Edit Distances for Comparing Merge Trees

Raghavendra Sridharamurthy* Adhitya Kamakshidasan† Vijay Natarajan‡

Department of Computer Science and Automation
Indian Institute of Science, Bangalore

Abstract

A merge tree captures the topology of sub-level and super-level sets in a scalar field. Estimating the similarity or dissimilarity between merge trees is an important problem with applications to visualization of time-varying and multi-field data. We present a tree edit distance based approach with a general subtree gap model to compare merge trees. The cost model is based on topological persistence. Experimental results on time-varying data show the utility of the method towards a feature-driven analysis of scalar fields.

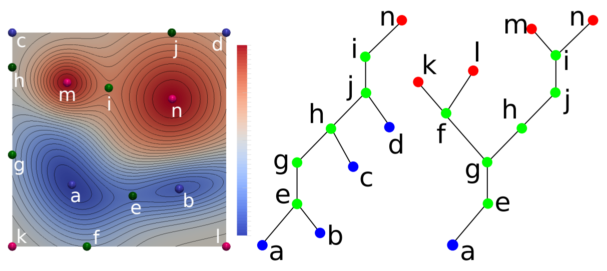
1 Introduction

Scalar functions are used to model many interesting aspects of scientific processes. Topological structures like merge trees [3] (see Figure 1) provide a succinct representation of a scalar function, which can then be used to further analyze the function and gather interesting insights. Multiple scenarios require a method to compare scalar functions. For example, tracking features in time-varying phenomena, identifying preserved features in ensemble simulations, or comparing simulated data against measured data. In all these scenarios, there is a need for a distance measure that captures similarity or dissimilarity. Such distance measures have been studied both from theoretical and practical perspectives. In terms of theoretical work, distance measures have been designed for comparing scalar fields represented by topological structures like the Reeb graph [1], contour tree and merge tree [5] and persistence diagram [4]. These measures satisfy provable properties like stability and discrimination power. But with few exceptions, the measures are not efficiently computable. From a practical perspective, algorithms are available to compute similarity between contour and merge trees [2], and also between extremum graphs [6].

Defining and computing a tree edit distance that allows for gaps is a well-studied problem. The computation has been shown to be NP-hard for arbitrary trees. However, for labeled binary trees a polynomial time, dynamic programming-based algorithm exists [8]. We focus on merge trees, which constitute a special subset of labeled binary trees. Further, we are interested in scenarios where the scalar functions being compared are not significantly different from each other. For example, functions from consecutive time-steps of time-varying data or a function compared against another obtained via a minor perturbation. Given the above assumptions, our aim is to design an effective distance measure using topological persistence. We demonstrate the utility of the measure using a few time-varying data sets.

2 Edit Distance

Informally, distance measures give us an estimate of how "far" two entities are. Tree edit distances are inspired by edit distances developed for strings and applied for string matching. Given two strings,



(a) 2D scalar field (b) join tree (c) split tree

Figure 1: Merge trees. (a) A 2D scalar field (b,c) Merge trees obtained by tracking the connectivity of sub-level sets (preimage of $f^{-1}(-\infty,c]$) or the super-level sets (preimage of $f^{-1}[c,\infty)$).

one is transformed into the other using a sequence of edit operations with non-negative associated costs. The distance is defined as the minimum cost over all transformations. Similar distance measure may be defined for labeled ordered trees with edit operations like relabeling, addition, and deletion of nodes.

Join trees consist of a collection of minima $M = \{m_i\}$, saddles $S = \{s_j\}$, and a global maximum. The structure of a join tree satisfies specific properties. Excluding the golbal maximum, which is the root of the tree, every node has either zero (if it is a minimum) or two children (if it is a saddle). All minima can be paired with saddles based on the notion of topological persistence [4] except for one that persists and can be paired to the lone global maximum. Each such pair (m,s) represents a feature and its persistence can be calculated as $pers(m) = pers(s) = f(s) - f(m)$. Split trees are defined similarly. Join/split trees are together referred to as merge trees.

Xu [8] describes a distance measure with a focus on correctness and worst case runtime analysis for computing the measure. We describe a distance measure that is also based on edit distances with a general subtree gap similar to Xu. However, our focus is on applicability to merge trees. The edit operations are (a) Relabel nodes, (b) Insert a subtree or gap, and (c) Delete a subtree or gap. A gap is defined as collection of nodes that are present in one tree but not in the other. We do not distinguish between starting gaps and continuing gaps.

A key property of the join tree is that it supports only a restricted set of insert/delete operations. Consider a minimum-saddle pair (m,s). Let l denote the parent of s and m' denote the child of s that is neither equal to m nor ancestor of m. If s is deleted, then m should also be deleted, and vice-versa, also m' is paired with l. But deletion of s does not result in the deletion of the entire subtree rooted at s.

Consider two join trees T_1 and T_2 representing scalar functions f_1 and f_2, whose ranges are normalized to lie within [0,1]. Let their vertex sets be V_1 and V_2, with $|V_1| = m, |V_2| = n$. The distance measure is defined on the preorder traversal of the trees. A relabel cost $r(i,j)$ is included if a node i from T_1 is relabeled to node j in T_2 and a gap cost $g(i)$ is included when a node part of a gap.

Given $1 \le i' \le i \le m$ and $1 \le j' \le j \le n$, the edit distance is defined as

*e-mail: raghavendrag@iisc.ac.in
†e-mail: adhitya@iisc.ac.in
‡e-mail: vijayn@iisc.ac.in

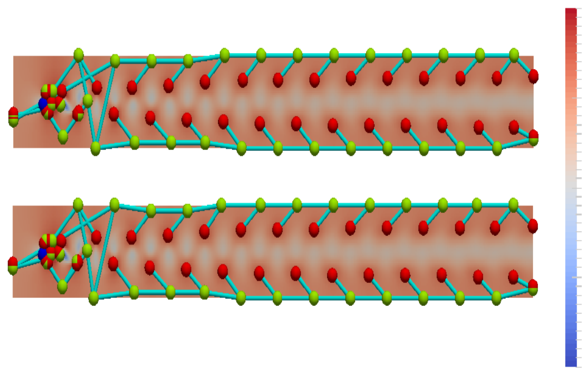


Figure 2: Time-step 0 (top) and 74 (bottom) of the flow around a cylinder simulation. The split tree along with the critical points is overlayed.

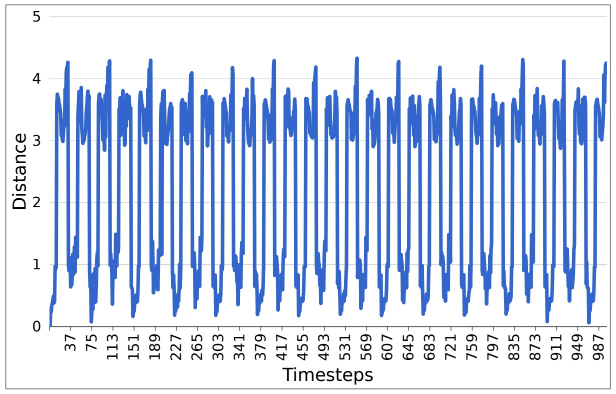


Figure 3: The plot shows distances computed between time step 0 and time steps 0-1000. The time steps and the distances are indicated on the x-axis and the y-axis respectively. From this plot, a time period of 74-75 can be identified.

$$\mathscr{D}[i'..i, j'..j] = \min \begin{cases} \mathscr{D}[i'..i-1, j'..j-1] + r(i,j), \\ \mathscr{D}_{\perp *}[i'..i, j'..j], \\ \mathscr{D}_{* \perp}[i'..i, j'..j], \end{cases}$$

If nodes corresponding to both i and j exist, then first expression gives the relabel cost, else depending on whether i or j is a gap node, $\mathscr{D}_{\perp *}$ or $\mathscr{D}_{* \perp}$ are used. $\mathscr{D}_{\perp *}$ and $\mathscr{D}_{* \perp}$ are defined based on the gap model and details are provided in Xu [8].

We propose to utilise well-known properties of merge trees to define appropriate costs for the edit operations.

Relabel cost. If the node $i \in V_1$ is matched with node $j \in V_2$, define the cost of relabelling i to j as $r(i,j) = |f_1(i) - f_2(j)|$

Add/delete gap cost. We define the cost of a gap node irrespective of whether it is a minimum or a saddle, as the persistence of the feature to which the node belongs to: $g(m) = g(s) = pers(m) = pers(s)$.

The overall cost is given by the minimum edit distance cost $d_e = \mathscr{D}[1..m, 1..n]$ from the algorithm. We compute d_e by incorporating the above-mentioned costs into $\mathscr{D}$ and computing it in a bottom up fashion.

3 Experimental Results

We demonstrate the potential utility of the distance measure by applying it to analyse time-varying data and to study symmetry in scalar fields .

Periodicity in time-varying data. To demonstrate the utility of the measure, we use Bénard von Kárman vortex street data-set formed

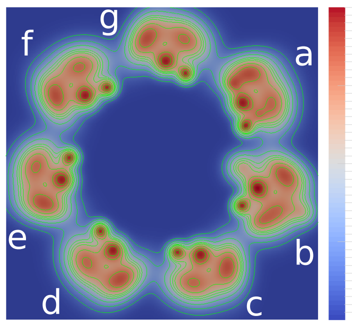


Figure 4: 2D synthetic data-set generated by sum of guassians.

by flow around a cylinder [7]. Figure 2 shows few timesteps of the data-set. The data-set is known to exhibit a periodicity of 75 and [6] detects another period of 38 along with 75. To identify periodicity, we compare the split tree of each time step with all 1000 time steps of the data-set. We plot the distances obtained by our measure in Figure 3 and observe a periodicity between 74 and 75.

Detecting symmetry or asymmetry. We use a synthetic data-set (see Figure 4) that contains seven regions out of which five are symmetrical and the other two (named as ‘a’ and ‘b’) are slightly perturbed to cause asymmetry. We apply our distance measure to compare each subtree corresponding a region with the other and we are able to distinguish between symmetric and asymmetric regions. Region d, for example has a distance close to 0 with regions c, e, f, g; 0.53 with region b and 0.45 with region a. This is consistent with the premise upon which the data is generated.

4 Conclusions

We present a distance measure based on edit distances. We define appropriate costs based on topological properties of the merge trees. The measure is parameter-free. The distance computation is however not real-time. We are currently working toward improving the efficiency of the computation. We believe that adding a geometric cost to the model will make the measure more disriminative. Analyzing the theoretical properties such as stability and optimality are also challenging open problems.

Acknowledgments

We acknowledge support from the Department of Science and Technology, India (DST/SJF/ETA-02/2015-16) and from the Joint Advanced Technology Programme, Indian Institute of Science (JATP/RG/PROJ/2015/16).

References

[1] U. Bauer, X. Ge, and Y. Wang. Measuring distance between reeb graphs. In *Proc. 13th Symp. Comp. Geom*, pages 464–474. ACM, 2014.

[2] K. Beketayev, D. Yeliussizov, D. Morozov, G. H. Weber, and B. Hamann. Measuring the distance between merge trees. In *TopoinVis III*, pages 151–165. Springer, 2014.

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

[4] H. Edelsbrunner and J. Harer. Persistent homology-a survey. *Contemporary mathematics*, 453:257–282, 2008.

[5] D. Morozov, K. Beketayev, and G. Weber. Interleaving distance between merge trees. *Discrete & Computational Geometry*, 2013.

[6] V. Narayanan, D. M. Thomas, and V. Natarajan. Distance between extremum graphs. In *PacificVis*, pages 263–270, 2015.

[7] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Trans. Vis. Comp. Graphics*, 16(6):1225–1234, 2010.

[8] H. Xu. An algorithm for comparing similarity between two trees. *arXiv preprint arXiv:1508.03381*, 2015.