Sketching Merge Trees for Scientific Visualization

Mingzhe Li, Sourabh Palande, Lin Yan, and Bei Wang

Abstract— Merge trees are a type of topological descriptors that record the connectivity among the sublevel sets of scalar fields. They are among the most widely used topological tools in visualization. In this paper, we are interested in sketching a set of merge trees using techniques from matrix sketching. That is, given a large set $\mathcal{T}$ of merge trees, we would like to find a much smaller set of basis trees $\mathcal{S}$ such that each tree in $\mathcal{T}$ can be approximately reconstructed from a linear combination of merge trees in $\mathcal{S}$. A set of high-dimensional vectors can be approximated via matrix sketching techniques such as principal component analysis and column subset selection. However, until now, there has not been any work on sketching a set of merge trees. We develop a framework for sketching a set of merge trees that combines matrix sketching with tools from optimal transport. In particular, we vectorize a set of merge trees into high-dimensional vectors while preserving their structures and structural relations. We demonstrate the applications of our framework in sketching merge trees that arise from time-varying scientific simulations. Specifically, our framework obtains a set of basis trees as representatives that capture the "modes" of physical phenomena for downstream analysis and visualization.

Index Terms—Merge trees, matrix sketching, topology in visualization, ensemble analysis

1 Introduction

Topological descriptors such as merge trees, contour trees, Reeb graphs, and Morse–Smale complexes serve to describe and identify characteristics associated with scalar fields, with many applications in the analysis and visualization of scientific data (e.g., see the surveys [37, 42]). Matrix sketching [69], on the other hand, is a class of mathematical tools that approximate a large data matrix with smaller and sparser matrices [24]. Principal component analysis (PCA) [54], for example, is a type of matrix sketching. We are interested in applying matrix sketching techniques to a set of topological descriptors, specifically merge trees, for scientific visualization.

We formulate our problem as follows: given a large set $\mathcal{T}$ of merge trees, we would like to find a much smaller set of basis trees $\mathcal{S}$ such that each tree in $\mathcal{T}$ can be approximately reconstructed from a linear combination of trees in $\mathcal{S}$. The set $\mathcal{T}$ may arise from a time-varying field or an ensemble of scientific simulations generated with varying parameters and/or different instruments. We aim to develop a merge tree sketching framework that:

- Identifies good representatives that capture topological variations in a set of merge trees as well as outliers; and
- Obtains a compressed representation of a large set of merge trees as a much smaller set of basis trees together with a coefficient matrix for downstream analysis and visualization.

A sketch of $\mathcal{T}$ with $\mathcal{S}$ gives rise to a significantly smaller representation of $\mathcal{T}$. Elements in $\mathcal{S}$ will serve as good representatives of $\mathcal{T}$, whereas elements with large sketching errors will be considered as outliers.

The ability to extract a basis set of merge trees is important for numerous applications, for which scientists are interested in detecting the "modes" of physical phenomena. This extraction could be achieved by computing a basis set, matching merge trees to the basis set, and computing the errors of each input tree w.r.t. that basis set. We could potentially uncover repeated phenomena that provide deep phenomenological insight. Our framework could recover cyclical phenomena for time-varying data or derive consensus sets for ensembles. Our contributions are:

- We combine tools from optimal transport with matrix sketching techniques to give a class of algorithms for sketching a set of merge trees. This is the first time matrix sketching is applied to a set of topological descriptors.
- We introduce a new distance between merge trees by adapting the Gromov-Wasserstein distance [18, 47, 55] in optimal transport.
- We provide experimental results that demonstrate the utility of our framework in sketching merge trees that arise from scientific simulations. Specifically, we show that understanding how a set of merge trees is approximated by a smaller set can be particularly useful for the study of time-varying scalar fields and ensembles, where our framework can be used to obtain compact representations for downstream analysis and visualization. The basis set extracted from matrix sketching can serve as good representatives in detecting the modes of physical phenomena.

Our framework offers an exciting direction of utilizing randomized linear algebra for topological descriptors in visualization.

2 An Overview and a Primer on Matrix Sketching

Data sketching is powerful in the analysis of massive datasets [44] and has enjoyed diverse and exciting advances in recent years. A *sketch* is a compressed mapping of the full dataset onto a smaller data structure that serves as a summary that retains certain properties of interest. A sketch is typically "easy to update with new or changed data and allows certain queries whose results approximate queries on the full dataset." [56]

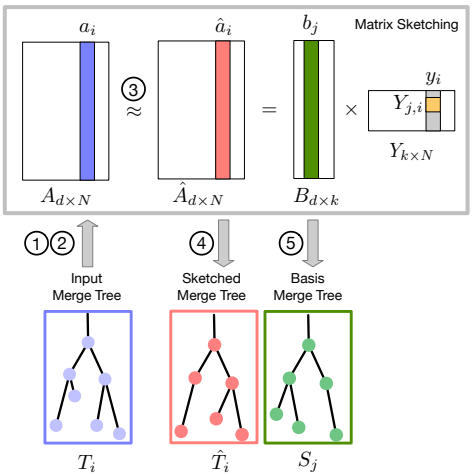

Fig. 1: The overall pipeline for sketching a set of merge trees.

- *Mingzhe Li is with the University of Utah. E-mail: mingzhel@sci.utah.edu.*
- *Lin Yan is with the Argonne National Laboratory. E-mail: lyan@anl.gov.*
- *Sourabh Palande is with the Michigan State University. E-mail: sourabh.palande@gmail.com.*
- *Bei Wang is with the University of Utah. E-mail: beiwang@sci.utah.edu.*

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

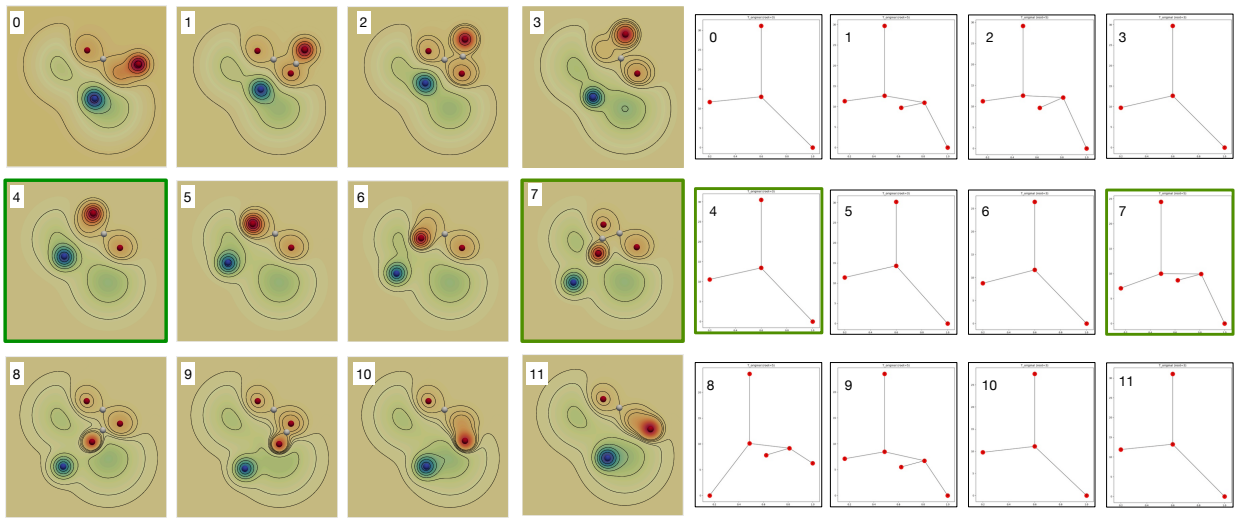

Fig. 2: *Rotating Gaussian*: Visualizing a time-varying mixture of Gaussian functions (left) together with their corresponding merge trees (right). Merge trees at time step 4 and 7 are selected as the representatives to describe the topology of the ensemble.

We are inspired by the idea of matrix sketching. A sketch of a matrix A is another matrix B that is significantly smaller than A, but still approximates it well [41]. Many matrix sketching techniques build upon numerical linear algebra and vector sketching. A set of high-dimensional vectors is *sketchable* via matrix sketching techniques such as principle component analysis (PCA) and column subset selection (CSS), as illustrated in Fig. 1 (gray box).

Given a dataset of N points with d features, represented as a $d \times N$ matrix A (with row-wise zero empirical mean), together with a parameter k, PCA aims to find a k-dimensional subspace $\mathbb{H}$ of $\mathbb{R}^d$ that minimizes the average squared distance between the points and their corresponding projections onto $\mathbb{H}$. For every column vector a_i of A, PCA finds a k-dimensional embedding y_i (a column vector of Y) along the subspace $\mathbb{H}$ to minimize $||A - \hat{A}||_F^2 = ||A - BY||_F^2$. B is a $d \times k$ matrix whose columns $b_1, b_2, \ldots, b_k$ form an orthonormal basis for $\mathbb{H}$. Y is a $k \times N$ coefficient matrix, whose column y_i encodes the coefficients for approximating a_i using the basis from B. That is, $a_i \approx \hat{a}_i = \sum_{j=1}^{k} b_j Y_{j,i}$.

Another technique we discuss is CSS, whose goal is to find a small subset of the columns in A to form B such that the projection error of A to the span of the chosen columns is minimized, that is, to minimize $||A-\hat{A}||_F^2 = ||A-BY||_F^2$, where we restrict B to come from columns of A. Such a restriction is important for data summarization, feature selection, and interpretable dimensionality reduction [8]. Thus, with PCA or CSS, given a set of high-dimensional vectors, we could find a set of basis vectors such that each input vector can be approximately reconstructed from a linear combination of the basis vectors.

Now, what if we replace a set of high-dimensional vectors by a set of objects that encode topological information of data, specifically topological descriptors? Until now, there has not been any work on sketching a set of merge trees. In this paper, we focus on merge trees, which are a type of topological descriptors that record the connectivity among the sublevel sets of scalar fields. We address the following question: Given a large set $\mathcal{T}$ of merge trees, can we find a much smaller basis set $\mathcal{S}$ as its "sketch"?

Our overall pipeline is illustrated in Fig. 1 and detailed in Sec. 6. In steps 1 and 2, given a set of N merge trees $\mathcal{T} = \{T_1, T_2, \cdots, T_N\}$ as input, we represent each merge tree T_i as a measure network and employ the Gromov-Wasserstein framework of Chowdhury and Needham [18] to map it to a column vector a_i in the data matrix A. In step 3, we apply matrix sketching techniques, in particular, column subset selection (CSS), to obtain an approximated matrix $\hat{A}$, where $A \approx \hat{A} = B \times Y$. In step 4, we convert each column in $\hat{A}$ into a merge tree (referred to as a sketched merge tree) using minimum spanning trees (MST). Finally, in step 5, we return a set of basis merge trees $\mathcal{S}$ by applying MST to each column b_j in B. Each entry $Y_{j,i}$ in the coefficient matrix Y defines the coefficient for basis tree S_j in approximating T_i. With the above pipeline, given a set of merge trees, we could find a set of basis trees such that each input tree can be approximately reconstructed from a linear combination of the basis trees.

3 A Simple Motivational Example

Before we dive into the technical details of our approach, we give a motivational example. A time-varying scalar field is generated as a mixture of 2D Gaussian functions that translate and rotate on the plane. We sample 12 scalar fields $\{f_0, \ldots, f_{11}\}$ across consecutive time steps, referred to as the *Rotating Gaussian* dataset, which gives rise to a set of merge trees $\mathcal{T} = \{T_0, \ldots, T_{11}\}$, as shown in Fig. 2. Each merge tree T_i is computed from $-f_i$; thus, its leaves correspond to the local maxima (red), internal nodes are saddles (white), and the root is the global minimum (blue).

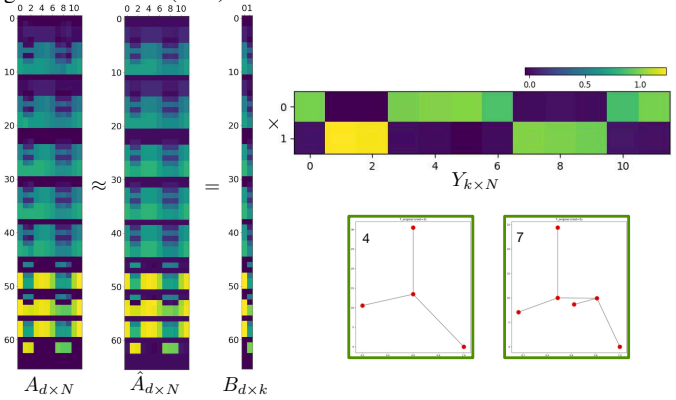

Fig. 3: *Rotating Gaussian*: Visualizing data matrices associated with the sketching, and the coefficient matrix.

We now apply matrix sketching to $\mathcal{T}$ using our pipeline described in Fig. 1. Since the dataset is quite simple, a couple of basis trees are sufficient to obtain very good sketching results. Using $k = 2$, we employ Iterative Feature Selection (IFS) – a type of column subset selection algorithm – from the matrix sketching toolbox. The algorithm produces a set of two basis trees, $\mathcal{S} = \{T_4, T_7\}$, which are highlighted with green boxes together with their corresponding scalar fields in Fig. 2. The topological structures of these two basis trees are noticeably distinct among the input trees. They clearly serve as good representatives of the entire set $\mathcal{T}$ and capture the structural variations.

We further visualize the data matrix A, $\hat{A}$, B, and highlight the coefficient matrix Y in Fig. 3 (cf. the gray box in Fig. 1). Y shows that each input tree (column) is well represented (with high coefficient) by one of the two basis trees. In particular, columns in the coefficient matrix with high (yellow or light green) coefficients (w.r.t. the given basis) may be

grouped together, forming two clusters $\{T_0, T_3, T_4, T_5, T_6, T_{10}, T_{11}\}$ and $\{T_1, T_2, T_7, T_8, T_9\}$ whose elements look structurally similar.

4 Related Work

Comparing merge trees. Merge trees record the connectivity among the sublevel sets of scalar fields (e.g., [6, 14]). They are rooted in Morse theory [52], which characterizes scalar field data by the topological changes in its sublevel sets at isolated critical points. A number of recent works focus on comparing merge trees and their variants (e.g., [6, 31, 57, 60, 67, 68]); see [72] for a survey. Recently, Pont et al. proposed a Wasserstein distance between merge trees [57] that equals to the L^2-Wasserstein distance between persistence diagrams. Wetzels et al. proposed variants of edit distances [67, 68] that are independent from branch decomposition trees.

In this paper, we treat merge trees as measure networks and introduce a Gromov-Wasserstein (GW) distance between merge trees based on optimal transport (Sec. 5). See Appendix E for a comparison with the Wasserstein distance [57]. Different from previous distances between merge trees, the GW distance is easy and efficient to compute, and provides explicit structural correspondences between the trees (Sec. 6). Our main focus is to use the GW distance to obtain alignments and vector representations of merge trees that interface with matrix sketching.

Gromov-Wasserstein distances. Gromov introduced Gromov-Hausdorff (GH) distances [34] while presenting a systematic treatment of metric invariants for Riemannian manifolds. GH distances can be employed as a tool for shape matching and comparison (e.g., [11, 45, 46, 49, 50]), where shapes are treated as metric spaces, and two shapes are considered equal if they are isometric. Memoli [47] modified the formulation of GH distances by introducing a relaxed notion of proximity between objects, thus generalizing GH distances to the notion of Gromov-Wasserstein (GW) distances for practical considerations. Since then, GW distances have had a number of variants based on optimal transport [62, 63] and measure-preserving mappings [48]. Apart from theoretical explorations [47, 61], GW distances have been utilized in the study of graphs and networks [38, 70, 71], machine learning [12, 28], and word embeddings [4]. Recently, Memoli et al. [51] considered the problem of approximating metric spaces using GW distance. Their goal was to approximate a (single) metric measure space modeling the underlying data by a smaller metric measure space. The work presented in this paper instead focuses on approximating a large set of merge trees – modeled as a set of metric measure networks – with a much smaller set of merge trees.

Aligning and averaging graphs. Graph alignment or graph matching is a key ingredient in performing comparisons and statistical analysis on the space of graphs (e.g., [27, 35]). It is often needed to establish node correspondences between graphs of different sizes. Edit distances have been used to align contour trees [43]. The approaches that are most relevant here are the ones based on the GW distances [18, 55], which employ *probabilistic matching* ("soft matching") of nodes. Information in a graph can be captured by a symmetric positive semidefinite matrix that encodes distances or similarities between pairs of nodes. Dryden et al. [25] described a way to perform statistical analysis and to compute the mean of such matrices. Agueh et al. [3] considered barycenters of several probability measures, whereas Cuturi et al. [20] and Benamou et al. [7] developed efficient algorithms to compute such barycenters. Peyre et al. [55] combined these ideas with the notion of GW distances [47] to develop GW averaging of distance/similarity matrices. Chowdhury and Needham [18] built upon the work in [55] and provided a GW framework to compute a Frechét mean among these matrices using measure couplings. In this paper, we utilize the GW framework [18] for probabilistic matching among merge trees.

Vectorizing topological descriptors. A number of recent works transform topological descriptors from data into feature vectors to be used as input to machine learning models; see [39] for a survey. A primary focus is on vectorizing persistence diagrams. Adams et al. introduced *persistence images* [2] that transform persistence diagrams into 2D images for classification tasks. Carrière et al. [16] used mappings between points in persistence diagrams to construct vector representations. A neural network layer was also used to embed persistence diagrams in vector spaces [15].

Our framework generates vectorized representations of merge trees using optimal transport to be interfaced with matrix sketching. Different from previous work, the vectorized merge trees preserve structural correspondences and there exist explicit mappings between pairs of merge trees. In addition, we can reconstruct input merge trees from basis trees, that is, we can reverse engineer merge trees from their vector representations. A number of previous works also utilize the latent representations of inputs from neuron networks as high-dimensional vector representations. However, these approaches often require extensive training and often do not generalize well across diverse datasets. In comparison, our approach is generalizable and does not require training.

Matrix sketching. Many matrix sketching techniques [56, 69] build upon linear algebra and vector sketching. For simplicity, we formulate the problem as follows: Given a $d \times N$ matrix A, we would like to approximate A using fewer columns, as a $d \times k$ matrix B such that A and B are considered to be *close* with respect to some problem of interest. Basic approaches for matrix sketching include truncated singular value decomposition (SVD), column or row sampling [22, 23], random projection [59], and frequent directions [33, 41]; see [56, 69] for surveys.

The column sampling approach carefully chooses a subset of the columns of A proportional to their *importance*, where the importance is determined by the squared norm (e.g., [22]) or the (approximated) leverage scores (e.g., [23]). The random projection approach takes advantage of the Johnson-Lindenstrauss (JL) Lemma [40] to create an $N \times k$ linear projection matrix S (e.g., [59]), where $B = AS$. The frequent directions approach [33, 41] focuses on replicating properties of the SVD. The algorithm processes each column of A at a time while maintaining the best rank-k approximation as the sketch.

5 Technical Background

We begin by reviewing the notion of a merge tree that arises from a scalar field. We then introduce the technical background needed to vectorize a merge tree as a column vector in the data matrix.

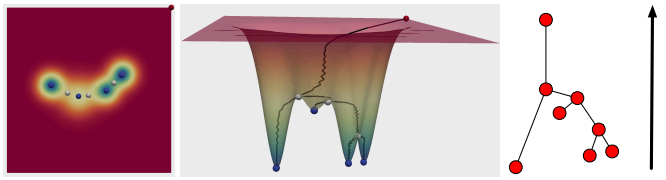

Fig. 4: An example of a merge tree from a height function. From left to right: 2D scalar field visualization, a merge tree embedded in the graph of the scalar field, and an abstract visualization of a merge tree as a rooted tree equipped with a height function.

Merge trees. Let $f : \mathbb{M} \to \mathbb{R}$ be a scalar field defined on the domain of interest $\mathbb{M}$, where $\mathbb{M}$ is a subset of $\mathbb{R}^2$ in our context. Merge trees capture the connectivity among the *sublevel sets* of f, i.e., $\mathbb{M}_a = f^{-1}(-\infty, a]$. Two points $x, y \in \mathbb{M}$ are *equivalent*, denoted by $x \sim y$, if $f(x) = f(y) = a$, and x and y belong to the same connected component of a sublevel set $\mathbb{M}_a$. The *merge tree*, $T(\mathbb{M}, f) = \mathbb{M}/\sim$, is the quotient space obtained by gluing together points in $\mathbb{M}$ that are equivalent under the relation $\sim$.

To construct a merge tree, we sweep the function value a from $-\infty$ to ∞, and create a new branch originating at a leaf node for each local minimum of f. As a increases, such a branch is extended as its corresponding component in $\mathbb{M}_a$ grows until it merges with another branch at a saddle point. If $\mathbb{M}$ is connected, all branches eventually merge into a single component at the global maximum of f, which corresponds to the root of the tree. For a given merge tree, leaves, internal nodes, and root node represent the minima, merging saddles, and global maximum of f, respectively. Fig. 4 displays an example. Abstractly, a merge tree T is a rooted tree equipped with a scalar function defined on its node set, $f : V \to \mathbb{R}$.

Gromov-Wasserstein distance for measure networks. Our framework utilizes tools from optimal transport, specifically, the GW distance

between measure networks. The GW distance was proposed by Memoli [46,47] for metric measure spaces. Peyre et al. [55] introduced the notion of a *measure network* and defined the GW distance between such networks. The key idea is to find a *probabilistic matching* between a pair of networks by searching over the convex set of couplings of the probability measures defined on the networks.

In our context, a finite merge tree T can be represented as a measure network using a triple (V, p, W), where V is the set of n nodes, p is a probability measure on V, and W is an $n \times n$ matrix capturing the relations between pairs of nodes. For our experiments, p is taken to be uniform, that is, $p = \frac{1}{n}\mathbf{1}_n$, where $\mathbf{1}_n = (1, 1, \ldots, 1)^T \in \mathbb{R}^n$. W may encode adjacency or shortest path relations (see Sec. 6).

Let $T_1(V_1, p_1, W_1)$ and $T_2(V_2, p_2, W_2)$ be a pair of merge trees with n_1 and n_2 nodes, respectively. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$. $V_1 = \{x_i\}_{i\in[n_1]}$ and $V_2 = \{y_j\}_{j\in[n_2]}$. A *coupling* between probability measures p_1 and p_2 is a joint probability measure on $V_1 \times V_2$ whose marginals agree with p_1 and p_2. That is, a coupling is represented as an $n_1 \times n_2$ non-negative matrix C such that each row sums up to $1/n_1$ and each column sums up to $1/n_2$. The *distortion* of a coupling C with an arbitrary loss function L is defined as [55]

$$\mathcal{E}(C) = \sum_{i,k\in[n_1],j,l\in[n_2]} L(W_1(i,k), W_2(j,l))C_{i,j}C_{k,l}. \quad (1)$$

Let $\mathcal{C} = \mathcal{C}(p_1, p_2)$ denote the collection of all couplings between p_1 and p_2. The *Gromov-Wasserstein discrepancy* [55] is defined as

$$\mathcal{D}(C) = \min_{C\in\mathcal{C}} \mathcal{E}(C). \quad (2)$$

In this paper, we consider the quadratic loss function $L(a, b) = \frac{1}{2}|a-b|^2$. The *Gromov-Wasserstein distance* [18,47,55] d_{GW} between T_1 and T_2 is defined as

$$d_{GW}(T_1, T_2) = \frac{1}{2}\min_{C\in\mathcal{C}} \sum_{i,k\in[n_1],j,l\in[n_2]} |W_1(i,k) - W_2(j,l)|^2 C_{i,j}C_{k,l}. \quad (3)$$

It follows from the work of Sturm [61] that such minimizers always exist and are referred to as *optimal couplings*.

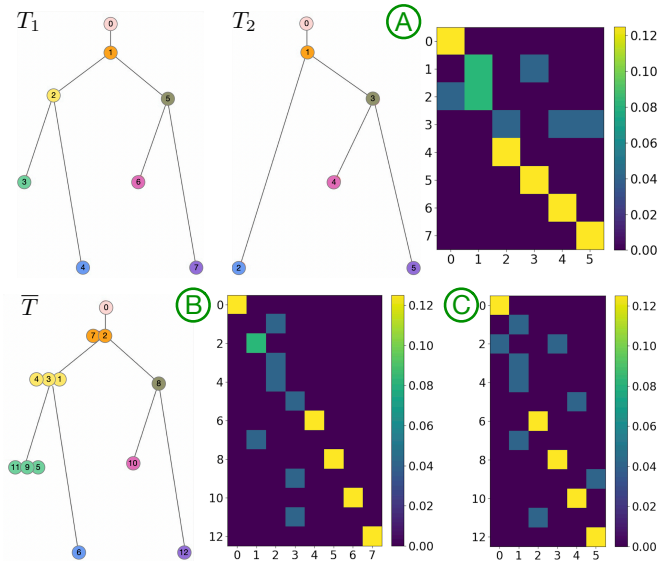

Fig. 5: An optimal coupling between two merge trees T_1 and T_2. The coupling matrix is visualized in (A): yellow means high and dark blue means low probability. Couplings between the Fréchet mean $\overline{T}$ with T_1 and T_2 are shown in (B) and (C), respectively.

We give a simple example involving a pair of merge trees in Fig. 5 (top). T_1 and T_2 contain 8 and 6 nodes, respectively, where nodes are labeled starting with a 0 index. The optimal coupling C is shown below and visualized in Fig. 5 (A):

$$C = \begin{bmatrix} 0.125 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.083 & 0 & 0.042 & 0 & 0 \\ 0.042 & 0.083 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.042 & 0 & 0.042 & 0.042 \\ 0 & 0 & 0.125 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.125 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.125 \end{bmatrix}$$

C is an 8×6 matrix, and it shows, for instance, that node 0 in T_1 is matched to node 0 in T_2 with the highest probability. Node 2 in T_1 is matched probabilistically with both node 0 and node 1 in T_2.

Alignment and blowup [18]. Given a pair of merge trees $T_1 = (V_1, p_1, W_1)$ and $T_2 = (V_2, p_2, W_2)$ with n_1 and n_2 nodes, respectively, a coupling $C \in \mathcal{C}(p_1, p_2)$ can be used to *align* their nodes. In order to do this, we will need to increase the size of T_1 and T_2 appropriately into their respective *blowup* trees T_1' and T_2', such that T_1' and T_2' contain the same n number of nodes (where $n_1, n_2 \leq n$).

Roughly speaking, let x be a node in T_1, and let n_x be the number of nodes in T_2 that have a nonzero coupling probability with x. The blowup tree $T_1' = (V_1', p_1', W_1')$ is created by making n_x copies of node x for each node in T_1, generating a new node set V_1'. The probability distribution p_1' and the weight matrix W_1' are updated from p_1 and W_1 accordingly. Similarly, we can construct the blowup $T_2' = (V_2', p_2', W_2')$ of T_2. An optimal coupling C between p_1 and p_2 expands naturally to a coupling C' between p_1' and p_2'. After taking appropriate blowups, C' is now an $n \times n$ matrix which can be used to align the nodes of the two blowup trees. With a bijective node alignment, we can permute C' to be a diagonal matrix whose marginals agree with p_1' and p_2' respectively. Since C' is a diagonal matrix, we have $p_1' = p_2' = diag(C')$. Finally, C' can be binarized to be an $n \times n$ permutation matrix (e.g., it has 1 where $C' > 0$, and 0 elsewhere). The GW distance is given by a formulation equivalent to Eqn. (3) based on an optimal coupling, following [18, Definition 2]:

$$d_{GW}(T_1', T_2') = \frac{1}{2}\sum_{i,j} |W_1'(i,j) - W_2'(i,j)|^2 p_1'(i)p_1'(j). \quad (4)$$

Fréchet mean. Given a collection of merge trees $\mathcal{T} = \{T_1, \ldots, T_N\}$, a *Fréchet mean* $\overline{T}$ of $\mathcal{T}$ is a minimizer of the functional $F(H, \mathcal{T}) = \frac{1}{N}\sum_{i=1}^N d_{GW}(T_i, H)$ over the space $\mathcal{N}$ of measure networks [18],

$$\overline{T} = \min_{H\in\mathcal{N}} \frac{1}{N}\sum_{i=1}^N d_{GW}(T_i, H). \quad (5)$$

Chowdhury and Needham [18] defined the directional derivative and the gradient of the functional $F(H, \mathcal{T})$ at H and provided a gradient descent algorithm to compute the Fréchet mean. Their iterative optimization begins with an initial guess H_0 of the Fréchet mean. At the k^{th} iteration, there is a two-step process: each T_i is first blown up and aligned to the current Fréchet mean, H_k; then H_k is updated using the gradient of the functional $F(H_k, \mathcal{T})$ at H_k. Such a two-step process is repeated until convergence where the gradient vanishes. For the complete algorithmic and implementational details, see [18]. If $\overline{T} = (\overline{V}, \overline{p}, \overline{W})$ is the Fréchet mean, then we have $\overline{W}(i,j) = \frac{1}{N}\sum_{k=1}^N W_k'(i,j)$, where W_k' is the weight matrix obtained by blowing up and aligning $T_k \in \mathcal{T}$ to $\overline{T}$. That is, when all trees in $\mathcal{T}$ are blown up and aligned to $\overline{T}$, the weight matrix of $\overline{T}$ is given by a simple elementwise average of the weight matrices of the merge trees.

In the example shown in Fig. 5 (bottom), we compute the Fréchet mean $\overline{T}$ of T_1 and T_2, which has 12 nodes. We align both T_1 and T_2 to $\overline{T}$ via their blowup trees. This alignment gives rise to a coupling matrix between $\overline{T}$ and T_1 (of size 12×8) in Fig. 5 (B), and a coupling matrix between $\overline{T}$ and T_2 (of size 12×6) in Fig. 5 (C), respectively. As shown in Fig. 5, root node 0 of $\overline{T}$ is matched probabilistically with root

node 0 of T_1 and root node 0 of T_2. Nodes 2 and 7 of $\overline{T}$ are matched probabilistically with node 1 in T_1. Now both T_1 and T_2 are blown up to be T_1' and T_2', each with 12 nodes, and can be vectorized into column vectors of the same size.

6 METHODS

Given a set $\mathcal{T}$ of N merge trees as input, our goal is to find a basis set $\mathcal{S}$ with $k \ll N$ merge trees such that each tree in $\mathcal{T}$ can be approximately reconstructed from a linear combination of merge trees in $\mathcal{S}$. We propose to combine the GW framework [18] with techniques from matrix sketching to achieve this goal. We detail our pipeline to compute $\mathcal{S}$, as illustrated in Fig. 1.

Step 1: Representing merge trees as measure networks. The first step is to represent merge trees as measure networks, as described in Sec. 5. Each merge tree $T \in \mathcal{T}$ can be represented using a triple (V, p, W). In this paper, we define p as a uniform distribution on V, and W as a shortest path distance matrix.

Recall that each node x in a merge tree is associated with a scalar value $f(x)$. For a pair of nodes $x, x' \in V$, if they are adjacent, we define $W(x, x') = |f(x) - f(x')|$, i.e., their absolute difference in function value; otherwise, $W(x, x')$ is the shortest path distance between them in T. By construction, the shortest path between two nodes goes through their lowest common ancestor in T. The node set of a merge tree is equipped with a function f; therefore, we define W in such a way to encode information from f.

Step 2: Merge tree vectorization via alignment to the Fréchet mean. The second step is to convert each merge tree into a column vector of the same size via blowup and alignment to the Fréchet mean.

Having represented each merge tree as a measure network, we can use the GW framework to compute a Fréchet mean of $\mathcal{T}$, denoted as $\overline{T} = (\overline{V}, \overline{p}, \overline{W})$. Let $n = |\overline{V}|$. In theory, n may become as large as $\prod_{i=1}^{N} |V_i|$. In practice, n is chosen to be much smaller. In our experiment, we choose n to be a small constant factor (e.g., 2 or 3) times the size of the largest input tree. The optimal coupling C between $\overline{T}$ and T_i is an $n \times n_i$ matrix with at least n nonzero entries. If the number of nonzero entries in each row is greater than 1, we keep only the largest value. That is, if a node of $\overline{T}$ has a nonzero probability of coupling with more than one node of T, we consider the mapping with only the highest probability, so that each coupling matrix C has exactly n nonzero entries. We then blow up each T_i to obtain $T_i' = (V', p', W')$, and align $\overline{T}$ with T_i'. The above procedure ensures that each blowup tree T_i' has exactly n nodes, and the binarized coupling matrix C_i' between $\overline{T}$ and T_i' induces a node matching between them.

We can now vectorize (i.e., flatten) each W_i' (an $n \times n$ matrix) to form a column vector $a_i \in \mathbb{R}^d$ of matrix A (where $d = n^2$), as illustrated in Fig. 1 (step 2). In practice, $d = (n+1)n/2$ as we store only the upper triangular matrix. Each a_i is a vector representation of the input tree T_i w.r.t. the Fréchet mean $\overline{T}$. Different from previous vectorization techniques, this process preserves (interpretable) structural correspondences between the vector and the tree.

Step 3: Merge tree sketching via matrix sketching. The third step is to sketch merge trees by applying matrix sketching to the data matrix A, as illustrated in Fig. 1 (step 3). By construction, A is a $d \times N$ matrix whose column vectors a_i are vector representations of T_i. We apply matrix sketching techniques to approximate A by $\hat{A} = B \times Y$. In our experiments, we use two linear sketching techniques from column subset selection (CSS). See Appendix B for implementation details.

Using CSS, the basis set is formed by sampling k columns of A. Let B denote the matrix formed by k columns of A and let $\Pi = BB^{+}$ denote the projection onto the k-dimensional space spanned by the columns of B. The goal of CSS is to find B such that $\|A - \Pi A\|_F$ is minimized. We experiment with two variants of CSS.

In the first variant of CSS, referred to as *Length Squared Sampling* (LSS), we sample (without replacement) columns of A with probabilities q_i proportional to the square of their Euclidean norms, i.e., $q_i = \|a_i\|_2^2 / \|A\|_F^2$. We modify the algorithm slightly such that before selecting a new column, we factor out the effects from columns that are already chosen, making the chosen basis as orthogonal as possible.

In the second variant of CSS, referred to as the *Iterative Feature Selection* (IFS), we use the algorithm proposed by Ordozgoiti et al. [53]. Instead of selecting columns sequentially as in LSS, IFS starts with a random subset of k columns. Then each selected column is either kept or replaced with another column, based on the residual after the other selected columns are factored out simultaneously.

Step 4: Reconstructing sketched merge trees. For the fourth step, we convert each column in $\hat{A}$ as a sketched merge tree. Let $\hat{A} = BY$, where matrices B and Y are obtained using CSS. Let $\hat{a} = \hat{a}_i$ denote the i^{th} column of $\hat{A}$. We reshape $\hat{a}$ as an $n \times n$ weight matrix $\hat{W}'$. We then obtain a tree structure $\hat{T}'$ from $\hat{W}'$ by computing its minimal spanning tree (MST). In particular, we treat $\hat{W}'$ as a pair-wise distance matrix, and the MST constructed from $\hat{W}'$ connects all the nodes and minimizes the sum of edge weights.

Step 5: Returning basis trees. Finally, we return a set of basis merge trees $\mathcal{S}$ using information encoded in the matrix B. Using CSS, each column b_j of B corresponds directly to a column in A; therefore, the set $\mathcal{S}$ is trivially formed by the corresponding merge trees from $\mathcal{T}$.

Error analysis. For each experiment, we compute the *global sketch error* $\epsilon = \|A - \hat{A}\|_F^2$, as well as *column-wise sketch error* $\epsilon_i = \|a_i - \hat{a}_i\|_2^2$, where $\epsilon = \sum_{i=1}^{N} \epsilon_i$. By construction, $\epsilon_i \leq \epsilon$. For merge trees, we measure the GW distance between each tree T_i and its sketched version $\hat{T}_i$, that is $\tau_i = d_{GW}(T_i, \hat{T}_i)$, referred to as the *column-wise GW loss*. The *global GW loss* is defined to be $\tau = \sum_{i=1}^{N} \tau_i$. For theoretical considerations, see discussions in Appendix A.

7 EXPERIMENTAL RESULTS

We demonstrate the applications of our sketching framework with merge trees that arise from three 2D and one 3D time-varying datasets from scientific simulations. The key takeaway is that, by applying matrix sketching, a large set $\mathcal{T}$ of merge trees is replaced by a much smaller basis set $\mathcal{S}$ such that trees in $\mathcal{T}$ are well approximated by trees in $\mathcal{S}$. Elements in the basis set $\mathcal{S}$ serve as good representatives that capture structural variations among the time instances, thus reflecting the "modes" of the underlying physical phenomena (Sec. 7.1 and Sec. 7.2). In addition, our framework also uncovers cyclical behavior of time-varying datasets that exhibit periodicity (Sec. 7.3 and Appendix C.1). See Appendix C for additional results and runtime analysis.

In practice, we simplify the scalar fields based on persistence before computing the merge trees. See Appendix B for details.

7.1 Heated Cylinder Dataset

Two of our datasets come from numerical simulations available online [1]. The first dataset, referred to as the *Heated Cylinder with Boussinesq Approximation* (*Heated Cylinder* in short), comes from the simulation of a 2D flow generated by a heated cylinder using the Boussinesq approximation [36, 58]. The dataset shows a time-varying turbulent plume containing numerous small vortices. We convert each time instance of the flow (a vector field) into a scalar field using the magnitude of its vertical (y) velocity component. We generate a set of split trees (i.e., the merge tree surrounding local maxima) from these scalar fields based on 31 time steps, which correspond to steps 600-630 from the original 2000 time steps. This set captures the evolution of small vortices over time.

Parameter. To choose the appropriate k number of basis trees for this dataset, we use the "elbow method" to determine k, similar to cluster analysis. We plot the global GW loss and global sketch error as a function of k, and pick the elbow of the curve as the k to use. As shown in Fig. 6, k is chosen to be three for the *Heated Cylinder* dataset. In subsequent sections, element-wise GW losses and sketch errors also reaffirm this choice (cf., Fig. 8).

Given 31 merge trees $\mathcal{T} = \{T_0, \ldots, T_{30}\}$ from the *Heated Cylinder* dataset, we apply two types of column subset selection (CSS) methods, namely IFS and LSS to obtain a set of basis trees $\mathcal{S}$ and reconstruct the sketched trees. Since we are using CSS, the basis trees are elements from the original input. We first demonstrate that the basis trees capture structural variations among the time-varying input. We then investigate

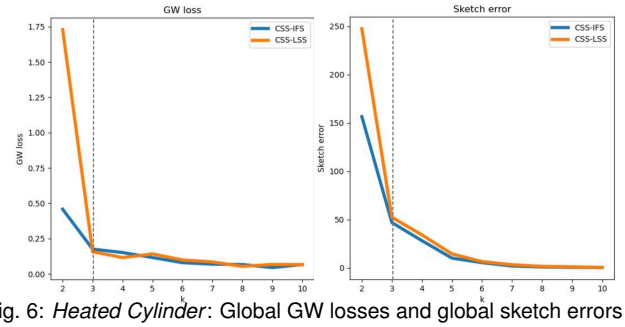

Fig. 6: *Heated Cylinder*: Global GW losses and global sketch errors for varying k, the number of basis trees, using IFS and LSS.

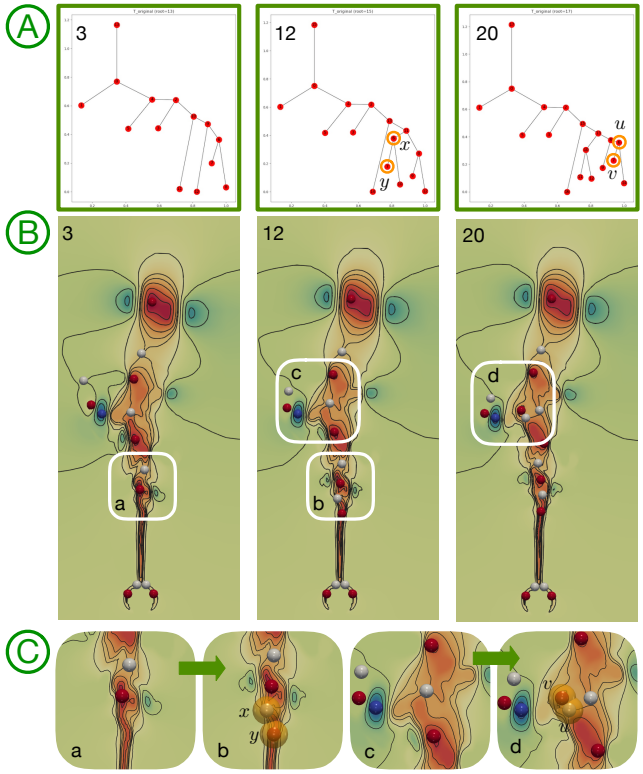

Fig. 7: Sketching the *Heated Cylinder* dataset with three basis trees using IFS: (A) basis trees where orange circles highlight topological changes w.r.t. nearby basis trees, (B) scalar fields that give rise to these basis trees. Areas with critical points appearances/disappearances are shown with zoomed views in (C).

the coefficient matrix and show that with only three basis trees, we can obtain sketched trees with small errors.

Basis trees as representatives with IFS. We first illustrate our sketching results using IFS. Based on our error analysis (Fig. 6), three basis trees appear to be the appropriate choice that strikes a balance between data summarization and structural preservation.

As shown in Fig. 7(A), IFS produces three basis trees, $\mathcal{S} = \{T_3, T_{12}, T_{20}\}$, which capture noticeable structural variations among the input merge trees. Specifically, moving from T_3 to T_{12} and T_{12} to T_{20}, a saddle-maxima pair appears in the merge trees, respectively (highlighted by orange circles). These changes in the basis trees reflect the appearances of critical points in the domain of the time-varying fields; see Fig. 7(B). In Fig. 7(C), we highlight (with orange balls) the appearances of these critical points in the domain. That is, from T_3 to T_{12}, critical points x and y appear in the scalar fields, whereas from T_{12} to T_{20}, critical points u and v appear. Therefore, the three basis trees capture structural changes in the time-varying data, thus reflecting the "modes" of the underlying phenomena. Such "modes" are also confirmed with the coefficient matrix (see Fig. 8), which is a byproduct of the sketching process.

Coefficient matrices with IFS. The coefficient matrix, column-wise sketch error, and GW loss are used to guide our investigation into the quality of individual sketched trees, see Fig. 8. Trees with small GW losses or sketch errors are considered well sketched w.r.t. the chosen basis. The coefficient matrix in Fig. 8(B) contains a number of yellow or light green blocks, indicating that consecutive input trees share similar coefficients w.r.t. the chosen basis; therefore, they are grouped together into three clusters, reflecting the three modes of the underlying phenomena. Such a blocked structure indicates that the chosen basis trees are good representatives of the clusters.

In comparison, using just two basis trees (T_3 and T_{22}) does not capture the structural variations as well as three basis trees. In Fig. 8(C), we see a slight degradation in the blocked structure and thus the sketching quality using two basis trees. In particular, trees in the red area of Fig. 8(D) (T_8 to T_{14}) are not well approximated due to a missing basis tree.

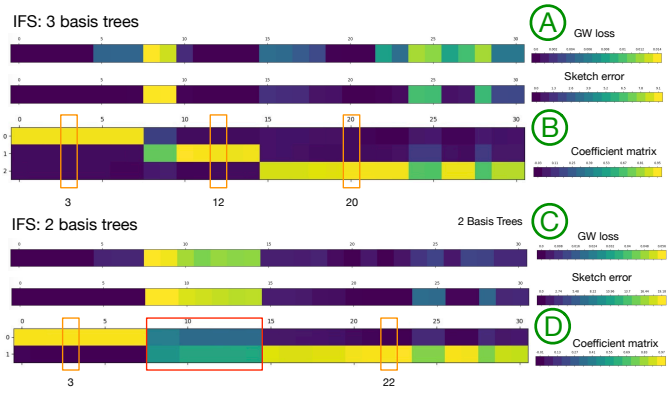

Fig. 8: Sketching the *Heated Cylinder* dataset with three (A-B) and two (C-D) basis trees using IFS. (A, C) column-wise sketch error and GW loss, (B, D) coefficient matrix. Orange boxes highlight basis trees.

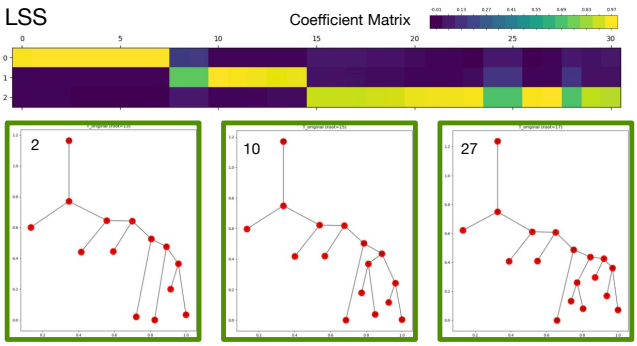

Fig. 9: *Heated Cylinder*: Coefficient matrices and basis trees used to sketch the dataset with three basis trees using LSS.

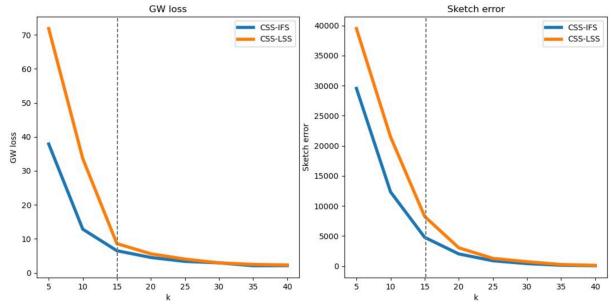

Fig. 10: *Corner Flow*: Global GW losses and global sketch errors for varying k, the number of basis trees, using IFS and LSS.

Sketching with LSS. Additionally, we include the sketching results using LSS as an alternative strategy, again with three basis trees according to the "elbow method" (Fig. 6). LSS gives basis trees T_2, T_{10} and T_{27} in Fig. 9, which are similar to the ones obtained by IFS (Fig. 7). In other words, for the *Heated Cylinder* dataset, variations in column selection methods do not affect the quality of sketching results.

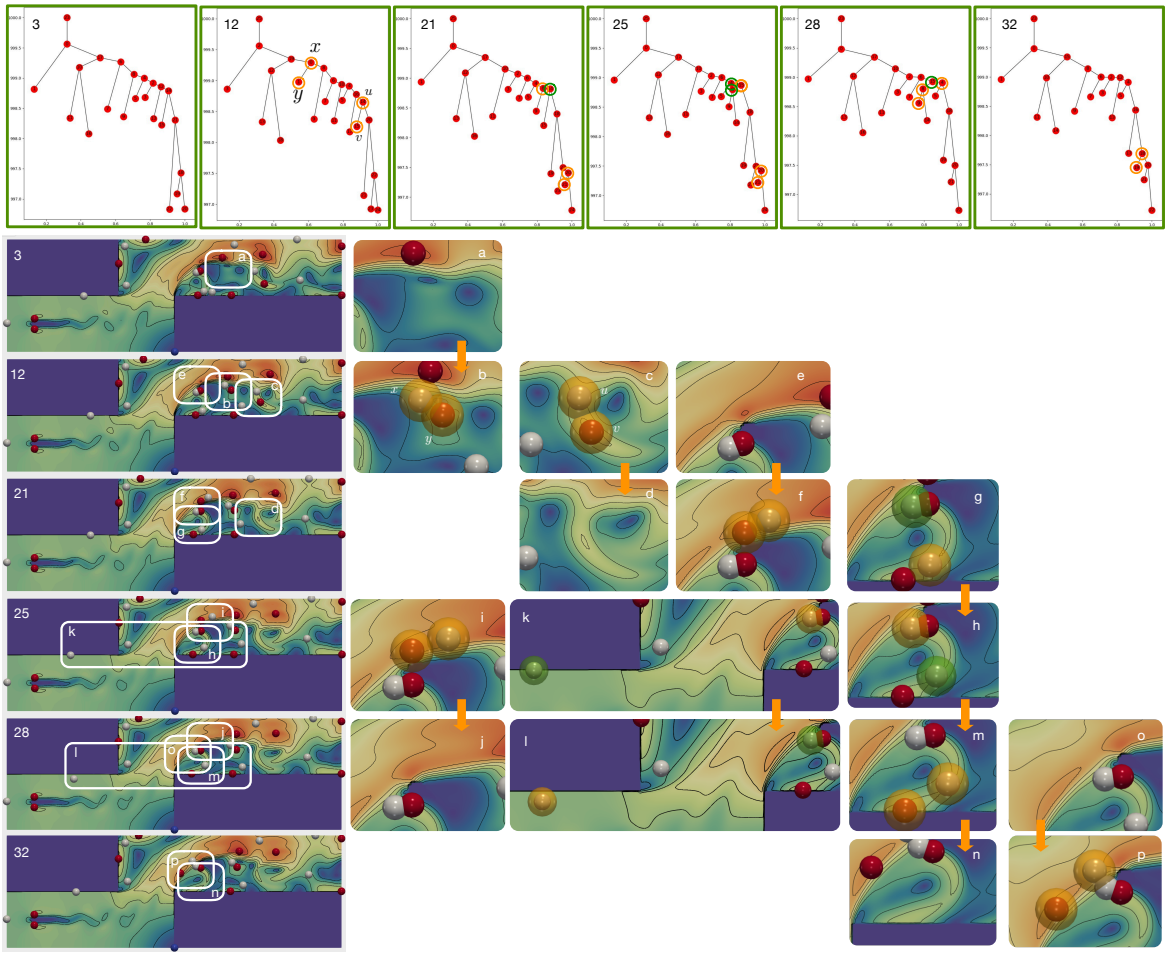

Fig. 11: Sketching the *Corner Flows* dataset with 15 basis trees with IFS: (Top) first 6 basis trees where orange circles highlight topological changes w.r.t. near basis trees, (Bottom Left) scalar fields that give rise to these basis trees, areas with critical points appearances/disappearances are shown with zoomed views in (Bottom Right).

7.2 Corner Flow Dataset

The second dataset, referred to as the *Cylinder Flow Around Corners* (*Corner Flow* in short), arises from the simulation of a viscous 2D flow around two cylinders [5, 58]. The channel into which the fluid is injected is bounded by solid walls. A vortex street is initially formed at the lower left corner, which then evolves around the two corners of the bounding walls. We generate a set of merge trees from the magnitude of the velocity fields of 100 time instances, which correspond to steps 801-900 from the original 1500 time steps. This dataset describes the formation of a one-sided vortex street on the upper right corner.

Parameter. Using the "elbow method", we choose $k = 15$ (see Fig. 10). Given a set of 100 merge trees, we first demonstrate that a set of 15 basis trees chosen with IFS gives sketched trees with a small error, based on the coefficient matrices and error analysis.

Coefficient matrices with IFS. We first compare the coefficient matrices generated using IFS, for $k = 10, 15$, respectively. Comparing Fig. 12(A) and (C), we see generally improved column-wise GW loss and sketch error using 15 instead of 10 basis trees. Furthermore, the coefficient matrix with 15 basis trees (B) contains better block structures than the one with 10 basis trees (D). In particular, using additional basis trees improves upon the sketching results in regions enclosed by red boxes in (D).

Basis trees as representatives. We thus report the sketching results with 15 basis trees under IFS. The basis trees are selected with labels 3, 12, 21, 25, 28, 32, 36, 40, 48, 53, 60, 65, 74, 81, 92; see Fig. 12(B) and the first 6 basis trees in Fig. 11(Top). Similar to the *Heated Cylinder*, we observe noticeable structural changes among pairs of adjacent basis trees, which lead to a partition of the input trees into clusters with similar structures; see the block structure in Fig. 12(B). Thus the basis

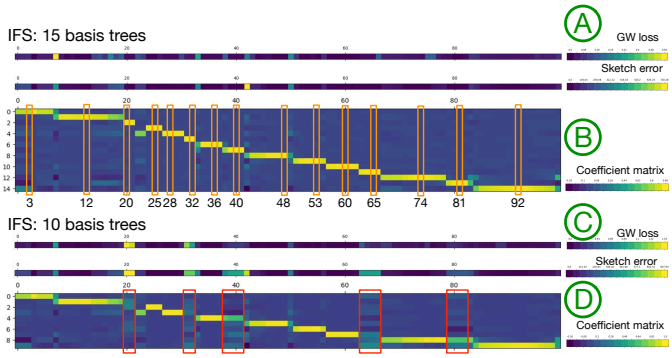

Fig. 12: Sketching the *Corner Flow* dataset with 15 (A, B) and 10 (C, D) basis trees using IFS. (A, C) column-wise sketch error and GW loss, (B, D) coefficient matrix. Orange boxes highlight basis trees. Red boxes in (D) indicate trees that are better sketched with 15 basis trees.

trees serve as good cluster representatives, as they are roughly selected one per block.

We highlight the structural changes among the first 6 adjacent basis trees in Fig. 11 Top (green boxes). We further highlight critical points involved in these structural changes in the domain (Fig. 11 Bottom Left) with zoomed views in Fig. 11 (Bottom Right). For instance, moving from T_3 to T_{12}, critical points x and y appear while u and v disappear, cf., Fig. 11 (Top) T_3 and T_{12} and (Bottom Right) a $\rightarrow$ b, c $\rightarrow$ d.

7.3 Flow Behind a Square Cylinder Dataset

We demonstrate the use of our method in distinguishing and summarizing different types of flow behavior using a flow behind a square

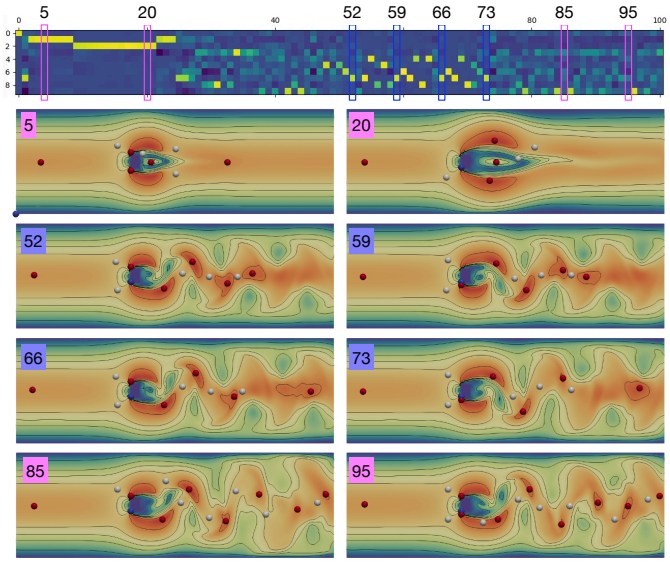

Fig. 13: Sketching the *Square Cylinder Flow* dataset with IFS. Top shows the coefficient matrix. Bottom shows instances of the scalar fields.

cylinder, referred to as the *Square Cylinder Flow* dataset. This dataset is a direct numerical Navier Stokes simulation by Simone Camarri and Maria-Vittoria Salvetti (University of Pisa), Marcelo Buffoni (Politecnico of Torino), and Angelo Iollo (University of Bordeaux) [13]. We use a uniformly resampled version that has been provided by Tino Weinkauf (`https://www.csc.kth.se/~weinkauf/notes/squarecylinder.html`) and used by von Funck et al. for smoke visualizations [65]. The magnitude of the velocity field is available as a scalar field on a $192 \times 64 \times 48$ grid over 101 time steps. We consider a slice perpendicular to the square cylinder. We report the sketching result by setting $k = 10$.

Observing different stages of flow evolution. As shown in the coefficient matrix in Fig. 13 (1st row), we observe four stages of the vortex flow evolution. For the first $0 \rightarrow 24$ instances (stage 1), the flow has not formed any visible vortices, and the topological structures of the scalar fields appear to be relatively simple; as shown in Fig. 13 (2nd row) for instances 5 and 20. The sketching framework identifies three basis trees in stage 1, corresponding to instances 0, 4 and 16, which capture structural variations. In stage 2 for instances $25 \rightarrow 51$, some vortices begin to form, and the merge trees generated from the scalar fields exhibit significant structural changes; however, periodic behaviors have not yet formed. During stage 3 from instances $52 \rightarrow 73$, the vortices demonstrate shedding behavior, that is, the vortices from both sides of the cylinder alternate and give rise to oscillating and thus periodic patterns. In particular, we highlight instances 52, 59, 66, 73 in Fig. 13 (3rd and 4th rows), where the crest and trough of vortices alternate at a fixed periodicity of ~ 7. For example, instance 52 is mirror-symmetrical to instance 59, and nearly identical to instance 66. Afterward, in stage 4 (instances $85 \rightarrow 100$), the vortices become significantly distorted and their periodic behaviors become less visible; see instances 85 and 95 in Fig. 13 (5th row) for reference.

Observed periodicity. The stage that shows periodicity includes instances $52 \rightarrow 73$. We show instances 52, 59, 66, and 73 that highlight the periodic behavior in the coefficient matrix (Fig. 13 1st row), all of which share high coefficients with the same basis. In addition, such a periodicity is further validated by the GW distance matrix, see Appendix C for details. These results illustrate the ability of our framework to simultaneously detect representatives (at an early stage) and capture periodicities (at a later stage) of the flow.

7.4 Isabel 3D Dataset

Our framework can be adapted to higher dimensional data. We demonstrate the application to a 3D *Isabel* dataset simulating Hurricane Isabel [66]. We use time steps $2 \rightarrow 5$, $30 \rightarrow 33$, and $45 \rightarrow 48$ for analysis, describing three key phases (i.e., formation, drift, landfall) of the hurricane. This is the same set of time steps processed by Pont et al. [57].

The underlying topology changes rapidly across this dataset. In Fig. 14, we see the first two time steps of each phase of the hurricane: between adjacent time steps, we see many differences between the merge trees. For example, see the areas inside blue, cyan, and purple circles, respectively. Across different hurricane phases, the difference in the underlying topology is even more significant: the merge tree for the time step 2, 30, and 45 has 22, 188, and 100 nodes, respectively.

Coefficient matrices. Using the prior knowledge that there are three phases of the hurricane in this dataset, we naturally desire to use three representatives to describe the overall topology. We apply our framework by setting $k = 3$. However, no time instances in the third phase are selected as representatives; see Fig. 14 (top right). Using the IFS strategy, we select time steps 2, 3, and 30 as representatives; using the LSS strategy, we select time steps 4, 5, and 33 as representatives. Both strategies select two time steps in the formation (first) phase and one time step in the drift (second) phase. Our conjecture is that since the number of topological features changes drastically during the formation phase, our framework selects more than one instance in this phase as the representatives to capture these drastic feature appearances.

Meanwhile, a representative is selected in the drift phase. Based on the selected basis, time instances in the drift phase are clearly grouped in the coefficient matrices using both CSS strategies (Fig. 14 top right), regardless of minor differences in the branch structures (e.g., see the cyan circles in Fig. 14).

However, our results do not imply that the GW distance fails to detect different phases of the hurricane. Instead, in Fig. 14 (bottom right), we can clearly see three block structures (in green squares) from the pairwise GW distance matrix. Each block indicates similar topological structures for all instances within, whereas the transition of blocks along the diagonal of the pairwise distance matrix indicates topological structure changes. In other words, using the GW distance, we see that time steps $2 \rightarrow 5$ share similar topological structures, as do time steps $30 \rightarrow 33$ and $45 \rightarrow 48$. The result matches the prior knowledge on this dataset.

Takeaway. The GW distance is able to identify a subset of representatives for 3D volume data with complex topological changes. The merge tree vectorization and sketching, however, is relatively sensitive to drastic topological changes, and, thus, its performance depends on the complexity of the data and the chosen number of basis.

8 Discussion: Alternative Vectorization Approaches

Given the promising performance of vectorizing and sketching merge trees using tools from optimal transport, we now discuss two alternative approaches to vectorize scalar field data. We apply the same sketching techniques (i.e., IFS and LSS) to the vectorized data and assess how much the basis vectors capture the topological variations in the data.

The first approach is to vectorize a scalar field f directly by unrolling its matrix representation into a vector. Suppose a scalar field f is uniformly sampled on a regular grid and represented as a matrix. We obtain a high-dimensional vector by unrolling the matrix in a row-major order (i.e., concatenating the rows).

The second approach is to vectorize the persistence diagram of a scalar field f (e.g., [2, 29]). A persistence image [2] is a vector representation of the persistent homology of f. We compute the persistence image of f that captures its 0-dimensional superlevel set persistent homology (i.e., behaviors of maxima and saddles). Since a persistence image is a matrix, we again vectorize it by matrix unrolling in a row-major order.

To keep all vectors of the same size, we fix the grid size of the scalar field domain in the first approach and the resolution of the persistence image in the second approach. We apply matrix sketching on the set of vectors that arise from the *Heated Cylinder* dataset. The same level of persistence simplification is applied for consistency. To sketch both types of vectors, we define the sketch error as the sum of the squared residuals between the original and its sketched version. We use the "elbow method" on the sketch error plots to select the number of basis.

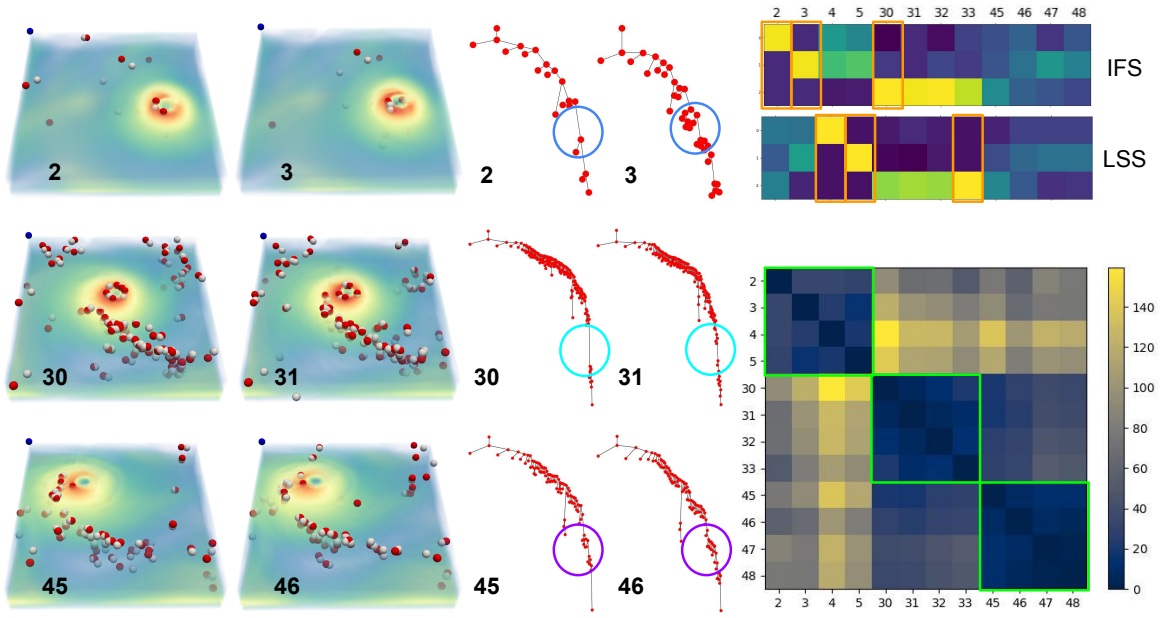

Fig. 14: Hurricane *Isabel* dataset. Left and middle: volume renderings and merge trees of the first two time steps of each phase of the hurricane. Right: coefficient matrix using IFS and LSS with three basis (top) and the pairwise GW distance matrix (bottom).

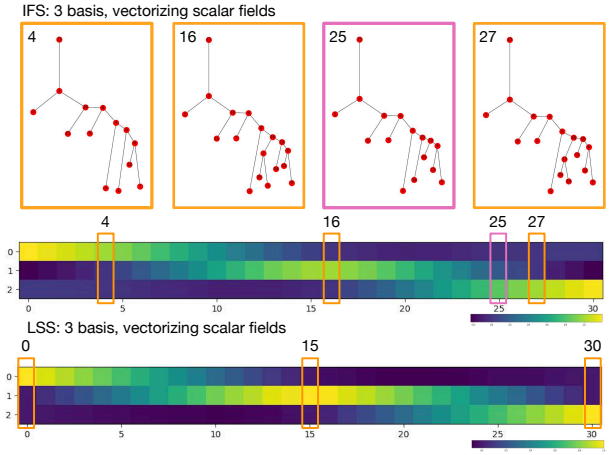

Fig. 15: Sketching the *Heated Cylinder* dataset using vectorized scalar fields. Orange boxes highlight selected basis vectors.

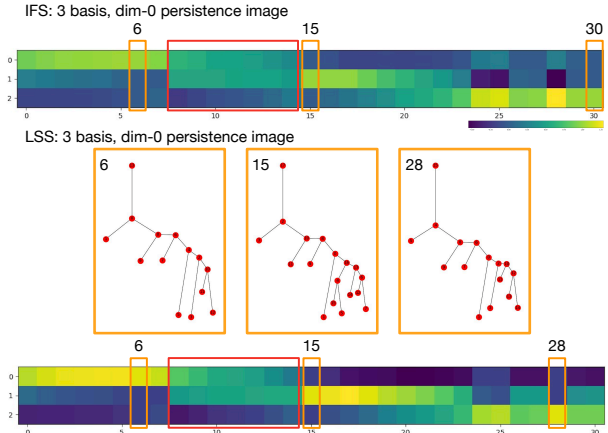

Fig. 16: Sketching the *Heated Cylinder* dataset using persistence images. Orange boxes highlight selected basis vectors.

Vectorizing scalar fields. Based on the error plot (see Appendix C), we set $k = 3$. Using IFS, we select time steps 4, 16, and 27 as the basis; see Fig. 15 (top). The selected basis vectors are oblivious to topological changes in the data. In particular, merge trees at time steps 16 and 27 are almost indistinguishable (c.f., observable topological changes at time step 25). In addition, we do not observe yellow or light green blocks in the coefficient matrix, indicating the lack of clustering/modes using these basis vectors. Furthermore, using LSS, we select time steps 0, 15, and 30 as the basis and observe that there are again no clear block structures in the coefficient matrix; see Fig. 15 (bottom). In short, representatives obtained by applying sketching to these scalar field vectorizations do not capture topological variations in the data.

Vectorizing persistence images. To sketch dimension-0 persistence images, we set $k = 3$ (Fig. 20 right). Using IFS and LSS, we observe some noisy block structures in the coefficient matrices, indicating that persistence images are partially successful in detecting modes in the data (especially using LSS), see Fig. 16. We display the chosen basis using LSS. These basis vectors also give rise to merge trees that are topologically distinguishable (see T_6, T_{15}, and T_{28}). However, these block structures are much noisier than those observed via the merge tree vectorizations (c.f., Fig. 8), and contain poorly sketched sections (enclosed by red boxes). We obtained similar results using dimension-1 persistence images (due to duality [26, page 164]). In summary, sketching persistence images produces results that are suboptimal, in comparison with those obtained by sketching merge trees.

9 Conclusion

We present a framework to sketch merge trees. Given a set $\mathcal{T}$ of merge trees of (possibly) different sizes, we compute a basis set of merge trees $\mathcal{S}$ such that each tree in $\mathcal{T}$ can be approximately reconstructed using $\mathcal{S}$. Our framework can be used to identify a basis set, which serves as a consensus set or a set of representatives that capture the modes of the underlying data. Such a basis set also provides a compact representation for downstream analysis. In addition, our framework could recover cyclical phenomena for time-varying data. Our framework demonstrates that, for the first time, matrix sketching could be applied to topological descriptors. This work starts an exciting direction of vectorizing topological descriptors using optimal transport and applying randomized linear algebra to these vector representations.

Our framework also has limitations. First, the computed Frechét mean depends on the initialization due to the optimization process. Second, the size of the blowup matrix may quickly become intractable [18] with large merge trees. Our approach is flexible enough to be generalized to sketch other topological descriptors such as contour trees, Reeb graphs, and Morse–Smale graphs (e.g., [17]), which is left for future work.

Acknowledgments

This work was partially funded by DOE DE-SC0021015 and NSF IIS-2145499. We thank Jeff Phillips for discussions involving column subset selection, Benwei Shi for his implementation on length squared sampling, and Ofer Neiman for sharing the code on low stretch spanning trees.

References

[1] Computer graphics laboratory. https://cgl.ethz.ch/research/visualization/data.php. 5

[2] H. Adams, T. Emerson, M. Kirby, R. Neville, C. Peterson, P. Shipman, S. Chepushtanova, E. Hanson, F. Motta, and L. Ziegelmeier. Persistence images: A stable vector representation of persistent homology. *The Journal of Machine Learning Research*, 18(1):218–252, 2017. 3, 8

[3] M. Agueh and G. Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011. 3

[4] D. Alvarez-Melis and T. Jaakkola. Gromov-Wasserstein alignment of word embedding spaces. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1881–1890, 2018. 3

[5] I. Baeza Rojo and T. Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):280–290, 2020. 7

[6] K. Beketayev, D. Yeliussizov, D. Morozov, G. Weber, and B. Hamann. Measuring the distance between merge trees. *Topological Methods in Data Analysis and Visualization III: Theory, Algorithms, and Applications, Mathematics and Visualization*, pages 151–166, 2014. 3

[7] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015. 3

[8] A. Bhaskara, S. Lattanzi, S. Vassilvitskii, and M. Zadimoghaddam. Residual based sampling for online low rank approximation. *IEEE 60th Annual Symposium on Foundations of Computer Science*, pages 1596–1614, 2019. 2

[9] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008. 12

[10] C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 968–977, 2009. 12

[11] A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM Journal on Scientific Computing*, 28(5):1812–1836, 2006. 3

[12] C. Bunne, D. Alvarez-Melis, A. Krause, and S. Jegelka. Learning generative models across incomparable spaces. *International Conference on Machine Learning*, pages 851–861, 2019. 3

[13] S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo. Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata*, pages 11–15, 2005. 8

[14] H. Carr, J. Snoeyink, and U. Axen. Computing contour trees in all dimensions. *Computational Geometry*, 24(2):75–94, 2003. 3

[15] M. Carriere, F. Chazal, Y. Ike, T. Lacombe, M. Royer, and Y. Umeda. Perslay: A neural network layer for persistence diagrams and new graph topological signatures. In S. Chiappa and R. Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2786–2796. PMLR, 26–28 Aug 2020. 3

[16] M. Carrière, S. Y. Oudot, and M. Ovsjanikov. Stable topological signatures for points on 3d shapes. *Computer Graphics Forum*, 34(5):1–12, 2015. 3

[17] M. J. Catanzaro, J. M. Curry, B. T. Fasy, J. Lazovskis, G. Malen, H. Riess, B. Wang, and M. Zabka. Moduli spaces of Morse functions for persistence. *Journal of Applied and Computational Topology*, 4:353–385, 2020. 9

[18] S. Chowdhury and T. Needham. Gromov-Wasserstein averaging in a Riemannian framework. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 842–843, 2020. 1, 2, 3, 4, 5, 9, 12, 13

[19] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 92(3):708–721, 2009. 15

[20] M. Cuturi and A. Doucet. Fast computation of Wasserstein barycenters. *Proceedings of the 31st International Conference on Machine Learning, PMLR*, 32(2):685–693, 2014. 3

[21] A. Deshpande and S. Vempala. Adaptive sampling and fast low-rank matrix approximation. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 292–303, 2006. 12

[22] P. Drineas, R. Kannan, and M. W. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36:158–183, 2006. 3, 12

[23] P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *Journal of Machine Learning Research*, 13:3441–3472, 2012. 3

[24] P. Drineas and M. W. Mahoney. RandNLA: Randomized numerical linear algebra. *Communications of the ACM*, 59(6):80–90, may 2016. 1

[25] I. L. Dryden, A. Koloydenko, and D. Zhou. Non-Euclidean statistics for covariance matrices, with applications to diffusion tensor imaging. *Annals of Applied Statistics*, 3(3):1102–1123, 2009. 3

[26] H. Edelsbrunner and J. Harer. *Computational Topology: An Introduction*. American Mathematical Society, 2010. 9

[27] F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346–347:180–197, 2016. 3

[28] D. Ezuz, J. Solomon, V. G. Kim, and M. Ben-Chen. GWCNN: A metric alignment layer for deep shape analysis. *Computer Graphics Forum*, 36:49–57, 2017. 3

[29] G. Favelier, N. Faraj, B. Summa, and J. Tierny. Persistence atlas for critical point variability in ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):1152–1162, 2018. 8

[30] C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011. 15

[31] E. Gasparovic, E. Munch, S. Oudot, K. Turner, B. Wang, and Y. Wang. Intrinsic interleaving distance for merge trees. arXiv preprint arXiv:1908.00063, 2019. 3

[32] S. Gerber, P.-T. Bremer, V. Pascucci, and R. Whitaker. Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 16:1271–1280, 2010. 12

[33] M. Ghashami, E. Liberty, J. M. Phillips, and D. P. Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM Journal of Computing*, 45(5):1762–1792, 2016. 3

[34] M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces*, volume 152 of *Progress in mathematics*. Birkhäuser, Boston, USA, 1999. 3

[35] S. Gu and T. Milenković. Data-driven network alignment. *PLoS ONE*, 15(7):e0234978, 2020. 3

[36] T. Günther, M. Gross, and H. Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics*, 36(4):141:1–141:11, 2017. 5

[37] C. Heine, H. Leitte, M. Hlawitschka, F. Iuricich, L. De Floriani, G. Scheuermann, H. Hagen, and C. Garth. A survey of topology-based methods in visualization. *Computer Graphics Forum*, 35(3):643–667, 2016. 1

[38] R. Hendrikson. Using Gromov-Wasserstein distance to explore sets of networks. Master's thesis, University of Tartu, 2016. 3

[39] F. Hensel, M. Moor, and B. Rieck. A survey of topological machine learning methods. *Frontiers in Artificial Intelligence*, 4, 2021. 3

[40] W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984. 3

[41] E. Liberty. Simple and deterministic matrix sketching. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 581–588, 2013. 2, 3

[42] S. Liu, D. Maljovec, B. Wang, P.-T. Bremer, and V. Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1249–1268, 2017. 1

[43] A.-P. Lohfink, F. Wetzels, J. Lukasczyk, G. H. Weber, and C. Garth. Fuzzy contour trees: alignment and joint layout of multiple contour trees. *Computer Graphics Forum (CGF)*, 39(3):343–355, 2020. 3

[44] M. W. Mahone and P. Drineas. Structural properties underlying high-quality randomized numerical linear algebra algorithms. In P. Bühlmann, P. Drineas, M. Kane, and M. v. d. Laan, editors, *Handbook of Big Data*, pages 137–154. Chapman and Hall, 2016. 1, 12

[45] F. Mémoli. *Estimation of distance functions and geodesics and its use for shape comparison and alignment: theoretical and computational results*. PhD thesis, University of Minnesota, 2005. 3

[46] F. Mémoli. On the use of Gromov-Hausdorff distances for shape comparison. *Eurographics Symposium on Point-Based Graphics*, pages 81–90, 2007. 3, 4

[47] F. Mémoli. Gromov-Wasserstein distances and the metric approach to object matching. *Foundations of Computational Mathematics*, 11(4):417–487, 2011. 1, 3, 4

[48] F. Mémoli and T. Needham. Gromov-Monge quasi-metrics and distance distributions. arXiv preprint arXiv:1810.09646, 2020. 3

[49] F. Mémoli and G. Sapiro. Comparing point clouds. *Proceedings of the Eurographics/ACM SIGGRAPH Symposiumon Geometry Processing*, pages 32–40, 2004. 3

[50] F. Mémoli and G. Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Foundations of Computational Mathematics*, 5:313–347, 2005. 3

[51] F. Memoli, A. Sidiropoulos, and K. Singhal. Sketching and clustering metric measure spaces. arXiv preprint arXiv:1801.00551, 2018. 3

[52] J. Milnor. *Morse Theory*. Princeton University Press, New Jersey, 1963. 3

[53] B. Ordozgoiti, S. G. Canaval, and A. Mozo. A fast iterative algorithm for improved unsupervised feature selection. *IEEE 16th International Conference on Data Mining*, pages 390–399, 2016. 5

[54] K. Pearson. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901. 1

[55] G. Peyré, M. Cuturi, and J. Solomon. Gromov-Wasserstein averaging of kernel and distance matrices. *Proceedings of the 33rd International Conference on Machine Learning, PMLR*, 48:2664–2672, 2016. 1, 3, 4

[56] J. M. Phillips. Coresets and sketches. In *Handbook of Discrete and Computational Geometry*, chapter 48. CRC Press, 3rd edition, 2016. 1, 3

[57] M. Pont, J. Vidal, J. Delon, and J. Tierny. Wasserstein distances, geodesics and barycenters of merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):291–301, 2022. 3, 8, 12, 15

[58] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004. 5, 7

[59] T. Sarlós. Improved approximation algorithms for large matrices via random projections. *Proceedings of 47th IEEE Symposium on Foundations of Computer Science*, pages 143–152, 2006. 3

[60] R. Sridharamurthy, T. B. Masood, A. Kamakshidasan, and V. Natarajan. Edit distance between merge trees. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1518–1531, 2020. 3, 13

[61] K.-T. Sturm. The space of spaces: curvature bounds and gradient flows on the space of metric measure spaces. arXiv preprint arXiv:1208.0434, 2012. 3, 4

[62] V. Titouan, N. Courty, R. Tavenard, and R. Flamary. Optimal transport for structured data with application on graphs. *International Conference on Machine Learning*, pages 6275–6284, 2019. 3

[63] V. Titouan, R. Flamary, N. Courty, R. Tavenard, and L. Chapel. Sliced Gromov-Wasserstein. *Advances in Neural Information Processing Systems*, pages 14726–14736, 2019. 3

[64] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. 16

[65] W. Von Funck, T. Weinkauf, H. Theisel, and H.-P. Seidel. Smoke surfaces: An interactive flow visualization technique inspired by real-world flow experiments. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1396–1403, 2008. 8

[66] W. Wang, C. Bruyere, B. Kuo, and T. Scheitlin. IEEE SciVis Contest. `https://sciviscontest.ieeevis.org/2004/`, 2004. 8

[67] F. Wetzels and C. Garth. A deformation-based edit distance for merge trees. In *2022 Topological Data Analysis and Visualization (TopoInVis)*, pages 29–38, 2022. 3

[68] F. Wetzels, H. Leitte, and C. Garth. Branch decomposition-independent edit distances for merge trees. *Computer Graphics Forum*, 41(3):367–378, 2022. 3, 15

[69] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014. 1, 3

[70] H. Xu, D. Luo, and L. Carin. Scalable Gromov-Wasserstein learning for graph partitioning and matching. *Advances in Neural Information Processing Systems*, pages 3046–3056, 2019. 3

[71] H. Xu, D. Luo, H. Zha, and L. Carin. Gromov-Wasserstein learning for graph matching and node embedding. *International Conference on Machine Learning*, pages 6932–6941, 2019. 3

[72] 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. *Computer Graphics Forum (CGF)*, 40(3):599–633, 2021. 3

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.

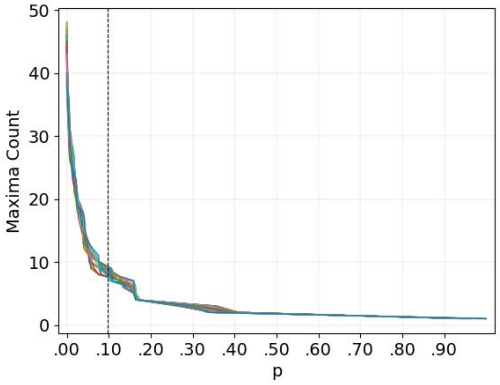

Fig. 17: *Heated Cylinder*: Persistence simplification using a persistence graph.

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.

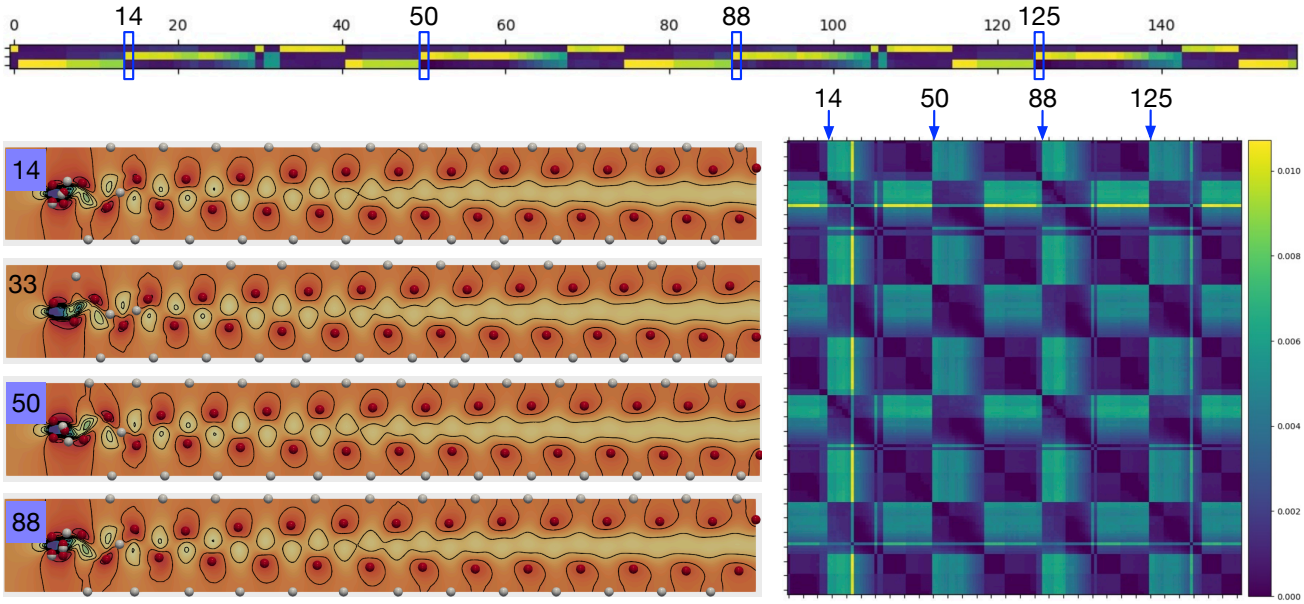

Fig. 18: *Vortex Street*: we highlight instance 14, 50, 88, and 125 in the coefficient matrix (TOP) to illustrate the periodicity of its topological structures, using IFS. The periodicity is also confirmed within the GW distance matrix (bottom right). Visualizations of selected scalar fields are shown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

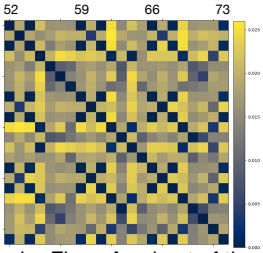

Fig. 19: *Square Cylinder Flow*: A subset of the pairwise GW distance matrix among T_{52} to T_{73}.

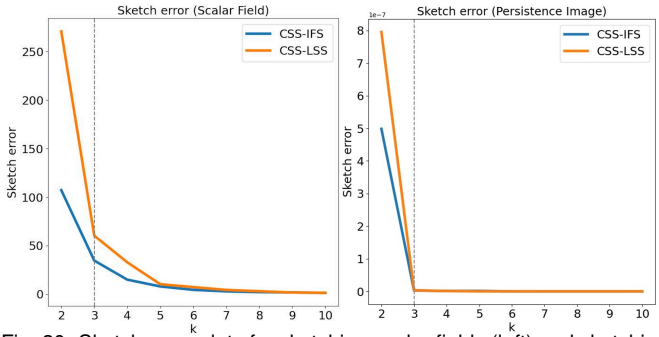

Fig. 20: Sketch error plots for sketching scalar fields (left) and sketching 0-dimensional persistence images (right).

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.

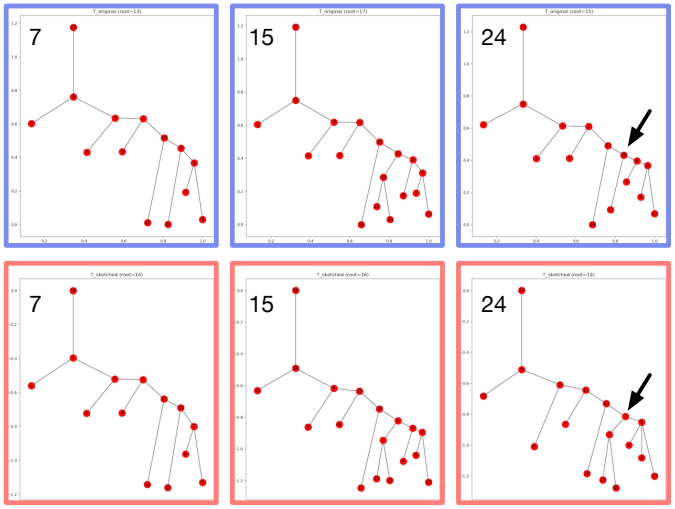

Fig. 21: *Heated Cylinder*: Comparing each sketched tree (red box) with its corresponding input tree (blue box), highlighting noticeable structural differences among subtrees (whose roots are pointed by black arrows) before and after sketching.

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.

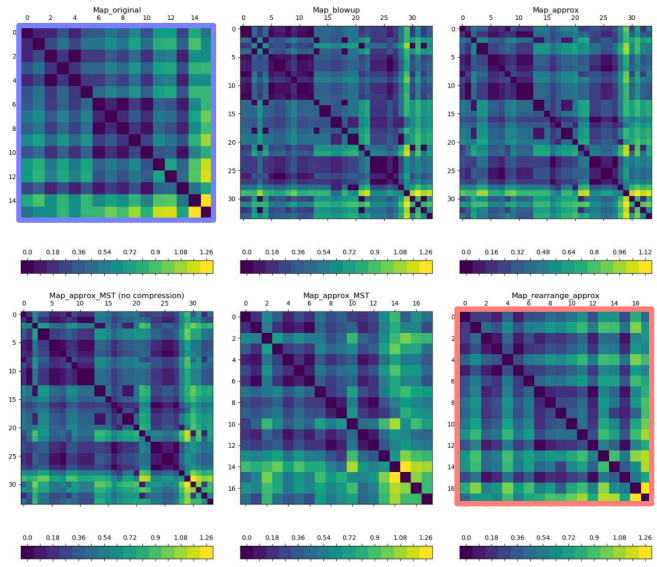

Fig. 22: *Heated Cylinder*: weight matrices associated with T_{24} during the sketching process with IFS.

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.

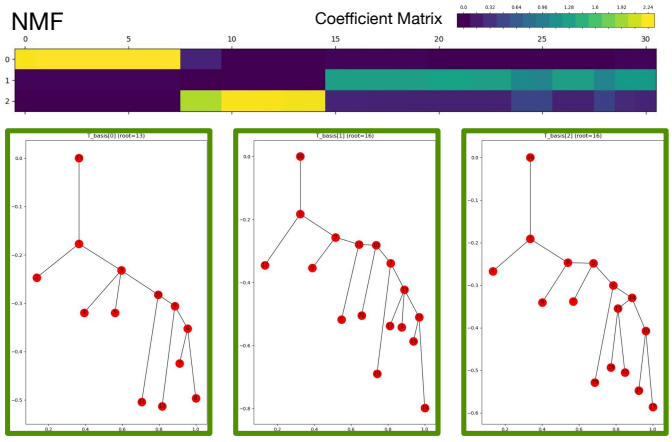

Fig. 23: *Heated Cylinder*: Coefficient matrices and basis trees used to sketch the dataset with three basis trees using NMF.

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

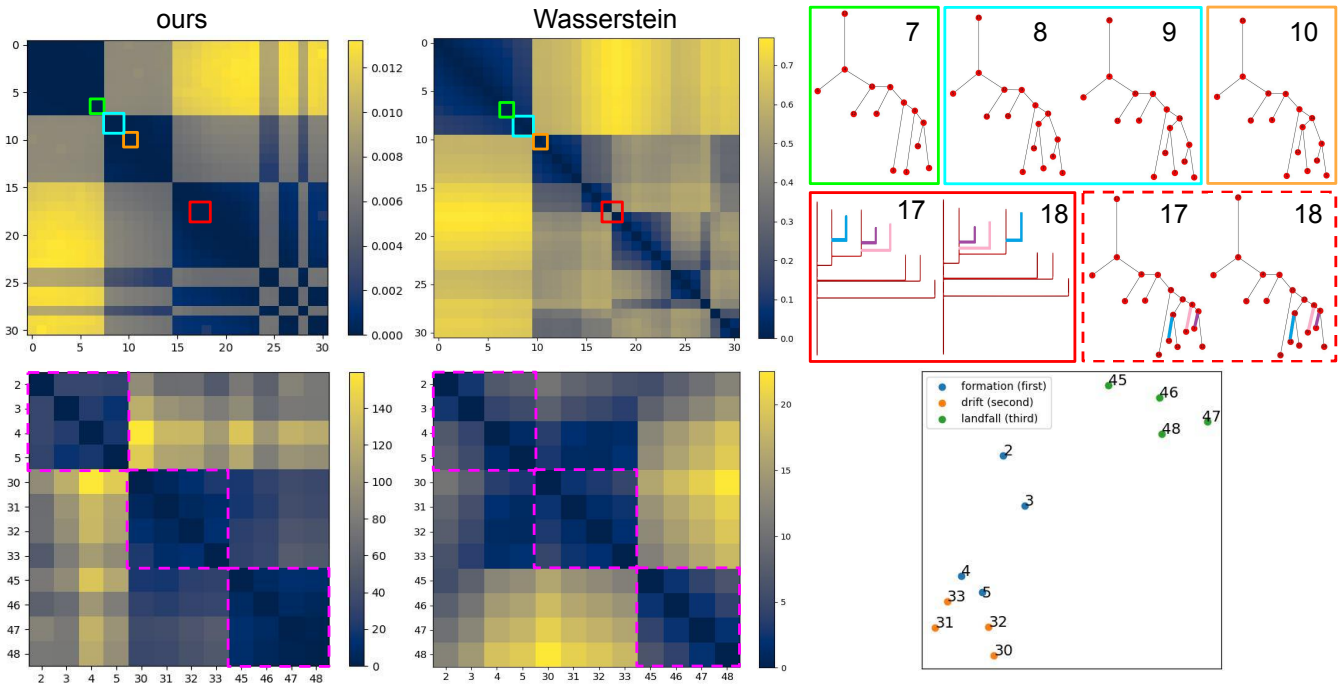

Fig. 24: Experimental comparison with Wasserstein distance for merge trees. Top row: *HeatedCylinder* dataset; left and middle: pairwise distance matrix for GW distance and Wasserstein distance, respectively; right: merge tree in binary tree layout at time steps 7 to 10, and merge tree in both branch 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.