Computing the Gromov-Hausdorff Distance for Metric Trees[⋆]

Pankaj K. Agarwal[1], Kyle Fox[1], Abhinandan Nath[1], Anastasios Sidiropoulos[2], and Yusu Wang[2]

[1] Duke University
{pankaj,kylefox,abhinath}@cs.duke.edu
[2] Ohio State University
sidiropoulos.1@osu.edu, yusu@cse.ohio-state.edu

Abstract. The Gromov-Hausdorff distance is a natural way to measure distance between two metric spaces. We give the first proof of hardness and first non-trivial approximation algorithm for computing the Gromov-Hausdorff distance for geodesic metrics in trees. Specifically, we prove it is NP-hard to approximate the Gromov-Hausdorff distance better than a factor of 3. We complement this result by providing a polynomial time $O(\min\{n, \sqrt{rn}\})$-approximation algorithm where r is the ratio of the longest edge length in both trees to the shortest edge length. For metric trees with unit length edges, this yields an $O(\sqrt{n})$-approximation algorithm.

1 Introduction

The Gromov-Hausdorff distance (or GH distance for brevity) [9] is one of the most natural distance measures between metric spaces, and has been used, for example, for matching deformable shapes [4, 14] and for analyzing hierarchical clustering trees [6]. Informally, the Gromov-Hausdorff distance measures the *additive* distortion suffered when mapping one metric space into another using a correspondence between their points. Multiple approaches have been proposed to estimate the Gromov-Hausdorff distance or provide alternatives to its computation [4, 13, 14].

Despite much effort, the problem of computing, either exactly or approximately, GH distance has remained elusive. On one hand, the problem is not known to be NP-hard, and on the other hand no polynomial-time approximation algorithm exists for graphic metrics [3] unless the graph isomorphism problem

[⋆] Work on this paper by P. K. Agarwal, K. Fox and A. Nath was supported by NSF under grants CCF-09-40671, CCF-10-12254, CCF-11-61359, and IIS-14-08846, and by Grant 2012/229 from the U.S.-Israel Binational Science Foundation. A. Sidiropoulos was supported by NSF under grants CAREER-1453472 and CCF-1423230. Y. Wang was supported by NSF under grant CCF–1319406.

[3] A graphic metric measures the shortest path distance between vertices of a graph with unit length edges.

is in P. (The metrics for two graphs have GH distance 0 if and only if the two graphs are isomorphic.) Motivated by this trivial hardness result, it is natural to ask whether GH distance becomes easier in more restrictive settings such as geodesic metrics over trees.

Our results. In this paper, we give the first non-trivial results on approximating the GH distance between metric trees. First, we prove (in Sect. 3) that the problem remains NP-hard even for metric trees via a reduction from 3-Partition. In fact, we show that there exists no algorithm with approximation ratio less than 3 unless P = NP. As noted above, we are not aware of any result that shows the GH distance problem being NP-hard even for general graphic metrics.

To complement our hardness result, we give an $O(\sqrt{n})$-approximation algorithm for the GH distance between metric trees with n nodes and *unit length* edges. Our algorithm works with arbitrary edge lengths as well; however, the approximation ratio becomes $O(\min\{n, \sqrt{rn}\})$ where r is the ratio of the longest edge length in both trees to the shortest edge length. Even achieving the $O(n)$-approximation ratio present here for arbitrary r is a non-trivial task.

Our algorithm uses a reduction, described in Sect. 4, to the similar problem of computing the *interleaving distance* [15] between two *merge trees*. Given a function $f : \mathbb{X} \to \mathbb{R}$ over a topological space $\mathbb{X}$, the merge tree T_f describes the connectivity between components of the sublevel sets of f. Morozov et al. [15] proposed the interleaving distance as a way to compare merge trees and their associated functions[4]. To take advantage of our reduction from GH distance, we describe, in Sect. 5, an $O(\min\{n, \sqrt{rn}\})$-approximation algorithm for interleaving distance between merge trees. Due to lack of space, most of the proofs have been provided in the full version [1].

Related work. Most work on associating points between two metric spaces involves *embedding* a given high dimensional metric space into an infinite host space of lower dimensional metric spaces. However, there is some work on finding a bijection between points in two given finite metric spaces that minimizes typically multiplicative distortion of distances between points and their images, with some limited results on additive distortion. See [10,12,16] for recent surveys.

The interleaving distance between merge trees [15] was proposed as a measure to compare functions over topological domains that is stable to small perturbations in a function. Distances for the more general Reeb graphs are given in [3,8]. These concepts are related to the GH distance (Section 4), which we will leverage to design an approximation algorithm for the GH distance for metric trees.

2 Preliminaries

Metric Spaces and the Gromov-Hausdorff Distance. A *metric space* $\mathcal{X} = (X, \rho)$ consists of a (potentially infinite) set X and a function $\rho : X \times X \to$

[4] In fact, our hardness result can be easily extended to the GH distance between discrete tree metrics and the interleaving distance between merge trees.

$\mathbb{R}_{\geq 0}$ such that the following hold: $\rho(x,y) = 0$ iff $x = y$; $\rho(x,y) = \rho(y,x)$; and $\rho(x,z) \leq \rho(x,y) + \rho(y,z)$.

Given sets A and B, a *correspondence* between A and B is a set $\mathcal{C} \subseteq A \times B$ such that: (i) $\forall a \in A, \exists b \in B$ such that $(a,b) \in \mathcal{C}$; and (ii)$\forall b \in B, \exists a \in A$ such that $(a,b) \in \mathcal{C}$. We use $\Pi(A,B)$ to denote the set of all correspondences between A and B.

Let $\mathcal{X}_1 = (X_1, \rho_1)$ and $\mathcal{X}_2 = (X_2, \rho_2)$ be two metric spaces. The *distortion* of a correspondence $\mathcal{C} \in \Pi(X_1, X_2)$ is defined as:

$$\mathrm{Dist}(\mathcal{C}) = \sup_{(x,y),(x',y') \in \mathcal{C}} |\rho_1(x,x') - \rho_2(y,y')| \ .$$

The *Gromov-Hausdorff distance* [13], d_{GH}, between $\mathcal{X}_1$ and $\mathcal{X}_2$ is defined as:

$$d_{GH}(\mathcal{X}_1, \mathcal{X}_2) = \frac{1}{2} \inf_{\mathcal{C} \in \Pi(X_1, X_2)} \mathrm{Dist}(\mathcal{C}) \ .$$

Intuitively, d_{GH} measures how close can we get to an *isometric* (distance-preserving) embeddding between two metric spaces. We note that there are different equivalent definitions of the Gromov-Hausdorff distance; see e.g, Theorem 7.3.25 of [5] and Remark 1 of [13].

Given a tree $T = (V,E)$ and a length function $l : E \to \mathbb{R}_{\geq 0}$, we associate a metric space $\mathcal{T} = (|T|, d)$ with T as follows. $|T|$ is a geometric realization of T. The metric space is extended to points in an edge such that each edge of length l is isometric to the interval $[0,l]$. For $x, y \in |T|$, define $d(x,y)$ to be the length of the path $\pi(x,y) \in |T|$ which is simply the sum of the lengths of the restrictions of this path to edges in T. It is clear that d is a metric. The metric space thus obtained is a *metric tree*. We denote $\mathcal{T} = (T, d)$, treating T as the same as $|T|$.

Merge Trees and the Interleaving Distance. Let $f : \mathbb{X} \to \mathbb{R}$ be a continuous function from a connected topological space $\mathbb{X}$ to the set of real numbers. The *sublevel set* at a value $a \in \mathbb{R}$ is defined as $F_{\leq a} = \{x \in \mathbb{X} \mid f(x) \leq a\}$. A *merge tree* T_f captures the evolution of the topology of the sublevel sets as the function value is increased continuously from $-\infty$ to $+\infty$. Formally, it is obtained as follows. Let epi$f = \{(x,y) \in \mathbb{X} \times \mathbb{R} \mid y \geq f(x)\}$. Let $\bar{f}$: epi$f \to \mathbb{R}$ be such that $\bar{f}((x,y)) = y$. We may say $\bar{f}((x,y))$ is the *height* of point $(x,y) \in \mathbb{X} \times \mathbb{R}$. For two points (x,y) and (x',y') in $\mathbb{X} \times \mathbb{R}$ with $y = y'$, let $(x,y) \sim (x',y')$ denote them lying in the same component of $\bar{f}^{-1}(y) (= \bar{f}^{-1}(y'))$. Then $\sim$ is an equivalence relation, and the merge tree T_f is defined as the quotient space $(\mathbb{X} \times \mathbb{R}) / \sim$.

Since two components at a certain height can only merge at a higher height and a component can never split as height increases, we get a rooted tree where the internal nodes represent the points where two components merge and the leaves represent the birth of a new component at a local minimum. Figure 1 shows an example of a merge tree for a 1-dimensional function. Note that the merge tree extends to a height of ∞, and our assumption that $\mathbb{X}$ is connected implies we have only one component in $F_{\leq \infty}$. We define the *root* of merge tree T_f to be the node with the highest function value.

Since each point $x \in T_f$ represents a component of a sublevel set at a certain

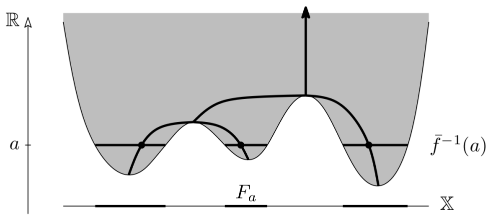

Fig. 1. Merge tree for a function from $\mathbb{R} \to \mathbb{R}$ (image by Morozov et al. [15]).

height, we can associate a height value $\hat{f}(x)$ with x. Given a merge tree T_f and $\epsilon \geq 0$, an ϵ-shift map $i^\epsilon : T_f \to T_f$ maps a point in the tree to its ancestor at height ϵ higher. We thus have $\hat{f}(i^\epsilon(x)) = \hat{f}(x) + \epsilon$. Given $\epsilon \geq 0$ and merge trees T_f and T_g with the associated shift maps i^ϵ and j^ϵ respectively, two continuous maps $\alpha^\epsilon : T_f \to T_g$ and $\beta^\epsilon : T_g \to T_f$ are said to be ϵ-compatible if they satisfy the following conditions

$$\begin{aligned} &\hat{g}(\alpha^\epsilon(x)) = \hat{f}(x) + \epsilon, \forall x \in T_f \ ; \qquad &&\hat{f}(\beta^\epsilon(y)) = \hat{g}(y) + \epsilon, \forall y \in T_g \ ; \\ &\beta^\epsilon \circ \alpha^\epsilon = i^{2\epsilon} \ ; &&\alpha^\epsilon \circ \beta^\epsilon = j^{2\epsilon} \ . \end{aligned}$$

The *interleaving distance* [15] is then defined as

$$d_I(T_f, T_g) = \inf\{\epsilon \mid \text{there exist } \epsilon\text{-compatible maps } \alpha^\epsilon \text{ and } \beta^\epsilon\} \ .$$

Remark. We can relax the requirements on α^ϵ and β^ϵ as follows. Instead of requiring *exact* value changes, we require $\hat{f}(x) \leq \hat{g}(\alpha^\epsilon(x)) \leq \hat{f}(x) + \epsilon$ and $\hat{g}(y) \leq \hat{f}(\beta^\epsilon(y)) \leq \hat{g}(y) + \epsilon$. In addition, as x moves toward the root of T_f, $\alpha^\epsilon(x)$ must move toward the root of T_g (although $\alpha^\epsilon(x)$ may remain constant for a range of x values) and we do not need them to be continuous. A similar rule applies for β^ϵ. Finally, $\beta^\epsilon(\alpha^\epsilon(x))$ must go to an ancestor of x and $\alpha^\epsilon(\beta^\epsilon(y))$ must go to an ancestor of y. Both definitions of interleaving distance are equivalent, and we may use either based on which is more convenient.

As shown in [15], the interleaving distance is a metric and has the desirable properties of being both stable to small function perturbations and more discriminative than the popular bottleneck distance between persistence diagrams [7].

In the remainder of the paper, we will frequently drop the superscript ϵ when it is clear from the context. Also, we may stop alluding to the underlying functions f and g of the merge trees T_f and T_g and simply refer to them as T_1 and T_2. We may also use f and g to sometimes denote the height of the points in the trees themselves.

3 Hardness of Approximation

We show a reduction from the following decision problem called UNRESTRICTED-PARTITION: given a multiset of positive integers $X = \{a_1, \ldots, a_n\}$ with $n = 3m$, is it possible to partition them into m multisets $\{X_1, \ldots, X_m\}$ such that all the elements in each multiset sum to the same quantity $S = \left(\sum_{i=1}^{n} a_i\right)/m$. This problem can be proved to be strongly NP-complete by a reduction from 3-Partition (see [1] for proof), so we can assume that the size of the integers is polynomial in the input.

We construct two trees T_1 and T_2 as follows. Let A and B be two sufficiently large numbers. Let $T_{s,t}$ denote a star graph having t edges of length s. T_1 consists of a node r_1 incident to an edge (r_1, r_1') of length B and to n edges $\{(r_1, p_1), \ldots, (r_1, p_n)\}$ of length 1, where p_i is the center of a copy of T_{A,a_i}. T_2 consists of a node r_2 incident to an edge (r_2, r_2') of length B and to m edges $\{(r_2, q_1), \ldots, (r_2, q_m)\}$ of length 2, where each q_i is the center of a distinct copy of $T_{A+1,S}$. See Fig. 2 for an illustration. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ denote the metric trees associated with T_1 and T_2 respectively. Clearly, this construction can be done in polynomial time.

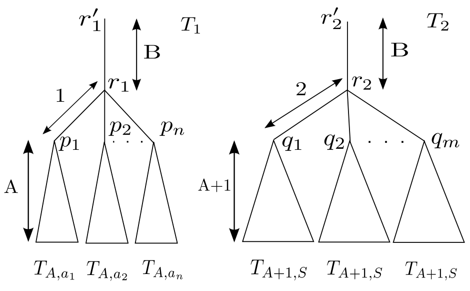

Fig. 2. The trees T_1 and T_2.

Lemma 1. *If the given instance of* UNRESTRICTED-PARTITION *is a* yes *instance, then* $d_{GH}(\mathcal{T}_1, \mathcal{T}_2) \leq 1$. *Otherwise,* $d_{GH}(\mathcal{T}_1, \mathcal{T}_2) \geq 3$.

Proof. (*Yes* instance) We construct a correspondence $\mathcal{C}$ between $\mathcal{T}_1$ and $\mathcal{T}_2$ with distortion at most 2, hence distance at most 1. A linearly interpolated bijection between the points of edges (r_1, r_1') and (r_2, r_2'), with r_1 mapping to r_2 and r_1' mapping to r_2', is added to $\mathcal{C}$. If a_i is assigned to X_j, the linearly interpolated bijection between edges (r_1, p_i) and (r_2, q_j) is added to $\mathcal{C}$. Also, the leaves of T_{A,a_i} are each mapped to a distinct leaf of $T_{A+1,S}$ attached to q_j such that there is a bijection between the leaves of T_1 and T_2 – this can be done since we have a *yes* instance. The interior points of the edges are mapped using linear interpolation. It can be easily verified that the distortion induced by $\mathcal{C}$ is at most 2.

(*No* instance) We show that any correspondence induces a distortion of at least 6, hence distance at least 3. Assume A and B are large enough so that for any correspondence with distortion ≤ 6, we can construct a bijection between the leaf edges of T_1 and T_2 such that two leaf edges are related if the correspondence sends the leaf of one edge to points on the other edge, with (r_1, r_1') mapping to (r_2, r_2'). Since we have a *no* instance, either no such bijection exists or there exists an i such that two leaves of T_{A,a_i} map to points inside leaf edges in $T_{A+1,S}$ attached to q_{j_1} and q_{j_2}, for some $j_1 \neq j_2$. Then the corresponding leaves attached to q_{j_1} and q_{j_2} (say l_1 and l_2 resp.) must map to points l_1' and l_2' inside T_{A,a_i} in T_1. We then have $d_1(l_1', l_2') \leq 2A$ while $d_2(l_1, l_2) = 2A + 6$. The distortion is at least 6.

We may also apply the reduction to metric trees with unit edge lengths by subdividing longer edges with an appropriate number of vertices. We thus have the following theorem.

Theorem 1. *Unless* P = NP*, there is no polynomial-time algorithm to approximate the Gromov-Hausdorff distance between two metric trees to a factor better than 3, even in the case of metric trees with unit edge lengths.*

4 Relating Gromov-Hausdorff and Interleaving Distances

Given a metric tree $\mathcal{T} = (T, d)$, let $V(T)$ denote the nodes of the tree. Given a point $s \in T$ (not necessarily a node), let $f_s : T \to \mathbb{R}$ be defined as $f_s(x) = -\, d(s, x)$. Equipped with this function, we obtain a merge tree T_{f_s} from $\mathcal{T}$. Intuitively, T_{f_s} has the structure of rooting T at s, and then adding an extra edge incident to s with function value extending to $+\infty$. The following theorem, proved in [1], connects the GH distance and the interleaving distance.

Theorem 2. *Let* $\gamma = \min_{u\in V(T_1), v\in V(T_2)} d_I(T_{1 f_u}, T_{2 f_v})$*. Then*

$$\tfrac{1}{2} d_{GH}(\mathcal{T}_1, \mathcal{T}_2) \le \gamma \le 10 d_{GH}(\mathcal{T}_1, \mathcal{T}_2) \ .$$

Corollary 1. *If there is a c-approximation algorithm for the interleaving distance between two merge trees, then there is a* $20c$*-approximation algorithm for the Gromov-Hausdorff distance between two metric trees.*

5 Computing the Interleaving Distance

We propose algorithms for the decision version of the interleaving distance problem, which is stated as follows : *Given two merge trees* T_1 *and* T_2 *and a value* $\epsilon \ge 0$*, compute an* ϵ*-compatible map between them if such a map exists; otherwise report that no such map exists*.

Given two merge trees T_1 and T_2, a *c-approximate decision procedure* for any $c \ge 1$ does the following: if $d_I(T_1, T_2) \le \epsilon$, it returns a pair of $c\epsilon$-compatible maps between T_1 and T_2; if $d_I(T_1, T_2) > \epsilon$ it will either return a pair of $c\epsilon$-compatible maps between T_1 and T_2 or report that no such maps exist. Using binary search, this gives us a c-approximation to $d_I(T_1, T_2)$.

If we know $\alpha^\epsilon(x)$ for a point x at height h, then we can compute $\alpha^\epsilon(y)$ for any ancestor y of x at height $h' \ge h$ by simply putting $\alpha^\epsilon(y) = j^{h'-h} \circ \alpha^\epsilon(x)$. A similar claim holds for β^ϵ. Thus specifying the maps for the leaves of the trees suffices, because any point in the tree is the ancestor of at least one of the leaves. Hence, these maps have a representation that requires linear space in the size of the trees.

We define the *length* of any edge in a merge tree other than the edge to infinity to be the height difference between its two end points. Given a parameter $\epsilon > 0$, an edge is called ϵ*-long*, or *long* for brevity, if its length is greater than 2ϵ. We first describe an exact decision procedure if all edges in both trees are long, and then describe an approximate decision procedure when there are short edges. Finally, we combine the two procedures to handle arbitrary merge trees.

Algorithm for trees with long edges. A *subtree* rooted at a point x in a merge tree T, denoted T^x, includes all the points in the merge tree that are descendants of x and an edge from x that extends upwards to height ∞. For every $x \in T$, the nearest descendant of x (including x) that is in $V(T)$, say $\tau(x)$, is the only node such that $T^x = T^{\tau(x)}$. For $u \in V(T)$, let $C(u)$ denote the children of u.

Assume $d_I(T_1, T_2) \leq \epsilon$, and let $\alpha : T_1 \to T_2$ and $\beta : T_2 \to T_1$ be a pair of ϵ-compatible maps. We define an indicator function $\Phi : T_1 \times T_2 \to \{0, 1\}$ such that $\Phi(u, v) = 1$ if $d_I(T_1^u, T_2^v) \leq \epsilon$ and 0 otherwise. We propose an algorithm to compute $\Phi(u, v)$ for all $u \in V(T_1), v \in V(T_2)$. If $\Phi(u, v) = 1$, the algorithm also computes a pair of ϵ-compatible maps between T_1^u and T_2^v. We are interested in $\Phi(r_1, r_2)$, where r_1 (resp. r_2) is the root of T_1 (resp. T_2).

Lemma 2. *If all the edges are long, the maps α and β induce a bijection between the subtrees rooted at the nodes of T_1 and the nodes of T_2.*

Proof. We define $\Psi_1 : V(T_1) \to V(T_2)$ as follows. Let $u \in V(T_1)$, and let u_p be its parent (for $u = r_1$ we set u_p to be an artificial node at height ∞ above r_1). Let u' be the ancestor of u at height $f(u_p) - 2\epsilon - \epsilon_0$ where ϵ_0 is such that all the children of u_p have height less than $f(u')$ and $\alpha(u') \notin V(T_2)$. We may use the same ϵ_0 for all $u \in V(T_1)$. Set $\Psi_1(u) = \tau(\alpha(u'))$. We prove that $|f(u) - g(\Psi_1(u))| \leq \epsilon$. This is true because all the points in T_1^u map to points in $T_2^{\Psi_1(u)}$ and vice versa, hence $d_I\left(T_1^u, T_2^{\Psi_1(u)}\right) \leq \epsilon$. If $|f(u) - g(\Psi_1(u))| > \epsilon$, the roots of T_1^u and $T_2^{\Psi_1(u)}$ are more than ϵ apart and at least one edge e incident to one of the roots will not be in the image of the corresponding ϵ-compatible map. However, the composition map applied to the lower node incident to e must map it to a point inside e (since the edges are longer than 2ϵ), a contradiction. Define Ψ_2 similarly.

We now prove $\Psi_2(\Psi_1(u)) = u$ for all $u \in V(T_1)$. We know $\beta(\alpha(u'))$ lies on the edge (u_p, u), because $f(u') < f(u_p) - 2\epsilon$. Therefore, $\beta(\Psi_1(u))$ is a descendant of u_p. Because $g(\Psi_1(u)) \geq f(u) - \epsilon$, we further conclude $\beta(\Psi_1(u))$ is an ancestor of u and $\Psi_2(\Psi_1(u))$ is an ancestor of u as well. Since $|f(u) - g(\Psi_1(u))| \leq \epsilon$ and $|g(v) - f(\Psi_2(v))| \leq \epsilon$ for all $u \in V(T_1)$ and $v \in V(T_2)$, we have $|f(u) - f(\Psi_2(\Psi_1(u))| \leq 2\epsilon$. All the edges are longer than 2ϵ, so $\Psi_2(\Psi_1(u)) = u$. We conclude Ψ_1 is a surjection, with Ψ_2 as its inverse. By symmetry, Ψ_2 must be surjective as well, making Ψ_1 a bijection.

Lemma 3. *Suppose all the edges in T_1 and T_2 are long. For any pair of nodes $u \in V(T_1), v \in V(T_2)$, $\Phi(u, v) = 1$ iff all of the following hold : (i) $|f(u) - g(v)| \leq \epsilon$; (ii) $|C(u)| = |C(v)|$; (iii) Let $C(u) = \{u_1, \ldots u_k\}$ and $C(v) = \{v_1, \ldots, v_k\}$, then there exists a permutation π of $[1 : k]$ such that $\Phi(u_i, v_{\pi(i)}) = 1$ for all $i \in [1 : k]$.*

See [1] for a proof. Using Lemma 3, we compute $\Phi(u, v)$ in a bottom-up manner. Suppose we have computed $\Phi(u_i, v_j)$ for all $u_i \in C(u)$ and $v_j \in C(v)$. We compute $\Phi(u, v)$ as follows. If (i) or (ii) of Lemma 3 does not hold for u and v, then we return $\Phi(u, v) = 0$. Otherwise we construct the bipartite graph

$G_{uv} = \{C(u) \cup C(v), E = \{(u_i, v_j) \mid \Phi(u_i, v_j) = 1\}\}$ and determine in $O(k^{5/2})$ time whether G_{uv} has a perfect matching, using the algorithm by Hopcroft and Karp [11]. If G_{uv} has a perfect matching $M = \{(u_1, v_{\pi(1)}), \ldots, (u_k, v_{\pi(k)})\}$, we set $\Phi(u, v) = 1$, else we set $\Phi(u, v) = 0$. If $\Phi(u, v) = 1$, we use the ϵ-compatible maps for $T_{u_i}, T_{v_{\pi(i)}}$, $i \in [1 : k]$, to compute a pair of ϵ-compatible maps between T_1^u and T_2^v, as discussed in the proof of Lemma 3. The theorem below follows (see [1] for the runtime analysis).

Theorem 3. *Given two merge trees T_1 and T_2 and a parameter $\epsilon > 0$ such that all edges of T_1 and T_2 are ϵ-long, then whether $d_I(T_1, T_2) \leq \epsilon$ can be determined in $O(n^{5/2})$ time. If the answer is yes, a pair of ϵ-compatible maps between T_1 and T_2 can be computed within the same time.*

Algorithm for short edges. Given two merge trees, a naive map is to map the lowest among all the leaves in both the trees to a point at height equal to the height of the higher root (see Fig. 3). Thus, all the points in one tree will be mapped to the infinitely long edge on the other tree. This map produces a distortion equal to the height of the trees, which can be arbitrarily larger than the optimum. Nevertheless, this simple idea leads to an approximation algorithm.

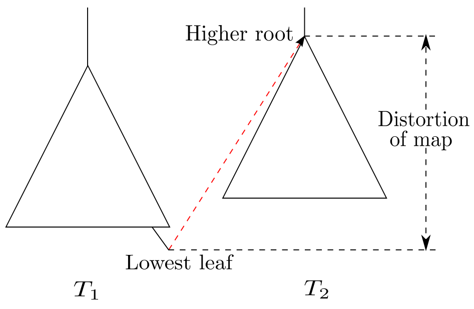

Fig. 3. A naive map.

Here is an outline of the algorithm. After carefully *trimming* off short subtrees from the input trees, the algorithm decomposes them into two kinds of regions – those with nodes and those without nodes. If the interleaving distance between the input trees is small, then there exists an isomorphism between trees induced by the regions without nodes. Using this isomorphism, the points in the nodeless regions are mapped without incurring additional distortion. Using a counting argument and the naive map described above, it is shown that the distortion incurred while mapping the regions with nodes and the trimmed regions is bounded.

More precisely, given T_1, T_2 and $\epsilon > 0$, define the *extent* $e(x)$ of a *point* x (which is not necessarily a tree node) in T_1 or T_2 as the maximum height difference between x and any of its descendants. Suppose each edge is at most $s\epsilon$ long. Let T_1' and T_2' be subsets of T_1 and T_2 consisting only of points with extent at least $2\sqrt{ns}\epsilon$, adding nodes to the new leaves of T_1' and T_2' as necessary. Note that T_1' and T_2' themselves are trees. For example, in Fig. 4 the red points in the left tree are those with extent less than a fixed value, and the right tree is obtained after trimming the red points.

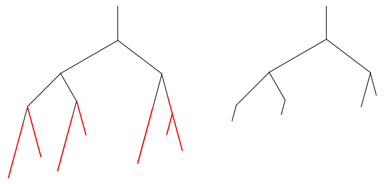

Fig. 4. Tree after trimming red points.

Lemma 4. *If $d_I(T_1, T_2) \leq \epsilon$, then $d_I(T'_1, T'_2) \leq \epsilon$.*

See [1] for a proof of the above lemma. We now define *matching points* in T'_1 and T'_2. A point x in T'_1 is a matching point if there exists a branching node x' in T'_1 or y' in T'_2 with function value $f(x)$ and there exist no branching nodes nor leaves in T'_1 or T'_2 with function value in the range $(f(x), f(x) + 2\epsilon]$. Matching points on T'_2 are defined similarly. By this definition, no two matching points share a function value within 2ϵ of each other unless they share the exact same function value. Furthermore, if x is a matching point, then all points with the same function value as x on both T'_1 and T'_2 are matching points. There are at most $O(n^2)$ matching points.

Suppose $d_I(T'_1, T'_2) \leq \epsilon$, and let $\alpha' : T'_1 \to T'_2$ and $\beta' : T'_2 \to T'_1$ be a pair of ϵ-compatible functions for T'_1 and T'_2. Call a matching point x in T'_1 and a matching point y in T'_2 with $f(x) = g(y)$ *matched* if $\alpha'(x)$ is an ancestor of y.

Lemma 5. *Let x be any matching point in T'_1. The matched relation is a bijective function between matching points in T'_1 with function value $f(x)$ and matching points in T'_2 with function value $f(x)$.*

See [1] for a proof. Let T_1^m be a rooted tree consisting of one node per matching point on T'_1. Let $p(v)$ be the matching point for node v. Tree T_1^m has node v as an ancestor of node u if $p(v)$ is an ancestor of $p(u)$ (see Fig. 5). Define T_2^m similarly. The size of T_1^m and T_2^m is $O(n^2)$.

Intuitively, T_1^m and T_2^m represent the trees induced by matching points. By the definition of interleaving distance and Lemma 5, T_1^m and T_2^m are isomorphic if T'_1 and T'_2 have interleaving distance at most ϵ.

Our algorithm finds an isomorphism between T_1^m and T_2^m in linear time [2]. If one does not exist, then the interleaving distance between T'_1 and T'_2 must be greater than ϵ; by Lemma 4, it thus reports that T_1 and T_2 have interleaving distance greater than ϵ.

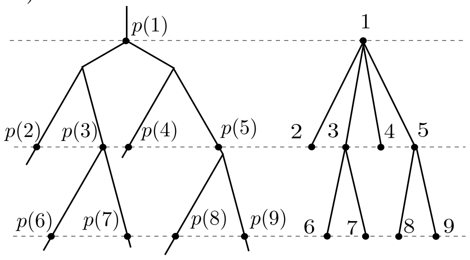

Fig. 5. The left tree shows matching points on tree T'_1 and the right tree shows T_1^m.

If an isomorphism between T_1^m and T_2^m does exist, then the following functions $\alpha : T_1 \to T_2$ and $\beta : T_2 \to T_1$ are returned. For each matched pair of matching points x and y, the algorithm sets $\alpha(x) = y$ and $\beta(y) = x$. Now, let (f_1, f_2) be any maximal range of function values without any branching points in T'_1 or T'_2 where $f_2 - f_1 > 2\epsilon$. Let x' be any point in T'_1 with $f(x') \in (f_1, f_2)$. Point x' has a unique matching point descendant x. The algorithm sets $\alpha(x')$ to the point y' in T'_2 where y' is the ancestor of $\alpha(x)$ with $g(y') = f(x')$, and it sets $\beta(y') = x'$. For every remaining point x'' in T'_1, the algorithm sets $\alpha(x'')$ to $\alpha(x)$ where x is the lowest matching point *ancestor* of x''. Assignment $\beta(y'')$ is defined similarly for remaining points y'' in T'_2. We call such points x'' and y'' *lazily assigned*. Finally, each point x''' in $T_1 - T'_1$ has $\alpha(x''')$ set to $\alpha(x)$ where x is the lowest ancestor of x''' on T'_1. Similar assignments are done for points in $T_2 - T'_2$.

One can verify that α and β meet all their desired properties except for how much a point's function value can change going from one tree to the other. A counting argument gives the following lemma, proved in [1].

Lemma 6. *For each lazily assigned point x'' in T_1', we have $g(\alpha(x'')) \leq f(x'') + 2\sqrt{ns}\epsilon$.*

Theorem 4. *Let T_1 and T_2 be two merge trees and $\epsilon > 0$ a parameter. There is an $O(n^2)$ time algorithm that returns a pair of $4\sqrt{ns}\epsilon$-compatible maps between T_1 and T_2, if $d_I(T_1, T_2) \leq \epsilon$ and the maximum length of a tree edge is $s\epsilon$. If $d_I(T_1, T_2) > \epsilon$, then the algorithm may return no or return a pair of $4\sqrt{ns}\epsilon$-compatible maps.*

Proof. By Lemma 6 and the symmetric lemma for T_2', each point in T_1' and T_2' has its function value changed by at most $2\sqrt{ns}\epsilon$. Points outside T_1' and T_2' have their function value changed by at most $2 \cdot 2\sqrt{ns}\epsilon$.

Remark. If $s = \Omega(n)$, we modify the above algorithm slightly – we skip the trimming step, but keep the rest same. It can be shown, as in Lemma 6, that the height of a point and its image differ by at most $2n\epsilon$.

Overall Algorithm. Given trees T_1 and T_2, let r denote the ratio between the lengths of the longest and the shortest edge in both trees. Our decision procedure works as follows. There are two cases –

Case 1. The shortest edge is longer than 2ϵ. We invoke the procedure for long edges and use Theorem 3.

Case 2. The shortest edge is at most 2ϵ. We invoke the procedure for short edges with $s = 2r$. Using Theorem 4 and the remark following it, we get a $\min(2n, 4\sqrt{2rn})$-approximate decision procedure.

Finally, by plugging this decision into a binary search over all possible candidate values for ϵ, we obtain an approximation algorithm for the interleaving distance. The following lemma, proved in [1], states that the number of candidate values for ϵ is only $O(n^2)$. Thus binary search takes $O(\log n)$ time, and Theorem 5 follows.

Lemma 7. *Let T_1 and T_2 be two merge trees with internal nodes I_1 and I_2 resp. and leaves L_1 and L_2 resp. Then the value of $d_I(T_1, T_2)$ is either*
(i) $|f(u) - g(v)|$ for some pair $(u, v) \in I_1 \times I_2 \cup L_1 \times L_2$, or
(ii) $\frac{1}{2}|f(u) - f(u')|$ for some $u \in L_1$, where u' is an ancestor node of u, or
(iii) $\frac{1}{2}|f(v) - f(v')|$ for some $v \in L_2$, where v' is an ancestor node of v.

Theorem 5. *Given two merge trees T_1 and T_2 with a total of n vertices, there exists an $O(n^{5/2} \log n)$ time $O(\min\{n, \sqrt{rn}\})$-approximation algorithm for computing the interleaving distance between them, where r is the ratio between the lengths of the longest and the shortest edge in both trees.*

Combining Theorem 5 with Corollary 1, we have:

Corollary 2. *Given two metric trees T_1 and T_2 with a total of n vertices, there exists an $O(n^{7/2} \log n)$ time $O(\min\{n, \sqrt{rn}\})$-approximation algorithm for computing the Gromov-Hausdorff distance between them, where r is the ratio between the lengths of the longest and the shortest edge in both trees.*

6 Conclusion

We have presented the first hardness results for computing the Gromov-Hausdorff distance between metric trees. We have also given a polynomial time approximation algorithm for the problem. But the current gap between the lower and upper bounds on the approximation factor is polynomially large. It would be very interesting to close this gap. In general, we hope that our current investigation will stimulate more research on the theoretical and algorithmic aspects of embedding or matching under additive metric distortion.

References

1. P. K. Agarwal, K. Fox, A. Nath, A. Sidiropoulos, and Y. Wang. Computing the Gromov-Hausdorff distance for metric trees. *CoRR*, abs/1509.05751, 2015.
2. A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.
3. U. Bauer, X. Ge, and Y. Wang. Measuring distance between Reeb graphs. In *30th Annual Sympos. on Comput. Geom.*, page 464, 2014.
4. A. M. Bronstein, M. M. Bronstein, and R. Kimmel. Efficient computation of isometry-invariant distances between surfaces. *SIAM J. on Sci. Comput.*, 28(5):1812–1836, 2006.
5. D. Burago, Y. Burago, and S. Ivanov. *A Course in Metric Geometry.* American Mathematical Society, 2001.
6. G. Carlsson and F. Mémoli. Characterization, stability and convergence of hierarchical clustering methods. *J. of Mach. Learn. Res.*, 11:1425–1470, 2010.
7. D. Cohen-Steiner, H. Edelsbrunner, and J. Harer. Stability of persistence diagrams. *Disc. Comput. Geom.*, 37(1):103–120, 2007.
8. V. de Silva, E. Munch, and A. Patel. Categorification of Reeb graphs. Preprint, 2014.
9. M. Gromov. *Metric Structures for Riemannian and Non-Riemannian Spaces.* Birkhäuser Basel, 2007.
10. A. Hall and C. Papadimitriou. Approximating the distortion. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, volume 3624 of *Lecture Notes in Computer Science*, pages 111–122. Springer Berlin Heidelberg, 2005.
11. J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. on Comp.*, 2(4):225–231, 1973.
12. C. Kenyon, Y. Rabani, and A. Sinclair. Low distortion maps between point sets. *SIAM J. on Comp.*, 39(4):1617–1636, 2009.
13. F. Memoli. On the use of Gromov-Hausdorff distances for shape comparison. In *Eurographics Symposium on Point-Based Graphics*, 2007.
14. F. Mémoli and G. Sapiro. A theoretical and computational framework for isometry invariant recognition of point cloud data. *Found. of Comput. Math.*, 5(3):313–347, 2005.
15. D. Morozov, K. Beketayev, and G. H. Weber. Interleaving distance between merge trees. In *Workshop on Topological Methods in Data Analysis and Visualization: Theory, Algorithms and Applications*, 2013.
16. C. Papadimitriou and S. Safra. The complexity of low-distortion embeddings between point sets. In *16th Annual ACM-SIAM Symp. on Discrete Algo.*, pages 112–118, 2005.