = f(x) + \varepsilon - g(\alpha_C(x))$
$[l_C(x), u_C(x)]$	$l_C(x) \in \max\{v \leq x \mid v \in \pi_T(C) \cup D\}$; $u_C(x)$ is either $u_C(x) = +\infty$ or $u_C(x) = \min\{v \geq x \mid v \in \pi_T(C) \cup D\}$
$\ell_T : \mathbf{T} \to V_T$	$\ell_T(x) = \max\{v \in V_T \mid v \leq x\}$
$\phi : V_T \to V_G$	$\phi(v) = \ell(\alpha(v))$
$\psi : V_G \to V_T$	$\psi(w) = \ell(\beta(w))$

B. Comparison with [3] and [25]

We now discuss the computational complexity and practical performance of our algorithms in comparison to those developed in [3] and [25]. To begin with, we note that, to the best of our knowledge, no public implementations of the latter two algorithms are currently available. As a result, we do not provide a case study to evaluate their runtimes or the reliability of their approximations. Similarly, the original works do not include any examples or simulations.

Moreover, these algorithms differ significantly in nature: ours is a BLP-based estimation algorithm equipped with a posteriori error bounds; [3] offers an approximation within a tree-dependent multiplicative factor; and [25] presents an exact fixed-parameter tractable (FTP) algorithm. Given these fundamental differences, making a direct comparison is far from straightforward.

Nonetheless, we attempt a comparison based on both theoretical and practical considerations, focusing on selected cases that, while specific, remain meaningful and illustrative.

We start by focusing on the algorithm in [25]. The authors of [25] show that determining if $d_I(T,G) \leq k$, can be run in $O((N+M+2)^2 \log^3(N+M+2)2^{2\tau}\tau^{\tau+2})$ (with N, M being as in Proposition 6), where τ depends on k, T, and G. Thus, in order make some explicit computations, we pick a situation in which we can parametrize τ in terms of tree complexity. To such extent, we consider a merge tree T with full binary tree structure, as in Proposition 6. We denote this tree by T_l, where $l = \mathrm{len}(T_l)$, and we assume that the height difference between each node and its child is exactly 1. In other words, the structure of T_l is fully described by l. Since each internal vertex has degree 3, for each $k < l$, we have

$\tau(k) \geq 3 \cdot 2^{k+1}$. In fact, k-balls, as defined in [25], are full binary trees of length k. Note that this is neither a worst-case scenario nor a best-case scenario analysis: it is just a reference situation in which we can make some explicit computations. For instance, having the length of the edges decay as $\mathrm{lvl}_{T_l}(v)$ grows would yield bigger values of $\tau(k)$.

Suppose that we want to check if the interleaving distance is bounded from above by $k = 1, 2, 3, 4, 5$. Then, the leading coefficient in the computational cost is, respectively, at least $10^{18}, 10^{43}, 10^{98}, 10^{223}, 10^{501}$, which then needs to be multiplied by $(N + M + 2)^2 \log^3(N + M + 2)$. Note that for $k = 5$, regardless of G, it is already challenging to compute the complexity of the algorithm, let alone running the algorithm itself to decide if $d_I(T_l, G) \leq k$. Plus, we point out that if G has one leaf, we have $d_I(T_l, G) = l$, and having $l = 4, 5$ is well within the range of complexity contained in our runtime simulations. See Appendix C.

To conclude this first comparison, we offer a complementary perspective by referencing an FTP algorithm developed in [4] for a specific edit distance between trees. More precisely, [4] addresses the computation of the edit distance between unordered, labeled trees with positive integer costs.

We stress that the following discussion is intended as a rough, qualitative comparison: to establish a meaningful basis for comparison between the algorithm in [25] and the one in [4], we will consider additional specific scenarios that help align the assumptions and problem settings of the two approaches.

This comparison is motivated by the following considerations:

1) In Section 9.2.9, we show that computing $\min \widetilde{W}^{\downarrow}_{x,y}$ is as expensive as computing the tree edit distance between two unlabeled weighted trees (with the same tree structures as the trees considered to compute the interleaving distance). Plus, the edit distance between unlabeled weighted trees can be framed as an edit distance between unordered labeled trees by setting relabeling, insertion, and deletion costs equal to the absolute difference between the weights of the involved edges (or zero, if only one edge is involved), and making sure that the labels sets of the two trees are disjoint;

2) To the best of our knowledge, [4] is the only work providing an FTP algorithm to exactly compute an edit distance between either unordered or unlabeled trees.

The results in [4] are not explicitly given as polynomials in $\max\{M, N\}$, but rather focus on the existence of a polynomial upper bound on the time complexity in terms of $\max\{M, N\}$. However, they do provide an approximation of the leading coefficient for such polynomial: $O(2.62^k)$. We set $a_k = 2.62^k$. We want to compare such coefficient with $b_k = 2^{2\tau(k)}\tau(k)^{\tau(k)+2} \geq 2^{6\cdot 2^{k+1}}(3 \cdot 2^{k+1})^{3\cdot 2^{k+1}+2}$, which is a lower bound for the leading coefficient of the algorithm in [25], assuming $T = T_l$. Clearly, directly comparing $d_I(T, G) < k$ and $d_E(T', G') < k$, with T' and G' having the same tree structures as, respectively, T and G, is not fair. In fact, d_I is a "sup" kind of distance, collecting only the biggest difference between trees, while d_E collects all the differences between trees, which are added to obtain the final value. Thus, when computing d_E, we multiply k by the number of edges in T' and G', i.e., we compare $d_I(T, G) < k$ with $d_E(T', G') < k(M + N)$. For $N = M = 20$, and $k = 1, 2, 3, 4, 5$, we obtain: $a_k \sim 10^8, 10^{16}, 10^{25}, 10^{33}, 10^{42}$ while $b_k \sim 10^{18}, 10^{43}, 10^{98}, 10^{223}, 10^{501}$.

Now, we focus on [3]. The authors of the paper provide a $O((M + N + 2)^{5/2} \log(M + N + 2))$ time algorithm, to approximate d_I up to a factor of $O(\min\{M + N + 2, \sqrt{r \cdot (M + N + 2)}\})$, with r being equal to the ratio of the longest edge across T and G, over the smallest edge across T and G. More precisely,

it takes $O((M+N+2)^2)$ to determine if $d_I(T,G) \leq k$ up to a multiplicative factor of:

$$c = 2\min\{(M+N+2), 2\sqrt{r\cdot(M+N+2)}+1\}.$$

Note that c depends on T and G, and cannot be controlled or improved by changing some parameters.

Let $\widetilde{d}(T,G)$ be the value returned by the algorithm in [3]. We know that $d_I(T,G)/\widetilde{d}(T,G) \leq c$, and thus:

$$\frac{1}{c}\cdot\widetilde{d}(T,G) \leq d_I(T,G) \leq c\cdot\widetilde{d}(T,G).$$

So, we can get a maximum range for the relative error introduced by $\widetilde{d}$, by taking $c-1/c$, similarly to what was done with Eq (10.1). We compute such error ranges across the first four datasets considered in Section 10.3 to compare them against the ones we obtained with Δ.

We obtained the following mean values for the error ranges: 35.19 ("Growth"), 30.63 ("Octane"), 39.93 ("NO_x"), 35.22 ("Tecator"). We point out that this means that, on average, the error range produced by $\widetilde{d}$ is more than 300% of $\widetilde{d}(T,G)$. Our approach, instead, produced much more reliable estimates, as for 95% of our distances, Δ was less than 0.008. Thus, we argue that, whenever feasible, our estimation should be preferred to the techniques in [3].

C. Computational runtimes

We now present a simulation study to showcase the computational runtimes of the implementation that was used in the case studies in Section 10. The data generating pipeline is the same as in Section 10.1, with the cardinality of the point cloud, i.e., the number of leaves in the trees, being $n_i = 5, 10, 15, 20, 25, 30, 35$. For each value of n_i, we sample 200 couples of point clouds, and compute d_u betwen the associated merge trees. We report the boxplots of the obtained runtimes in Figure 15.

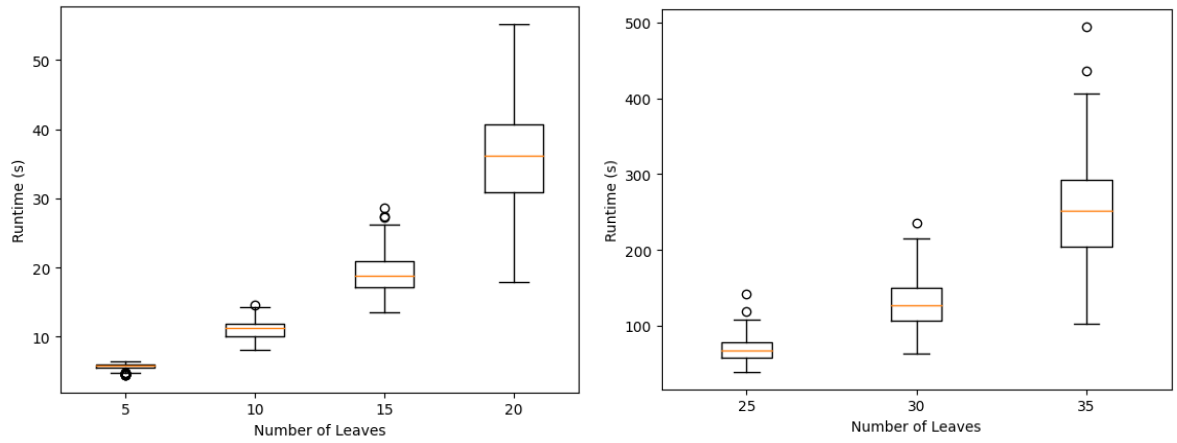


Figure 15. Boxplots of the runtimes, as a function of the number of leaves in each merge tree, for the simulation in Appendix C.

We close the simulation with the following disclaimer: given the nature of the present work, our focus has not been on optimizing the code or the computational environment. For instance, we used a non-commercial version of the Gurobi 12 solver, without having ample RAM dedicated exclusively to these computations. Consequently, the reported runtimes do not accurately reflect the potential

efficiency of a fully optimized implementation. See state-of-the-art implementations of metrics with similar complexity, such as [31].

D. Proofs

D.1. Proof of Proposition 1

We consider separately the different combinations arising from the considered vertices belonging to $\pi_T(C)$ or D.

- The thesis is obvious if $x, x' \in \pi_T(C)$;
- If $x \in \pi_T(C)$ and $x' \in D$, then $w \geq \chi(x')$, with $(x, w) \in C$. Otherwise, (C3) is violated since $\#\Lambda(x) > 1$, and so, $\alpha_C(x) \geq \alpha_C(x')$;
- If $x' \in \pi_T(C)$ and $x \in D$, then $\chi(x) \leq w$ with $(x', w) \in C$;
- Lastly, suppose $x, x' \in D$ and consider the following cases:
  - If $\#\Lambda(x) = \#\Lambda(x') = 0$, then we have $\varphi(x) = \varphi(x')$, and, thus, $f(x) + \frac{1}{2}(f(\varphi(x)) - f(x)) \leq f(x') + \frac{1}{2}(f(\varphi(x')) - f(x'))$. This entails $g(\alpha_C(x)) \geq g(\alpha_C(x'))$;
  - If $\#\Lambda(x), \#\Lambda(x') > 1$, then we have $\chi(x) \leq \chi(x')$, and the thesis clearly follows;
  - Lastly, suppose that $\#\Lambda(x) = 0$ and $\#\Lambda(x') = 1$; then, $\eta(x) \leq \chi(x')$.

D.2. Proof of Lemma 1

Given $x \in \mathbf{T}$, if $\max \pi_T(C) \in \{v \in \mathbf{T} \mid v \geq x\}$, we have a well-defined upper extreme $u_C(x)$. Moreover, since $\{v \in \mathbf{T} \mid v \geq x\}$ is totally ordered, $u_C(x)$ is unique. If $\max \pi_T(C) \notin \{v \in \mathbf{T} \mid v \geq x\}$, i.e., $\mathrm{LCA}(x, \max \pi_T(C)) > \max \pi_T(C)$, then $u_C(x) = +\infty$.

Now, consider $x \in L_T$. Since $sub_T(x) = \{x\}$, then $x \notin U$. Thus, also $l_C(x)$ is well-defined. On top of that, since by hypotheses, we are considering only vertices such that $x \notin \pi_T(C) \cup D$, $x \neq l_C(x) \neq u_C(x)$. Thus, $[l_C(x), u_C(x)]$ is a nondegenerate sequence of edges.

Suppose $\{v \in U \mid l_C(x) \leq v \leq x\} = \emptyset$ and $v', v'' \in \max\{v \leq x \mid v \in \pi_T(C) \cup D\}$. Clearly, $[v', x] \bigcap [v'', x] \bigcap V_T \neq \emptyset$; so consider $p \in [v', x] \bigcap [v'', x] \bigcap V_T$. We know $p \notin U$, thus, $p \in \pi_T(C) \cup D$, which is absurd since $v', v'' < p$.

Lastly, suppose $x \in U$ (and so $\#\Lambda(x) = 1$) and $\max\{v \leq x \mid v \in \pi_T C \cup D\} \subset D$. Then, $\#\Lambda(l_C(x)) \neq 1$ for any $l_C(x)$. Let $\{v\} = \Lambda(x)$ and consider $[v, x]$. If $v' \in U$ for all $v < v' < x$, we are done. Clearly, there cannot be vertices v' with $v < v' < x$ which are in $\pi_T(C)$. So suppose there is $v' \in D$, with $v < v' < x$. Since $v \in \Lambda(v')$, we have $\#\Lambda(v') > 0$, but then $\#\Lambda(v') > 1$, which means $v, v'' \in \Lambda(v')$ for some $v'' < v'$. Clearly, in $[v'', x]$ there can be no vertex contained in $\pi_T(C)$ apart from v''. Thus, $\#\Lambda(x) > 1$, which is absurd.

D.3. Proof of Proposition 3

We need to check continuity. Consider $x_n \to x$ in $\mathbf{T}$. We know that their order relationships is preserved by α_C and α_C^{ε}. On top of that, $f(x_n) \to f(x)$ yields $f(x_n) + \varepsilon \to f(x) + \varepsilon$ and the result follows.

D.4. Proof of Lemma 2

We know that $\#\Lambda(x), \#\Lambda(y) > 1$. Thus, x, y are either deleted or coupled. Note that $\chi(x) \geq y$ and $\chi(y) \geq x$. If both of them are coupled, then $(x, y) \in C$. Suppose x is coupled with $\delta(y)$ and y is deleted. Then, $cost(y) =\mid f(\chi(y)) - g(y) \mid< \varepsilon$ and $cost(x) =\mid f(x) - g(\delta(y)) \mid< \varepsilon$. Since $\delta(y) \geq y$ and $\chi(y) \geq x$, we obtain the thesis.

D.5. Proof of Theorem 1

First, we check continuity, and then we check (P1)–(P3).

- α_C^ε is continuous by Proposition 3.
- (P1) holds by Proposition 3.
- Now, we prove (P2). Suppose we have $\alpha_C^\varepsilon(v) < \alpha_C^\varepsilon(v')$ and set $x = \text{LCA}(v, v')$.

  We can suppose $\#\Lambda(x) > 1$ and $\text{LCA}(\Lambda(x)) = x$; otherwise, at least one between $\varphi(v) \geq x$ and $\varphi(v') \geq x$ holds. Suppose the second one holds, then $\#\Lambda(v') = 0$ and $\varepsilon \geq (f(\varphi(v')) - f(v'))/2$ and $\varphi(v') \geq x$. Thus, (P2) holds. The same if the first one holds.

  Now, we show that if $\#\Lambda(x) > 1$, we can find $a, b \in V_T$ such that:

  – $x = \text{LCA}(a, b)$;
  – $(a, a'), (b, b') \in C$;
  – $\alpha_C(v) \geq \alpha_C(a)$ and $\alpha_C(v') \geq \alpha_C(b)$.

  Note that, in this case, upon calling $y = \text{LCA}(a', b')$, we have: $\mid f(x) - g(y) \mid\leq \varepsilon$ by Lemma 2, $\alpha_C^\varepsilon(v') \geq \{\alpha_C(v), \alpha_C(a), \alpha_C(b)\}$, and so, $\alpha_C(v') \geq y$, which means that $f(v') + \varepsilon \geq g(y)$. Thus, $f(x) - f(v') \leq 2\varepsilon$.

  We enumerate all the possible situations for v; clearly, the same hold for v':

  – $v \in \pi_T(C)$: then $a = v$;
  – $\Lambda(v) = 0$: then $a = v''$ with $(v'', \eta(v)) \in C$; by hypothesis, $\varphi(v) \leq x$ and $v'' < x$;
  – if $v \notin \pi_T(C)$ and $\#\Lambda(v) > 0$: then $a \in \Lambda(v)$.

- Now, we prove (P3). If $w \notin Im(\alpha_C^\varepsilon)$, then $w \in D_C^G$. In fact, if $(x, w') \in C$ with $w' < w$, then $[w', w] \subset Im(\alpha_C^\varepsilon)$, since there are $l(w) \geq w'$ and $u(w) \leq r_G$. Then, we know that there is $w'' = \min\{y \in V_G \mid y > w \text{ and } y \notin D_C^G\}$ with $g(w'') - g(w) \leq \varepsilon$.

D.6. Proof of Corollary 1.

We know that $w := \alpha(v) \leq w' := \alpha(v')$; thus, $\max\{y \in V_G \mid y \geq w\} \leq \max\{y \in V_G \mid y \geq w'\}$, and the thesis follows.

D.7. Proof of Theorem 2

We build C by subsequently adding couples starting from an empty set. The proof is divided in sections which should help the reader in following the various steps.

Before starting, following [25], we can take $\beta : \mathbf{G} \to \mathbf{T}$ induced by α so that α and β are ε-compatible.

Step 1. Leaves of T

Step 1.1. Selecting the coupled leaves

We consider the following set of leaves:

$$\mathcal{L}_T = \{v \in L_T \mid \nexists v' \in L_T \text{ such that } \alpha(v) < \alpha(v')\}. \tag{D.1}$$

We give a name to the condition:

(a) $\nexists v' \in L_T$ such that $\alpha(v) < \alpha(v')$,

so that we can more easily use it during the proof. Note that we can avoid treating the case $\alpha(v) = \alpha(v')$ thanks to (G).

The set $\mathcal{L}_T$ is the set of leaves which will be coupled by C, while all other leaves will be deleted: we add to C all the couples of the form $(v, \phi(v))$ with $v \in \mathcal{L}_T$. We characterize those couples with the following proposition.

Lemma 5. Given $v, v' \in \mathcal{L}_T$, then $\phi(v) \geq \phi(v')$ if and only if $v = v'$. Moreover, for every $v' \in L_T$ such that (a) does not hold, there is $v \in \mathcal{L}_T$ such that $\alpha(v) < \alpha(v')$.

Proof. The first part of the proof reduces to observing that $\phi(v) \leq \phi(v')$ if, and only if, $\alpha(v) \leq \alpha(v')$.

Now, consider $v' \in L_T$ such that (a) does not hold. We know there is v_0 such that $\alpha(v_0) < \alpha(v')$. If $v_0 \in \mathcal{L}_T$, we are done; otherwise, there is v_1 such that $\alpha(v_1) < \alpha(v_0) < \alpha(v')$. Note that $f(v_1) < f(v_0)$. Thus, we can carry on this procedure until we find $v_i \in \mathcal{L}_T$. Note that $\arg\min_{V_T} f \in \mathcal{L}_T$; thus, in a finite number of steps, we are done. □

Step 1.2. Cost bound on couples

Now, we want to prove the following proposition, which gives an upper bound for the cost of the couples added to C.

Lemma 6. Given $v \in \mathcal{L}_T$, then $| f(v) - g(\phi(v)) | \leq \varepsilon$.

Proof. Suppose the thesis does not hold. Since $g(\phi(v)) \leq f(v) + \varepsilon$, contradicting the thesis means that we have $v \in \mathcal{L}_T$ such that:

(b) $g(\phi(v)) + \varepsilon < f(v)$.

Let $w = \phi(v)$. If (b) holds, then $g(father(w)) - g(w) > g(\alpha(v)) - g(w) > 2\varepsilon$. Let $v' = \psi(w) \leq \beta(w)$. Note that $f(v') < f(v)$. We have $\phi(v') \leq \alpha(v') \leq \alpha(\beta(w)) = s_G^{2\varepsilon}(w)$. Since $g(father(w)) - g(w) > 2\varepsilon$, we also have $\alpha(v') \leq \alpha(v)$ with $v' \neq v$, which is absurd by Lemma 5. □

Step 1.3. Cost bound on deletions

In this step, we prove the following proposition which gives an analogous bound to the one of Lemma 6, but for the deleted leaves of T.

Lemma 7. Given $v \in L_T - \mathcal{L}_T$, then there exists $x > v$ such that:

- There is $v' < x$ such that $v' \in \mathcal{L}_T$;
- $f(x) \leq f(v) + 2\varepsilon$.

Proof. Since (a) does not hold for v, we use Lemma 5 to obtain $v' \in \mathcal{L}_T$ such that $\alpha(v') < \alpha(v)$. However, since α is an ε-good map, we have $s_T^{2\varepsilon}(v') \leq s_T^{2\varepsilon}(v)$ which implies $f(\mathrm{LCA}(v, v')) \leq f(v) + 2\varepsilon$. Thus, $x = \mathrm{LCA}(v, v')$ ends the proof. □

Lemma 7 implies that, using the notation of the proposition, $\varphi(v) \leq x$. Then, $f(v) < f(v')$ implies that $g(\phi(v)) \leq f(v') + \varepsilon < f(v) + \varepsilon$. Thus, $g(\eta(v)) < f(v) + \varepsilon$. Since $f(x) \leq f(v) + 2\varepsilon$, we have that the cost of deleting any $x' < x$ with $\#\Lambda(x') = 0$ is less then ε.

Step 2. Leaves of G

Lemma 8. Given $w \in L_G$, there exists $y \geq w$ such that:

- There is $w' = \phi(v)$ with $v \in \mathcal{L}_T$ and $w' < y$;
- $g(y) \leq g(w) + 2\varepsilon$.

Proof. Consider $\beta(w)$. Let $v \leq \beta(w)$ leaf. We have $\alpha(\beta(w)) \geq \mathrm{LCA}(\alpha(v), w)$. If $v \in \mathcal{L}_T$ we are done for $\phi(v) \leq \alpha(v)$. If (a) does not hold by Lemma 5, it means that there is $v' \in \mathcal{L}_T$ such that $\alpha(v') < \alpha(v)$. We are done since $g(\alpha(\beta(w))) = g(w) + 2\varepsilon$. □

Note that if $w < \phi(v)$ for some $v \in \mathcal{L}_T$, then, by Lemma 8, $g(\phi(v)) \leq g(w) + 2\varepsilon$. In fact, using the notation of Lemma 8, in this case, we have $w' = y = \phi(v)$ by definition. As in Step 1.3, we have that the cost of the deletion of any w such that $\#\Lambda(w) = 0$ and $w \notin \pi_G(C)$ is at most ε.

Step 3. Internal vertices

Now, we need to extend the coupling C taking into account the internal vertices of T. We will do so after simplifying our merge trees in two different ways: First, we remove all vertices which are deleted with $\#\Lambda(p) = 0$, and then we take out all inessential internal vertices.

Step 3.1. Pruning

Let $T_0 = T$ and $G_0 = G$. We define T_1 as the merge tree obtained from T_0 deleting the following set of vertices (and the corresponding edges): x satisfying both $\#\Lambda(x) = 0$ and $x \notin \pi_T(C)$. Note that, for any $x \in V_T$, either there is $v \leq x$ with $v \in \mathcal{L}_T$ or, for any leaf below x, we can apply Lemma 7 and the consequential observations.

The tree G_1 is obtained from G_0 in an analogous way: any time we have $w \in V_G$ satisfying both $\#\Lambda(w) = 0$ and $w \notin \pi_G(C)$, w is deleted from G_0, along with the edge $(w, father(w))$.

Before proceeding, we point out that, by construction, the leaves of T_1 are exactly $\mathcal{L}_T$.

Step 3.2. Restricting α

Thanks to Corollary 1, we have that anytime we delete some vertex in G_0 to obtain G_1 and that vertex is in the image of ϕ, we are sure that also its counterimage is deleted from T_0.

Now, for every $v \in \mathbf{T}$, by construction we have that $\alpha(v)$ belongs to an edge removed from G_0 if, and only if, $\phi(v)$ is deleted from G_0. Consider $v' = \ell(v) \in V_T$. Then, $\phi(v') \leq \phi(v)$, and thus, $\phi(v')$ is deleted as well. This entails that v' is deleted as well. All of this put together implies that we can restrict α to $\mathbf{T}_1$ (the metric tree obtained from T_1), and its image lies in $\mathbf{G}_1$ (obtained from G_1).

Step 3.3. ε-Good restriction

We define $\alpha_1 := \alpha_{|\mathbf{T}_1} : \mathbf{T}_1 \to \mathbf{G}_1$. We want to prove that α_1 is still an ε-good map. Clearly, $(P1)$ and $(P2)$ still hold upon restricting the domain. We just need to show $(P3)$. Suppose there is $w \in V_{G_0}$ such that $w \notin \alpha_1(\mathbf{T}_1)$. We distinguish between two cases: (1) $w \notin \phi(\mathbf{T}_1)$ and (2) $w \in \phi(\mathbf{T}_1)$. Consider scenario (1): $w \notin \phi(\mathbf{T}_1)$ clearly implies that $\#\Lambda(w) = 0$, and so, w is deleted; scenario (2) instead means that there is $\alpha(v) = \min\{\alpha(v') > w \mid v' \in \mathbf{T}_1\}$ with $\ell(\alpha(v)) = \phi(v) = \ell(w)$. Clearly, $\{w' \in \mathbf{G}_0 \mid w' < w\} \bigcap \alpha(\mathbf{T}_1) = \emptyset$, and, thus, v is a leaf of T_1, which means $v \in \mathcal{L}_T$. This also implies that $w \notin \alpha(\mathbf{T})$, and, in particular, $\alpha(v) = \min\{\alpha(v') > w \mid v' \in \mathbf{T}_0\}$, thus condition $(P3)$ is satisfied.

Step 3.4. Properties of $\phi : \mathbf{T}_1 \to \mathbf{G}_1$ and removal of inessential vertices

We know that $\phi : L_{T_1} \to L_{G_1}$ is injective. On top of that, we have proved that, if $w \notin \alpha_1(\mathbf{T}_1)$, then $\ell(w) = \phi(v)$ for some $v \in \mathcal{L}_T$. Thus, $\phi : L_{T_1} \to L_{G_1}$ is a bijection.

From now on we will ignore any vertex v such that $\#child(v) = 1$. Formally, we introduce T_2 (and G_2) obtained from T_1 (G_1) removing all the vertices $v \in V_{T_1}$ such that $\#child(v) = 1$, (similarly, $w \in V_{G_1}$ such that $\#child(w) = 1$): consider $\{v'\} = child(v)$. We remove v from V_{T_1} and replace the edges (v', v) and $(v, father(v))$ with the edge $(v', father(v))$. We do this operation recursively until no vertices such that $\#child(v) = 1$ can be found.

Step 3.5: Coupling and deleting internal vertices

We start with the following lemma:

Lemma 9. For every $x \in V_{T_2}$, and $y \in V_{G_2}$, we have $| x - \chi(x) | \leq \varepsilon$ and $| y - \chi(y) | \leq \varepsilon$.

Proof. Let $x \in V_{T_2} - L_{T_2}$. Then, $x = \text{LCA}(v_1, \ldots, v_n)$ with $v_1, \ldots, v_n$ being the leaves of $sub_{T_2}(x)$. Then, $\alpha(x) > \alpha(v_i)$ for every i, and, thus, $\alpha(x) \leq w = \chi(x) = \text{LCA}(\alpha(v_1), \ldots, \alpha(v_n))$. Clearly, $x \leq \beta(w)$ for analogous reasons. Thus, $| x - w | \leq \varepsilon$. For $y \in V_{G_2}$, we can make an analogous proof. □

Let $v_1, \ldots, v_n \in L_{T_2}$ and $\phi(v_1), \ldots, \phi(v_n) \in L_{G_2}$. Let $x = \text{LCA}(v_1, \ldots, v_n)$ and $y = \text{LCA}(\phi(v_1), \ldots, \phi(v_n))$. We know that $\alpha(x) \geq y$ and $\beta(y) \geq x$, and so $| f(x) - g(y) | \leq \varepsilon$.

Thus, we proceed as follows: we order all the internal vertices of T_2 according to their height values and we start from the higher one (the root of T_2; note that this, in general, is not the root of T_0), coupling it with the other root (thus, $(C1)$ is verified). Consider a lower x, and let $v_1, \ldots, v_n$ be the leaves of $sub_{T_2}(x)$, (and, thus, $x = \text{LCA}(v_1, \ldots, v_n)$). Let $w = \text{LCA}(\phi(v_1), \ldots, \phi(v_n))$. If the leaves of $sub_{G_2}(w)$ are $\phi(v_1), \ldots, \phi(v_n)$, we add the couple (x, w), otherwise, we skip x which will be deleted, with $\#\Lambda(x) > 1$. Note that $w = \chi(x)$. Moreover, by Lemma 9, $| f(x) - g(w) | \leq \varepsilon$. Going from the root downward we repeat this procedure for every x.

We clearly have:

- (C3) is satisfied;
- $cost((v, w)) \leq \varepsilon$;
- If $v \in V_{T_2}$ is deleted, then $\#\Lambda(v) > 1$ and $cost(v) = | f(v) - g(\chi(v)) | \leq \varepsilon$;
- If $w \in V_{G_2}$ is deleted, then $\#\Lambda(w) > 1$ and $cost(w) = | g(w) - f(\chi(w)) | \leq \varepsilon$ by Lemma 9.

Step 4. Coupling Properties and Costs

First, we verify that C is a coupling:

$(C1)$ See Step 3.5;

($C2$) See Step 1 and Step 3.5, we carefully designed the coupling so that no vertex is coupled two times;

($C3$) It is explicitly proven in Step 3.5;

($C4$) In Step 3.4, we remove all vertices such that $\#child(x) = 1$ to obtainT_2 and G_2; all the couples in C are either leaves of vertices in T_2 and G_2. So, since all leaves of T_2 and G_2 are coupled, its impossible to have a vertex p belonging to any tree such that $\#\Lambda(p) = 1$.

Lastly, we verify its costs:

- If $(x, y) \in C$, then $| f(x) - g(y) | \leq \varepsilon$ as verified in Step 1.2 for $x \in L_T$, and in Step 3.5 for x being an internal vertex;
- If $x \in V_T \cap D$, with $\#\Lambda(x) = 0$, it is verified that $| f(x) - \phi(x) | \leq 2\varepsilon$ in Step 1.3; if instead $y \in V_G \cap D$, we verify $| g(y) - \phi(y) | \leq 2\varepsilon$ in Step 2; in both cases, we have a vertex lower than x (y) which, by construction, is coupled, and the cost of the couple is less then ε. Thus, the deletion of x (y) costs less than or equal to ε;
- If $p \in D$, with $\#\Lambda(p) > 1$, then we verify in Step 3.5 that the cost of this deletion is less than or equal to ε.

This concludes the proof.

D.8. Proof of Theorem 3

First, we prove the second part of the statement.

Given $C \in C(T, G)$ optimal coupling such that $\{(r, r')\} = \max C$, we define $\mathcal{M}(C) := \max(C - \{(r, r')\})$. Clearly, $\mathcal{M}(C) \in C^*(T, G)$. For any $(x, y) \in \mathcal{M}(C)$, consider $C^o_{x,y} \in C^o_R(T_x, G_y)$ such that $\| C^o_{x,y} \|_\infty \leq \| C_{|(x,y)} \|_\infty$ with $C_{|(x,y)} := \{(v, w) \in C \mid (v, w) < (x, y)\}$. By hypothesis, we know that $C^o_{x,y}$ exists for every x, y. Note that $C_{|(x,y)} \in C_R(T_x, G_y)$.

We want to prove that the extension $C' := \{(r, r')\} \cup (\bigcup_{(x,y)\in\mathcal{M}(C)} C^o_{x,y})$ satisfies $\| C' \|_\infty \leq \| C \|_\infty$. The set C' clearly is a coupling since for every $x, x' \in \pi_T(\mathcal{M}(C))$, $sub_T(x)$ and $sub_T(x')$ are disjoint, and the same for $y, y' \in \pi_G(\mathcal{M}(C))$. We need to consider the costs of different kinds of vertices separately. In particular, we indicate with (a) whenever we have $v \in V_T$ such that $v \leq x$ for some $x \in \pi_T(\mathcal{M}(C))$. If this condition does not hold, we say that (b) holds for v. The same definitions apply also for vertices in G. For instance, (a) holds for all vertices in $\pi_T(\mathcal{M}(C))$ and $\pi_G(\mathcal{M}(C))$. Similarly, if (b) holds for v, then it holds also for all $v' > v$.

- Consider $v \in V_T$ such that (a) holds for some $(x, y) \in \mathcal{M}(C)$; we have that $\chi_{C'}(v) \leq y$, $\varphi_{C'}(v) \leq x$ and $\eta_{C'}(v) \leq y$, so $cost(v)$ depends only on $C^o_{(x,y)}$. Thus, by construction, $cost_{C'}(v) = cost_{C^o_{(x,y)}}(v) \leq \| C_{|(x,y)} \|$;
- Suppose now (b) holds for $v \in V_T$; by construction, in this case, we have $\chi_C(v) = \chi_{C'}(v)$, $\Lambda_{C'}(v) = \Lambda_C(v)$, and, thus, $\varphi_C(v) = \varphi_{C'}(v)$. Suppose $\Lambda_{C'}(v) = 0$: (a) holds for $\eta_{C'}(v)$ by means of some $y \in \pi_G(C(M))$, and, thus, $g(\eta_{C'}(v)) = \min_{w' \leq y} g(w') \leq g(\eta_C(v))$ (thanks to the properties of couplings in $C^o_R(T_x, G_y)$). Since $\varphi_C(v) = \varphi_{C'}(v)$, we then have $cost_{C'}(v) \leq cost_C(v)$. Lastly, consider $\#\Lambda_{C'}(v) > 1$. In this case, we have $\chi_C(v) = \chi_{C'}(v) = r'$, and, thus, $cost_{C'}(v) = cost_C(v)$. Note that in all these situations, $cost_{C'}(v)$ does not depend on the chosen extension.

Exchanging the role of T and G, we obtain the same results for the vertices of G.

The first part of the statement follows with analogous reasoning, by dropping the special coupling constraints and focusing on the vertices $v \in \pi_T(e(C^*))$ or $\#\Lambda_{e(C^*)}(v) > 0$. In particular, the special couplings constraints are necessary only if $\#\Lambda_{e(C^*)}(v) = 0$.

D.9. Proof of Lemma 3

1) This is due to the fact that all vertices of $P\varepsilon(T)$ are coupled, and, thus, $P\varepsilon(T)$ is still a rooted tree and (P) does not alter the order relationships of the remaining vertices;

2) Since (P) removes at most one leaf from the previous tree and adds no new leaves, it is clear that $L_{P_\varepsilon(T)} \subset L_T$. Consider now $v \in L_T$. Define:

$$v_- = \max\{v' \in V_T \mid v' \geq v \text{ and } f(v') < f(v) + \varepsilon\},$$

$$v_+ = \min\{v' \in V_T \mid v' \geq v \text{ and } f(v') \geq f(v) + \varepsilon\}.$$

Note that both vertices are well-defined and $(v_-, v_+) \in E_T$.

Let $v' = \arg\min_{v'' \leq v_+} f(v'')$. We show that v' can't be deleted by the recursive application of (P), which also implies the thesis. Suppose it is deleted.

By construction, v' is the last leaf, in terms of recursive applications of (P) and among the leaves below v_-, that are deleted. In fact, at any step along the recursion, any sibling v'' of v' will always have a lower distance to the common father. Thus, v' cannot be deleted until it has siblings, i.e., all other leaves below v_- are deleted as well. This also means that, at some point along the recursion, v_- becomes the father of v'. After all the siblings of v' are removed from the tree, finally, v_- becomes a degree 2 vertex with v' as an only child, and it is removed as well. So, v_+ becomes the father of v'. At this point we have:

$$f(v_+) - f(v') \geq f(v_+) - f(v) \geq \varepsilon,$$

which is absurd as, for v' to be deleted, we must have $f(father(v')) - f(v') < \varepsilon$;

i) Consider $v \in V_T - V_{P_\varepsilon(T)}$; let $\Lambda_{C_\varepsilon}(v) = \{a_1, \ldots, a_n\}$. By construction, $v \geq x = \text{LCA}(\Lambda_{C_\varepsilon}(v))$. We know that a vertex is removed from T if, at a certain point along the recursive application of (P), it becomes a degree 2 vertex or a small-weight leaf. Thus, the vertex x is not removed from T unless $n \leq 1$, which is absurd because then $\Lambda_{C_\varepsilon}(v) = \{x\}$. Thus, $\#\Lambda(v) \leq 1$. As a consequence, $v \in D$ if, and only if, $\#\Lambda(v) = 0$.

The vertex $\varphi_{C_\varepsilon}(v)$ is the first vertex x above v such that there is a leaf v', $v' < x$, with $f(x) - f(v') \geq \varepsilon$. If $f(x) - f(v) \geq \varepsilon$, then by point (2) there exist $v' \in L_{P_\varepsilon(T)}$ with $\text{LCA}(v', v) < \varphi_{C_\varepsilon}(v)$ which is absurd;

ii) Consider two leaves $v, v' \in L_T$ with $f(v) < f(v')$, with $v \notin V_{P_\varepsilon(T)}$ and $v' \in V_{P_\varepsilon(T)}$. On top of that, suppose $v' < \varphi_{C_\varepsilon}(v)$. By point (3) we have $f(\varphi_{C_\varepsilon}(v)) - f(v') < \varepsilon$ which means that v' is a small-weight leaf, which is absurd;

iii) Combining points (3) and (4), we see that we only have deletions with $\#\Lambda_{C_\varepsilon}(v) = 0$ and $2 \cdot cost_{C_\varepsilon}(c) = f(\varphi_{C_\varepsilon}(v)) - f(v) \leq \varepsilon$.

D.10. Proof of Proposition 4

Consider $C \in C(P_\varepsilon(T), P_\varepsilon(G))$. Then, C can be seen also as a coupling $C \in C(T, G)$. Let C_ε be as in Lemma 3.

We partition the vertices V_T into three sets:

- The case $v \in V_{P_\varepsilon(T)}$ is not a concern since the cost doesn't change when considering $v \in V_T$ or $v \in V_{P_\varepsilon(T)}$;
- If $v \in U^T_{C_\varepsilon}$, then $v \in U^T_C$ or $v \in D^T_C$. We ignore the first case. In the second case, $\{v'\} = \max\{v'' < v \mid v'' \in V_{P_\varepsilon(T)}\}$, is such that $v' \in D^T_C$, and so $cost_C(v) < cost_C(v')$;
- If $v \in D^T_{C_\varepsilon}$, then $v \in D^T_C$ and $\#\Lambda_C(v) = 0$; by construction $\varphi_{C_\varepsilon}(v) \leq \varphi_C(v)$. If $\varphi_{C_\varepsilon}(v) < \varphi_C(v)$, then also η_{C_ε} is deleted and, since $f(\eta_{C_\varepsilon}(v)) < f(v)$, $cost_C(v) < cost_C(\eta_{C_\varepsilon}(v))$.

  So, we are left with the case $\varphi_{C_\varepsilon}(v) = \varphi_C(v)$. If $g(\eta_C(v)) - f(v) > 0.5 \cdot (f(\varphi_C(v) - f(v)))$, then, again, $cost_C(v) < cost_C(\eta_{C_\varepsilon}(v))$, for $f(\eta_{C_\varepsilon}(v)) < f(v)$. Thus, we always have $cost_C(v) \leq \max\{\varepsilon/2, cost_C(\eta_{C_\varepsilon}(v))\}$.

Exchanging the role of T and G, and repeating the same observations, we obtain that, if we consider $C \in C^o(T, G)$, we have $d_I(T, G) \leq \| C \|_\infty \leq \max\{d_I(P_\varepsilon(T), P_\varepsilon(G)), \varepsilon/2\}$.

D.11. Proof of Lemma 4

- If $v \leq r$, then $r' \geq \chi(v), \varphi(v)$ and $r' \geq \eta(v)$, so $cost(v)$ depends only on $C_{r,r'}$;
- If $v \not\leq r$, then it is either unused, if $v > r$, or it is deleted with $\#\Lambda(v) = 0$. Then the cost of such deletion is either $0.5(f(v_r) - f(v))$ or $g(\eta(v)) - f(v)$. If $C_{r,r'} \in C^o_R(T_r, G_{r'})$, $g(\eta(v)) = g_r$, and so we are done. Otherwise, we simply have $g(\eta(v)) \geq g_r$, and, thus, $g(\eta(v)) - f(v) \geq g_r - f(v)$ implying the lower bound that we needed to prove.

D.12. Proof of Proposition 5

We just explore the different pieces of the cost function to assess the thesis:

- In the case of $(x, y) \in C_{\mathcal{V}}$, the cost of coupling T_x and G_y is given by $\sum_y a_{x,y} \| C_{x,y} \|_\infty$;
- If $u_x = 1$, we have $| g(r_G) - f(x) | u_x$ which is the cost of deleting x with $\#\Lambda(x) > 1$. That is, any time $x \geq x'$ with $x' = \mathrm{LCA}(v, v')$ for $v, v' \in \pi_T(C_{\mathcal{V}})$. Recall that, in this case, we have $\chi(x) \leq r_G$;
- Deleting x with $\#\Lambda(x) = 0$ is taken care by the remaining part the cost function. For a vertex x, we indicate with x_f its father, and set $A_x = 0.5(f(x_f) - f_x)(1 - d_x)$ and $B_x = \{\sum_y a_{v,y}\left(g_y - f_x\right) - Kd_x\}_{v \in E_T, v < x_f}$. Now, we try to unveil the meaning of A_x and B_x. Recall that deleting x with $\#\Lambda(x) = 0$ corresponds to having $d_x = 0$. If $d_x = 1$, then $A_x = 0$ and $\max B_x < 0$; while if $d_x > 1$, both A_x and $\max B_x$ are negative. Consider now $d_x = 0$. Then, $\varphi_{C_m}(x) = x'_f$, with x'_f being father of some $x' \geq x$. Clearly, $d_{x'} = 0$ as well and $A_{x'} > A_x$. In particular, $A_{x'} = \max_{v \leq x'} 0.5\left(f(\varphi_{C_m}(v)) - f_v\right)$ (the same holds for C^o). Now, we turn to B_x. If $x < x'$, then $x_f < \varphi_{C^o}(x) = x'_f$, and so $d_{x_f} = 0$, entailing $\max B_x = \min B_x = 0$. Instead, if $x = x'$, we have by construction $v < x'_f$ with $a_{v,y} = 1$. Then, $\max\{\sum_y a_{v,y} g_y \mid v < x_f\} \geq g(\eta_{C^o}(x'))$ and so:

$$A_{x'} \leq \max_{v \leq x'} cost_{C^*}(v) \leq \max\{A_{x'}, \max B_{x'}\}. \tag{D.2}$$

D.13. Proof of Corollary 3

Given $C^* \in C^*(T, G)$, with $r = \mathrm{LCA}(\pi_T(C^*))$ and $r' = \mathrm{LCA}(\pi_G(C^*))$, thanks to Corollary 2, we can approximate with $\Gamma^{\uparrow}$ and $\Gamma^{\downarrow}$ the costs of the vertices in T_r and $T_{r'}$. By Lemma 4, $H_{r,r'}$ then takes care of the vertices $v \not\leq r$ and $w \not\leq r'$.

AIMS Press