Merge Tree Geodesics and Barycenters with Path Mappings

Florian Wetzels, Mathieu Pont, Julien Tierny, Christoph Garth

Merge Tree Geodesics and Barycenters with Path Mappings

Florian Wetzels, Mathieu Pont, Julien Tierny, and Christoph Garth

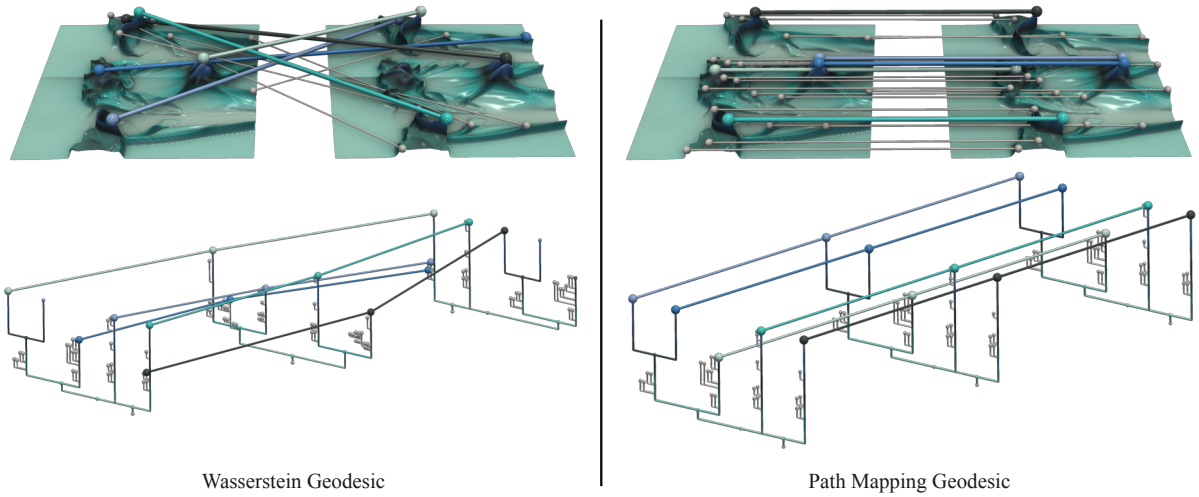

Fig. 1: Two merge tree geodesics with their corresponding mappings. The top row shows the mappings of critical points (restricted to maxima) embedded into the domain. In the bottom row the mapping between the two corresponding merge trees and their interpolated geodesic is shown. The most persistent mapped features are highlighted and colored to show the correspondence between embedding and planar layout. The Wasserstein geodesic can be seen on the left, the path mapping geodesic on the right. Clearly, the middle tree on the right is a more meaningful interpolation of the two input trees than the middle tree on the left; and yields a more intuitive correspondence between the nodes. Note that the used layout differs between the two figures. However, the two trees in front are indeed identical in both figures, as are the trees in the back.

Abstract—Comparative visualization of scalar fields is often facilitated using similarity measures such as edit distances. In this paper, we describe a novel approach for similarity analysis of scalar fields that combines two recently introduced techniques: Wasserstein geodesics/barycenters as well as path mappings, a branch decomposition-independent edit distance. Effectively, we are able to leverage the reduced susceptibility of path mappings to small perturbations in the data when compared with the original Wasserstein distance. Our approach therefore exhibits superior performance and quality in typical tasks such as ensemble summarization, ensemble clustering, and temporal reduction of time series, while retaining practically feasible runtimes. Beyond studying theoretical properties of our approach and discussing implementation aspects, we describe a number of case studies that provide empirical insights into its utility for comparative visualization, and demonstrate the advantages of our method in both synthetic and real-world scenarios. We supply a C++ implementation that can be used to reproduce our results.

Index Terms—Topological data analysis, merge trees, scalar data, ensemble data

1 Introduction

The comparison of scalar fields through edit distances or related similarity measures is an area of increasing interest in scientific visualization. Specifically in the growing field of ensemble visualization, comparative analysis techniques are of high interest, as the data becomes more and more complex. Since the size of scalar fields acquired from real-world experiments or simulation quickly makes direct comparison on the domains infeasible, topological abstractions such as merge trees, contour trees or persistence diagrams are used to derive efficient distance measures. This has lead to a rapid development in comparison techniques on merge trees in recent years [4, 35, 43, 46, 50, 61, 62].

Out of those, two prominent examples are Wasserstein barycenters for merge trees and branch decomposition-independent edit distances. The Wasserstein barycenter framework by Pont et al. [43] introduced new analysis techniques based on barycenters and geodesics, in regards to the Wasserstein distance for merge trees. Intuitively, barycenter merge trees are an interpolated mean for a collection of merge trees, based on a given metric, while a geodesic is a shortest continuous path in this metric space of trees. The distance is an extension of the well-known Wasserstein distance for persistence diagrams and is based on the so-called degree-1 edit distance on branch decomposition trees (BDT). Applications are, e.g., k-means [10, 17] clustering or temporal reduction of time series. In contrast to that, the work of Wetzels et al. introduced two novel comparison measures for merge trees: the branch mapping distance [62] and the path mapping distance [61]. These are edit distances tailored specifically to merge trees yielding better quality comparison at the cost of increased complexity. The branch mapping distance introduced the concept of branch decomposition-independent edit distances, however, it does not fulfill the metric properties. The path mapping distance is an improved variant which forms a metric and represents an intuitive edit distance based on deformation retractions on merge trees as continuous topological spaces.

- *Florian Wetzels and Christoph Garth are with University of Kaiserslautern-Landau. E-mail: {wetzels | garth}@rptu.de*
- *Mathieu Pont and Julien Tierny are with CNRS / Sorbonne University. E-mail: {mathieu.pont | julien.tierny}@sorbonne-universite.fr*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

In this paper, we propose a combination of these techniques such that the merge tree barycenter framework can take advantage of the improved distance measures. More specifically, we implement barycenters and geodesics based on the path mapping distance, since it is a metric, making it better suited than the branch mapping distance. We thereby increase the flexibility of the framework: depending on the type, size and structure of the data, it is suited for high quality analysis using the more precise but also more complex path mapping distance as well as efficient and less precise analysis based on Wasserstein distances. We then evaluate the new method in terms of quality and performance on synthetic and real-world datasets. For the evaluation, we used three visualizations and analysis tasks on which barycenters and geodesics can be applied: ensemble summarization, ensemble clustering and temporal reduction of time series. An example for significantly improved interpolation on merge trees is illustrated in Fig. 1. We also briefly compare path mapping barycenters to the only other method we know of computing a merge tree representing a whole ensemble: the contour tree alignment [30]. Here, we illustrate the fact the barycenter merge trees are indeed valid merge trees, whereas (in general) the alignment is not. In particular, our contributions are:

- We describe an algorithm to compute both path mapping geodesics and path mapping barycenters.
- We provide an experimental evaluation for the utility of the new method. Our experiments are – to our knowledge – the first that show advantages of branch decomposition-independent comparison of real-world datasets, and advantages of path mappings over branch mappings. Previously, the former had only been shown on synthetic examples [61, 62]).
- We provide an open source implementation that is based on the *Topology ToolKit* (TTK) [56], extending its barycenter framework.

We conclude this section with an overview over related work. In Sec. 2, we give the formal background and recap path mappings and the Wasserstein barycenter framework. Sec. 3 describes the new algorithm for path mapping barycenters and geodesics. In Sec. 4, we present the results of our experiments, before Sec. 5 concludes the paper.

Related Work

Topological abstraction or descriptors for scalar fields are a well-established tool in scientific visualization with applications covering a vast area of tasks such as assisting rendering and interaction [9, 41, 52], comparing data [67] or deriving abstract visualizations [30, 59]. An overview over these methods can be found in the survey by Heine et al. [23]. We identified three areas specifically related to our work: topology-based comparison of scalar fields, feature tracking (or more general analysis of time series) and ensemble visualization.

Similarity Measures. Scalar field comparison through distances on merge trees include several edit distances by Saikia et al. [46], Sridharamurthy et al. [50, 51], Pont et al. [43] and Wetzels et al. [61, 62]. Examples for other merge tree-based distances are the works of Beketayev et al. [4], Morozov et al. [37] or Bollen et al [5].

Apart from merge trees, scalar field comparison is often done via distances on other topological descriptors such as contour trees [30, 54], persistence diagrams [11–14], reeb graphs [2, 3, 19] or extremum graphs [38,55]. Some distances also combine geometric and topological measures [21, 66, 68]. A survey on the methods can be found in [67].

Feature Tracking and Time Series. We classify topological methods for feature tracking into two major categories. One uses mappings between topological descriptors to derive mappings between the features of consecutive time steps. Examples based on merge trees are the works of Lohfink et al. [29], Saikia et al. [46] or Pont et al. [43], while other descriptors are used as well [15, 40]. The other class of methods uses topological descriptors for feature identification and then applies geometric techniques such as gradient or overlap mapping to find correspondences. Examples are the works of Lukasczyk et al. [32–34], Bremer et al. [6, 7, 64] or others [39, 47–49]. A method by Yan et al. [66] combines geometric and topological measures. Furthermore, topological methods can also be used for more advanced analysis of time series such as temporal merge tree maps [26] or geodesics [43, 57].

Ensemble and Uncertainty Visualization. Topological methods for ensemble analysis include the above mentioned similarity measures on topological descriptors. Apart from that, a typical approach is, given an ensemble of topological descriptors, to compute a representative summarizing the set of member descriptors. Examples are fuzzy contour trees [30], merge tree 1-centers [68], the uncertain contour tree layout [65], coherent contour trees [18], and Wasserstein barycenters of merge trees [43] or persistence diagrams [58]. Based on the merge tree barycenters, advanced statistical tools have also been proposed, such as principal geodesic analysis for merge trees [44]. Other methods for ensemble or uncertain data are found in [22, 27, 57, 65].

2 Preliminaries

In this section, we quickly recap the concepts and definitions of merge trees, the path mapping distance and the Wasserstein barycenters. In most cases, we simply re-state the definitions from [43, 61, 62].

2.1 Merge Trees

Given a domain $\mathbb{X}$ with a continuous scalar function $f : \mathbb{X} \to \mathbb{R}$, the *merge trees* of $\mathbb{X}, f$ represent the connectivity of sublevel or superlevel sets. Depending on the direction, we call them *join tree* or *split tree*; both form rooted trees. In this paper, we work on discrete representations of these rooted trees, called abstract merge trees and defined in the following. The nodes of an abstract merge tree are critical points of the domain $\mathbb{X}, f$ and the edges represent classes of connected components of level sets. For simplicity and w.l.o.g. we restrict to abstract split trees in the theoretic discussions (all arguments and definitions are easy to adapt for join trees). A detailed introduction is given in [16, 36].

A rooted, unordered tree T is a connected, directed graph with vertex set $V(T)$ and edge set $E(T)$, containing no undirected cycles and a unique sink, which we call the root, denoted $\text{root}(T)$. For an edge $(c, p) \in E(T)$, we call c the child of p and p the parent of c, i.e. we use parent pointers in our directed representation.

As for general graphs, a *path* of length k in a rooted tree T is a sequence of vertices $p = v_1 \dots v_k \in V(T)^k$ with $(v_i, v_{i-1}) \in E(T)$ for all $2 \leq i \leq k$ and $v_i \neq v_j$ for all $1 \leq i < j \leq k$. Note the strict root-to-leaf direction of the vertex sequence: we only consider monotone paths. For a path $p = v_1 \dots v_k$, we denote its first vertex by $\text{start}(p) := v_1$, its last vertex by $\text{end}(p) := v_k$. We write $v \in p$ if $v = v_i$ for some $1 \leq i \leq k$ and $e \in p$ if $e = (v_i, v_{i-1})$ for some $2 \leq i \leq k$. We denote the set of edges of p by $\text{edges}(p) := \{(v_i, v_{i-1})\}$. We denote the set of all paths of a tree T by $\mathscr{P}(T)$.

Merge trees inherit the scalar function from their original domain and thus are labeled trees. Usually, they are considered as node-labeled trees, with the labels being defined through the scalar function on the critical points. All merge trees have certain properties in common, which we now use for the definition of an abstract merge tree.

Definition 1. *An unordered, rooted tree T of (in general) arbitrary degree (i.e. number of children) with node labels $f : V(T) \to \mathbb{R}_{>0}$ is an* Abstract Merge Tree *if the following properties hold:*

- *the root node has degree one,* $\deg(\text{root}(T)) = 1$,
- *all inner nodes have a degree of at least two,* $\deg(v) \neq 1$ *for all* $v \in V(T)$ *with* $v \neq \text{root}(T)$,
- *all nodes have a larger scalar value than their parent node,* $f(c) > f(p)$ *for all* $(c, p) \in E(T)$.

For the path mapping distance and the corresponding geodesics, we work on edge-labeled trees. Here, edge labels represent the length of the scalar range of the edge. These two representations are interchangeable; given a node label function $f : V(T) \to \mathbb{R}_{>0}$, we define the corresponding edge label function $\ell_f : E(T) \to \mathbb{R}_{>0}$ by $\ell_f((u, v)) = |f(u) - f(v)|$. Given an edge label function, we can again define f_ℓ by placing the root node at a fixed scalar value, e.g. 0.

We lift the edge label function ℓ of a merge tree T from edges to paths in the following way: $\ell(v_1 \dots v_k) = \sum_{2 \leq i \leq k} \ell((v_i, v_{i-1}))$. For the implicit edge labels ℓ_f, we get $\ell_f(v_1 \dots v_k) = |f(v_1) - f(v_k)|$. To define edit distances on merge trees, we also need to define a cost function

to compare edges. Since we use $\mathbb{R}_{>0}$ as the label set for abstract merge trees, we use 0 as the blank symbol, i.e. the label of an empty or non-existing edge. Then, we define the cost function as the euclidean distance on $\mathbb{R}_{\geq 0}$: $c(l_1,l_2) = |l_1 - l_2|$ for all $l_1, l_2 \in \mathbb{R}_{\geq 0}$.

A *branch* of an abstract merge tree T is a path that ends in a leaf. A branch $b = b_1...b_k$ is a parent branch of another branch $a = a_1...a_\ell$ if $a_1 = b_i$ for some $1 < i < k$. We also say a is a child branch of b. A set of branches $B = \{B_1,...,B_k\}$ of a merge tree T is called a *Branch Decomposition* of T if $\{\text{edges}(B_1),...,\text{edges}(B_k)\}$ is a partition of $E(T)$. We also call the length of a branch its *persistence*. The parent-child relations of the branches in a branch decomposition B of T form a tree structure by themselves. The tree built from the vertex set $V = B$ and edge set E, with $(a,b) \in E$ if and only if b is a parent branch of a, is called the *Branch Decomposition Tree* (BDT) of B. In practice, the nodes of the BDT are labeled with the the birth and death values of its branch (i.e. the scalar values of the first and last vertex [14]). Most of the time, the branch decomposition derived by the elder rule (giving longer/more persistent branches higher priority, see [14] for details) is used. We denote it by $\mathscr{B}(\mathbb{X}, f)$ for a scalar field $\mathbb{X}, f$. For simplicity, we often write $b \in \mathscr{B}(\mathbb{X}, f)$ instead of $b \in V(\mathscr{B}(\mathbb{X}, f))$.

2.2 Path Mapping Distance

We now recap the concepts of *path mappings* and the *path mapping distance* between merge trees. Both were introduced in [61] to capture a constrained variant of the deformation based edit distance defined in the same paper. In contrast to classic edit mappings that map the nodes of two trees onto each other, path mappings (as the name suggests) map paths of one tree to paths of another tree. Similar to classic edit mappings, they do so in a structure-preserving way. We now re-state the definition of path mappings and the corresponding distance measure.

Definition 2. *Given two abstract merge trees T_1, T_2, a path mapping between T_1 and T_2 is a mapping $M \subseteq \mathscr{P}(T_1) \times \mathscr{P}(T_2)$ such that*

1. *(one-to-one) for all $p_1, q_1 \in \mathscr{P}(T_1)$, $p_2, q_2 \in \mathscr{P}(T_2)$ with $(p_1,p_2) \in M$ and $(q_1,q_2) \in M$, $p_1 = q_1$ if and only if $p_2 = q_2$,*
2. *(paths do not overlap) $|p_1 \cap q_1| \leq 1$ and $|p_2 \cap q_2| \leq 1$ for all $(p_1,p_2),(q_1,q_2) \in M$,*
3. *(paths form a connected subtree) for all $(p,q) \in M$,*
   - *(paths only meet at start/end) either there are paths $p' \in \mathscr{P}(T_1)$ and $q' \in \mathscr{P}(T_2)$ such that $(p',q') \in M$ and $\text{start}(p) = \text{end}(p')$ and $\text{start}(q) = \text{end}(q')$,*
   - *or $\text{start}(p) = \text{root}(T_1)$ and $\text{start}(q) = \text{root}(T_2)$.*

For an edge $e \in E(T_1)$ (or a vertex $v \in V(T_1)$), we write $e \notin M$ ($v \notin M$), if there is no pair $(p_1,p_2) \in M$ with $e \in p_1$ ($v \in p_1$). We use the same notation for edges or vertices of T_2.

For a path mapping M between two abstract merge trees T_1, ℓ_1 and T_2, ℓ_2, we also define its corresponding edit operations $\text{edits}(M)$*. They consist of the corresponding relabel, insert and delete operations:*

$$\begin{aligned}\text{rel}(M) &= \{(\ell_1(p_1), \ell_2(p_2)) \mid (p_1,p_2) \in M\},\\ \text{ins}(M) &= \{(0, \ell_2(e_2)) \mid e_2 \in E(T_2),\ e_2 \notin M\},\\ \text{del}(M) &= \{(\ell_1(e_1), 0) \mid e_1 \in E(T_1),\ e_1 \notin M\}.\end{aligned}$$

Then, we have $\text{edits}(M) = \text{rel}(M) \cup \text{ins}(M) \cup \text{del}(M)$*. Moreover, we define the costs of a mapping via corresponding edit operations and the path mapping distance as the costs of an optimal mapping:*

$$c(M) = \sum_{(l_1,l_2) \in \text{edits}(M)} c(l_1,l_2), \quad \delta_P(T_1,T_2) = \min_{M \text{ between } T_1,T_2} \{c(M)\}. \tag{1}$$

2.3 Wasserstein Distance and Barycenters

Next, we formalize the Wasserstein distance between merge trees as well as Wasserstein barycenters and the algorithm computing them. Pont et al. [43] introduced a metric between BDTs, formulated as a generalization of the celebrated Wasserstein distance between persistence diagrams [14], which we briefly recap here. Let us first consider the simple case where in both BDTs to compare (denoted $\mathscr{B}_1 := \mathscr{B}(\mathbb{X}_1, f_1)$ and $\mathscr{B}_2 := \mathscr{B}(\mathbb{X}_2, f_2)$), all nodes are direct children of the root. This situation corresponds to the classical formulation of the Wasserstein distance between persistence diagrams. Each branch in $\mathscr{B}_1$ and $\mathscr{B}_2$ can be represented as a 2D point by using its birth value as X coordinate and its death value as Y coordinate. This yields, for each persistence diagram, a 2D point cloud in the so-called birth/death plane. In this representation, all the branches appear above the diagonal (since a feature dies only after it was created).

To measure the distance between such BDTs $\mathscr{B}_1$ and $\mathscr{B}_2$, a first pre-processing step consists in augmenting each BDT with projections $\Delta(b)$ for all off-diagonal points b in the other BDT:

$$\mathscr{B}'_1 = \mathscr{B}_1 \cup \{\Delta(b_2) \mid b_2 \in \mathscr{B}_2\},\ \mathscr{B}'_2 = \mathscr{B}_2 \cup \{\Delta(b_1) \mid b_1 \in \mathscr{B}_1\},$$

where $\Delta(b) = (\frac{x+y}{2}, \frac{x+y}{2})$ stands for the diagonal projection (i.e. the closest point on the the diagonal) of the off-diagonal point $b = (x,y)$. Intuitively, this augmentation phase inserts dummy features in the BDT (with zero persistence, along the diagonal), hence preserving the topological information. This augmentation guarantees that the two BDTs now have the same number of points ($|\mathscr{B}'_1| = |\mathscr{B}'_2|$), which facilitates the evaluation of their distance, as described next.

Given two points $b_1 = (x_1,y_1) \in \mathscr{B}'_1$ and $b_2 = (x_2,y_2) \in \mathscr{B}'_2$, the ground distance d_2 in the 2D birth/death space is given by:

$$d_2(b_1,b_2) = (|x_2 - x_1|^2 + |y_2 - y_1|^2)^{1/2} = \|b_1 - b_2\|_2.$$

By convention, $d_2(b_1,b_2)$ is set to zero between diagonal points ($x_1 = y_1$ and $x_2 = y_2$). Then, the L^2-Wasserstein distance $W_2^{\mathscr{T}}$ is:

$$W_2^{\mathscr{T}}(\mathscr{B}_1,\mathscr{B}_2) = \min_{\phi \in \Phi} \Big(\sum_{b_1 \in \mathscr{B}'_1} d_2(b_1, \phi(b_1))^2 \Big)^{1/2}, \tag{2}$$

where Φ is the set of all possible assignments ϕ mapping a point $b_1 \in \mathscr{B}'_1$ to a point $b_2 \in \mathscr{B}'_2$ (possibly its diagonal projection, indicating the destruction of the corresponding feature).

To account for the general case where the BDTs have an arbitrary structure, one simply needs to update the above formulation by considering a smaller search space of possible assignments, noted $\Phi' \subseteq \Phi$, constrained to describe (rooted) partial isomorphisms [43] between $\mathscr{B}_1$ and $\mathscr{B}_2$. Intuitively, $W_2^{\mathscr{T}}$ can be understood as a variant of the Wasserstein distance between persistence diagrams, which takes into account the structures of the BDTs when evaluating candidate assignments in the optimization of Eq. 2. In the remainder, we note $\mathbb{B}$ the metric space induced by the Wasserstein metric between BDTs.

Once distances for BDTs are available, the notion of *barycenter* can be introduced. Given a set $\mathscr{S}_{\mathscr{B}} = \{\mathscr{B}_1,\dots,\mathscr{B}_N\}$, let $E_F(\mathscr{B}) = \sum_{i=1}^N W_2^{\mathscr{T}}(\mathscr{B},\mathscr{B}_i)^2$ be the Fréchet energy of a candidate BDT $\mathscr{B} \in \mathbb{B}$. Then a BDT $\mathscr{B}^* \in \mathbb{B}$ which minimizes E_F is called a *Wasserstein barycenter* of the set $\mathscr{S}_{\mathscr{B}}$ (or its Fréchet mean under the metric $W_2^{\mathscr{T}}$).

E_F can be optimized with an iterative algorithm [43], alternating *assignment* and *update* phases. The algorithm is reminiscent of Lloyd's relaxation algorithm [28]. Specifically, given a candidate barycenter $\mathscr{B}$, the assignment step consists in computing the optimal assignments ϕ_i (w.r.t. Eq. 2) between $\mathscr{B}$ and each input BDT $\mathscr{B}_i$. Next, the update step aims at optimizing E_F under the current set of assignments ϕ_i, which is achieved by moving, in the birth/death plane, each branch b of $\mathscr{B}$ to the arithmetic mean of its matched branches $\phi_i(b)$. This assignment/update sequence is then iterated, decreasing the Fréchet energy at each iteration. Note that this procedure is not guaranteed to converge to a global minimum, i.e. the Fréchet energy is not necessarily globally minimized. Iteration is stopped when the Fréchet energy change is less than 1% between two steps. The initial candidate is chosen as a random member of $\mathscr{S}_{\mathscr{B}}$, see [43] for details. This choice might influence which local minimum is found.

To ensure that the interpolated barycenter trees are indeed structurally valid BDTs (i.e. invertible into a valid merge trees), Pont et al. introduce a local normalization [43] in a pre-processing step. Specifically, a branch $b = (x,y)$ with parent branch $b' = (x',y')$ is valid/invertible if

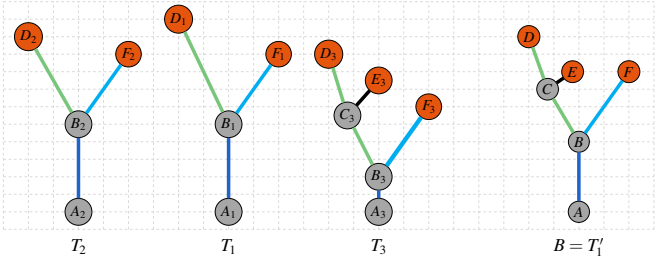

Fig. 2: Example of barycenter assignment and update for member trees T_1, T_2, T_3 and initial candidate T_1. Optimal path mappings are illustrated through the edge colors. Edge lengths can be read from the grid.

$[x,y] \subseteq [x',y']$. Thus, the persistence of each branch $b_i \in \mathscr{B}_i$ is normalized with regard to that of its parent $b_i' \in \mathscr{B}_i$, displacing b_i in the 2D birth/death space such that its position is given relatively to the scalar range of b_i'. After this pre-normalization of the input trees, any interpolated barycenter BDT is reverted into a valid merge tree by recursively reverting the normalization, thereby guaranteeing the validity of the interpolated BDT and merge tree.

3 Deformation Based Geodesics and Barycenters

In this section, we describe the construction of geodesics and barycenters based on the path mapping distance. We first give an intuition for the barycenter construction before describing the algorithms in detail. While the barycenter and geodesic construction is more technical than for the Wasserstein barycenters, the intuition behind it is similar and should be easy to grasp. Furthermore, our algorithm does not depend on a prior normalization to obtain valid merge trees after interpolation. Note that the geodesic is just a special case of the barycenter (as it is in the Wasserstein barycenter framework). Thus, we first focus on the more generic case: we give an intuitive and algorithmic description of the barycenter construction. In Sec. A of the suppl. material, we then formalize the interpolation which results for the case of two input trees (i.e. the resulting path of merge trees) and show that it indeed forms a geodesic in the metric space based on the path mapping distance.

Intuition. The core idea of our approach is identical to the Wasserstein barycenters: after initializing with a random member, we alternate an *assignment* and *update* step and continue the iteration until a stable barycenter is reached. Given a barycenter candidate and the ensemble of merge trees, the assignment step computes the optimal path mapping between the candidate and each member tree. The update step interpolates the lengths of mapped paths, where unmapped paths are interpolated with imaginary edges of length 0. Thus, our approach is an adaptation of the Wasserstein barycenter method where we replace interpolation of mapped branches by interpolation of mapped paths.

An example can be seen in Fig. 2. We interpolate the paths A_1B_1 (length 5), A_2B_2 (length 5) and A_3B_3 (length 2) to AB (length 4). We interpolate the paths B_1D_1 (length 5), B_2D_2 (length 6) and $B_3C_3D_3$ (length 7) to BCD (length 6). The node C in the barycenter is kept in the same relative position as C_3 in T_3. The intuition behind this, in terms of the corresponding deformation, is that shortening $B_3C_3D_3$ to BCD can also be seen as uniformly shortening both B_3C_3 to BC and C_3D_3 to CD, which keeps the relative position constant. The path BF is handled exactly as AB. As a last step, we shorten C_3E_3 to $\frac{1}{3}$ of its length, which intuitively means interpolating with two imaginary edges of length 0 (as it does not appear in the other two trees).

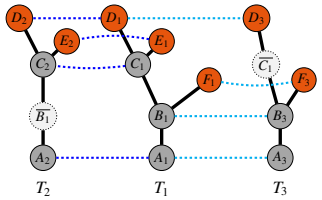

Fig. 3: Example of the barycenter update: Three merge trees T_1, T_2 and T_3 and two path mappings (dashed lines) are shown. T_2 and T_3 contain an imaginary node, highlighted through the dotted strokes.

However, this manner of interpolation is not necessarily well-defined for all combinations of path mappings. The partition of the edges of the barycenter candidate may vary between the different mappings, which might make a meaningful interpolation impossible. An example is shown in Fig. 3. Here, the setup is similar to Fig. 2. The mappings (here illustrated through the induced vertex maps) between T_1 and T_2 (M_2) and between T_1 and T_3 (M_3) are considered. M_2 contains the path $A_1B_1C_1$, while M_3 only contains B_1C_1. Here, it is unclear which of the paths A_1B_1 or B_1C_1 should be interpolated.

We fix this problem by splitting the mapped paths through insertion of imaginary nodes into the member trees until a proper interpolation is possible, i.e. until all mappings are pure edge mappings. These splits can be designed such that the resulting fine-grained edge-mappings are still structure-preserving and thus can be interpolated into a meaningful merge tree. Fig. 3 shows these imaginary nodes in the example trees, too. This yields the following adapted mappings: M_2 now maps A_1B_1 to $A_2\overline{B_1}$ and B_1C_1 to $\overline{B_1}C_2$ instead of $A_1B_1C_1$, while M_2 now maps B_1C_1 to $B_3\overline{C_1}$ and C_1D_1 to $\overline{C_1}D_3$ instead of $B_1C_1D_1$.

Note that the example only considers the case where the two contradicting paths follow the same "direction". They could also branch away from each other, e.g. one mapping could contain $A_1B_1C_1$ whereas the other could contain $A_1B_1F_1$. Indeed, this case is also solved by splitting the paths until a pure edge mapping is reached.

In the following, we describe the barycenter algorithm in detail. In contrast to the Wasserstein barycenters, our algorithm does not necessarily guarantee monotonic decrease of the Fréchet energy between its iterations. However, in Sec. 4.7, we show empirical convergence to a local energy minimum on example datasets.

Barycenters Construction. Given a barycenter candidate B and path mappings $M_i \subseteq \mathscr{P}(B) \times \mathscr{P}(T_i)$ for each $1 \leq i \leq N$, we build the next candidate B^* as follows. First, any nodes or edges in B not matched in any M_i are removed. Next, we relabel the nodes in B. Lastly, we insert all unmatched nodes and edges from the M_i with adapted scalar values. We describe these three procedures in more detail in Alg. 1.

The subroutine `RemoveUnmatched` in Alg. 1 removes unmatched nodes from the barycenter candidate in a straightforward manner by iterating over all mappings to check which edges are part of mapped paths and removing those that do not appear.

The subroutine `RelabelBarycenter` in Alg. 1 is more involved. Recall that we want to split the mapped paths in the mappings through imaginary nodes to obtain pure edge mappings. The algorithm does this implicitly. It again iterates over all mappings and all pairs of paths in these mappings. Let (p,p') be such a pair where p is a path in the barycenter candidate and p' is a path in a member tree. For each barycenter edge e in p, it computes the length of the corresponding segment in p', i.e. the same fraction of length. In particular, the segment that corresponds to e in p' has length $\frac{\ell(e)}{\ell(p)}\ell(p')$. Then, the algorithm computes the average of all these segments.

As a last step, subroutine `AddUnmatched` in Alg. 1 adds unmatched subtrees of the member trees to the barycenter. Note that these subtrees are always complete subtrees rooted in some edge, since the path mapping distance only allows for insertions and deletions of leaf edges. The algorithm also iterates over all path mapping and all pairs of paths in them. For a mapped pair (p,p'), p being the path in the barycenter, we add each node $v \in p'$ to p and define its scalar value such that its relative position to start and end point is the same in p and p'. Each child of v that is not on p' is the root of a deleted subtree. Thus, we insert these trees and define the length (in the barycenter) of each inserted edge to be $\frac{1}{k}$ of its original length (in the member tree).

As for the Wasserstein barycenters in [43], the iteration resembles Lloyd's method and is a generalization of the geodesic. In contrast to the iteration for Wasserstein barycenters, this update procedure does not necessarily decrease the Fréchet energy under the metric δ_P (defined as $\sum_{i=1}^{N} \delta_P(B,T_i)$) in each step. Thus, we opted to show the convergence

Algorithm 1: Subroutines of the update phase

Function `RemoveUnmatched` $(B, M_1, \ldots, M_k, T_1, \ldots, T_k)$**:**
- $B' = B$
- Initialize EdgesMapped with 0 for each edge in $E(B')$
- **foreach** $i = 1 \ldots k$ **do**
  - **foreach** $(p, p') \in M_i$ **do**
    - **foreach** Edge $e \in p$ **do**
      - EdgesMapped[e] = 1
- **foreach** Edge $e = (c, p) \in E(B')$ in post-order **do**
  - **if** EdgesMapped[e] = 0 **then**
    - Remove e from $E(B')$ and c from $V(B')$
- **foreach** Node $v \in V(B')$ **do**
  - **if** $\deg(v) = 1$ **then**
    - Replace $(c, v) \in E(B')$ with (c, p) for p the parent of v
- **return** B'

Function `RelabelBarycenter` $(B, M_1, \ldots, M_k, T_1, \ldots, T_k)$**:**
- $B' = B$
- Initialize EdgeLengths as 0 for each edge in $E(B')$
- **foreach** $i = 1 \ldots k$ **do**
  - **foreach** $(p, p') \in M_i$ **do**
    - **foreach** Edge $e \in p$ **do**
      - EdgeLengths[e] += $\frac{\ell(e)}{\ell(p)} \cdot \ell(p')$
- **foreach** Edge $e \in E(B')$ **do**
  - Set $\ell(e)$ to EdgeLengths[e] $\cdot \frac{1}{k}$
- **return** B'

Function `AddUnmatched` $(B, M_1, \ldots, M_k, T_1, \ldots, T_k)$**:**
- $B' = B$
- **foreach** $i = 1 \ldots k$ **do**
  - **foreach** $(p, p') \in M_i$ **do**
    - **foreach** Node $v \in p'$ **do**
      - Add v to p in B' with $f_\ell(v)$ such that $\frac{f_\ell(v)}{\ell(p)} = \frac{f_\ell(v)}{\ell(p')}$
      - **foreach** $e = (c, v) \in E(T_i)$ **do**
        - **if** $c \notin p$ **then**
          - Add the tree T_e rooted in e to B' with labels $\frac{1}{k}\ell(e')$ for each edge $e' \in E(T_e)$
- **return** B'

(to a local energy minimum) experimentally, see Sec. 4.7.

To see why the update procedure does not necessarily decrease the Fréchet energy, consider Eqs. 1 and 2. Note that to obtain the path mapping distance, we simply take the sum of the single edit costs, whereas for the Wasserstein distance they are first squared and then we take the square root after summing up. Since the mean value minimizes squared distances, not the sum of distances, the update procedure given in Alg. 1 is locally (i.e., for a set of interpolated edges) not optimal.

The sum of distances is minimized by the median, not the mean. Thus, we can take the median instead in `RelabelBarycenter` and `AddUnmatched` to circumvent this problem. The resulting barycenter is, however, of inferior quality than the one based on the mean. Using the median implies skipping `AddUnmatched` completely (the median will always be one of the $k-1$ imaginary edges of length 0), which means no edges are ever added to the barycenter. If the initial candidate does not contain all features present in the ensemble, we get a poor quality barycenter, which we showcase on an example in Sec. 4.6.

4 Experiments

In this section, we study the utility of our methods on three visualization and analysis tasks: ensemble summarization, ensemble clustering, and time series temporal reduction. For each, we compare results obtained using the path mapping distance to results achieved with the Wasserstein distance. We briefly show the advantages of both barycenter methods over contour tree alignments in the appendix.

We describe implementation and experiment setup for each task in detail, and study five datasets as well as convergence and performance.

4.1 Implementation and Experiment Setup

Since the geodesic and barycenter constructions described in the previous section are just an adaptation of the Wasserstein geodesic and barycenter algorithms, we integrated our implementation into the original one in TTK [56]. In both cases, the barycenter computation is a generalization of the geodesic computation. Thus, it suffices to adapt the barycenter method. We extended the existing barycenter filter in TTK by adding our algorithm as an alternative option.

This enables us to use them as a drop-in in more advanced methods based on geodesics and barycenters as well. TTK contains filters to compute a k-means [10, 17] clustering based on merge tree distances and barycenters as well as filters for temporal reduction of merge tree sequences and the corresponding reconstruction. Details on these methods can be found in [43]. In all of them, it is now possible to use the path mapping distance instead of the Wasserstein distance, which enables users to reuse old workflows when applying the new method. For our experiments, we used the TTK implementation and ParaView [1] on the following tasks (executed on an Intel Core i7-7700 with 64GB of RAM).

Ensemble Summarization. Given an ensemble of k scalar fields $f_1, f_2, \ldots, f_k$, we first compute the k merge trees $T_1, \ldots, T_k$. Then, we compute the barycenter merge tree B of $T_1, \ldots, T_k$. This barycenter tree should visually summarize the ensemble to give the user an overview of the existing features and the overall topological structure of the scalar fields in it. The barycenter tree should contain all important features of the members and should not contain prominent structures that cannot be mapped back to features in at least some of the members. We will rate the results on a purely visual basis.

Ensemble Clustering. Using the iterative barycenter computation, it is possible to apply existing clustering strategies such as the k-means algorithm. We added our implementation of the path mapping distance to the existing merge tree clustering framework in TTK [43].

Then, given an ensemble of k scalar fields $f_1, f_2, \ldots, f_k$, we first compute the k merge trees $T_1, \ldots, T_k$. Next, we apply the k-means algorithm to $T_1, \ldots, T_k$. We compare the resulting cluster assignment to a previously known ground truth. We consider the result as correct, if no two members assigned to the same cluster belong to different ground truth clusters. An addition to counting the number of fully correct results, we compute the adjusted rand index [25] (ARI). Since the k-means implementation in TTK can utilize randomized initialization, we did multiple runs of both the path mapping based and the Wasserstein distance based clustering. We compare the percentage of correct runs as well as the average ARI.

Temporal reduction. Given an ensemble of k scalar fields $f_1, f_2, \ldots, f_k$, we first compute the k merge trees $T_1, \ldots, T_k$. This time series of merge trees is then reduced to k' key frames. The target number k' is given as an input by the user. We greedily remove merge trees from the time series, until only k' merge trees are left, as described in [43]. To reconstruct the original time series, we compute geodesic merge trees of the adjacent key frames.

To rate the quality of the reduction and reconstruction, we compute the distance between the original merge trees and the corresponding reconstructed ones. We compute the key frames and the reconstruction with both the path mapping distance and the Wasserstein distance and compare quality of the two reconstructions. The comparison can be done visually (i.e. do the reconstructed merge trees look like good interpolations) or based on the distance between the original and reconstructed trees. The latter is done using both distance measures.

In the remainder of this section, we study the behavior of the path mapping based approaches on these three tasks. We go through five different datasets and discuss the results in the corresponding subsection. As a last step, we study convergence and running times in practice.

4.2 Analytical Example

The analytical example is an ensemble consisting of 20 members. Each member is a 2-dimensional regular grid with a scalar function defined on the vertices. It was first used in [62] to showcase the advantages of branch decomposition-independent edit distances over those using

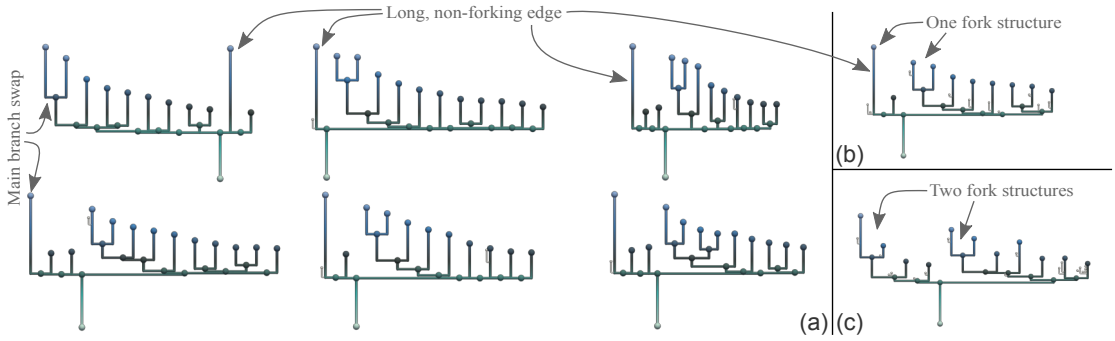

Fig. 4: The six member split trees of the starting vortex ensemble with the two barycenters on the right. Branches of low persistence are uncolored and drawn thinner. All member trees (a) and the path mapping barycenter (b) contain one edge of high persistence without a fork structure as well as a fork structure of slightly lower persistence. The long edge also forms the main branch of the branch decomposition in all but one of the member trees. In contrast to that, the Wasserstein barycenter (c) creates a fork structure within this main branch, thus having two high-persistence forks. The reason is that the fork structure is the main branch in one member which leads to bad mappings.

fixed BDTs. It is a synthetic dataset designed specifically to provoke instabilities for BDT-based methods, which we explain in the following.

Each member field consists of four main peaks, one of which has five smaller peaks arising from it (see Fig. 5, together with the corresponding merge trees). The positions of the maxima as well as their heights are all chosen randomly within small ranges, leading to differences in the nesting of the persistence-based branch hierarchy. More specifically, the order of the four main maxima is "shuffled" in the BDT. In contrast, the five side maxima (although also slightly perturbated) stay on the same hill. As a consequence, the corresponding noisy sub-tree (located on the front-most hill in Fig. 5) travels in the BDTs, depending on the maximum value reached by its main hill. Therefore, structure-preserving mapping on BDTs can either map all four main peaks correctly or the five side peaks, but not both.

In Fig. 5, we illustrate the difference in the BDTs through the position of the corresponding arcs in the planar layout. They are placed according to the order in the branch decomposition from left to right. In the remainder of this paper, we will refer to this behavior as a maximum swap. Note that each of the 20 member trees is of the form of one of the four merge trees in Fig. 5 and we omit to show them all.

We used this dataset for the summarization task, applying both the path mapping and the Wasserstein distance. The resulting path mapping barycenter strongly resembles the member trees: it has four main maxima and there is exactly one of them which has side branches leading to smaller maxima. The barycenter based on the Wasserstein distance duplicates the side branches: each main maximum has attached some side peaks. The reason is that the Wasserstein distance utilizes structure-preserving mappings on BDTs, which fail as explained above. In contrast, the path mapping distance is branch decomposition-independent, working purely on the merge trees (which are highly similar). Thus, the Wasserstein barycenter is a less precise summarization of the orginal member trees. Fig. 5 also shows the two barycenters.

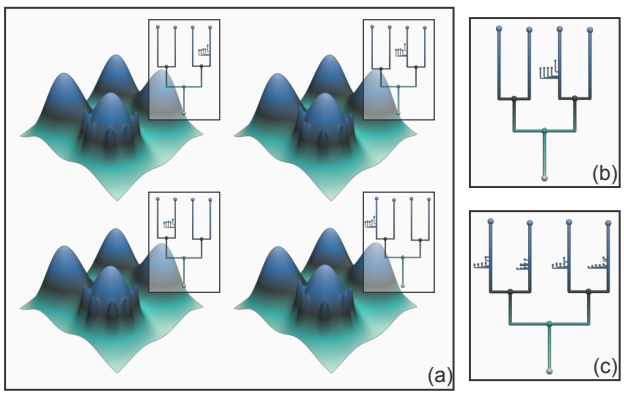

Fig. 5: Four member fields from the analytical example with their split trees, shown in (a). On the right, the two barycenters are shown. The barycenter in (b) was computed using the path mapping distance, the one in (c) using the Wasserstein distance.

4.3 Starting Vortex

The starting vortex dataset is an ensemble of 2D regular grids with a scalar function on the vertices representing flow turbulence behind a wing. It was generated using the Gerris flow solver [45] and has been used in [20, 43, 58]. Each member represents a different inclination angle of the wing. In [43] it was successfully used for the clustering task, where the two clusters were defined by two different ranges of angles, consisting of six consecutive integer values. Here, we use one of these clusters and compute the barycenter split tree. We applied a topological simplification with a threshold of 5% of the scalar range. Fig. 4 shows the member trees and both the path mapping barycenter and the Wasserstein barycenter. The path mapping barycenter is clearly a better representative of the ensemble than the Wasserstein barycenter. We should note that the behavior depends on the initial barycenter candidate. For the path mapping distance, all six resulting barycenters are of good quality, whereas for the Wasserstein distance, five of six initial candidates produce low-quality results, again due to a maximum swap (cf. Fig. 4), as explained for the first dataset. We provide additional renderings for all possible barycenter outcomes in Sec. B of the suppl. material.

4.4 TOSCA Shape Matching Ensemble

The TOSCA non-rigid world dataset [8] is a shape matching ensemble containing several shapes (different humans and animals) in various poses. The meshes have an average geodesic distance field [24] attached. We use the same preprocessed scalar fields as Sridharamurthy et al. in [50] and applied topological simplification with a threshold of 2% of the scalar range in a pre-processing step. We picked the first five poses for four different shapes (David, Michael, seahorse and centaur) and defined the following clusters as ground truth: the two human shapes David and Michael form one cluster, the seahorse another one, as well as the centaur.

We applied the k-means clustering to this set of scalar fields and compared the assigned clusters to the ground truth. We used a target number of 3 clusters for the basic experiment and performed runs for a target number of 4 as well. In the latter case, we counted runs with split clusters as correct, but not if two clusters were mixed.

The path mapping distance clearly outperforms the Wasserstein distance. Fig. 6 shows two typical outputs in ParaView. Since k-means

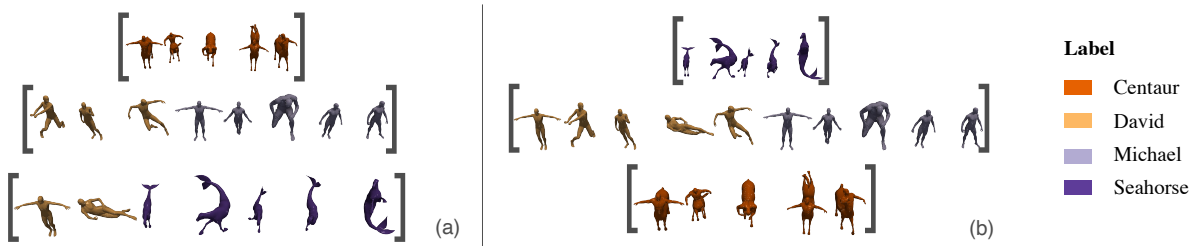

Fig. 6: Two screenshots of the clustering output for the TOSCA ensemble in ParaView. The meshes are colored by their shape name (note that the human cluster consists of two different shapes) and arranged according to the assigned cluster. The screenshot in (b) shows a correct result, which assigns all human shapes into one cluster, as well as all centaurs and all seahorses. It was computed using the path mapping barycenters. The screenhot in (a) shows an incorrect clustering computed by the Wasserstein barycenters.

implementation is randomized, we performed 100 runs for each algorithm and checked whether the assigned clusters are completely correct. Here, the path mapping barycenters produced 57 correct clusterings and an ARI of 0.75 when using a target number of 3, whereas the clustering based Wasserstein barycenters was never correct and achieved an ARI of 0.34. Interestingly, we get an even better accuracy (in terms of completely correct runs) for a target number of 4. Here, the number of correct runs for the path mapping distance was 78 and 1 for the Wasserstein distance. The ARIs were 0.7 and 0.43. Overall, we observed accuracies of the path mapping barycenters to be significantly better in comparison to the Wasserstein barycenters.

Furthermore, we took the ten human poses forming the largest cluster and computed the barycenter tree as a representative. Surprisingly, while path mappings show significantly better results for the clustering, they fail to produce a good representative for the human cluster. Fig. 7 illustrates this behavior. The reason is that there are saddle swap instabilities within the human shapes. While both distances are unable to handle those *in general* (although for both we can preprocess the data by merging close saddles), the Wasserstein distance is able to handle *some* instabilities by working on unordered BDTs. It thereby allows to match swapped saddles, as long as they are children of the same parent branch. This is not possible for path mappings. In [62], this was discussed in more detail for branch mappings (but the arguments translate to path mappings as well). There, it says that the search spaces of the branch mapping distance and Wasserstein distance are orthogonal to each other. The subset of the TOSCA ensemble considered in this section is a good example for exactly this orthogonality.

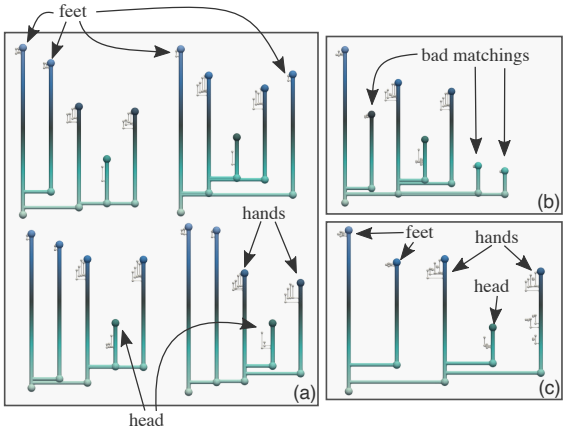

Fig. 7: Dataset summarization on the TOSCA dataset: four example members of the human cluster are shown in (a), the path mapping barycenter of this cluster can be seen in (b), the Wasserstein barycenter in (c). The Wasserstein barycenter is a good representative of the ensemble with clearly identifiable head, hands and feet. The path mapping fails to do so and contains branches that are hard to interpret, stemming from poor quality mappings.

4.5 Ionization Front

The ionization front dataset [53] consists of 2D slices from a time dependent scalar field representing ion concentration. The scalar fields were preprocessed with a topological simplification using a threshold of 5% of the scalar range. We used this dataset as a time series as well as an ensemble for clustering by picking three clusters (different phases of the simulation) of consecutive time points.

The examples presented here mainly focus around a specific main maximum swap between the time steps 126 and 127. This main maximum swap is shown in Fig. 1. There, it is easy to see that the Wasserstein distance fails to produce a good mapping, which also leads to a poor quality geodesic tree. While the layouts of the three trees on the right are aligned to show the correct mapping, the layouts on the left are based on the branch decomposition. The branch decomposition based layout shows the main maximum swap (in the first tree, the short fork is the most persistent branch feature, in the last one it is the long fork), which is the reason for the bad mapping.

In the following, we use this observation to showcase the influence on two applications. First, we show that the bad mapping leads to bad clustering with the Wasserstein distance if the ground truth clusters contain the main maximum swap. This is not the case for the path mapping distance. Next, we show that the same holds for the temporal reduction: using the Wasserstein distance, more time steps are needed to get the same reconstruction quality as with the path mapping distance.

For the clustering, we picked three sets of four consecutive time steps each. We then applied the k-means algorithm using both the path mapping distance and the Wasserstein distance. Again, the path mapping distance shows significantly better results. To better understand the reason for this, we also plotted the distance matrices for the two distances in Fig. 9. It is easy to see that the Wasserstein distance splits two of the three ground truth clusters into two smaller clusters each, leading to five clusters in total. This happens due to a maximum swap within these clusters. We observed 87% of the runs with the path mapping distance to be correct, and 0% with the Wasserstein distance. The ARIs were 0.92 and 0.46.

For the temporal reduction task, Fig. 11 illustrates the results of both methods on a subsequence around the described maximum swap. The time series used here consists of 12 time steps around the above explained saddle swap: 6 time steps before and after the swap, respectively. We used 3 as the target number of key frames. As expected, in the first half of the series (before the swap), both methods yield very precise reconstructions, since in both cases the middle key frame is chosen *before* the swap. In contrast, in the second half, only the path mapping geodesic yields precise reconstructions, since the bad mapping of the Wasserstein distance (as discussed above) between the second and last keyframe also leads to bad interpolation. In fact, the Wasserstein geodesic needs 4 key frames for a precise reconstruction (the first time step, the last, one right before, and one right after the swap), whereas the path mapping geodesic already gives good results for only 2 key frames (see Sec. D of the suppl. material).

Apart from a visual or qualitative comparison, we also provide a quantitative comparison, i.e. we compared the actual distances to the original member trees. To avoid a bias through the used distances, we

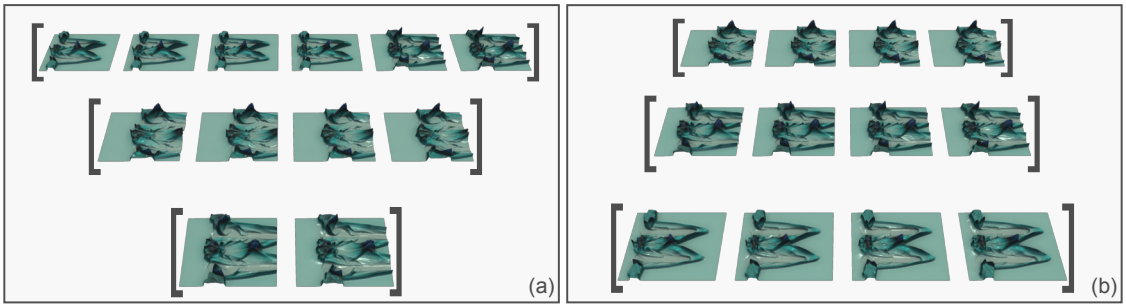

Fig. 8: Two clustering results for the ionization front ensemble. The result for the path mapping distance is shown on the right. The assigned clusters correspond to the three ground truth phases of the simulation, which are visually clearly distinguishable. The earliest one is shown in the bottom cluster, the middle one in the center cluster, the last one in the top clusters. On the left, the clustering result for the Wasserstein distance can be seen. The mid phase is split up and mixed with the earliest phase (top). The last phase is identified correctly (center line).

provided both the path mapping and the Wasserstein distance for both reconstructions each. Table 1 of Sec. D of the suppl. material shows the results. In the first half of the series, the distances are generally small. In the second half, the path mapping geodesics yield much smaller distances, independent of the chosen distance measure. This intuitively fits with the observations in Fig. 11.

4.6 Heated Cylinder

The heated cylinder ensemble consists of 23 time-dependent scalar fields describing flow around a heated pole. Each member was created using small-scale perturbations of the initial conditions. It was used in [30] for the computation of fuzzy contour trees. We again apply

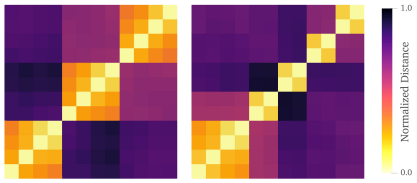

Fig. 9: The distance matrices on the clustering ensemble for the path mapping distance (left) and the Wasserstein distance (right). The rows and columns are ordered by time step. The path mapping distance clearly shows the three ground truth clusters, whereas the Wasserstein distance has five clusters in total. This leads to the poor results when applying a k-means clustering.

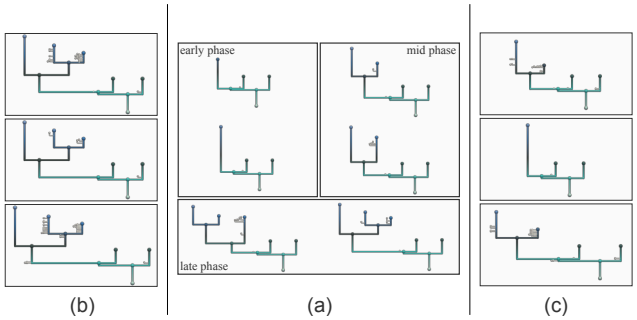

Fig. 10: Six member trees from the heated cylinder ensemble (a) with several barycenters. The six example members come from three different phases of one single run. The barycenters on the right (c) were computed on a consecutive subsequence of a single run, whereas the barycenters on the left (b) were computed on one fixed time step from the late phase for each run. The top row barycenters in (b) and (c) were computed using the path mapping distance and the typical mean method. The center row barycenters are based on the path mapping distance in the median variant. The bottom row barycenters are the Wasserstein barycenters.

topological simplification (5% of the scalar range). We compute the barycenter tree for both a fixed time point with varying runs (23 trees) and a fixed run with varying time points (30 trees). The results are shown in Fig. 10. The latter case shows that the median-based barycenter fails if the initial candidate contains few features, since missing features will never be added. In contrast to that, on a fixed time step, all member trees are very similar and all three methods perform well.

We also compare the barycenter summarization of the ensemble to the fuzzy contour tree [30] in Sec. C of the suppl. material. We used the publicly available notebook [31] and the TTK implementation [30, 56] of the contour tree alignment and fuzzy contour tree algorithms.

4.7 Convergence and Runtime Performance

We now study the convergence of the barycenter iteration experimentally. We ran the barycenter algorithm for 100 iterations on four different datasets and on each dataset we used different initial candidates. The plots can be found in Fig. 12. As can be seen, the Fréchet energy converges in less than 10 iterations on all datasets. Although the first iteration often increases the energy, it is overall significantly decreased. However, for some initial candidates of the heated cylinder ensemble, it converges at an energy higher than in the initial state. Furthermore, the plots show that the convergence energy can differ within one dataset depending on the initial candidate.

Next, we consider the runtime performance/complexity of the new method. Branch decomposition-independent edit distances have an asymptotic running time of $\mathcal{O}(n^4)$ and need the same amount of memory, in contrast to $\mathcal{O}(n^2)$ for classic tree edit distances. This limits the practical application of these distances on very large merge trees containing thousands of nodes. E.g. distance computation on the unsimplified asteroid dataset [42] requires in excess of 500GB of RAM and should therefore be performed on advanced hardware. Running times in the range of seconds for single distance computation are observed for merge trees of up to a few hundred nodes (see [61, 62]). Since finding the geodesic only requires the computation of a single path mapping, this performance analysis can also be applied here.

For the barycenter computation, multiple path mappings have to be computed in each iteration. The barycenter candidate (initially a member tree) can get significantly larger than the member trees, since it may contain many unmapped features. Thus, we now focus on the runtime performance of the barycenter iteration, considering the amount of ensemble members, the size of the barycenter and the iteration number in addition to the size of the input trees.

In Table 1, running times for the barycenter computation is given for multiple ensembles. Here, we used the datasets from above (with different levels of simplification) as well as a jet flow simulation dataset and the vortex street dataset from [50] (simulated by Weinkauf [60] using the Gerris flow solver [45]). With the path mapping barycenters, feasible running times in the range of seconds are reached on all datasets with input trees of less than 100 nodes, even if the barycenter size gets up to over 400 nodes. On the Jet dataset, with an average member

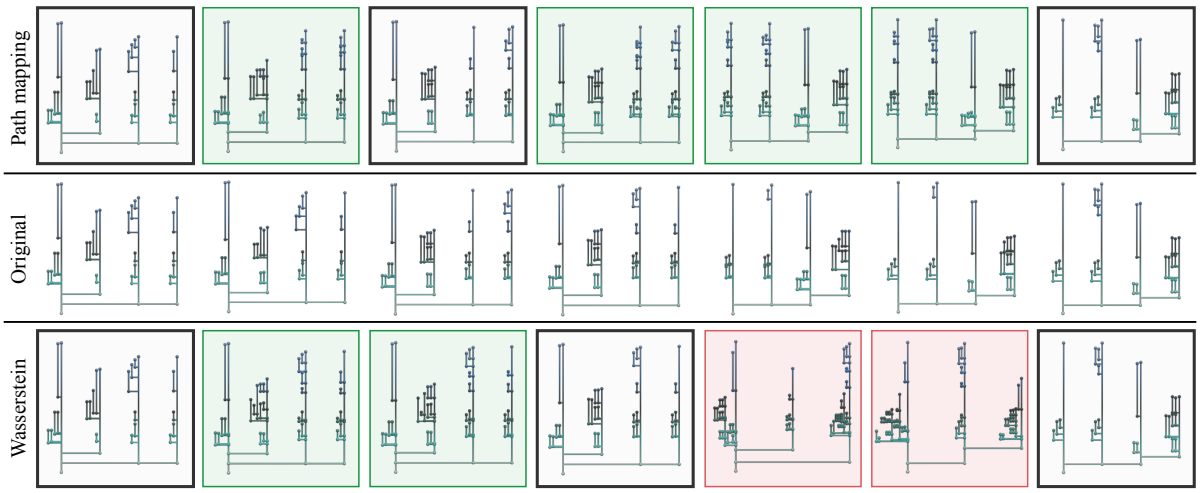

Fig. 11: Two reconstructions (top and bottom line) of a time series of merge trees (center line) after temporal reduction. For both the Wasserstein distance and the path mapping distance, we used three key frames in the encoding phase (highlighted through black frames). We then used path mapping geodesics and Wasserstein geodesics to reconstruct the sequence. Visually good reconstructions are highlighted in green, bad ones in red. A numerical comparison can be found in Table 1 of Sec. D of the suppl. material.

tree size of roughly 150 and barycenter sizes of up to 400 vertices, the running times were still in the range of minutes. Regarding the effect of the ensemble size, a super-linear increase in runtime can be observed. This is probably due to the fact that the barycenter size scales with the number of members: not only the number of distance computations increases, but their complexity does, too. However, this happens for both the Wasserstein and the path mapping distance.

The running times of the Wasserstein barycenter do not suffer from such an increase, as expected. All running times are below 1s. The comparison on the analytical example suggests that, in case of decreased barycenter sizes through the better mapping quality, the path mapping distance can sometimes be quicker.

Overall, the numbers suggest that a topological simplification should be applied prior to the path mapping barycenter computation. Since one of the main purposes of the barycenter is a visual summarization, a few hundred vertices seems to be a reasonable limitation, as merge trees containing more nodes are hard to use as a visual summary anyway.

5 Results

In this paper, we defined new notions of geodesics and barycenters of merge trees based on the metric space defined by the path mapping distance. We gave an algorithm heuristically computing barycenters as well as accurate geodesics. We implemented the algorithm in TTK and integrated it into the existing Wasserstein barycenter implementation, yielding a unified framework. We then provided experimental evidence for the improved quality of path mapping barycenters over Wasserstein barycenters on five different datasets: the path mapping barycenters are often better summarizations of ensembles, lead more frequently to correct clustering results and can further reduce time series of merge trees while retaining or even improving reconstruction quality. Furthermore, we highlighted limitations that are summarized in the following paragraph. These should be considered in future work.

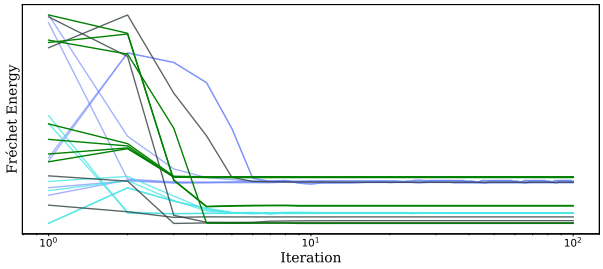

Fig. 12: (Relative) Fréchet energy at each iteration of the barycenter algorithm for two member series of the heated cylinder ensemble (blue/cyan), the starting vortex ensemble (gray) and the human cluster of the TOSCA ensemble (green). Since absolute value of the energy depends on the given scalar function, we omit quantitative labels on the y-axis.

Table 1: Runtime performance of the path mapping and Wasserstein barycenters for various datasets. $|T|$ denotes the average size of the member trees, k the amount of members. The columns labeled $|B|$ show the minimum and maximum barycenter size, averaged over the different runs. The average number of iterations is denoted by n and average runtime by t.

Dataset	$\|T\|$	k	$\|B_P\|$	n_P	t_P	$\|B_W\|$	n_W	t_W
Analytic Ex.	18	20	18-18	3	0.009s	70-150	3	0.02s
Starting Vortex	33	6	70-78	3.4	0.05s	90-100	4.4	0.01s
TOSCA	35	10	61-75	4.5	0.05s	92-118	3.6	0.017s
TOSCA	48	10	93-136	5	0.28s	156-185	5	0.04s
Ionization	47	12	105-196	6.3	0.77s	167-261	5	0.07s
Ionization	71	12	191-313	5.7	3.5s	330-456	5	0.165s
Ionization	89	12	257-406	4.8	14.5s	448-579	4.3	0.22s
Vortex Street	69	10	156-172	4.9	1.84s	168-174	4.6	0.04s
Jet	149	10	242-377	7	348s	257-411	5.6	0.29s
Heat. Cylinder	19	23	76-145	5	0.24s	129-174	3.8	0.03s
Heat. Cylinder	19	46	94-234	5.8	1.0s	175-297	5.1	0.12s
Heat. Cylinder	19	69	125-405	5.6	3.7s	304-477	3.8	0.24s

Limitations. Our method shows increased runtimes, both asymptotically ($\mathcal{O}(n^4)$ vs $\mathcal{O}(n^2)$) and practically (single threaded times are in the range of minutes on merge trees with 100-200 nodes) when compared to the Wasserstein framework. There is also the possibility to reduce result quality on datasets containing specific forms of saddle swaps. Furthermore, it remains open whether our barycenter algorithm fulfills some form of formal convergence property, or if this can be achieved with adapted strategies. The influence of the random initialization should also be studied in more detail.

Conclusion. Overall, we gave strong evidence that path mapping barycenters (though having limitations) should be considered in practical applications and gave users the option to do so (e.g. choosing the preferred method based on sizes of the input trees) in an established framework.

Acknowledgments

The authors wish to thank Raghavendra Sridharamurthy for providing the pre-processed TOSCA dataset and Marvin Petersen for helpful discussions. This work is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 442077441 - as well as by the European Commission grant ERC-2019-COG *"TORI"* (ref. 863464, https://erc-tori.github.io/).

Supplementary Material

This manuscript is accompanied by supplementary material:

- The publicly available source code [63] is provided together with detailed instructions to compile it and reproduce the images shown in Fig. 1. This implementation will be contributed as open source to TTK in the future.

  The companion ZIP file contains an archive of the repository.

- We provide a supplementary PDF that contains additional images and a proof that the proposed interpolation yields a geodesic.

References

[1] J. P. Ahrens, B. Geveci, and C. C. Law. Paraview: An end-user tool for large-data visualization. In C. D. Hansen and C. R. Johnson, eds., *The Visualization Handbook*, pp. 717–731. Academic Press / Elsevier, 2005. doi: 10.1016/b978-012387582-2/50038-1 5

[2] U. Bauer, B. D. Fabio, and C. Landi. An edit distance for reeb graphs. In A. Ferreira, A. Giachetti, and D. Giorgi, eds., *9th Eurographics Workshop on 3D Object Retrieval, 3DOR@Eurographics 2016, Lisbon, Portugal, May 8, 2016*. Eurographics Association, 2016. doi: 10.2312/3dor.20161084 2

[3] U. Bauer, X. Ge, and Y. Wang. Measuring distance between reeb graphs. In S. Cheng and O. Devillers, eds., *30th Annual Symposium on Computational Geometry, SoCG'14, Kyoto, Japan, June 08 - 11, 2014*, pp. 464–473. ACM, 2014. doi: 10.1145/2582112.2582169 2

[4] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann. Measuring the distance between merge trees. In P. Bremer, I. Hotz, V. Pascucci, and R. Peikert, eds., *Topological Methods in Data Analysis and Visualization III, Theory, Algorithms, and Applications*, pp. 151–165. Springer, 2014. doi: 10.1007/978-3-319-04099-8_10 1, 2

[5] B. Bollen, P. Tennakoon, and J. A. Levine. Computing a stable distance on merge trees. *IEEE Trans. Vis. Comput. Graph.*, 29(1):1168–1177, 2023. doi: 10.1109/TVCG.2022.3209395 2

[6] P. Bremer, G. H. Weber, V. Pascucci, M. S. Day, and J. B. Bell. Analyzing and tracking burning structures in lean premixed hydrogen flames. *IEEE Trans. Vis. Comput. Graph.*, 16(2):248–260, 2010. doi: 10.1109/TVCG.2009.69 2

[7] P. Bremer, G. H. Weber, J. Tierny, V. Pascucci, M. S. Day, and J. B. Bell. Interactive exploration and analysis of large-scale simulations using topology-based data segmentation. *IEEE Trans. Vis. Comput. Graph.*, 17(9):1307–1324, 2011. doi: 10.1109/TVCG.2010.253 2

[8] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. *Numerical Geometry of Non-Rigid Shapes*. Monographs in Computer Science. Springer, 2009. doi: 10.1007/978-0-387-73301-2 6

[9] H. A. Carr and J. Snoeyink. Path seeds and flexible isosurfaces - using topology for exploratory visualization. In *5th Joint Eurographics - IEEE TCVG Symposium on Visualization, VisSym 2003, Grenoble, France, May 26-28, 2003*, pp. 49–58. Eurographics Association, 2003. doi: 10.2312/VisSym/VisSym03/049-058 2

[10] M. E. Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Syst. Appl.*, 40(1):200–210, 2013. doi: 10.1016/j.eswa.2012.07.021 1, 5

[11] F. Chazal, D. Cohen-Steiner, M. Glisse, L. J. Guibas, and S. Oudot. Proximity of persistence modules and their diagrams. In J. Hershberger and E. Fogel, eds., *Proceedings of the 25th ACM Symposium on Computational Geometry, Aarhus, Denmark, June 8-10, 2009*, pp. 237–246. ACM, 2009. doi: 10.1145/1542362.1542407 2

[12] D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Discret. Comput. Geom.*, 37(1):103–120, 2007. doi: 10.1007/s00454-006-1276-5 2

[13] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, and Y. Mileyko. Lipschitz functions have L_p-stable persistence. *Found. Comput. Math.*, 10(2):127–139, 2010. doi: 10.1007/s10208-010-9060-6 2

[14] H. Edelsbrunner and J. Harer. *Computational Topology - an Introduction*. American Mathematical Society, 2010. 2, 3

[15] H. Edelsbrunner, J. Harer, A. Mascarenhas, and V. Pascucci. Time-varying reeb graphs for continuous space-time data. In J. Snoeyink and J. Boissonnat, eds., *Proceedings of the 20th ACM Symposium on Computational Geometry, Brooklyn, New York, USA, June 8-11, 2004*, pp. 366–372. ACM, 2004. doi: 10.1145/997817.997872 2

[16] H. Edelsbrunner, D. Letscher, and A. Zomorodian. Topological persistence and simplification. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pp. 454–463. IEEE Computer Society, 2000. doi: 10.1109/SFCS.2000.892133 2

[17] C. Elkan. Using the triangle inequality to accelerate k-means. In T. Fawcett and N. Mishra, eds., *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pp. 147–153. AAAI Press, 2003. 1, 5

[18] M. Evers, M. Herick, V. Molchanov, and L. Linsen. Coherent topological landscapes for simulation ensembles. In K. Bouatouch, A. A. de Sousa, M. Chessa, A. Paljic, A. Kerren, C. Hurter, G. M. Farinella, P. Radeva, and J. Braz, eds., *Computer Vision, Imaging and Computer Graphics Theory and Applications - 15th International Joint Conference, VISIGRAPP 2020 Valletta, Malta, February 27-29, 2020, Revised Selected Papers*, vol. 1474 of *Communications in Computer and Information Science*, pp. 223–237. Springer, 2020. doi: 10.1007/978-3-030-94893-1_10 2

[19] B. D. Fabio and C. Landi. The edit distance for reeb graphs of surfaces. *Discret. Comput. Geom.*, 55(2):423–461, 2016. doi: 10.1007/s00454-016-9758-6 2

[20] G. Favelier, N. Faraj, B. Summa, and J. Tierny. Persistence atlas for critical point variability in ensembles. *IEEE Trans. Vis. Comput. Graph.*, 25(1):1152–1162, 2019. doi: 10.1109/TVCG.2018.2864432 6

[21] E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. *CoRR*, 1908.00063, 2019. doi: 10.48550/arXiv.1908.00063 2

[22] D. Günther, J. Salmon, and J. Tierny. Mandatory critical points of 2d uncertain scalar fields. *Comput. Graph. Forum*, 33(3):31–40, 2014. doi: 10.1111/cgf.12359 2

[23] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. D. Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Comput. Graph. Forum*, 35(3):643–667, 2016. doi: 10.1111/cgf.12933 2

[24] M. Hilaga, Y. Shinagawa, T. Komura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In L. Pocock, ed., *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2001, Los Angeles, California, USA, August 12-17, 2001*, pp. 203–212. ACM, 2001. doi: 10.1145/383259.383282 6

[25] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985. doi: 10.1007/BF01908075 5

[26] W. Köpp and T. Weinkauf. Temporal merge tree maps: A topology-based static visualization for temporal scalar data. *IEEE Trans. Vis. Comput. Graph.*, 29(1):1157–1167, 2023. doi: 10.1109/TVCG.2022.3209387 2

[27] M. Kraus. Visualization of uncertain contour trees. In P. Richard and J. Braz, eds., *IMAGAPP 2010 - Proceedings of the International Conference on Imaging Theory and Applications and IVAPP 2010 - Proceedings of the International Conference on Information Visualization Theory and Applications, Angers, France, May 17 - 21, 2010*, pp. 132–139. INSTICC Press, 2010. doi: 10.5220/0002817201320139 2

[28] S. P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28(2):129–136, 1982. doi: 10.1109/TIT.1982.1056489 3

[29] A. P. Lohfink, F. Gartzky, F. Wetzels, L. Vollmer, and C. Garth. Time-varying fuzzy contour trees. In *2021 IEEE Visualization Conference, IEEE VIS 2021 - Short Papers, New Orleans, LA, USA, October 24-29, 2021*, pp. 86–90. IEEE, 2021. doi: 10.1109/VIS49827.2021.9623286 2

[30] A. P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth. Fuzzy contour trees: Alignment and joint layout of multiple contour trees. *Comput. Graph. Forum*, 39(3):343–355, 2020. doi: 10.1111/cgf.13985 2, 8

[31] A. P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth. Source Code for Fuzzy Contour Trees: Alignment and Joint Layout of Multiple Contour Trees. https://github.com/scivislab/

Fuzzy-Contour-Trees, 2021. 8

[32] J. Lukasczyk, G. Aldrich, M. Steptoe, G. Favelier, C. Gueunet, J. Tierny, R. Maciejewski, B. Hamann, and H. Leitte. Viscous fingering: A topological visual analytic approach. In *Physical Modeling for Virtual Manufacturing Systems and Processes*, vol. 869 of *Applied Mechanics and Materials*, pp. 9–19. Trans Tech Publications Ltd, 9 2017. doi: 10.4028/www.scientific.net/AMM.869.9 2

[33] J. Lukasczyk, C. Garth, G. H. Weber, T. Biedert, R. Maciejewski, and H. Leitte. Dynamic nested tracking graphs. *IEEE Trans. Vis. Comput. Graph.*, 26(1):249–258, 2020. doi: 10.1109/TVCG.2019.2934368 2

[34] J. Lukasczyk, G. H. Weber, R. Maciejewski, C. Garth, and H. Leitte. Nested tracking graphs. *Comput. Graph. Forum*, 36(3):12–22, 2017. doi: 10.1111/cgf.13164 2

[35] D. Morozov, K. Beketayev, and G. H. Weber. Interleaving distance between merge trees. In *TopoInVis*. 2014. 1

[36] D. Morozov and G. H. Weber. Distributed merge trees. In A. Nicolau, X. Shen, S. P. Amarasinghe, and R. W. Vuduc, eds., *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, PPoPP '13, Shenzhen, China, February 23-27, 2013*, pp. 93–102. ACM, 2013. doi: 10.1145/2442516.2442526 2

[37] D. Morozov and G. H. Weber. Distributed contour trees. In P. Bremer, I. Hotz, V. Pascucci, and R. Peikert, eds., *Topological Methods in Data Analysis and Visualization III, Theory, Algorithms, and Applications*, pp. 89–102. Springer, 2014. doi: 10.1007/978-3-319-04099-8_6 2

[38] V. Narayanan, D. M. Thomas, and V. Natarajan. Distance between extremum graphs. In S. Liu, G. Scheuermann, and S. Takahashi, eds., *2015 IEEE Pacific Visualization Symposium, PacificVis 2015, Hangzhou, China, April 14-17, 2015*, pp. 263–270. IEEE Computer Society, 2015. doi: 10.1109/PACIFICVIS.2015.7156386 2

[39] E. Nilsson, J. Lukasczyk, W. Engelke, T. B. Masood, G. Svensson, R. Caballero, C. Garth, and I. Hotz. Exploring cyclone evolution with hierarchical features. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pp. 92–102, 2022. doi: 10.1109/TopoInVis57755.2022.00016 2

[40] P. Oesterling, C. Heine, G. H. Weber, D. Morozov, and G. Scheuermann. Computing and visualizing time-varying merge trees for high-dimensional data. In H. Carr, C. Garth, and T. Weinkauf, eds., *Topological Methods in Data Analysis and Visualization IV*, pp. 87–101. Springer International Publishing, 2017. doi: 10.1007/978-3-319-44684-4_5 2

[41] V. Pascucci, K. Cole-McLaughlin, and G. Scorzelli. Multi-resolution computation and presentation of contour trees. In *IASTED*, 2004. 2

[42] J. Patchett and G. R. Gisler. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2018/, 2018. 8

[43] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Trans. Vis. Comput. Graph.*, 28(1):291–301, 2022. doi: 10.1109/TVCG.2021.3114839 1, 2, 3, 4, 5, 6

[44] M. Pont, J. Vidal, and J. Tierny. Principal geodesic analysis of merge trees (and persistence diagrams). *IEEE Trans. Vis. Comput. Graph.*, 29(2):1573–1589, 2023. doi: 10.1109/TVCG.2022.3215001 2

[45] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004. 6, 8

[46] H. Saikia, H. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Comput. Graph. Forum*, 33(3):41–50, 2014. doi: 10.1111/cgf.12360 1, 2

[47] H. Saikia and T. Weinkauf. Global feature tracking and similarity estimation in time-dependent scalar fields. *Comput. Graph. Forum*, 36(3):1–11, 2017. doi: 10.1111/cgf.13163 2

[48] A. Schnorr, D. N. Helmrich, D. Denker, T. W. Kuhlen, and B. Hentschel. Feature tracking by two-step optimization. *IEEE Trans. Vis. Comput. Graph.*, 26(6):2219–2233, 2020. doi: 10.1109/TVCG.2018.2883630 2

[49] Q. Shu, H. Guo, J. Liang, L. Che, J. Liu, and X. Yuan. Ensemblegraph: Interactive visual analysis of spatiotemporal behaviors in ensemble simulation data. In C. Hansen, I. Viola, and X. Yuan, eds., *2016 IEEE Pacific Visualization Symposium, PacificVis 2016, Taipei, Taiwan, April 19-22, 2016*, pp. 56–63. IEEE Computer Society, 2016. doi: 10.1109/PACIFICVIS.2016.7465251 2

[50] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Trans. Vis. Comput. Graph.*, 26(3):1518–1531, 2020. doi: 10.1109/TVCG.2018.2873612 1, 2, 6, 8

[51] R. Sridharamurthy and V. Natarajan. Comparative analysis of merge trees using local tree edit distance. *IEEE Trans. Vis. Comput. Graph.*, 29(2):1518–1530, 2023. doi: 10.1109/TVCG.2021.3122176 2

[52] S. Takahashi, Y. Takeshima, and I. Fujishiro. Topological volume skeletonization and its application to transfer function design. *Graphical Models*, 66(1):24–49, 2004. doi: 10.1016/j.gmod.2003.08.002 2

[53] R. Taylor, A. Chourasia, D. Whalen, and M. L. Norman. The IEEE SciVis Contest. http://sciviscontest.ieeevis.org/2008/, 2008. 7

[54] D. M. Thomas and V. Natarajan. Symmetry in scalar field topology. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2035–2044, 2011. doi: 10.1109/TVCG.2011.236 2

[55] D. M. Thomas and V. Natarajan. Detecting symmetry in scalar fields using augmented extremum graphs. *IEEE Trans. Vis. Comput. Graph.*, 19(12):2663–2672, 2013. doi: 10.1109/TVCG.2013.148 2

[56] J. Tierny, G. Favelier, J. A. Levine, C. Gueunet, and M. Michaux. The topology toolkit. *IEEE Trans. Vis. Comput. Graph.*, 24(1):832–842, 2018. doi: 10.1109/TVCG.2017.2743938 2, 5, 8

[57] K. Turner, Y. Mileyko, S. Mukherjee, and J. Harer. Fréchet means for distributions of persistence diagrams. *Discret. Comput. Geom.*, 52(1):44–70, 2014. doi: 10.1007/s00454-014-9604-7 2

[58] J. Vidal, J. Budin, and J. Tierny. Progressive wasserstein barycenters of persistence diagrams. *IEEE Trans. Vis. Comput. Graph.*, 26(1):151–161, 2020. doi: 10.1109/TVCG.2019.2934256 2, 6

[59] G. H. Weber, P. Bremer, and V. Pascucci. Topological landscapes: A terrain metaphor for scientific data. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1416–1423, 2007. doi: 10.1109/TVCG.2007.70601 2

[60] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Trans. Vis. Comput. Graph.*, 16(6):1225–1234, 2010. doi: 10.1109/TVCG.2010.198 8

[61] F. Wetzels and C. Garth. A deformation-based edit distance for merge trees. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pp. 29–38, 2022. doi: 10.1109/TopoInVis57755.2022.00010 1, 2, 3, 8

[62] F. Wetzels, H. Leitte, and C. Garth. Branch decomposition-independent edit distances for merge trees. *Comput. Graph. Forum*, 41(3):367–378, 2022. doi: 10.1111/cgf.14547 1, 2, 5, 7, 8

[63] F. Wetzels, M. Pont, J. Tierny, and C. Garth. Merge tree geodesics and barycenters with path mappings (supplementary source code). https://doi.org/10.5281/zenodo.8160990, 2023. 10

[64] W. Widanagamaachchi, C. Christensen, P. Bremer, and V. Pascucci. Interactive exploration of large-scale time-varying data using dynamic tracking graphs. In R. S. Barga, H. Pfister, and D. H. Rogers, eds., *IEEE Symposium on Large Data Analysis and Visualization, LDAV 2012, Seattle, Washington, USA, 14-15 October, 2012*, pp. 9–17. IEEE Computer Society, 2012. doi: 10.1109/LDAV.2012.6378962 2

[65] K. Wu and S. Zhang. A contour tree based visualization for exploring data with uncertainty. *International Journal for Uncertainty Quantification*, 3:203–223, 2013. doi: 10.1615/Int.J.UncertaintyQuantification.2012003956 2

[66] L. Yan, T. Bin Masood, F. Rasheed, I. Hotz, and B. Wang. Geometry aware merge tree comparisons for time-varying data with interleaving distances. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–1, 2022. doi: 10.1109/TVCG.2022.3163349 2

[67] L. Yan, T. B. Masood, R. Sridharamurthy, F. Rasheed, V. Natarajan, I. Hotz, and B. Wang. Scalar field comparison with topological descriptors: Properties and applications for scientific visualization. *Comput. Graph. Forum*, 40(3):599–633, 2021. doi: 10.1111/cgf.14331 2

[68] L. Yan, Y. Wang, E. Munch, E. Gasparovic, and B. Wang. A structural average of labeled merge trees for uncertainty visualization. *IEEE Trans. Vis. Comput. Graph.*, 26(1):832–842, 2020. doi: 10.1109/TVCG.2019.2934242 2

Merge Tree Geodesics and Barycenters with Path Mappings Supplementary Material

A Geodesic Proof

In this section, we provide a formal proof that for two input merge trees the interpolation defined in Sec. 3 indeed forms a geodesic between the input trees. We now give a formal description of the interpolated geodesic trees to make formal arguments easier. Let $(T_0, f_0), (T_1, f_1)$ be two merge trees, $M \subseteq \mathcal{P}(T_0) \times \mathcal{P}(T_1)$ a path mapping between T_0 and T_1 and let $(T_\alpha, f_\alpha)_{\alpha\in(0,1)}$ as follows. We write $\ell_0, \ell_1, \ell_\alpha$ for $\ell_{f_0}, \ell_{f_1}, \ell_{f_\alpha}$.

Intuitively, for a given α, we interpolate the two trees with coefficients α and $(1-\alpha)$. We first interpolate all mapped paths and move the nodes on these paths such that their relative position on the path remains constant. Then, we interpolate the inserted/deleted edges from/to length zero. Note that the problem of contradicting paths described before does not arise here, as only one mapping is considered. Structurally, the interpolated tree is the supertree induced by the path mapping. To define scalars, we first interpolate the labels of the matched nodes, i.e. the start and end vertices of the matched paths. Then, we move the nodes on these paths such that their relative position stays the same. Furthermore, we contract all deleted or inserted edges to $1-\alpha$ or α of their original lengths. An example is shown in Fig. 1.

For a formal definition, recall that we say a vertex $v \in V(T_0)$ is present in M (and write $v \in M$) if there is a pair of paths $(p, p') \in M$ such that v is in p (and analogously for vertices of T_1). Furthermore, we assume that $V(T_0)$ and $V(T_1)$ are disjoint and denote the scalar function on the union $V(T_0) \dot\cup V(T_1)$ of nodes by f. Now we define $V(T_\alpha)$ to be the set

$$\{(v, v') \mid (v, \dots, u, v', \dots, u') \in M\}$$
$$\cup \{v_i, u_j \mid (v_0, \dots, v_k, u_0, \dots, u_{k'}) \in M, 1 \le i < k, 1 \le j < k'\}$$
$$\cup \{v \in V(T_0) \mid v \notin M\} \cup \{v \in V(T_1) \mid v \notin M\}.$$

In the example in Fig. 1, the ellipse nodes form the first set, nodes C_0, C_1, F_0 form the second set and E_0, E_1, H_0 form the last set.

Next, we define the edge set of T_α. For a pair of mapped paths $(v_1 \dots v_k, u_1 \dots u_{k'}) \in M$ (an example path is highlighted in Fig. 1), let $s_1, s_2, \dots, s_{k+k'-4}$ be the sorted union of the inner nodes of the two paths, i.e. $f(s_1) \le f(s_2) \le \dots \le f(s_{k+k'-4})$ (in the example, this sequence is C_0C_1). For each such path in M, we then include the edges $((v_1, u_1), s_1)$, $(s_{k+k'-4}, (v_k, u_{k'}))$ and (s_i, s_{i+1}) for each $1 \le i < k+k'-4$. Furthermore, we include the edge $(v, v') \in E(T_0)$ if $v \notin M$ and the same way for edges of T_1. In the case where v' is the start or end vertex of a mapped path, we have to replace it by its corresponding node in $V(T_\alpha)$ (the resulting path in T_α is also highlighted in the example).

As a last step, we need to define the scalar function on the new nodes. For the matched nodes (v, v') (i.e. v, v' are the start or end nodes on two mapped paths), we define $f_\alpha((v, v')) = (1-\alpha)\cdot f_0(v) + \alpha \cdot f_1(v')$. For easier notation, we also write $f_\alpha(v)$ or $f_\alpha(v')$ instead of $f_\alpha((v, v'))$. For a node p_i on a path $p_1 \dots p_k$ matched to $p'_1 \dots p_{k'}$ with $1 \ne i \ne k$,

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

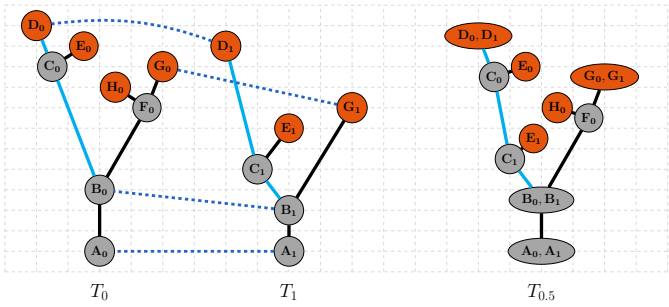

Fig. 1: Merge trees T_0 and T_1 with barycenter $T_{0.5}$. The optimal path mapping between T_0 and T_1 is illustrated by the dotted lines. Two mapped paths and interpolated path in the geodesic tree are highlighted in cyan. Scalar values and edge lengths can be read from the grid.

we define $f_\alpha(p_i) = \frac{f_0(p_i)-f_0(p_1)}{f_0(p_k)-f_0(p_1)} \cdot (f_\alpha(p_k) - f_\alpha(p_1)) + f_\alpha((p_1, p_k))$. For a deleted node $v \notin M$ with parent p, we define $f_\alpha(v) = f_\alpha(p) + (1-\alpha)\cdot(f_0(v) - f_0(p))$. For an inserted node $v \notin M$ with parent p, we define $f_\alpha(v) = f_\alpha(p) + \alpha \cdot (f_0(v) - f_0(p))$. Note that at least one pair of paths containing the roots of both trees is in the optimal mapping and thus the recursive definition above is well-defined. Furthermore, the described tree is indeed the result of the first iteration barycenter computation in Sec. 3 when the number of inputs is two.

Based on this definition, we can now show that the merge trees T_α $(0 \le \alpha \le 1)$ define a geodesic between T_0 and T_1. Clearly, the tree T_α can be created from T_0, T_1 in linear time.

Recall that, for a metric d, a continuous path $P = (T_\alpha)_{0\le\alpha\le1}$ between two trees T_0, T_1 is a geodesic if its length

$$\mathcal{L}(P) = \sup_{n;0=t_0\le t_1\le\dots\le t_n=1} \sum_{k=0}^{n-1} d(T_{t_k}, T_{t_{k+1}})$$

is exactly the distance $d(T_1, T_2)$ between T_1 and T_2.

Now consider two time points $s, t \in [0, 1]$. With the above definition, we can derive a path mapping $M_{s,t}$ between T_s and T_t from M. We have to make a case distinction on whether the one of the two time points is 0 or 1.

For $0 < s \le t < 1$, the two trees are structurally the same (only the labels differ). We define $M_{s,t}$ to be the identity mapping on the edges, which is obviously a valid path mapping. Thus, $\delta(T_s, T_t) \le c(M_{s,t})$. Next, we determine the cost of $M_{s,t}$.

Let I and D be the inserted and deleted edges of M. We have $(e, e) \in M_{s,t}$ for each $e \in I \cup D$. Each $e \in I$ contributes $c(0, \ell_1(e)) = \ell_1(e)$ in $c(M)$, whereas they contribute $c(\ell_s(e), \ell_t(e))$ in $c(M_{s,t})$. By definition, $\ell_s(e) = s \cdot \ell_1(e)$ and $\ell_t(e) = t \cdot \ell_1(e)$ and therefore $c(\ell_s(e), \ell_t(e)) = (t-s)\cdot\ell_1(e)$. Analogously, each $e \in D$ contributes $c(\ell_0(e), 0) = \ell_0(e)$ in $c(M)$, whereas they contribute $c(\ell_s(e), \ell_t(e))$ in $c(M_{s,t})$. By definition, $\ell_s(e) = (1-s)\cdot\ell_0(e)$ and $\ell_t(e) = (1-t)\cdot\ell_0(e)$ and therefore $c(\ell_s(e), \ell_t(e)) = (t-s)\cdot\ell_0(e)$.

Now consider the rest of the edges in T_s and T_t. They are constructed from a mapped path as described above. A pair of mapped paths (we assume leaf-to-root direction in a split tree) $(p_1 \dots p_k, p'_1 \dots p'_{k'}) \in M$ contributes

$$||f_0(p_1) - f_0(p_k)| - |f_1(p'_1) - f_1(p'_{k'})||$$

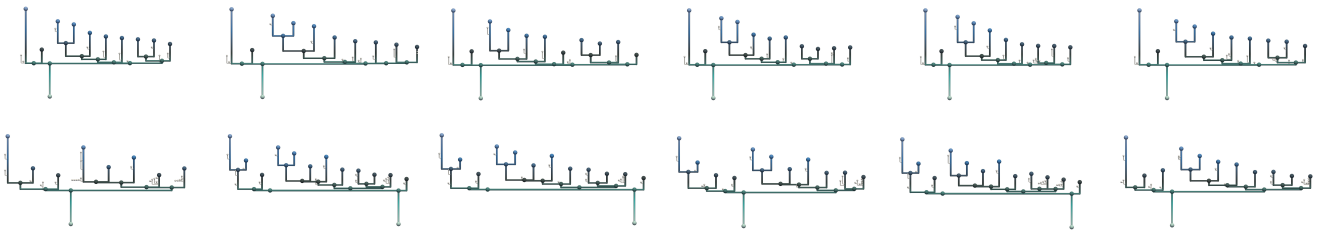

Fig. 2: All possible barycenter results on the starting vortex ensemble. The top row shows the path mapping barycenters for each of the six initial candidates. The bottom row shows the corresponding Wasserstein barycenters.

Path mapping distance	TP0	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10	TP11	TP12
Wasserstein Geodesic	**0.0**	0.63	0.85	1.2	1.02	0.74	**0.0**	1.59	2.59	3.54	3.09	1.6	**0.0**
Path mapping Geodesic	**0.0**	0.52	0.62	0.86	0.78	**0.0**	0.68	0.96	0.98	1.04	0.75	0.53	**0.0**
Wasserstein distance	TP0	TP1	TP2	TP3	TP4	TP5	TP6	TP7	TP8	TP9	TP10	TP11	TP12
Wasserstein Geodesic	**0.0**	0.06	0.17	0.18	0.11	0.04	**0.0**	1.34	0.87	0.49	0.22	0.06	**0.0**
Path mapping Geodesic	**0.0**	0.26	0.37	0.09	0.06	**0.0**	0.01	0.03	0.04	0.05	0.03	0.02	**0.0**

Table 1: Comparison of the temporal reduction results based on path mapping geodesics and Wasserstein geodesics. The first table shows the path mapping distance between the original and reconstructed merge trees for each time point and both methods. The second table depicts the Wasserstein distances. Keyframes are again highlighted in bold.

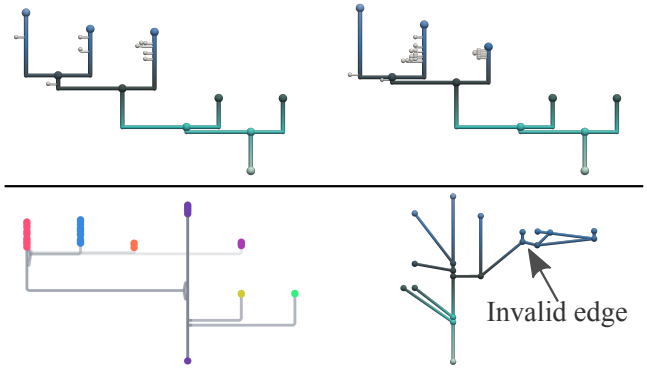

Fig. 3: Comparison of merge tree barycenters and contour tree alignments: The top left image shows the path mapping barycenter, the top right image the Wasserstein barycenter. The bottom row shows the fuzzy contour tree on the left and the ParaView rendering of the alignment tree on the right. The latter illustrates the problem of the fuzzy contour tree summarization: the ensemble representative is not a valid merge tree.

$$= f_0(p_1) - f_0(p_k) - (f_1(p'_1) - f_1(p'_{k'}))$$

to $c(M)$. Instead of $p_1 ... p_k$ and $p'_1 ... p'_{k'}$ in T_0, T_1, we have in T_α the nodes

$$s_0 := (p_1, p'_1), s_1, s_2, ..., s_{k+k'-4}, s_{k+k'-3} := (p_k, p'_{k'})$$

as well as the edges (s_i, s_{i+1}) for each $0 \leq i \leq k + k' - 4$.

Each mapped edge in $M_{s,t}$ contributes

$$||f_s(s_i) - f_s(s_{i+1})| - |f_t(s_i) - f_t(s_{i+1})||$$

to $c(M_{s,t})$. Thus, the whole path contributes

$$\sum_{0 \leq i \leq k+k'-4} ||f_s(s_i) - f_s(s_{i+1})| - |f_s(s_i) - f_s(s_{i+1})||.$$

Note that the whole path and thus each single edge gets either shorter or longer. So we either have:

- $|f_s(s_0) - f_s(s_{k+k'-3})| > |f_t(s_0) - f_t(s_{k+k'-3})|$ and $|f_s(s_i) - f_s(s_{i+1})| > |f_t(s_i) - f_t(s_{i+1})|$ for each i or
- $|f_s(s_0) - f_s(s_{k+k'-3})| < |f_t(s_0) - f_t(s_{k+k'-3})|$ and $|f_s(s_i) - f_s(s_{i+1})| < |f_t(s_i) - f_t(s_{i+1})|$ for each i.

W.l.o.g. we have $|f_s(s_i) - f_s(s_{i+1})| > |f_t(s_i) - f_t(s_{i+1})|$ for each i. In total, we get for the cost of mapping the whole path:

$$\sum_{0 \leq i \leq k+k'-4} ||f_s(s_i) - f_s(s_{i+1})| - |f_s(s_i) - f_s(s_{i+1})||$$

$$= \sum_{0 \leq i \leq k+k'-4} (f_s(s_i) - f_s(s_{i+1})) - (f_t(s_i) - f_t(s_{i+1}))$$

$$= \sum_{0 \leq i \leq k+k'-4} (f_s(s_i) - f_s(s_{i+1})) - \sum_{0 \leq i \leq k+k'-4} (f_t(s_i) - f_t(s_{i+1}))$$

$$= f_s(s_0) - f_s(s_{k+k'-3}) - (f_t(s_0) - f_t(s_{k+k'-3}))$$

$$= f_s(p_1) - f_s(p_k) - (f_t(p'_1) - f_t(p'_{k'}))$$

$$= (1-s)f_0(p_1) + sf_1(p'_1) - ((1-s)f_0(p_k) + sf_1(p'_{k'}))$$

$$-((1-t)f_0(p_1) + tf_1(p'_1) - ((1-t)f_0(p_k) + tf_1(p'_{k'})))$$

$$= (1-s)(f_0(p_1) - f_0(p_k)) + s(f_1(p'_1) - f_1(p'_{k'}))$$

$$-((1-t)(f_0(p_1) - f_0(p_k)) + t(f_1(p'_1) - f_1(p'_{k'})))$$

$$= (t-s)(f_0(p_1) - f_0(p_k)) + (s-t)(f_1(p'_1) - f_1(p'_{k'}))$$

$$= (t-s)(f_0(p_1) - f_0(p_k)) - (t-s)(f_1(p'_1) - f_1(p'_{k'})).$$

In total, we get that for each deleted or inserted edge as well as each mapped path that contributes x to $c(M)$ contributes $(t-s)x$ to $c(M_{s,t})$. Thus, $c(M_{s,t}) = (t-s)c(M)$.

Now consider the case where $0 = s < t$. Since T_t is structurally a supertree of T_0, we can define the mapping $M_{0,t}$ to be the embedding from T_0 in T_t. Consider some inserted (in M) edge $e \in I$. Since $e \notin E(T_0)$, it is also inserted in the embedding $M_{0,t}$. Thus, it contributes $c(0, \ell_t(e)) = t \cdot \ell_1(e) = (t-s) \cdot \ell_1(e)$ to $M_{0,t}$, whereas it contributes $\ell_1(e)$ to M. A deleted (in M) edge $e \in D$ is mapped to itself in the embedding $M_{0,t}$. Thus, it contributes $c(\ell_0(e), \ell_t(e)) = \ell_0(e) - (1-t) \cdot \ell_0(e) = t \cdot \ell_0(e) = (t-s) \cdot \ell_0(e)$ to $M_{0,t}$, whereas it contributes $\ell_0(e)$ to M.

For mapped paths, all arguments are analogous to the previous case. Thus, in total we again have that for each deleted or inserted edge as well as each mapped path that contributes x to $c(M)$ contributes $(t-s)x$ to $c(M_{s,t})$, and therefore $c(M_{s,t}) = (t-s)c(M)$. The same holds for the case where $s < t = 1$ with analogous arguments.

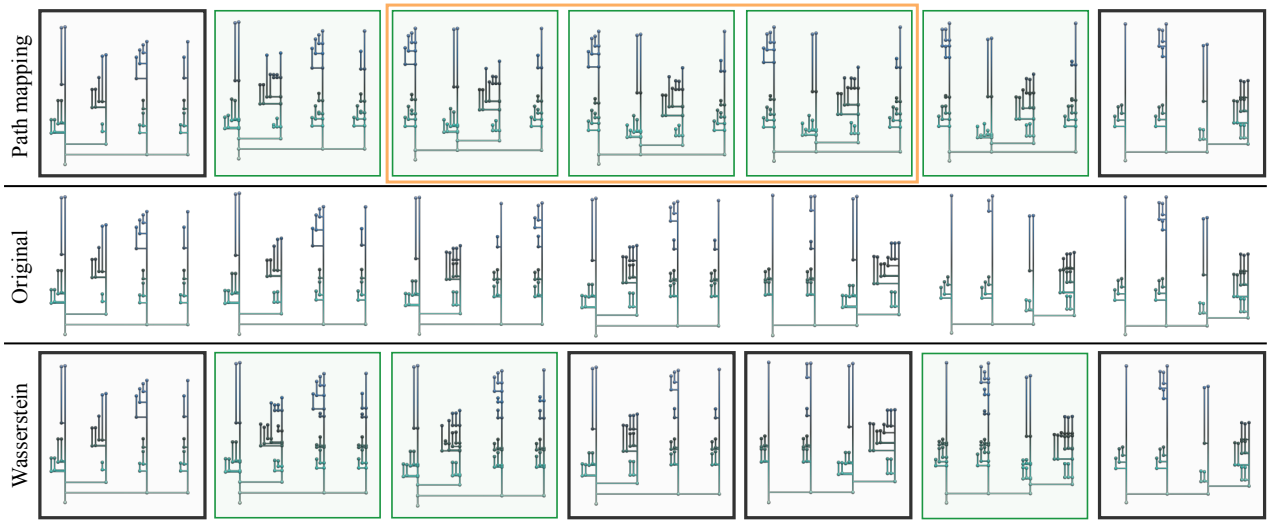

Fig. 4: Exemplary time steps of the temporal reduction and reconstruction. The original time series is shown in the middle row. The top row shows the result of the path mapping geodesic, the bottom row of the Wasserstein geodesics. The path mapping reconstruction produces merge trees with a diiferent branch decomposition (to the original series) in the time steps highlighted in yellowm, which is not the case for the Wasserstein geodesics. In particular, the original trees and their Wasserstein reconstructions have the long fork structure as main branch, whereas the path mapping reconstruction has a different one. However, this does not change the fact the path mapping reconstructions are very similar to the original series (when ignoring the branch order).

So we have $c(M_{s,t}) = (t-s)c(M)$ for any two time points $0 \leq s < t \leq 1$. From this, we can conclude that $\delta(T_s, T_t) \leq (t-s)\delta(T_0, T_1)$. We can now show that for $P = (T_\alpha)_{\alpha\in[0,1]}$, it holds that $\mathcal{L}(P) = \delta(T_0, T_1)$.

Using the metric property of δ, we know that for any $0 \leq s < t \leq 1$:

$$\delta(T_0, T_1) \leq \delta(T_0, T_s) + \delta(T_s, T_t) + \delta(T_t, T_1)$$

$$\leq ((s-0) + (t-s) + (1-t))\delta(T_0, T_1) = \delta(T_0, T_1).$$

We can conclude that $\delta(T_s, T_t) = (t-s)\delta(T_0, T_1)$ and for any subdivision $0 = t_0 < t_1 < \cdots < t_n = 1$ of P, we have:

$$\sum_{k=0}^{n-1} \delta(T_{t_k}, T_{t_{k+1}}) = \delta(T_0, T_1).$$

Thus, for the length of P we get

$$\mathcal{L}(P) = \sup_{n, 0=t_0<t_1<\cdots<t_n=1} \sum_{k=0}^{n-1} \delta(T_{t_k}, T_{t_{k+1}}) = \delta(T_0, T_1).$$

B Starting Vortex Barycenters

We now provide further screenshots of the barycenters computed on the starting vortex ensemble. Fig. 2 shows the barycenter merge trees for each possible initial candidate and both methods. For five out of six initial candidates, the Wasserstein barycenter contains two fork structures of high persistence, which is not the case in the member trees (see Fig. 4), whereas only one contains a long, non-forking edge. In contrast, all six path mapping barycenters are a good summary of the ensemble.

C Comparison to contour tree alignments

Next, we quickly illustrate the advantages of path mapping and Wasserstein barycenters over contour tree alignments. We computed barycenter merge trees, the contour tree alignment and the fuzzy contour tree layout for an ensemble consisting of a fixed time steps (in the late phase of the simulation, see Fig. 10) from different runs of the heated cylinder dataset. We applied topological simplification with a threshold of 2% of the scalar range. Fig. 3 shows both barycenters, the fuzzy contour tree rendering (see [30] for details) as well as a ParaView rendering of the alignment tree. While the branch decomposition layout of the fuzzy contour tree summarizes the ensemble well, the ParaView rendering reveals that the alignment tree is not a valid merge tree. This is due to a different averaging technique based on the nodes instead of arcs, branches or paths. It is therefore harder to use for further analysis tasks.

D Temporal Reduction

In this section, we provide more detailed results for the temporal reduction on the ionization front time series. In Sec. 4, we compared the reconstructed series of the path mapping geodesics and the Wasserstein geodesics with three key frames for both methods. The quantitative comparison in terms of actual distances between the original and reconstructed trees are given in Table 1.

Furthermore, the path mapping geodesics yield good reconstructions already for two key frames, since it can compute meaningful mappings even between the first and last time step. The Wasserstein geodesics need four key frames to produce a good reconstruction, since it can not map the first tree to the last one in a meaningful manner. It therefore needs the time step right before the maximum swap to correctly interpolate the first half of the series and the time step right after the maximum swap to correctly interpolate the second half. We illustrate this behavior on example time steps in Fig. 4. The path mapping reconstruction with two key frames is of similar quality to the one with three key frames (rated on a purely visual basis), whereas the Wasserstein reconstruction is significantly improved with four keyframes (visually bad interpolation as highlighted in red in in Fig. 11 do no longer happen).