A Theoretical Considerations

We discuss some theoretical considerations in sketching merge trees. In the first two steps of our framework, we represent merge trees as metric measure networks and vectorize them via blow-up and alignment to a Fréchet Mean using the GW framework [18]. Each merge tree $T = (V, p, W) \in \mathcal{T}$ is mapped to a column vector a in matrix A, where W captures the shortest path distances using function value differences as weights. The computation of the Fréchet mean $\overline{T}$ is an optimization process, but the blow-up of T and its alignment to $\overline{T}$ does not change the underlying distances between the tree nodes, which are encoded in W. Therefore, reshaping the column vector a back to a pairwise distance matrix and computing its corresponding MST fully recover the original input merge tree.

In the third step, we sketch the matrix A using CSS. It is also possible to apply another matrix sketching technique, namely, non-negative matrix factorization (NMF). Both CSS and NMF (albeit with different constraints) aim to obtain an approximation $\hat{A} = BY$ of A that minimizes the error $\epsilon = \|A - \hat{A}\|_F$. Let A_k denote the (unknown) best rank-k approximation of A. In the case of CSS, the theoretical upper bound is given as a multiplicative error of the form $\epsilon \leq \epsilon_k \cdot \|A - A_k\|_F$, where ϵ_k depends on the choice of k [10, 21], or it is given as an additive error $\epsilon \leq \|A - A_k\|_F + \epsilon_{k,A}$, where $\epsilon_{k,A}$ depends on k and $\|A\|_F$ [22, 44]. $\|A - A_k\|_F$ is often data dependent. In the case of NMF, a rigorous theoretical upper bound on ϵ remains unknown.

Given an approximation $\hat{A}$ of A, the next step is to reconstruct a sketched merge tree from each column vector $\hat{a}$ of $\hat{A}$. We reshape $\hat{a}$ into an $n \times n$ matrix $\hat{W}$ and construct a sketched tree $\hat{T}$ by computing the MST of $\hat{W}$. The distance matrix $\hat{D}$ of the sketched tree $\hat{T}$ thus approximates the distance matrix W' of the blow-up tree T'.

When a sketched merge tree is obtained via a MST, the theoretical bounds on $\|\hat{W} - \hat{D}\|_F$ are unknown, although MST does provide good sketched trees in practice. Finally, although the smoothing process does not alter the tree structure significantly, it does introduce some error in the final sketched tree, whose theoretical bound is not yet established.

A practical consideration is the *simplification* of a sketched tree $\hat{T}'$ coming from NMF. $\hat{T}'$ without simplification is an approximation of the blow-up tree T'. It contains many more nodes compared to the original tree T. Some of these are internal nodes with exactly one parent node and one child node. In some cases, the distance between two nodes is almost zero. We further simplify $\hat{T}'$ to obtain a final sketched tree $\hat{T}$ by removing internal nodes and nodes that are too close to each other; see Appendix B for details. To return a basis tree using NMF, we obtain each basis tree by applying MST to columns b_j of B with appropriate simplification, as illustrated in Fig. 1 (step 5).

Therefore, although we have obtained good experimental results in sketching merge trees, there is still a gap between theory and practice for individual sketched trees. Filling such a gap is left for future work.

B Implementation

In this section, we provide some implementation details for various algorithms employed in our merge tree sketching framework.

Persistence simplification. We apply *persistence simplification* to each dataset before computing the merge trees. The simplification level p is chosen based on the *persistence graph* [32], where the x-axis represents the persistence in proportion to the maximum persistence across all instances in a dataset, the y-axis captures the number of local maxima (in our setting), and a plateau implies a stable range of scales to separate features from noise. We simplify the scalar field so that critical points with persistence less than the chosen simplification level are removed. See Fig. 17 for the persistence graph of the *Heated Cylinder* dataset. p is chosen to be 9.7% of the max persistence. Similarly, p for the *Corner Flow*, *Vortex Street*, and *Square Cylinder Flow* dataset is 4.85%, 2%, 6%, respectively. For the *Isabel* dataset, we choose $p = 2\%$ to be consistent with the work by Pont et al. [57].

Initializing the coupling probability distribution. In Sec. 6, we introduce the blowup procedure that transforms a merge tree T to a larger tree T'. This procedure optimizes the probability of coupling between T and $\overline{T}$, the Fréchet mean. Since the optimization process is finding a coupling matrix that is a local minimum of the loss function, similar input trees may give different coupling matrices due to the optimization process, which may affect the ordering of nodes in the blown-up trees, leading to completely different vectorization results and large sketch errors. Specifically for time-varying data, to ensure that adjacent trees are initialized with similar coupling probabilities w.r.t. $\overline{T}$, we use the coupling probability between T_{i-1} and $\overline{T}$ to initialize the coupling probability between T_i and $\overline{T}$, for $1 \leq i \leq N-1$. This strategy is based on the assumption that merge trees from adjacent time instances share similar structures.

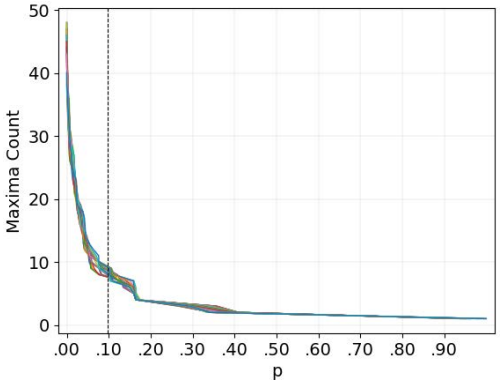

Fig. 17: *Heated Cylinder*: Persistence simplification using a persistence graph.

Matrix sketching algorithms. In the main paper, we use two variants of column subset selection (CSS) algorithms. We may also consider non-negative matrix factorization (NMF) to sketch the data matrix A. Here, we provide pseudocode for these matrix sketching algorithms.

- Modified Length Squared Sampling (LSS)
  1. $s \leftarrow 0$, B is an empty matrix, $A' = A$.
  2. $s \leftarrow s + 1$. Select column c from A' with the largest squared norm (or select c randomly proportional to the squared norm) and add it as a column to B. Remove c from A'.
  3. For each remaining column c' in A' (i.e., $c' \neq c$), factor out the component along c as:
     (a) $u \leftarrow c/\|c\|$
     (b) $c' \leftarrow c' - \langle u, c' \rangle u$
  4. While $s < k$, go to step 2.
- Iterative Feature Selection (IFS)
  1. Choose a subset of k column indices $r = \{i_1, i_2, \ldots, i_k\}$ uniformly at random.
  2. Construct subset $B_r = [a_{i_1}, a_{i_2}, \ldots, a_{i_k}]$ of A with columns indexed by r.
  3. Repeat for $j = 1, 2, \ldots, k$:
     (a) Let X_{jl} denote matrix formed by replacing column a_{i_j} with column a_l in B_r, where $l \in [n] \setminus r$. Let X_{jl}^+ denote its Moore-Penrose pseudoinverse.
     (b) Find $w = \text{argmin}_{l \in [n] \setminus r} \|A - X_{jl} X_{jl}^+ A\|_F$.
     (c) $B_r \leftarrow X_{jw}$.
     (d) $r \leftarrow (r \setminus \{i_j\}) \bigcup \{w\}$.
- Non-Negative Matrix Factorization (NMF)
  1. Given A and k, initialize $B \in \mathbb{R}^{d \times k}$, $Y = X^T \in \mathbb{R}^{k \times N}$ using the non-negative double singular value decomposition algorithm of Boutsidis and Gallopoulos [9].
  2. Normalize columns of B and X to unit L_2 norm. Let $E = A - BX^T$.
  3. Repeat until convergence: for $j = 1, 2, \ldots, k$,
     (a) $Q \leftarrow E + b_j x_j^T$.

(b) $x_j \leftarrow [Q^T b_j]_+$.
(c) $b_j \leftarrow [Q x_j]_+$.
(d) $b_j \leftarrow b_j / \|b_j\|$.
(e) $E \leftarrow Q - b_j x_j^T$.

Here, $[Q]_+$ means that all negative elements of the matrix Q are set to zero.

Merge tree simplification. To reconstruct a sketched tree, we reshape the sketched column vector $\hat{a}$ of $\hat{A}$ into an $n \times n$ matrix $\hat{W}'$, and obtain a tree structure $\hat{T}'$ by computing its MST. $\hat{T}'$ is an approximation of the blown-up tree T'. To get a tree approximation closer to the original input tree T, we further simplify $\hat{T}'$ as described below.

The simplification process has two parameters. The first parameter α is used to merge internal nodes that are too close ($\leq \alpha$) to each other. Let R be the diameter of $\hat{T}'$ and n the number of nodes in $\hat{T}'$. α is set to be $c_\alpha R/n^2$ for $c_\alpha \in \{0.5, 1, 2\}$. The second parameter $\beta = c_\beta R/n$ is used to merge leaf nodes that are too close ($\leq \beta$) to the parent node, where $c_\beta \in \{0.5, 1, 2\}$. Let $\hat{W}'$ be the weight matrix of $\hat{T}'$. The simplification process is as follows:

1. Remove from $\hat{T}'$ all edges (u, v) where $\hat{W}'(u, v) \leq \alpha$.
2. Merge all leaf nodes u with their respective parent node v if $\hat{W}'(u, v) \leq \beta$.
3. Remove all the internal nodes.

The tree $\hat{T}$ obtained after simplification is the final sketched tree.

Merge tree layout. To visualize both input merge trees and sketched merge trees, we experiment with a few strategies. To draw an input merge tree T equipped with a function defined on its nodes, $f : V \to \mathbb{R}$, we set each node $u \in V$ to be at location (x_u, y_u); where $y_u = f(u)$, and x_u is chosen within a bounding box while avoiding edge intersections. The edge (u, v) is drawn proportional to its weight $W(u, v) = |f(u) - f(v)| = |y_u - y_v|$.

To draw a sketched tree as a merge tree, we perform the following steps:

1. We fix the root of the sketched tree at $(0, 0)$.
2. The y-coordinate of each child node is determined by the weight of the edge between the node and its parent.
3. The x-coordinate is determined by the left-to-right ordering of the child nodes. We consider ordering the child nodes that share the same parent node by using a heuristic strategy described below.
   (a) Sort the child nodes by their size of the subtrees of which the child node is the root in ascending order. This sorting tries to keep larger subtrees on the right so the overall shape of the tree is protected and straightforward to read.
   (b) If the sizes of multiple subtrees are the same, we apply the following strategy: we sort child nodes by their distances to the parent node in descending order. Suppose the order of child nodes after sorting is $c_1, c_2, \ldots, c_t$. If t is odd, we reorder the nodes from left to right as $c_t, c_{t-2}, c_{t-4}, \ldots, c_3, c_1, c_2, c_4, \ldots, c_{t-3}, c_{t-1}$. If t is even, we reorder the nodes as $c_{t-1}, c_{t-3}, c_{t-5}, \ldots, c_3, c_1, c_2, c_4, \ldots, c_{t-2}, c_t$.

The idea is to keep the child nodes that have a larger distance to the parent near the center to avoid edge crossings between sibling nodes and their subtrees.

Our layout strategy assumes that the trees are rooted. However, $\hat{T}$, which is our approximation of T, is not rooted. In our experiments, we use two strategies to pick a root for $\hat{T}$ and align T and $\hat{T}$ for visual comparison.

Using the **balanced layout** strategy, we pick the node u of $\hat{T}$ that minimizes the sum of distances to all other nodes. Set u to be the *balanced root* of $\hat{T}$. Similarly, we find the balanced root v of the input tree T. T and $\hat{T}$ are drawn using the balanced roots.

Using the **root alignment** strategy, we obtain the root node of the sketched tree by keeping track of the root node during the entire sketching process. We can get the root node of T' because it is either a duplicate node or the same node of the root node in T. Then we can get the root node in $\hat{T}'$, as the labels in the sketched blown-up tree are identical to T'. Lastly, by keeping track of the process of merge tree simplification, we can know the label of the root of $\hat{T}$.

Other details. Our framework is mainly implemented in Python. The code to compute MST from a given weight matrix is implemented in Java. For data processing and merge tree visualization, we use Python packages, including numpy, matplotlib, and networkx. In addition, the GW framework of Chowdhury and Needham [18] requires the Python Optimal Transport (POT) package.

C Additional Results and Runtime Analysis

C.1 Vortex Street Dataset

We demonstrate the use of our framework in detecting cyclic behaviors using the classic time-varying 2D von Kárman *Vortex Street* dataset. We use the velocity magnitude field (as used in a previous work [60]) and compute its split tree. There are 157 time instances, which give rise to a set of merge trees.

Coefficient matrix. The coefficient matrix generated with IFS is shown in Fig. 18 (Top) using $k = 3$. We observe a periodicity of $36 \sim 38$ time steps. To show this periodicity, we highlight instances 14, 50, 88, and 125 (blue boxes) in the coefficient matrix. We can see that all these instances indicate the starting points of four long yellow blocks on the second row of the coefficient matrix.

We further compare the corresponding scalar fields (instances 14, 50, and 88) highlighted with local maxima (red points) and saddles (white points) in Fig. 18 (bottom left). The scalar field visualization indicates that instances 14 and 88 have nearly identical structures, whereas instance 50 appears to be a mirror image of instance 14. The relation between instances 14 and 50 is not surprising, as our measure network formulation of the merge tree encodes only its intrinsic information, and thus the GW distance between T_{14} and T_{50} is considered to be near-zero within our sketching framework. For comparison, there is a clear structural change between instance 14 and 33 in Fig. 18 (bottom left), where instance 33 is chosen within a particular period.

GW distance matrix. The observed periodicity is further confirmed with the GW distance matrix, as shown in Fig. 18 (bottom right). We compute pairwise GW distances between pairs of instances and obtain a 157×157 matrix, where yellow means high and blue means low distance values. Similar to our observations with the coefficient matrix, we clearly see a repeating pattern in the GW distance matrix with a periodicity of $36 \sim 38$.

C.2 Square Cylinder Flow

For the *Square Cylinder Flow* dataset, we provide additional experimental results on the periodicity observed from the GW distance matrix.

The observed periodicity can be further validated by the GW distance matrix. As shown in Fig. 19, we zoom into this chosen period and observe repeated patterns in the distance matrix, which indicates structural similarities among these merge trees.

C.3 Sketch Error Plots for Alternative Approaches

We include the sketch error plots for sketching scalar fields and sketching 0-dimensional persistence images in Fig. 20 for completeness.

C.4 Runtime Analysis

We report the runtime in computing the pairwise Gromov-Wasserstein distances between merge trees for all real-world datasets in Tab. 1. The number of comparisons is equal to $\frac{t \times (t-1)}{2}$, where t is the number of time steps. Average runtime is equal to $\frac{total\ runtime}{\#\ of\ comparisons}$. All these distances are easy and efficient to compute.

The runtime was collected using Python hosted by Jupyter Notebook on a Windows 11 system with a 12th Gen Intel(R) Core(TM) i9-12900H 2.50 GHz CPU with 32 GB memory.

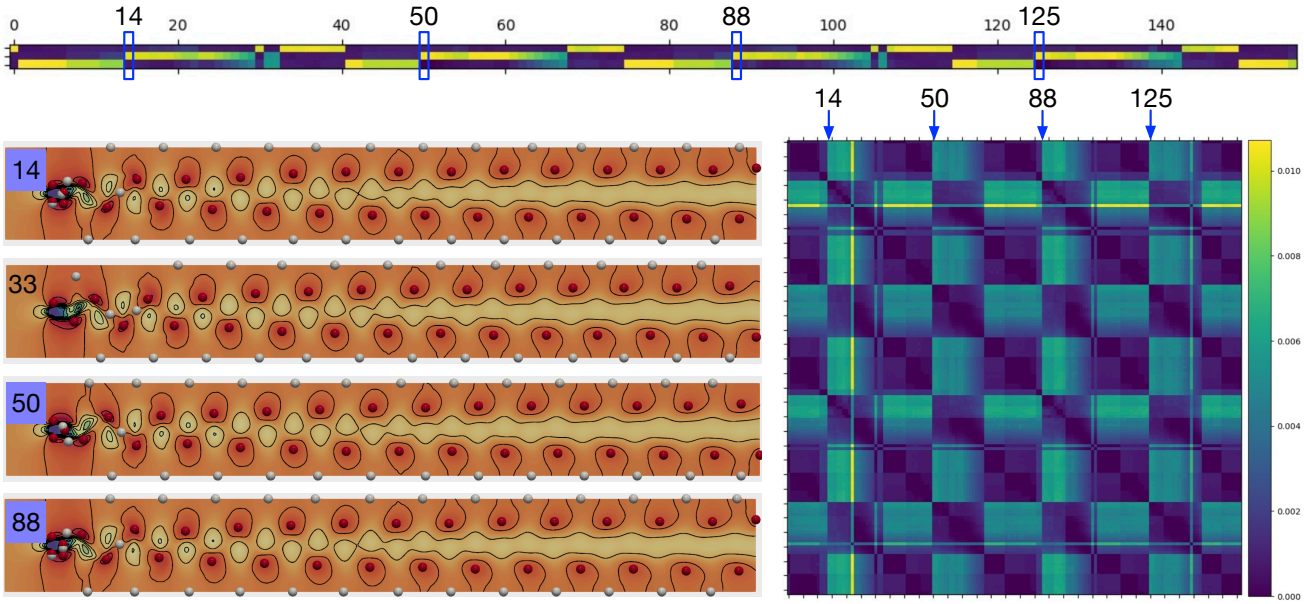

Fig. 18: *Vortex Street*: we highlight instance 14, 50, 88, and 125 in the coefficient matrix (TOP) to illustrate the periodicity of its topological structures, using IFS. The periodicity is also confirmed within the GW distance matrix (bottom right). Visualizations of selected scalar fields are shown in the Bottom Left.

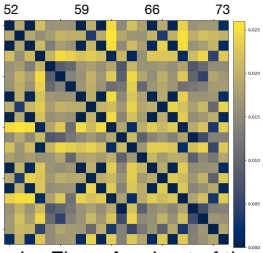

Fig. 19: *Square Cylinder Flow*: A subset of the pairwise GW distance matrix among T_{52} to T_{73}.

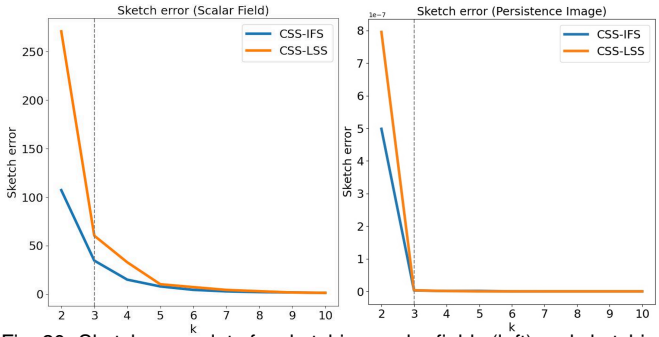

Fig. 20: Sketch error plots for sketching scalar fields (left) and sketching 0-dimensional persistence images (right).

Dataset	Max # of Nodes	# of Comparisons	Total Runtime	Average Runtime
HeatedCylinder (2D)	18	465	1.2967	0.0028
CornerFlow (2D)	30	4950	21.1775	0.0043
VortexStreet (2D)	62	12246	303.5248	0.0248
SquareCylinderFlow (2D)	18	5050	12.1553	0.0024
Isabel (3D)	194	66	4.2124	0.0638

Table 1: Runtime (in seconds) for pairwise GW distances between merge trees across all real-world datasets.

D Discussions on Sketched Trees and NMF

In this paper, we apply matrix sketching to a set of merge trees and utilize the basis set as a consensus set that captures the modes from the underlying phenomena. As a byproduct, the sketching framework also produces sketched trees. In this section, we discuss the sketched trees, as well as additional matrix sketching techniques beyond column subset selection.

Investigation of sketched trees. We validate the claim that given three basis trees in $\mathcal{S}$, each tree in $\mathcal{T}$ can be approximately reconstructed from a linear combination of trees in $\mathcal{S}$. For the *Heated Cylinder* dataset, we compare a subset of input trees (blue boxes) against their sketched versions (red boxes) using IFS in Fig. 21. Even though we use only three basis trees, a large number of input trees—such as T_7, T_{15}—and their sketched versions are indistinguishable with small errors. Even though T_{24} is considered an outlier relative to other input trees, its sketched version does not deviate significantly from the original tree. We highlight the subtrees with noticeable structural differences before and after sketching for T_{24}, whose roots are pointed by black arrows.

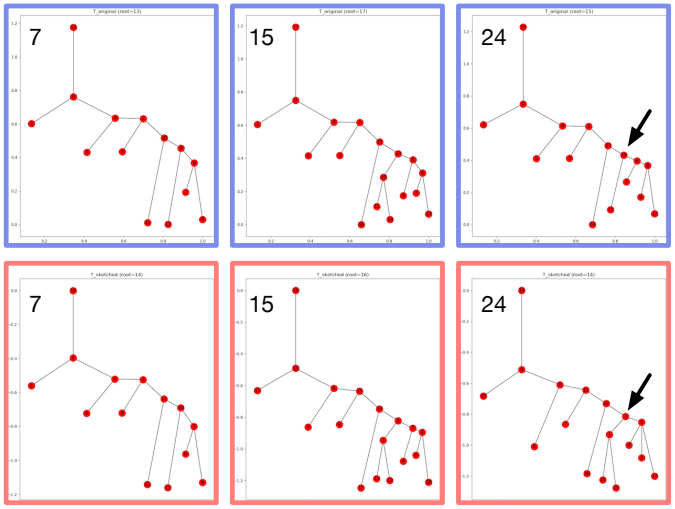

Fig. 21: *Heated Cylinder*: Comparing each sketched tree (red box) with its corresponding input tree (blue box), highlighting noticeable structural differences among subtrees (whose roots are pointed by black arrows) before and after sketching.

In Fig. 22, we further investigate the weight matrices from different stages of the sketching pipeline for tree $T = T_{24}$. From left to right, we show the weight matrix W of the input tree, its blow-up matrix W' (which is linearized to a column vector a), the approximated column vector $\hat{a}$ after sketching (reshaped into a square matrix), the weight matrix $\hat{W}'$ of the MST derived from the reshaped $\hat{a}$, the weight matrix

of the MST after simplification, and root alignment $\hat{W}$ w.r.t. T. We observe minor changes between W (blue box) and $\hat{W}$ (red box), which explain the structural differences before and after sketching in Fig. 21.

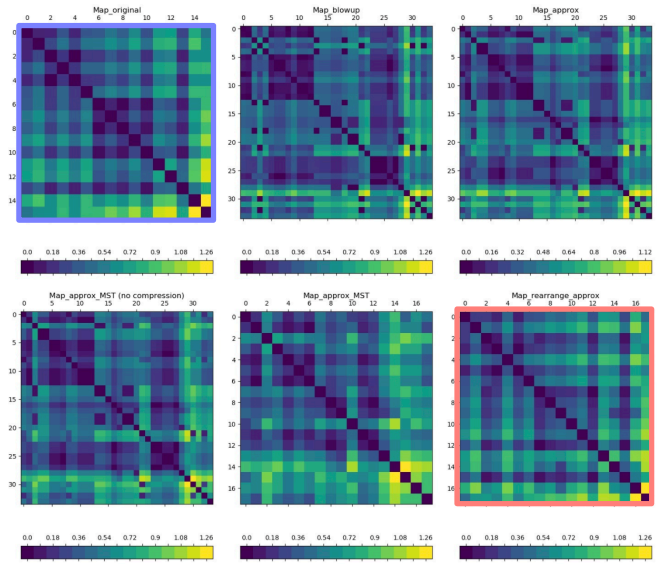

Fig. 22: *Heated Cylinder*: weight matrices associated with T_{24} during the sketching process with IFS.

Sketching with NMF. For the *Heated Cylinder* dataset, we also discuss the sketching results using matrix sketching techniques beyond CSS such as non-negative matrix factorization (NMF). In NMF, the goal is to compute non-negative matrices B and Y such that $\|A - \hat{A}\|_F = \|A - BY\|_F$ is minimized. We use the implementation provided in the decomposition module of the scikit-learn package [19, 30]. The algorithm initializes matrices B and $X = Y^T$ and minimizes the residual $Q = A - BX^T + b_j x_j^T$ alternately with respect to column vectors b_j and x_j of B and X, respectively, subject to the constraints $b_j \geq 0$ and $x_j \geq 0$.

Using NMF, we show the three basis trees together with a coefficient matrix in Fig. 23. Although these basis trees are generated by matrix factorization, that is, they do not correspond to any input trees, they nicely pick up the structural variations in the input and are shown to resemble the basis trees chosen by column selections (cf., Fig. 7 and Fig. 9). This observation shows that even though these matrix sketching techniques employ different (randomized) algorithms, they all give rise to reasonable choices of basis trees, which leads to reasonable sketching results.

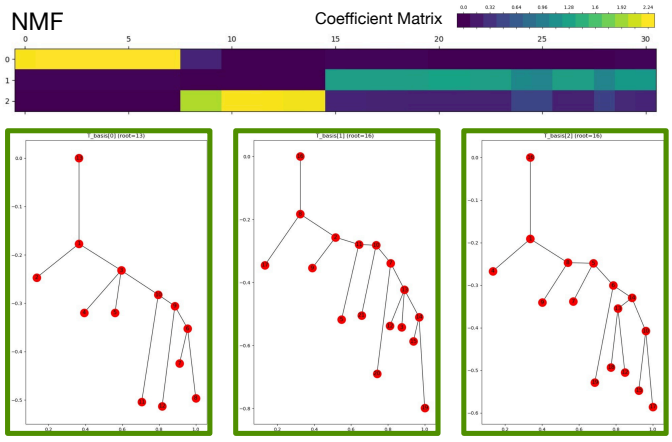

Fig. 23: *Heated Cylinder*: Coefficient matrices and basis trees used to sketch the dataset with three basis trees using NMF.

Potential applications. Although neither the sketched trees nor NMF are the main focus in studying the time-varying datasets discussed in this paper, we may consider other potential applications. For instance, our framework could be applied to a set of plant root systems (e.g., `https://roots.ornl.gov/`), where each plant root may be digitalized and modeled as merge trees. Our framework may be used to characterize different root classes where NMF can be used to obtain new representatives from the set that are not part of the original input. Furthermore, the distance between a sketched tree and the original tree captures how much a particular input tree could be approximated by the basis set. This is left for future work.

E Comparison with Wasserstein Distances

In this section, we perform experimental comparisons with Wasserstein distance for merge trees [57] by Pont et al. We visualize the distance matrices using our GW distance and the Wasserstein distance to compare their performances in identifying topological similarities and differences. We use the *HeatedCylinder* and *Isabel* datasets. For fair comparisons, both methods use only split trees (describing maxima and saddle relations).

HeatedCylinder dataset. Even though the topological changes in the *HeatedCylinder* dataset are relatively simple, we observe obvious differences between two pairwise distance matrices.

One noticeable difference is the similarity between time steps 8 and 9 (in cyan boxes) and their adjacent time steps (in green and orange boxes, respectively); see Figure 24 (top left and top middle). The GW distance indicates that time steps 8 and 9 are similar to time step 10 and obviously different from time step 7, whereas the Wasserstein distance reaches the opposite conclusion. We visualize the merge tree structures for time steps 7 to 10, as in the green, cyan, and orange boxes on the top right of Figure 24. Apparently, the merge tree structures among time steps 8, 9, and 10 are similar, whereas the merge tree at time step 7 has one fewer branch. In other words, the Wasserstein distance fails to detect the topological change from time step 7 to 8, and in the following raises a false-positive change from time step 9 to 10.

We elaborate on such a "false-positive" from the Wasserstein distance using another example. In the pairwise distance matrix for Wasserstein distance (see Figure 24 top middle), the red box highlights a topological feature change from time steps 17 to 18. However, this transition is not detected in the GW distance matrix (Figure 24 top left). We now use the branch decomposition layout to visualize the merge tree at time steps 17 and 18 in Figure 24 (middle right, red solid box). In the merge trees at both time steps, we use blue, purple, and pink to highlight three branches; branches in the same color indicate the same pair of critical points in the scalar field. In the transition from time steps 17 to 18, the branch decomposition hierarchy of the three highlighted branches is largely shifted, which is detected by the Wasserstein distance as topological changes. However, in the binary tree layout of these two merge trees, we see that the merge tree structure is almost unchanged, see Figure 24 (middle right, red dotted box), indicating that the detected topological change is false-positive. In this example, we show that even though there are only small function value perturbations across time steps, the branch decomposition result can change greatly. Therefore, involving branch decomposition in computation, the Wasserstein distance for merge trees suffers from instability of branch decomposition from small-scale perturbations, which is already mentioned by other works [68]. In comparison, our method is more robust than the Wasserstein distance against such instabilities.

Isabel dataset. Recall that the time steps for the *Isabel* dataset can be clustered into three phases of the hurricane, each of which contains four consecutive time steps. As shown in Figure 24 (bottom left), we use magenta-dotted boxes to identify time instances in the same phase as the ground truth for clustering. The GW distance successfully captures the topological variations across three known phases of the Hurricane Isabel.

In contrast, the Wasserstein distance fails to distinguish the topological variation between the formation (first) and the drift (second) phase of the hurricane: there is no block structure transitioning between time steps 5 and 30. Besides, in the MDS plot visualizing the Wasserstein

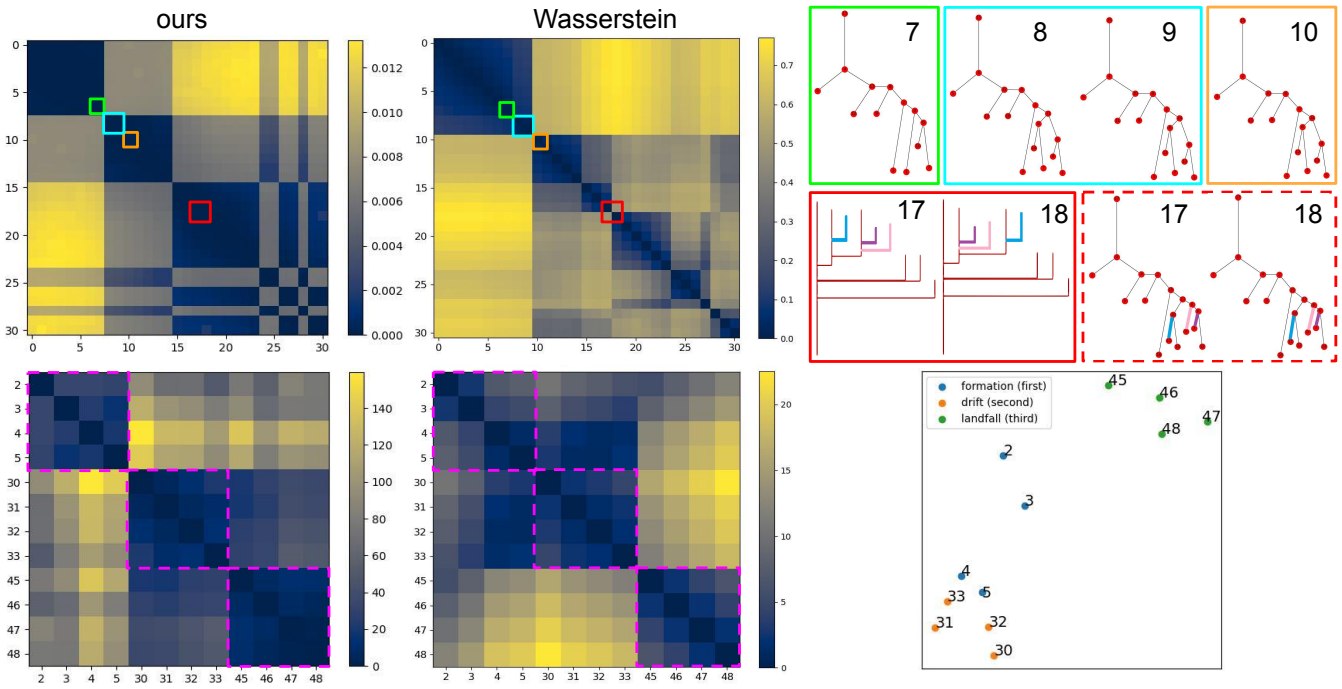

Fig. 24: Experimental comparison with Wasserstein distance for merge trees. Top row: *HeatedCylinder* dataset; left and middle: pairwise distance matrix for GW distance and Wasserstein distance, respectively; right: merge tree in binary tree layout at time steps 7 to 10, and merge tree in both branch decomposition layout and in binary tree layout at time steps 17 and 18. Bottom row: *Isabel* dataset; left and middle: pairwise distance matrix for GW distance and Wasserstein distance, respectively; right: MDS scatter plot [64] using the Wasserstein distance, in which colors represent the phase of time instances.

distance (Figure 24 bottom right), time steps 4 and 5 in the formation phase of the hurricane are closer to the cluster of time steps 30 to 33 in the drift phase than to time steps 2 and 3 in the formation phase. The result shows that the Wasserstein distance cannot detect the phase transition between the formation and the drift phase of the hurricane. On the other hand, the Wasserstein distance does provide a clearer distinction in the landfall (third) phase compared with the GW distance.