The power of two min-hashes for similarity search among hierarchical data objects

Sreenivas Gollapudi
Microsoft Research
Mountain View, CA 94043
sreenig@microsoft.com

Rina Panigrahy
Microsoft Research
Mountain View, CA 94043
rina@microsoft.com

Abstract

In this study we propose sketching algorithms for computing similarities between hierarchical data. Specifically, we look at data objects that are represented using leaf-labeled trees denoting a set of elements at the leaves organized in a hierarchy. Such representations are richer alternatives to a set. For example, a document can be represented as a hierarchy of sets wherein chapters, sections, and paragraphs represent different levels in the hierarchy. Such a representation is richer than viewing the document simply as a set of words. We measure distance between trees using the best possible super-imposition that minimizes the number of mismatched leaf labels. Our distance measure is equivalent to an Earth Mover's Distance measure since the leaf-labeled trees of height one can be viewed as sets and can be recursively extended to trees of larger height by viewing them as set of sets. We compute sketches of arbitrary weighted trees and analyze them in the context of locality-sensitive hashing (LSH) where the probability of two sketches matching is high when two trees are similar and low when the two trees are far under the given distance measure. Specifically, we compute sketches of such trees by propagating min-hash computations up the tree. Furthermore, we show that propagating one min-hash results in poor sketch properties while propagating two min-hashes results in good sketches.

1 Introduction

The proliferation of information on the internet creates a huge amount of data in different formats. In the absence of rigid standards, *ad hoc* methods are used by different entities to represent data. For instance, a book may be represented hierarchically as a collection of chapters and these in turn as a collection of sections. Such an abstraction may be represented as a tree in the emerging XML standard. Similarly directory structures may be viewed as hierarchical entities that can be communicated in different formats. It then becomes a challenge to compare two such information corpora. For example, the same information can be represented under different isomorphic permutations of the underlying directory structure. A database record could be presented with attributes in different order. In such cases, rather than looking at the exact tree structure representing the objects, it may be more appropriate to check the similarity of the two tree structures under all possible isomorphic permutations. Again, to measure the similarity between two directory structures, it may be more appropriate to ignore the directory names and check if there is an ordering of the directory structure that minimizes mismatch between the file contents. However, computing the ordering that minimizes the mismatch on the tree data becomes intractable as the degree and height of the tree increase.

Sketching is a powerful tool for concise representation and comparison of complex and large data. For instance, strings and sets can be compressed into points in hamming space. This typically yields sketches represented by bit-vectors. Such sketches, although are not identical to the original data, preserve many of their properties. For instance sketches can be used to measure containment and distance [2]. Given the plethora of applications for sketching, there have been several attempts to compute sketches of more complex data such as trees and graphs. Some examples include file system directories, XML DOM trees, and phylogenetic trees [19, 14, 1, 6, 18]. Other works of research have presented sketch based measurement of similarity between trees [1, 5]. Another important application of sketching sets is comparison of documents such as web pages wherein a document is viewed as a set of words. However, we lose the structure of the document by viewing it as a set. A hierarchical structure is richer in capturing context in which words occur such as a paragraph, chapter, etc. In this work, we propose a sketch based algorithm via an Earth Mover's Distance metric for trees

with provable guarantees.

1.1 Related Work

Similarity among trees has been widely studied in the context of edit distance between trees, i.e., the number of edits required to transform one tree into the other [13, 17, 18] while others have approached this problem of matching via alignment of trees and effectively computing the best alignment that minimized the objective function [12]. The distance function adopted in this work is similar in spirit and is based on the optimal super-imposition that minimizes the number of mismatches in the set of leaf labels between the two trees. A more recent work of Augsten et al [1] uses pq-grams to match hierarchical data. A sketch of the tree is composed of a set of pq-gram profiles and sketch similarity is computed using well known set similarity measures. This approach, however, is not resistant to permutations in the leaf labels when $q = 1$. The other case of $q > 1$ generates mismatches that are amplified by the shingling approach adopted to generate the pq-grams, thus resulting in a poor match between trees even when they are similar.

1.2 Contributions of this study

We introduce the distance between arbitrary weighted trees as the best possible super-imposition that results in the maximum number of matches between the trees. Specifically, we look at data objects that are represented using leaf-labeled trees denoting a set of elements at the leaves organized in a hierarchy. Our distance measure is equivalent to an Earth Mover's Distance measure since the leaf-labeled trees of height one can be viewed as sets and can be recursively extended to trees of larger height by viewing them as set of sets. For weighted trees, this is equivalent to recursive fractional weighted matching between the leaves according to the tree hierarchy. We propose sketching algorithms that generalize the concept of min-wise independent permutations to trees. We analyze the sketching algorithms in the framework of locality-sensitive hashing and show how our sketch functions can be used to perform nearest-neighbor search among trees.

Specifically, we use sketching algorithms on sets to compute their min-hashes and then propagate the min-hashes up the hierarchy. We show that propagating only one min-hash can result in poor similarity guarantees. For example, two trees that are far apart can produce very similar sketches using this scheme. Finally, we show that by propagating two min-hashes for each set of labels, we can compute similarity accurately between two trees in the context of locality-sensitive hashing. Although, our provable guarantees hold only under the assumption that the trees are of uniform height and leaves are uniquely labeled, the algorithms presented are useful in more general practical scenarios.

Specifically, for our distance measures we show how to find an approximate nearest neighbor of a query tree. Given a database of N trees, we can construct a data structure of size $N^{1+\rho}$ that can be used to compute approximate nearest neighbor in time N^{ρ}, where $\rho = O(\frac{1-\log(1-\delta)}{\log(1/\epsilon)})$. Given a query point whose nearest neighbor is within distance δ, our search algorithm will return a neighbor at distance at most $i - \epsilon$ from the query point; essentially, our algorithm returns a $\frac{1-\epsilon}{\delta}$-approximate nearest neighbor when the query point has a δ-near neighbor.

2 Models and Definitions

In this section, we present the preliminaries for hierarchies as well as introduce notions of similarities between hierarchies. We classify trees as either labeled or unlabeled and ordered or unordered. In the labeled case, all nodes in the tree are labeled. A common example of such a hierarchy is a file system directory structure. Leaf-labeled trees where only the leaves are labeled have no labels on the internal nodes. For example, we could consider only the names and content of all files under a directory to compute the similarity between two directories. In such a case, we ignore the names of the directories themselves. An ordered tree has all leaves in a defined order. For a tree of height h (a path from the root to a leaf has h edges), we will say that the leaves are at height 0 and the root is at height h. For example, a string can be considered an ordered tree of height one.

Observe that in practice two trees are considered similar even if one is obtained by a isomorphic reordering of nodes. In this case, we assume the trees to be unordered. We also assume that leaves are uniquely labeled.

Before we analyze the similarity measures for leaf-labeled trees, we introduce well-studied notions of similarities between unordered entities such as sets and how these notions have been successfully used in defining similarity measures between ordered sequences such as strings. Set similarity is computed using measures such as intersection and (symmetric) difference. String similarity is measured using edit distances. While there are many efficient algorithms for sketching sets such as min-hash computation [3], obtaining efficient sketching algorithms for strings with strong guarantees is much harder. The best known algorithm for sketching strings work by embedding strings into L_1 and the best known

embedding has distortion $2^{O(\sqrt{\log n \log\log n})}$ [15] which is far less efficient compared to sketching algorithms for sets which produce almost no distortion.

Many of these algorithms are based on shingling [2], which essentially transform an ordered sequence of characters into a set of substrings known as *shingles*. Then by applying standard sketching algorithms on these sets the similarity between the original strings is estimated.

In this work, we present algorithms for unordered leaf-labeled weighted trees. We believe our algorithms may be extended to ordered trees just as shingling based methods extend algorithms on sets to algorithms on strings. Furthermore, we consider general trees with arbitrary shape with non-uniform degrees but of the same height. More generally, we allow the nodes to be weighted where the total weight of all children under any given node is 1.

Before we analyze similarity measures for trees, we introduce some of the basic concepts we employ in our sketching algorithms. First is the concept of sketching sets using the technique of min-wise independent permutations which are often referred to as min-hash [3]. The commonly used similarity measure for sets A and B is $sim(A,B) = \frac{|A\cap B|}{|A\cup B|}$ [3]. A simple method for estimating the similarity between two sets (or bags) is the *min-hash* technique introduced by [10].

A weighted tree of height one can be viewed as a weighted set (a bag) instead of a set with leaves at height 0. In this case, the similarity measure is computed via min-hash computations on bags. This extension can be easily introduced by replacing a bag $A = \{(a_1,\alpha_1),\ldots,(a_n,\alpha_n)\}$ with integral weights by a set $\tilde{A} = \{a_{1,1}, a_{1,2},\ldots,a_{1,\alpha_1},\ldots,a_{n,1},a_{n,2},\ldots,a_{n,\alpha_n}\}$ where $a_{i,j}$ is an element obtained by concatenating the bag element a_i with a frequency j. Note that this transformation can be extended to bags with real weights as well. Under this transformation, we can easily show $|A\cap B| = |\tilde{A}\cap\tilde{B}|$ and $|A\cup B| = |\tilde{A}\cup\tilde{B}|$ [10, 9] where intersections(unions) over bags is obtained by assigning to each element a weight equal to the minimum(maximum) of its weights in the two bags $\tilde{A}$ and $\tilde{B}$. Thus, under a given permutation $\pi : [|U|] \to [|U|]$, we have $\Pr[MH_\pi(A) = MH_\pi(B)] = \frac{|A\cap B|}{|A\cup B|} = \Pr[MH_\pi(\tilde{A}) = MH_\pi(\tilde{B})] = \frac{|\tilde{A}\cap\tilde{B}|}{|\tilde{A}\cup\tilde{B}|}$ [10, 9].

Definition 1. *Let U denote the universal set. Given a set $A \subseteq U$ and a permutation $\pi : [|U|] \to [|U|]$, we define the min-hash $MH_\pi(A)$ to be $argmin_x\{\pi(x)|x \in A\}$. Essentially, $MH_\pi(A)$ is the element in A whose value in the permutation is the minimum. Alternately, let $f(x), x \in U$ be a real valued (hash) function that maps elements from the universe U to a real number randomly and uniformly in the interval $[0,1]$. Then $MH_f(A) = argmin_x\{f(x)|x \in A\}$. Therefore, $MH_f(A)$ is the element in A whose hash value into the interval $[0,1]$ is minimum. Note that this definition of min-hash can be applied to weighted sets or bags as well.*

A common measure used to compute similarity between two sets of points, where the points are chosen from a metric space, is the *Earth Mover's Distance*. Here each set can be viewed as a distribution of weights over the metric space where the sum of the weights adds up to 1. We now define the Earth Mover's Distance between two such distributions.

Definition 2 ([4])**.** *Let (X,d) be a metric on a set X of weighted elements (x_i,w_i), $1 \le i \le n$, where $d(.,.)$ is the underlying distance measure between elements. Let $P(X)$ denote the a distribution of non-negative weights $u_1,u_2,\ldots,u_n$ on X such that $\sum_i u_i = 1$. The Earth Mover's Distance is a measure between two such distribution $P(X) = u_1,u_2,\ldots,u_m$ and $Q(X) = v_1,v_2,\ldots,v_m$. Specifically, the Earth Mover's Distance defines the optimal cost of transforming $P(X)$ into $Q(X)$ and can be formally stated as*

$$\begin{aligned} EMD(P(X),Q(X)) = \quad & min \sum_{ij} f_{ij} d(i,j) \\ \forall i \textstyle\sum_j f_{ij} = u_i & \\ \forall j \textstyle\sum_i f_{ij} = v_j, \qquad & and \\ \forall i,j \ f_{ij} \ge 0 & \end{aligned}$$

In fact the EMD corresponds to a fractional weighted matching in a bi-partite graph with the nodes on either side of an edge corresponding to elements of X with weights u_i and u_j respectively. Thus, the total weight of the matching edges at a node equals the weight of the point. This distance measure can be generalized to a distance measure on leaf-labeled trees of same height. Observe that a tree of height one can be viewed as a weighted set $(X,P(X))$ where X is the set of leaf labels and $P(X)$ denotes the weights on the leaves. The above EMD measure defines the distance between two such trees $(X,P(X))$ and $(X,Q(X))$ of height one. This can be extended to trees of larger height easily by treating X as the set of subtrees at height $h-1$ and the weights being assigned to subtrees instead of leaves in trees of height one (leaves are at height 0).

Sketching functions can be used to estimate similarity between two objects by comparing their sketches. Given a sketching function that results in a low matching probability when the underlying objects are dissimilar and a higher probability when the objects are similar, we can construct efficient data structures for approximate nearest neighbor search on a database of objects. Such sketching functions are also called *locality sensitive hash*(LSH) functions[7]. For a domain X

of points with distance measure d, an LSH family of functions is defined as follows.

Definition 3. *A family $\mathcal{H} = h : X \to U$ is called (r_1, r_2, p_1, p_2)-sensitive for d if for any $v, q \in X$*

- *if $d(v,q) \leq r_1$, then $\Pr_{\mathcal{H}}[h(q) = h(v)] \geq p_1$.*
- *if $d(v,q) > r_2$, then $\Pr_{\mathcal{H}}[h(q) = h(v)] \leq p_2$.*

Indyk and Motwani [11] show how LSH can be used for nearest neighbor searches in high dimensions.

Theorem 1 ([7]). *Suppose there is a (R, cR, p_1, p_2)-sensitive family $\mathcal{H}$ for a distance measure d. Given a query point with a nearest neighbor within distance R, there is an algorithm to compute a c-approximate nearest neighbor, i.e., a neighbor within distance cR using an index of $O(N^{1+\rho})$ space, and query time dominated by $O(N^{\rho})$ distance computations where N is the size of the database of points and $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$ for $N > 1/p_2$.*

An information theoretic formulation of LSH resulting in linear size data structures has been studied in [16].

3 Similarity measures for trees

In this study we focus on weighted leaf-labeled trees of arbitrary shape with a given height h. The notion of similarity between two such trees is measured by the best possible super-imposition that minimizes the number of mismatched leaf labels. Given two trees of height h, we view them as weighted sets of trees of height $h-1$. Given that the total weight of all children under any given node is at most 1, we can recursively extend the distance measure on trees of height h as the Earth Mover's Distance between their weighted sets of trees of height $h-1$.

Definition 4 (distance). *For two trees T_1 and T_2 of height zero (trees are singleton leaves), the distance $d(T_1, T_2) = 0$ if the singleton leaves have the same label and 1 otherwise. Using this base case, we can recursively define the distance between two trees of height h by viewing such trees as a weighted set of trees of height $h-1$. If X denotes the space of all leaf-labeled trees of height $h-1$, then any such weighted set with total weight 1 can be represented by a distribution. So, for two trees T_1 and T_2 of height h, we can obtain the corresponding distributions $P(X)$ and $Q(X)$ over trees of height $h-1$. We then define $d(T_1, T_2) = EMD(P(X), Q(X))$ where the underlying metric uses the distance between subtrees of height $h-1$ recursively. More precisely, let $T_1 = \{(a_1, \alpha_1), \ldots, (a_n, \alpha_n)\}$ where a_i is a subtree of height $h-1$ with weight α_i and let $T_2 = \{(b_1, \beta_1), \ldots, (b_n, \beta_n)\}$ with b_j and β_j correspondingly defined. Then, $d(T_1, T_2) = \min \sum_{ij} f_{ij} d(a_i, b_j)$ where $\forall i, \sum_j f_{ij} = \alpha_i$, $\forall j \sum_i f_{ij} = \beta_j$, and $\forall i, j \ f_{ij} \geq 0$.*

Definition 5 (similarity). *For trees T_1 and T_2 of height h, the similarity between them is defined as $sim(T_1, T_2) = 1 - d(T_1, T_2)$. Alternately, similarity can also be defined recursively as the maximum weighted fractional matching between the weighted sets of trees of height $h-1$.*

We say that a tree T_2 is a δ-near neighbor of tree T_1 if $d(T_1, T_2) < \delta$. Similarly, a tree T_2 is $1-\epsilon$-far from a tree T_1 if $sim(T_1, T_2) < \epsilon$.

Under our distance measure, the distance between any two trees lies in $[0, 1]$. Given two trees of height 1, let $\tilde{A}$ and $\tilde{B}$ be their weighted set of leaves (with total weight 1 each). Then the above similarity measure coincides with the bag intersection, i.e. $sim(T_1, T_2) = |\tilde{A} \cap \tilde{B}|$. This implies $|\tilde{A} \cap \tilde{B}| = 1 - d(T_1, T_2)$ and $|\tilde{A} \cup \tilde{B}| = 1 + d(T_1, T_2)$. We also note that for complete unweighted trees of uniform degree, the distance measure $d(.,.) \in [0, 1]$ is indeed the fraction of unmatched leaf labels under the best possible super-imposition. For two unweighted trees of height 1, with the same number of leaves, the EMD measure is the same as the fraction of unmatched leaf labels. This is because for such trees, the best fractional matching becomes an integral matching. The same reasoning applies recursively for larger heights for two complete trees of uniform degree.

Since the object of interest in this study are trees, our goal is to study tree sketching algorithms in the LSH framework. We will refer to such a sketch of tree T as *tree-hash* $TH(T)$. The framework studies the probability of the sketches matching under a gap in the distance between the trees. Using theorem 1, this framework can be used to find a near neighbor of a query tree. Given a database of N trees, we can construct a data structure of size $N^{1+\rho}$ that can be used to compute approximate nearest neighbor in time N^{ρ}, where $\rho = O(\frac{1-\log(1-\delta)}{\log(1/\epsilon)})$. Given a query point whose nearest neighbor is within distance δ, our search algorithm will return a neighbor at distance at most $i - \epsilon$ from the query point; essentially, our algorithm returns a $\frac{1-\epsilon}{\delta}$-approximate nearest neighbor when the query point has a δ-near neighbor. Formally, we will lower bound the probability p_1 of the sketches matching when the $d(T_1, T_2) < \delta$ and upper bound the probability p_2 when the $d(T_1, T_2) > 1 - \epsilon$ (correspondingly, $sim(T_1, T_2) \leq \epsilon$). Given two trees T_1 and T_2 of same uniform height, let s_{ij} denote $sim(T_{1i}, T_{2j})$, where T_{ki} denotes the ith sub-tree of tree T_k. We have,

Lemma 1. *$\sum_i s_{ij} \leq 1$ and $\sum_j s_{ij} \leq 1$.*

Proof. We will say that two trees are disjoint if their set of leaf labels are disjoint. Since all the leaves of a tree are uniquely labeled, we observe that all the sub-trees of T_1 of a given height are disjoint. The lemma follows if we show that for any tree U of height h and a collection of disjoint trees (leaf sets are disjoint), $V_1, V_2, \ldots, V_n$, of same height h, $\sum_i sim(U, V_k) \leq 1$. We will prove this by induction on height h. The base case when $h = 0$ is obvious since for $h = 0$, each tree V_k is a singleton leaf with a distinct label. Therefore, $\sum_i sim(U, V_k) \leq 1$ since U's label can match the label of at most one of V_k's. We use induction on the height of the tree. Note that U can be viewed as a weighted set of trees u_i of height $h-1$ with weight α_i. Therefore $U = \{(u_1, \alpha_1), \ldots, (u_n, \alpha_n)\}$. Similarly, let v_{kj} be the sub-trees of height $h-1$ of V_k. Now, consider the fractional matching f_{ij}^k which corresponds to the similarity $sim(U, V_k)$. Then,

$$sim(U, V_k) = \sum_{ij} f_{ij}^k sim(u_i, v_{kj})$$

But $f_{ij}^k \leq \alpha_i$ since $\sum_j f_{ij}^k \leq \alpha_i$. So,

$$\begin{aligned} \sum_k sim(U, V_k) &= \sum_k \sum_{ij} f_{ij}^k sim(u_i, v_{kj}) \\ &\leq \sum_i \sum_{kj} \alpha_i sim(u_i, v_{kj}) \\ &= \sum_i \alpha_i \sum_{kj} sim(u_i, v_{kj}) \end{aligned}$$

Observe that trees v_{kj} are disjoint. So, by induction $\sum_{kj} sim(u_i, v_{kj}) \leq 1$ giving us $\sum_k sim(U, V_k) \leq \sum_i \alpha_i \leq 1$. □

4 Sketching algorithms for trees of height two

Trees of height one are same as sets and therefore, sketching such trees are equivalent to sketching a set. We note that trees of height two are good abstractions for recursing the similarity computations from a set to set of sets. Therefore, they present a good starting point for experimenting with different approaches. We will show a sketching algorithm for trees of height two where the sketches match with high probability if the trees are similar and with low probability if the trees are far; if the trees are δ-near, the probability that the sketches differ is at most $O(\delta)$. On the other hand, if the trees are $(1-\epsilon)$-far, the probability that the sketches match is at most $O(\epsilon \log(1/\epsilon))$. In the following section, we contrast different algorithms starting from naive extensions of sketching algorithms on sets to algorithms on set of sets. We generalize many of the concepts introduced for trees of height two to trees of larger height in Section 5. We show how algorithms for sketching sets can be extended recursively to such trees. We start with a naive approach and show that it does not admit accurate similarity computations. We then show an approach that does produce good sketches resulting in effective similarity computations.

4.1 Propagating one min-hash does not work

A naive recursive extension is to compute the min-hash for each sub-tree of height one and then compute the min-hash of the resulting min-hashes at level one. Where applicable, a tree T of height one may also be viewed as a weighted set of its leaves. Furthermore, we use a different min-hash permutation at each level. We will show that this method does not work as two completely different trees can result in the same min-hash with high probability.

Let π_1 and π_2 denote the min-hash permutation used on the leaves and nodes at height one respectively. Then, if $T_1, T_2, \ldots, T_n$ are the height one subtrees of T, the tree hash $TH(T) = MH_{\pi_2}(MH_{\pi_1}(T_1), \ldots, MH_{\pi_1}(T_n))$. The following two propositions show that a random assignment of a given set of labels results in very different trees, but with a same tree-hash with significant probability. For this we will consider trees of height two that are unweighted and have degree n with n^2 leaves.

Proposition 1. *For given permutations π_1 and π_2 on the two levels of the tree and a set of n^2 labels, if the labels are randomly assigned without replacement to the n^2 leaves, there is some fixed label among these so that with at least constant probability $\Omega(1)$, the min-hash of the tree will result in that label.*

Proof. Let $l_1, l_2, \ldots, l_{n^2}$ denote the labels according to the min-hash permutation order π_1. We will show that with constant probability the tree-hash will be equal to $MH_{\pi_2}(l_1, l_2, \ldots, l_{n^2})$. Essentially, we will show that the set of min-hashes at level one has a big overlap with the set $H = \{l_1, l_2, \ldots, l_n\}$. Precisely, if A is the set of min-hashes at level one, then $E[|A \cap H|] \geq n/2$. This is because the probability of l_1 being present in H is 1; l_2 is present in H if l_1 is not in the same subtree which happens with probability at least $1 - 1/n$. Similarly, l_i $(i \leq n)$ is present in H with probability at least $1 - (i-1)/n$. By linearity of expectations, the $E[|A \cap H|] \geq 1 + (1 - 1/n) + \ldots + 0 \geq n/2$.

From set similarity, it follows that $\Pr[MH_{\pi_2}(A) = MH_{\pi_2}(H)] = |A \cap H|/|A \cup H| > \frac{n/2}{2n} = 1/4$. □

Proposition 2. *Given two different random assignments (without replacement) of a given set of n^2 labels, then w.h.p., the distance between the resulting trees is $1 - o(1)$.*

Proof. Consider one set of leaves in a subtree. The expected number of leaves from this set present in any given subtree of the other tree is 1. By Chernoff bounds, w.h.p, the overlap between any two subtrees across the trees is $O(\log n)$. So, even under the best super-imposition, the distance is $1 - \frac{O(\log n)}{n}$. □

Given two random trees of degree n and height two, they have the same hash value with high probability $\Omega(1)$ but have a large distance with high probability.

4.2 Propagating multiple min-hashes at each height

We now consider the following modification to the naive algorithm wherein we compute more than one min-hash at all nodes in the tree. Specifically, each height one node computes c min-hashes of the set of leaves in its subtree. The compound min-hash, $CMH(T)$ of the tree T rooted at the height one node is computed by concatenating all the c min-hashes. Then the root node computes one min-hash from the compound min-hashes at height one nodes as shown in Algorithm 1. We note that we keep the number of permutations c constant and can hence be treated as an implicit parameter in all our analysis.

Algorithm 1 TH(T)

1: $\Pi \leftarrow \{\pi_{1,1}, \pi_{2,1}, \ldots, \pi_{c,1}\}$ be the permutations used by height one nodes
2: $\pi_{r,1}$ be the permutation used by the root node; $n \leftarrow$ degree of T
3: **for all** subtree T_i **do**
4: $\quad L_i \leftarrow leafs(T_i)$
5: $\quad CMH(T_i) \leftarrow MH_{\pi_{1,1}}(L_i) \bullet MH_{\pi_{2,1}}(L_i) \bullet \ldots \bullet MH_{\pi_{c,1}}(L_i)$
6: **end for**
7: **return** $MH_{\pi_{r,1}}\{CMH(T_1), \ldots, CMH(T_n)\}$

We now present the bounds on the probability of sketches matching for this case.

4.2.1 Lower bound p_1 on the matching probability

We now prove a lower bound on $\Pr[TH(T_1) = TH(T_2)]$ given $d(T_1, T_2) \leq \delta$. Let $T_1 = \{(a_1, \alpha_1), \ldots, (a_n, \alpha_n)\}$ where a_i is a subtree of height one with weight α_i and let $T_2 = \{(b_1, \beta_1), \ldots, (b_n, \beta_n)\}$.

Lemma 2. *Given $d(T_1, T_2) \leq \delta$, $\Pr[TH(T_1) = TH(T_2)] \geq \max\{2^{-(c+1)}(1-\delta)^c, 1 - 4c\delta\}$.*

Proof. From the distance measure, $EMD(T_1, T_2)$, it follows that there is a fractional matching f_{ij} so that the distance between T_1 and T_2 is equal to $\sum_{ij} f_{ij} d(a_i, b_j) \leq \delta$. Using this bound on the distance, we need to prove a lower bound on the probability that the two sketches match.

For a subtree of height one, a_i from T_1 and b_j from T_2, the probability that a min-hash computation will result in the same hash value for the two trees is $|a_i \cap b_j|/|a_i \cup b_j| = (1 - d(a_i, b_j))/(1 + d(a_i, b_j))$. So the probability that all c min-hashes match for the two subtrees is $\geq (\frac{1-d(a_i,b_j)}{1+d(a_i,b_j)})^c$. Let A and B denote the weighted sets of compound min-hashes in the tree-hash computation of T_1 and T_2 respectively. Then the probability that the final sketches match is $E[\frac{|A \cap B|}{|A \cup B|}] = E[\frac{|A \cap B|}{2 - |A \cap B|}] \geq \frac{E[|A \cap B|]}{2 - E[|A \cap B|]}$ (by concavity of $x/(2-x)$ in the range $[0, 1]$). But, $E[|A \cap B|] \geq \sum_{ij} f_{ij} \Pr[CMH(a_i) = CMH(b_j)] \geq \sum_{ij} f_{ij} (\frac{1-d(a_i,b_j)}{1+d(a_i,b_j)})^c$. By concavity of $(\frac{1-x}{1+x})^c$ and Jensen's inequality, this expectation is at least $(\frac{1-\sum_{ij} f_{ij} d(a_i,b_j)}{1+\sum_{ij} f_{ij} d(a_i,b_j)})^c \geq (\frac{1-\delta}{1+\delta})^c \geq (1 - 2c\delta)$. Since, $x/(2-x)$ is an increasing function of x, $E[\frac{|A \cap B|}{|A \cup B|}] \geq \frac{1-2c\delta}{2-(1-2c\delta)} = \frac{1-2c\delta}{1+2c\delta} \geq (1 - 4c\delta)$. Therefore the probability that the sketches match is at least $(1 - 4c\delta)$ giving us the proof for the second part. The proof for the first part follows by observing that $E[\frac{|A \cap B|}{|A \cup B|}] \geq E[|A \cap B|]/2 \geq 1/2(\frac{1-\delta}{1+\delta})^c \geq 2^{-(c+1)}(1-\delta)^c$. □

Computing the upper bound is dependent on whether we consider the overlap between *every* pair of subtrees to bounded by ϵ or we use the more general case wherein the average overlap is bounded by ϵ. We consider both the cases next.

4.2.2 Upper bound p_2 on the matching probability

In this section, we will upper bound $\Pr[TH(T_1) = TH(T_2)]$ given $d(T_1, T_2) > 1 - \epsilon$.

Lemma 3. *Given two trees T_1 and T_2 of height two and similarity ϵ, the probability their sketches match is at most $O(\epsilon \log(1/\epsilon))$.*

Let d_{ij} denote the distance $d(a_i, b_j)$. Then for all f_{ij} satisfying $\sum_j f_{ij} \leq \alpha_i$ and $\sum_i f_{ij} \leq \beta_j$, we have $\sum_{ij} f_{ij} d_{ij} \geq 1 - \epsilon$. Let A and B denote the weighted sets of min-hashes at level one in the tree-hash computation of T_1 and T_2 respectively. Let $s_{ij} = 1 - d_{ij}$. Then, $E[|A \cap B|] \leq \sum_{ij} \min(\alpha_i, \beta_j) \Pr[CMH(A_i) = CMH(B_j)] = \sum_{ij} \min(\alpha_i, \beta_j)(s_{ij}/(2 - s_{ij}))^c \leq \sum_{ij} \theta_{ij} s_{ij}^c$,

where $\theta_{ij} = \min(\alpha_i, \beta_j)$. We have the upper bound $p_2 = E[\frac{|A \cap B|}{|A \cup B|}] \leq E[|A \cap B|] \leq \sum_{ij} \theta_{ij} s_{ij}^c$.

Special case - To simplify, we first bound p_2 assuming that the similarity s_{ij} between *every* pair of subtrees is bounded by ϵ and root nodes have degree n.

We know that s_{ij} is at most ϵ and $\sum_i s_{ij} \leq 1$ and $\sum_j s_{ij} \leq 1$. The sum $\sum_{ij} s_{ij}^c$ is maximized when each s_{ij} is either ϵ or 0. Exactly, n/ϵ will be non-zero giving a maximum value of $\frac{n}{\epsilon}\epsilon^c$ for the sum. So the probability is bounded above by ϵ^{c-1}. This shows that for $c = 1$, the algorithm works poorly to distinguish between very different trees. In fact, as shown in Proposition 2, if we take two trees with the same set of leaves appearing in random order, for $c = 1$, they will result in the same sketch with high probability.

Next, we compute the upper bound p_2 in the general case.

General case - We eliminate the assumption in the special case.

In this case, all that we are given is that the distance between the two trees is at least $1-\epsilon$. By our definition of tree distance, this means for all fractional matchings, f_{ij}, $\sum_{ij} f_{ij} s_{i\pi(i)} \leq \epsilon$. Thus, the maximum weight fractional matching between the two sets of subtrees is at most ϵ where the weight is set to be the similarity measure. Now, we need to bound $\sum_{ij} \theta_{ij} s_{ij}^c$.

To prove the existence of this bound, we represent the subtrees of height one T_1 and T_2 as the left and right nodes in a weighted bi-bipartite, respectively. The weights on the edges between the nodes denotes the similarity between the sub-trees. This naturally leads to an optimization problem on a complete bipartite graph with a given maximum weight matching formulation.

Problem Definition 1. *Given a complete bipartite graph with edge weights w_{ij} and weights α_i, β_j on the left and right vertices, what is the maximum value of $\sum_{ij} \theta_{ij} w_{ij}^c$ where $\theta_{ij} = \min(\alpha_i, \beta_j)$ given*

1. *the weighted degree of any node in G is bounded by 1, i.e., $\sum_i w_{ij} \leq 1$, and $\sum_j w_{ij} \leq 1$,*
2. *$\sum_i \alpha_i = 1$ and $\sum_j \beta_j = 1$,*
3. *the maximum weight fractional matching is at most ϵ, that is $\forall f_{ij}$, if $\sum_j f_{ij} \leq \alpha_i$ and $\sum_i f_{ij} \leq \beta_j$ then $\sum_{ij} f_{ij} w_{ij} \leq \epsilon$.*

The following lemma shows that for the above problem the maximum value of $\sum_{ij} \theta_{ij} w_{ij}^c$ is at most $O(\epsilon \log(1/\epsilon))$.

Lemma 4. *Given a complete weighted bipartite graph $G = (V, E)$ on m left vertices and n right vertices with weight $w_{ij} \in [0, 1]$ on each edge and weights α_i, β_j on the left and right vertices such that $\sum_i \alpha_i = 1$ and $\sum_j \beta_j = 1$, and given that the maximum fractional weight matching of G is bounded by ϵ where $\epsilon > 0$, then for any $c \geq 2$, $\sum_{ij} \theta_{ij} w_{ij}^c \leq O(\epsilon \log(1/\epsilon))$.*

Proof. We will partition the edges into two sets - those with weights less than ϵ and those with weights more than ϵ. So, $\sum_{ij} \theta_{ij} w_{ij}^c = \sum_{ij} \theta_{ij} w_{ij}^c + \sum_{ij} \theta_{ij} w_{ij}^c$.

Part 1 - We have,

$$\begin{aligned}
\sum_{ij} \theta_{ij} w_{ij}^c &= \sum_{ij} \theta_{ij} w_{ij} w_{ij}^{c-1} \\
&\leq \sum_{ij} \theta_{ij} w_{ij} \epsilon^{c-1} \\
&\leq \epsilon^{c-1} \sum_{ij} \alpha_i w_{ij} \\
&= \epsilon^{c-1} \sum_i \left(\alpha_i \sum_j w_{ij} \right) \\
&\leq \epsilon^{c-1} \sum_i \alpha_i \\
&\leq \epsilon^{c-1}
\end{aligned}$$

Part 2 - We now bound the contribution from edges with weights in the range $[\epsilon, 1]$. We round up all weights to the nearest power of $1/2$ and group the edges by their weights. This gives us $\log(1/\epsilon)$ groups where all the edges in the k-th group have weight $\gamma_k = (1/2)^k$. We will show that contribution from each group is $O(\epsilon)$.

Look at the sub-graph obtained by considering all edges from the k-th group, G_k. Since the total weight at node is at most 1 (in fact, it'll be at most 2 after the rounding, but does not affect our analysis), the degree of any node in this sub-graph is at most $1/\gamma_k$. We know that such a sub-graph can be decomposed into at most $1/\gamma_k$ matchings. Therefore, it suffices to bound the contribution from each matching. For any matching M, $\sum_{ij \in M} \theta_{ij} w_{ij} \leq \epsilon$. To see this we set $f_{ij} = \theta_{ij}$ if (i, j) is in the matching and 0 otherwise. These values of f_{ij} are a valid fractional matching satisfying condition (3) in the problem definition. Summing over all the matchings in group G_k, we get $\sum_{ij \in G_k} \theta_{ij} w_{ij} \leq \epsilon/\gamma_k$. Now,

$$\begin{aligned}
\sum_k \sum_{ij \in G_k} \theta_{ij} w_{ij}^c &\leq \sum_k \sum_{ij \in G_k} \theta_{ij} w_{ij} \gamma_k^{c-1} \\
&\leq \sum_k \sum_{ij \in G_k} \theta_{ij} w_{ij} \gamma_k \text{ for } c \geq 2 \\
&\leq \sum_k \epsilon \\
&= \epsilon \log(1/\epsilon)
\end{aligned}$$

Adding the bounds on the two parts completes the proof. □

Observe that choosing any value of $c > 2$ does not improve the bound on p_2. Therefore $c = 2$ is good choice for our algorithm. We summarize the result with the following theorem.

We are ready to prove Lemma 3.

Proof of Lemma 3. Recall that the probability that the hash values of T_1 and T_2 match is at most $\sum_{ij} \theta_{ij} s_{ij}^c$. We now map this to our problem on bi-partite graphs by setting $w_{ij} = s_{ij}$. Condition (1) of our problem is satisfied Lemma 1. Condition (2) follows from the definition of θ_{ij} and condition (3) follows from the fact that the similarity between the two trees is at most ϵ. So, Lemma 4 implies the bound on the probability. □

4.3 Connections to Locality-Sensitive Hashing

Given that $p_1 = 1 - 4c\delta$ and $p_2 = \epsilon log(\frac{1}{\epsilon})$, for $c = 2$, the above probabilities give $\rho = O(\frac{\delta}{\log(1/\epsilon)})$. From theorem 1, we obtain the following result. Essentially, we have a sub-linear time algorithm to find the nearest neighbor in a setting where the nearest neighbor is much closer than all the other points.

Theorem 2. *Given a database of N trees of height 2, we can construct a data structure of size $N^{1+\rho}$ that can be used to compute approximate nearest neighbor in time N^ρ, where $\rho = O(\frac{\delta}{\log(1/\epsilon)})$. Given a query point whose nearest neighbor is within distance δ, our search algorithm will return a $\frac{1-\epsilon}{\delta}$-approximate nearest neighbor; essentially, our algorithm returns a neighbor of the query point within a distance at most $1 - \epsilon$.*

5 Sketching algorithms for trees of larger height

We continue the analysis for weighted trees of larger height with arbitrary shape and present a generalization to compute the bounds for the matching probabilities between such trees.

We now generalize the algorithm for computing the tree-hash, $TH(T^h)$, of a tree T^h, of height h. If $h = 1$, then tree-hash is equal to the min-hash of the leaf set. For $h > 1$, we consider c random instances of the tree-hash function with independent coin tosses. We use these functions to compute the compound tree-hash for every subtree at height $h - 1$. Thus, a compound tree-hash, $CTH(.)$, itself is a concatenation of all the c tree-hash values obtained using the c functions. The tree-hash for a subtree at height h is computed from the n compound tree-hashes at level $h - 1$. In practice, to avoid min-hashes that are concatenation of many labels, we can simply use a hash value of the concatenation instead. However, for the analysis, we will conceptually use the concatenated value.

Remark 1. *In the current definition of Algorithm 2, the number of min-hash computations at the leaf level becomes 2^h for $c = 2$ because of the recursion. This can be avoided by using the same set of c min-hashes at each level resulting in $2h$ min-hash computations for $c = 2$. Under this assumption, the analysis of the lower bound can be easily extended, while the analysis of the upper bound remains an open question.*

Algorithm 2 TH(T^h)

1: $\pi^r \leftarrow$ permutation used by the root node; $h \leftarrow height(T^h)$; $n \leftarrow degree(root(T^h))$
2: **if** $h > 1$ **then**
3: use c random instances of tree-hash functions, $TH_1, \ldots, TH_c$ with independent coin tosses
4: **for** subtree T_i^{h-1} at height $h-1$ **do**
5: $CTH(T_i^{h-1}) \leftarrow TH_1(T_i^{h-1}) \bullet TH_2(T_i^{h-1}) \bullet \ldots \bullet TH_c(T_i^{h-1})$
6: **end for**
7: **return** $MH_{\pi^r}\{CTH(T_1^{h-1}), \ldots, CTH(T_n^{h-1})\}$
8: **else if** $h = 1$ **then**
9: $L \leftarrow leafs(T^h)$
10: **return** $MH_{\pi^r}(L)$
11: **end if**

Using the above recursive formulation for computing the tree-hash of a tree with height greater than 2, we compute the probability bounds $p_{1,h}$ and $p_{2,h}$. We show inductively that the bounds hold for all levels in the tree. Since there is an super-exponential dependence on h, our analysis should be viewed in the context where h is a small constant.

Theorem 3 (Lower bound p_1). *Given two trees T_1 and T_2 of height h with distance δ and the probability that their tree-hashes match is $p_{1,h}(\delta)$, then for $h \geq 1$ and $c = 2$,*

$$p_{1,h}(\delta) \geq \frac{(1-\delta)^{2^{h-1}}}{2^{2^h-1}} \tag{1}$$

Proof. Let $T_1 = \{(a_1, \alpha_1), \ldots, (a_n, \alpha_n)\}$ be a tree of height h where a_i is a subtree of height $h-1$ with weight α_i in T_1 and similarly, let $T_2 = \{(b_1, \beta_1), \ldots, (b_n, \beta_n)\}$ be a tree of height h with b_i defined correspondingly. It follows from the definition of $EMD(T_1, T_2)$, there is a fractional matching f_{ij} so that the distance between T_1 and T_2 is equal to $\sum_{ij} f_{ij} d(a_i, b_j) \leq \delta$ We want to lower bound $\Pr[TH(T_1) = TH(T_2)]$ given the

distance bound $d(T_1, T_2) \leq \delta$. Let d_{ij} denote the distance between two sub-trees a_i and b_j. This gives the matching probability of the compound min-hashes of a_i and b_j to be $p^c_{1,h-1}(d_{ij})$. Let A and B denote the weighted sets of compound min-hashes in the tree-hash computation of T_1 and T_2 respectively. We know that $\Pr[TH(T_1) = TH(T_2)] = E[\frac{|A\cap B|}{|A\cup B|}] \geq E[\frac{|A\cap B|}{2}]$ (by concavity of $x/(2-x)$ in the range $[0,1]$). But, $E[|A \cap B|] \geq \sum_{ij} f_{ij} \Pr[CMH(a_i) = CMH(b_j)] \geq \sum_{ij} f_{ij} p_{1,h-1}{}^c(d_{ij})$. By concavity of the $p_{1,h-1}(d_{ij})$ in equation 1 and Jensen's inequality, this expectation is at least $p^c_{1,h-1}(\sum_{ij} f_{ij} d_{ij}) \geq p^c_{1,h-1}(\delta)$. Given this recurrence, for $h > 0$ and $c = 2$, we have

$$p_{1,h}(\delta) \geq \frac{p^2_{1,h-1}(\delta)}{2} = \frac{(1-\delta)^{2^{h-1}}}{2^{2^h-1}}$$

□

Theorem 4 (Upper bound p_2). *Given two trees T_1 and T_2 of level h with similarity ϵ and the probability that their tree-hashes match is $p_{2,h}$, for $h \geq 2$ and $c = 2$, there exists some constants c_1 and c_2 for which*

$$p_{2,h} \leq c_1 \epsilon \log(\frac{1}{\epsilon}) c_2^{3^h} (\log(\frac{n}{\epsilon}))^{2^{h-1}-2}$$

Proof. Let A and B denote the set of compound tree-hashes for the child subtrees of T_1 and T_2 respectively. The probability that the sketches match is $E[|A\cap B|/|A\cup B|] \leq E[|A \cap B|]$. Let $p_{ij} = \Pr[TH(T_i^h) = TH(T_j^h)]$, i.e., probability that the tree-hashes of two given subtrees match. Now, given p_{ij}, we want to bound $p_{2,h} = \Pr[TH(T_1^h) = TH(T_2^h)]$ recursively.

Similar to two-level trees, $p_{2,h} = E[|A \cap B|/|A \cup B|] \leq E[|A \cap B|]$. Thus, we have

$$\begin{aligned} E[|A \cap B|] &\leq \sum_{ij} \theta_{ij} \Pr[CTH(A_i) = CTH(B_j)] \\ &= \sum_{ij} \theta_{ij} (\Pr[TH(A_i) = TH(B_j)])^2 \\ &\leq \sum_{ij} \theta_{ij} p_{ij}^2 \end{aligned}$$

Therefore, by induction,

$$E[|A\cap B|] \leq \sum_{ij} \theta_{ij} (c_1 s_{ij} \log(\frac{1}{s_{ij}}) c_2^{3^h} (\log(\frac{n}{s_{ij}}))^{2^{h-1}-2})^2.$$

By concavity and Jensen's inequality, we have,

$$E[|A \cap B|] \leq \sum_{ij} \theta_{ij} c_1^2 R \log^2(\frac{1}{R}) c_2^{3^h 2} (\log^2(\frac{n}{R}))^{2^h-4}$$

where, $R = \sum_{ij} \theta_{ij} s_{ij}^2 / \sum_{ij} \theta_{ij}$. Viewing s_{ij} as weights in a weighted bipartite graph which satisfy conditions of Lemma 4, we get

$$\sum_{ij} \theta_{ij} s_{ij}^2 \leq O(\epsilon \log(1/\epsilon)),$$

since ϵ is the normalized maximum weight matching. Using $m = \sum_{ij} \theta_{ij} < n$ gives us $p_{2,h+1}$

$$\begin{aligned} &\leq c_1^2 m \tfrac{\epsilon \log(1/\epsilon)}{m} \log^2(\tfrac{m}{\epsilon \log(1/\epsilon)}) c_2^{3^h 2} (\log(\tfrac{nm}{\epsilon \log(1/\epsilon)}))^{2^h-4} \\ &\leq c_1^2 \epsilon \log(1/\epsilon) \log^2(\tfrac{n}{\epsilon \log(1/\epsilon)}) (4c_2)^{3^h 2} \log(\tfrac{n}{\epsilon \log(1/\epsilon)}))^{2^h-4} \\ &= c_1^2 \epsilon \log(1/\epsilon) 2^{2^h-4} c_2^{3^h 2} \log(\tfrac{n}{\epsilon \log(1/\epsilon)}))^{2^h-2} \end{aligned}$$

It follows that $c_1 2^{2^h-4} c_2^{3^h 2} \leq c_2^{3^{h+1} 2}$ for small h and $c_2 > c_1$ and $c_2 > 4$. For the specified values of h, c_2 and c_1, we have,

$$p_{2,h+1} \leq c_1 \epsilon \log(1/\epsilon) c_2^{3^h} (\log(n/\epsilon))^{2^h-2}$$

Thus, we have shown inductively that the bound for $p_{2,h+1}$ holds given $p_{2,h}$. □

Given the upper and lower bounds for matching probabilities, we have for trees of small height, $\rho = \frac{\log 1/p_{1,h}}{\log 1/p_{2,h}}$. Clearly for small values of h, $1/p_{2,h}$ is dominated by $O(\epsilon \log(1/\epsilon))$. This gives us a value of $\rho = O(\frac{1-\log(1-\delta)}{\log(1/\epsilon)})$ which gives us the following result as a consequence of theorem 1.

Theorem 5. *Given a database of N trees, we can construct a data structure of size $N^{1+\rho}$ that can be used to compute approximate nearest neighbor in time N^ρ, where $\rho = O(\frac{1-\log(1-\delta)}{\log(1/\epsilon)})$. Given a query point whose nearest neighbor is within distance δ, our search algorithm will return a $\frac{1-\epsilon}{\delta}$-approximate nearest neighbor; essentially, our algorithm returns a neighbor of the query point within a distance at most $1-\epsilon$.*

6 Conclusions

We analyze sketching algorithms to compute similarities between trees. Specifically, we study our algorithms using the framework of locality-sensitive hashing introduced in [8]. This allows us to find an approximate nearest neighbor among trees in time N^ρ, where N is the size of the database and δ and $1-\epsilon$ are well separated upper and lower bounds of distance in our algorithms.

References

[1] N. Augsten, M. Bohlen, and J. Gamper. Approximate matching of hierarchical data using *pq*-grams. In *Proc. of the 31st VLDB Conference*, pages 301–312, 2005.

[2] Andrei Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of SEQUENCES SEQS: Sequences '91*, 1998.

[3] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, 2000.

[4] Moses Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Annual ACM Symposium on Theory of Computing*, pages 380–388, 2002.

[5] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change detection in hierarchically structured information. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 493–504, 1996.

[6] W. Chen. New algorithms for ordered tree-to-tree correction problem. *Journal of Algorithms*, 40(2):135–158, August 2001.

[7] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. of the 20th ACM Symposium on Computational Geometry*, pages 253–262, 2004.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. of 25th International Conference on Very Large Data Bases, VLDB*, pages 518–529, 1999.

[9] S. Gollapudi and R. Panigrahy. Exploiting asymmetry in hierarchical topic extraction. In *Proc. of 13th Conference on Information and Knowledge Management*, 2006.

[10] Taher H. Haveliwala, Aristides Gionis, and Piotr Indyk. Scalable techniques for clustering the web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000.

[11] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proc. of 30th ACM Symposium on Theory of Computing (STOC)*, pages 604–613, 1998.

[12] T. Jiang, L. Wang, and K. Zhang. Alignment of trees - an alternative to tree edit. In *Proc. Intl Conference on Combinatorial Pattern Matching*, pages 75–86, 1994.

[13] K. Kailing, H-P. Kriegel, S. Schonauer, and T. Seidl. Efficient similarity search for hierarchical data in large databases. In *Proc. 9th Intl Conference on Extending Database Technology*, pages 676–693, 2004.

[14] T. Margush and F. R. McMorris. Consensus n-trees. *Bulletin of Mathematical Biology*, 3:239–244, 1981.

[15] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 218–224, 2005.

[16] Rina Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proc. of 17th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006*, pages 1186–1195, 2006.

[17] K. Zhang. A constrained editing distance between unordered labeled trees. *Algorithmica*, 15:205–222, 1996.

[18] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262, 1989.

[19] Li Zhang. On matching nodes between trees. Technical Report HPL-2003-67, HP Laboratories, Palo Alto, CA, April 2003.