algorithms

MDPI

Article

Efficient Approaches to the Mixture Distance Problem

Justie Su-Tzu Juan [1], Yi-Ching Chen [2], Chen-Hui Lin [1] and Shu-Chuan Chen [3,*]

1 Department of Computer Science and Information Engineering, National Chi Nan University, Puli, Nantou 54561, Taiwan; jsjuan@ncnu.edu.tw (J.S.-T.J.); tedlinct@gmail.com (C.-H.L.)
2 Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10617, Taiwan; d94010@csie.ntu.edu.tw
3 Department of Mathematics and Statistics, Idaho State University, Pocatello, ID 83209, USA
* Correspondence: scchen@isu.edu

Received: 31 August 2020; Accepted: 23 November 2020; Published: 28 November 2020

Abstract: The ancestral mixture model, an important model building a hierarchical tree from high dimensional binary sequences, was proposed by Chen and Lindsay in 2006. As a phylogenetic tree (or evolutionary tree), a mixture tree created from ancestral mixture models, involves the inferred evolutionary relationships among various biological species. Moreover, it contains the information of time when the species mutates. The tree comparison metric, an essential issue in bioinformatics, is used to measure the similarity between trees. To our knowledge, however, the approach to the comparison between two mixture trees is still unknown. In this paper, we propose a new metric named the mixture distance metric, to measure the similarity of two mixture trees. It uniquely considers the factor of evolutionary times between trees. If we convert the mixture tree that contains the information of mutation time of each internal node into a weighted tree, the mixture distance metric is very close to the weighted path difference distance metric. Since the converted mixture tree forms a special weighted tree, we were able to design a more efficient algorithm to calculate this new metric. Therefore, we developed two algorithms to compute the mixture distance between two mixture trees. One requires $O(n^2)$ and the other requires $O(nh_1h_2)$ computational time with $O(n)$ preprocessing time, where n denotes the number of leaves in the two mixture trees, and h_1 and h_2 denote the heights of these two trees.

Keywords: phylogenetic tree; evolutionary tree; ancestral mixture model; mixture tree; mixture distance; tree comparison

1. Introduction

Phylogeny reconstruction involves reconstructing the evolutionary relationship from biological sequences among species. Nowadays it has become a critical issue in molecular biology and bioinformatics. Several existing methods, such as neighbor-joining methods [1] and maximum likelihood methods [2], have been proposed to reconstruct a phylogenetic tree. A novel and natural method, ancestral mixture models [3], was developed by Chen and Lindsay to deal with such a problem. The mixture tree, a hierarchical tree created from the ancestral mixture model, induces a sieve parameter to represent the evolutionary time. Chen, Rosenberg and Lindsay (2011) then developed MixtureTree algorithm [4], a linux based program written in C++, which employed the ancestral mixture models to reconstruct mixture tree from DNA sequences. With the information provided by the mixture tree, one can identify when and how a mutation event of species occurs. An example of the mixture tree created by MixtureTree algorithm [3] is shown in Figure 1. The data from Griffiths and Tavare (1994) [5] are a subset of the mitochondrial DNA sequences which first appeared in Ward et al. (1991) [6]. To study the mitochondrial diversity within the Nuu-Chuah-Nulth, an Amerindian tribe from Vancouver Island, Ward et al. (1991) [6] sequenced 360 nucleotide segments of the mitochondrial control region for 63 individuals from the Nuu-Chuah-Nulth. Griffiths' and Tavares' subsample consisted of 55 of the 63 distinct sequences and 18 segregating sites, including 13 pyrimidines (C, T) and five purines (A, G). Each linage represents a distinct sequence—that is, there are lineages a through n. The time scale on the tree can be represented by $-\log(1-2p)$, where p is a parameter, the mutation rate. The number on the tree represents the site of the lineage whereat the mutation occurs. For example, when $p = 0.01$, lineages e and f merge because mutation occurs at site 5 of lineage f.

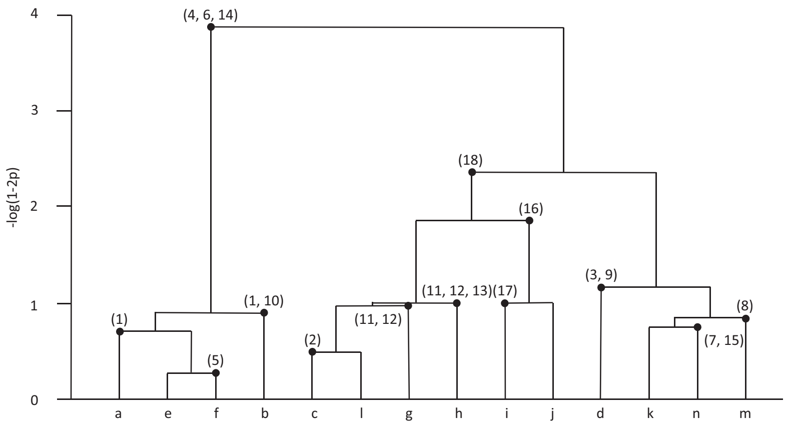

Figure 1. An example of the mixture tree [3].

Distinct methods may produce distinct trees, even though the methods adopt an identical dataset [7]. To uncover a well-represented tree involved in evolutionary relationship among species it is quite important to estimate how similar (or different) trees are. The tree distance between two trees is a general measurement for the similarity of the trees.

The tree distance problem is a traditional issue in mathematics. Several metrics have been proposed to measure the similarity between two trees, such as the partition metric (also called the Robison–Foulds metric or RF distance for short) [8], the quartet metric [9], the nearest neighbour interchange metric [10] and the nodal distance metric [11]. Those metrics all compare two trees by considering the tree structure only, and do not mention any parameter in the tree. Thus, those metrics are not suitable for computing the similarity between two mixture trees. Therefore, we propose a novel metric named the mixture distance metric to measure the similarity of two mixture trees in this paper. Among the above metrics, the metric from the nodal distance algorithm is similar to our proposed metric. In 2003, John Bluis and Dong-Guk Shin [11] presented the nodal distance algorithm which is used to measure the distances from leaves to all other leaves in a tree. The metric is defined as follows: $\text{Distance}(T_1, T_2) = \sum_{x,y \in L(T_1)=L(T_2)} |D_{T_1}(x,y) - D_{T_2}(x,y)|$, where $D_{T_i}(x,y)$ denotes the distance of leaf x to leaf y in the tree T_i. The nodal distance algorithm was developed for this metric. Anyway, using this metric to measure the distance between two mixture trees is not conformable.

For the metric of the mixture distance, the time parameter indicating when a mutation event of species occurs plays an important role in the tree similarity, which is, however, not considered by those previous metrics. If the weight of an edge in a mixture tree is defined as the difference in time parameters between its two endpoints, a mixture tree can be regarded as a weighted tree. We can design metrics to calculate the distance between two weighted trees. Some literature discusses the distance problem between two weighted trees. For example, take the weighted RF metric [12], geodesic distance [13] and the path difference metric [14]. However, the weight on each edge is considered to be the number of base changes between the sequences of the species represented by its incident vertices in these documents. Since the weights of each edge in those weighted trees may be different, the algorithm must spend more time to calculate those distances between those two weighted trees. For example, although there is an linear time algorithm to compute RF distance [15], and a randomized algorithm has been shown to approximate the RF distance with a bounded error in sublinear time [16], the complexity of the weighted RF distance still needs $O(n^2)$. Some papers have studied algorithms for calculating the geodesic trees distances [17–19]. The best one already known is $O(n^4)$ [19]. Due to the characteristics of the time parameter of a mixture tree, any two edges connecting two leaves to the same parent will have the same "weight" in a mixture tree. This helped us to design a better metric and algorithm. We further developed two algorithms to compute the mixture distance between two mixture trees. One requires $O(n^2)$ and the other requires $O(nh_1h_2)$ computational time with $O(n)$ preprocessing time, where n denotes the number of leaves in these two mixture trees, and h_1 and h_2 denote the heights of these two trees. If we use the nodal distance algorithm with the mixture distance metric, the time complexity will be $O(n^3)$ for binary unrooted trees. Comparisons with some previous methods show our method performs better.

2. Mixture Distance Metric

A tree $T = (V(T), E(T))$ is a connected and acyclic graph with a node set $V(T)$ and an edge set $E(T)$. T is a rooted tree if exactly one node of T has been designated the root. A node $v \in V(T)$ is a leaf if it has no child; otherwise, v is an internal node. A node $v \in V(T)$ is called in level i, denoted by $level(v) = i$, which means the number of edges on the path between the root and v is i. Let $L(T)$ denote a subset of node set $V(T)$, where each member is a leaf in T and $n = |L(T)|$. Let $height(T)$ denote the height of tree T, which is $\max\{level(v)|v \in L(T)\}$. T is a full binary tree if each node of T either has two children or it is a leaf. A complete binary tree is a full binary tree in which every level, except possibly the last, is completely filled, and all nodes are as far left as possible. Let $h_1 = height(T_1)$, $h_2 = height(T_2)$.

For a mixture tree T, each leaf is associated with a species, and every internal node v is associated with a mutation time $m_T(v)$ that represents the time when a mutation event occurs on the species node. In fact, the mutation time of an internal node in a mixture tree can be regarded as the distance between the node and any leaf of its descendants. Any two mixture tress T_1 and T_2 are comparable if $L(T_1) = L(T_2)$. Throughout this paper, a tree refers to a rooted full binary tree and each internal node of the tree is associated with its mutation time, if not mentioned particularly.

Given any two nodes $u, v \in V(T)$, the least common ancestor or lowest common ancestor (abbreviated LCA) of u and v is an ancestor of both u and v with the smallest mutation time. (It is also called the most recent common ancestor (abbreviated MRCA), or the last common ancestor (abbreviated LCA) in biology and genealogy.) Let $P_T(u, v)$ denote the mutation time $m_T(w)$ of the LCA w of two leaves u and v in T. The mixture distance metric, a metric for the mixture tree, is formally defined as follows.

The mixture distance between two comparable mixture trees T_1 and T_2, denoted by $d_m(T_1, T_2)$, is defined as the sum of difference of the mutation times with respect to the LCAs of any two leaves in T_1 and T_2. That is, $d_m(T_1, T_2) = \sum_{u,v \in L(T_1)=L(T_2)} |P_{T_1}(u, v) - P_{T_2}(u, v)|$.

The significance of the mixture distance metric is to measure the similarity between two mixture trees, considering the mutation times (molecular clock) and mutation sites simultaneously. The study sought to develop two algorithms for efficiently computing the mixture distance between two comparable mixture trees. Before we go into the algorithms, three properties of the mixture distance matric are demonstrated. Felsenstein [20] derived three mathematical properties—reflexivity, symmetry and triangle inequality—required for a well-defined metric. We show that the mixture distance is well-defined in Theorem 1.

Theorem 1. *The mixture distance d_m satisfies:*

1. *Reflexivity: for any two comparable mixture trees T_1 and T_2, $d_m(T_1, T_2) = 0$ if and only if T_1 and T_2 are identical.*
2. *Symmetry: for any two comparable mixture trees T_1 and T_2, $d_m(T_1, T_2) = d_m(T_2, T_1)$.*

3. *Triangle inequality: for any three comparable mixture trees T_1, T_2 and T_3, $d_m(T_1, T_2) + d_m(T_2, T_3) \geq d_m(T_1, T_3)$.*

Proof. 1. Due to $T_1 = T_2$, for any two nodes $u, v \in L(T_1) = L(T_2)$, we have $P_{T_1}(u, v) = P_{T_2}(u, v)$. Therefore, $d_m(T_1, T_2) = 0$ can be concluded. On the other hand, if $d_m(T_1, T_2) = 0$ for any two comparable mixture trees T_1 and T_2. We have $P_{T_1}(u, v) - P_{T_2}(u, v)$ for any $u, v \in L(T_1) = L(T_2)$ by the definition. Then we can prove $T_1 = T_2$ by induction on the height of T_1 (or T_2).

2. For any two nodes $u, v \in L(T_1) = L(T_2)$, $P_{T_1}(u, v) - P_{T_2}(u, v) = -(P_{T_2}(u, v) - P_{T_1}(u, v))$. Thus, $d_m(T_1, T_2) = \sum_{u,v \in L(T_1) = L(T_2)} |P_{T_1}(u, v) - P_{T_2}(u, v)| = \sum_{u,v \in L(T_1) = L(T_2)} |P_{T_2}(u, v) - P_{T_1}(u, v)| = d_m(T_2, T_1)$.

3. The triangle inequality is always satisfied for any three nonnegative numbers $a, b, c \in \Re^+ \cup 0$; that is, $|a - b| + |b - c| \geq |a - c|$. Therefore, $|P_{T_1}(u, v) - P_{T_2}(u, v)| + |P_{T_2}(u, v) - P_{T_3}(u, v)| \geq |P_{T_1}(u, v) - P_{T_3}(u, v)|$ holds. Further, we have

$$\sum_{u,v \in L(T_1)} |P_{T_1}(u, v) - P_{T_2}(u, v)| + \sum_{u,v \in L(T_2)} |P_{T_2}(u, v) - P_{T_3}(u, v)|$$

$$\geq \sum_{u,v \in L(T_1)} |P_{T_1}(u, v) - P_{T_3}(u, v)|.$$

Consequently, $d_m(T_1, T_2) + d_m(T_2, T_3) \geq d_m(T_1, T_3)$ can be concluded. □

3. An $O(nh_1h_2)$-Time Algorithm

Let T_1 and T_2 denote two comparable mixture trees of n leaves for each tree. Note that the mixture distance of T_1 and T_2 can be solved in $O(n^2)$-time: As when given two comparable mixture trees T_1 and T_2 each with n leaves, there are $O(n^2)$ pairs of leaves separately in T_1 and T_2. In fact, the LCA of any pair of leaves can be found by adopting the $O(1)$-time algorithm with $O(n)$-time preprocessing [21].

In the following, another $O(n^2)$-time algorithm, named Algorithm MIXTUREDISTANCE, is proposed to compute the mixture distance between T_1 and T_2, which will help us to realize the next $O(nh_1h_2)$-time algorithm, the main result.

3.1. Algorithm MixtureDistance

Algorithm MIXTUREDISTANCE, as shown on Algorithm 1, proceeds the nodes of T_1 by breadth-first search. For each internal node v in T_1, we find out the leaves of T_1 such that v is exactly the LCA of each pair of leaves, and then compute the LCA u of the leaves in T_2 which are mapped into the found leaves of T_1. Finally, the difference of the mutation times between u and v is calculated. For convenience, we define $(a, b) * (c, d) = ad + bc$ for any two ordered pairs (a, b) and (c, d) in this algorithm, where a, b, c and d are any four integers.

The algorithm adopts a 2-coloring method [22] on the leaves in T_1 and T_2 for easy implementation. For each iteration associated with an internal node v of T_1 in line 4, the leaves of the left and right subtrees rooted by v are colored by red and green, respectively. The mapped leaves in T_2 have the same coloring as one in T_1. The mixture distance between each internal node u in T_2 and v is calculated according the coloring scheme in T_2 (in lines 16–17), and the coloring information of u would be derived for the computation of its parent node (in line 18).

The coloring information of u, denoted by $color(u)$, indicates the coloring information of the subtree in T_2 rooted by u. $color(u)$ includes two numbers of u's descendant leaves colored by red ($color(u)[0]$) and green ($color(u)[1]$), respectively. $color(u)$ is derived by the coloring information of its two children. That is, $color(u)[0] = color(u_L)[0] + color(u_R)[0]$ and $color(u)[1] = color(u_L)[1] + color(u_R)[1]$, where u_L and u_R separately denote the left and right children of u in T_2.

Algorithm 1: MIXTUREDISTANCE(T_1, T_2).

```
Input: Two comparable mixture trees $T_1$ and $T_2$, with mutation times $m_{T_1}(v)$ ($m_{T_2}(u)$,
  respectively) for every internal node $v$ of $T_1$ ($u$ of $T_2$, respectively).
Output: The mixture distance $\mathcal{D}$ between $T_1$ and $T_2$.
1  $\mathcal{D} = 0$.
2  Traverse $T_1$ by the breadth-first search from its root and keep a list $\mathcal{I}_1$ of
   the internal nodes in order.
3  Traverse $T_2$ by the breadth-first search from its root and keep a list $\mathcal{I}_2$ of
   the internal nodes in reverse order.
4  for each node $v \in \mathcal{I}_1$ do
5      In $T_1$, color red the leaves of the left subtree rooted by $v$ and green the
       leaves of the right subtree rooted by $v$.
6      for each node $u \in \mathcal{I}_2$ do
           // Initialize the coloring information of $u$'s children
7          for each child $w$ of $u$ in $T_2$ do
8              if $w$ is a leaf then
9                  if $w$ is colored by red in $T_1$ then
10                     $color(w) = (1, 0)$.
11                 else if $w$ is colored by green in $T_1$ then
12                     $color(w) = (0, 1)$.
13                 else
14                     $color(w) = (0, 0)$.
15         Let $u_L$ and $u_R$ be the left and right children of $u$ in $T_2$, respectively.
           // Calculate the difference of the mutation times of $u$ and $v$ and
              sum them up for computing mixture distance
16         $number(u) = color(u_L) * color(u_R)$.
17         $\mathcal{D} = \mathcal{D} + |m_{T_1}(v) - m_{T_2}(u)| \times number(u)$.
           // Calculate the coloring information of $u$
18         $color(u) = color(u_L) + color(u_R)$.
```

In line 16, $number(u)$ is achieved by the special product of the color vectors of u's two children, $number(u) = color(u_L)[0] \times color(u_R)[1] + color(u_L)[1] \times color(u_R)[0]$, which means the number of times that u is an LCA of a red leaf and a green leaf. We multiply the difference of their mutation times by $number(u)$ in line 17, for computing the mixture distance between each internal node u in T_2 and v. At the end of Algorithm MIXTUREDISTANCE, $\mathcal{D}$ indicates the mixture distance of T_1 and T_2.

Since the numbers of internal nodes in T_1 and T_2 ($= \mathcal{I}_1$ and $\mathcal{I}_2$) are both equal to $n - 1$, two for-loops will take $O(n)$ time, and the innermost for-loop always takes 2 (a constant) time units. Therefore, Algorithm MIXTUREDISTANCE requires $O(n^2)$ computational time.

3.2. Modified Algorithm

After introducing Algorithm MIXTUREDISTANCE, we can give a $O(nh_1h_2)$ computational time algorithm for computing the mixture distance between two mixture trees in the following part. In Algorithm MIXTUREDISTANCE, when the leaves of the subtree rooted by an internal node v in T_1 are colored, other leaves in T_1 have no color, as do the mapped leaves in T_2. That is, $color(w) = (0, 0)$ for $w \in L(T_2)$. However, Algorithm MIXTUREDISTANCE still processes the ancestors of such leaves in T_2. In the following, we propose an algorithm for disregarding the nodes without meaningful coloring information, and reduce the time complexity from $O(n^2)$ to $O(nh_1h_2)$.

The algorithm contains three main stages, as follows:

1. Rank the leaves in T_1 and T_2.
2. Construct a minimal subtree T_2' of T_2 involved in colored leaves with respect to node v, for each internal node v in T_1.
3. Compute the mixture distance between v and each internal node in T_2'.

In stage 1, the nodes of T_2 are ranked in postorder, and the leaves of T_1 are assigned by the same rank of the mapped leaves in T_2. In Figure 2, red numbers nearby leaves in two given comparable mixture trees T_1 and T_2 indicate the ranking achieved by stage 1 of the algorithm. Note that the number within the nodes means the mutation time $m_{T_i}(v)$ of the associated node v for $i = 1$ or 2.

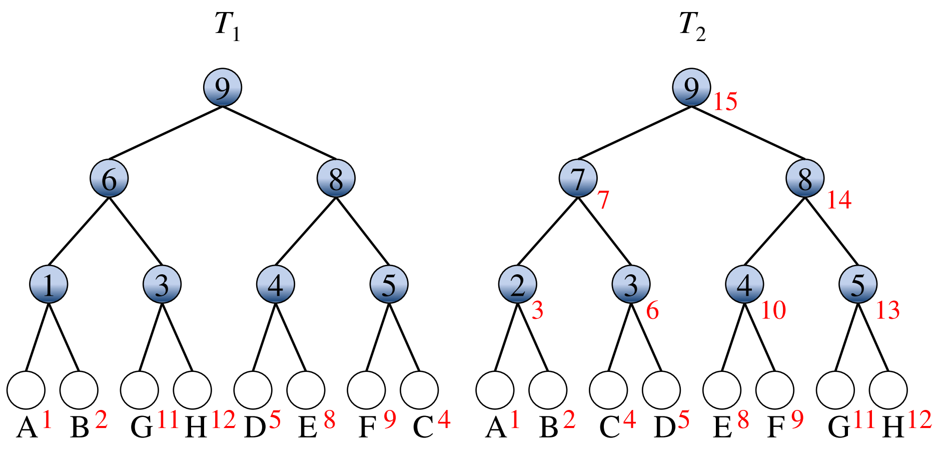

Figure 2. An example of ranking leaves of T_1 and T_2.

The algorithm proceeds to stage 2 for each internal node v of T_1 in the reverse order of breadth-first search. When v in T_1 is processed, stage 2 seeks to construct a minimal subtree T_2' of T_2 involved in colored leaves with respect to node v. For node v, a nondecreasing list of the leaves of the subtree rooted by v, denoted by $leaf(v)$, is obtained from the leaf lists of its two children, where the leaves in the list are sorted by their ranks. Suppose that there are k ordered nodes in $leaf(v)$, that is, $leaf(v) = \{w_1, w_2, \ldots, w_k\}$. With the list $leaf(v)$, the subtree T_2' can be constructed as follows.

Let $lca(w_i, w_j)$ denote the LCA of leaves w_i and w_j in T_2, for any $i, j \in \{1, 2, \ldots, k\}$. The subtree $T_2' = (V', E')$ is initialized by $V' = \{w_1, w_2, lca(w_1, w_2)\}$, $E' = \{\overline{lca(w_1, w_2)w_1}, \overline{lca(w_1, w_2)w_2}\}$ and $root = lca(w_1, w_2)$. For node w_i, $i \in \{1, 2, \ldots, k-2\}$,

$$V' = V' \cup \{lca(w_{i+1}, w_{i+2}), w_{i+2}\} \text{ and}$$

$$E' = E' \cup \{\overline{lca(w_{i+1}, w_{i+2})w_{i+2}}\}$$

Moreover, if the mutation time (the number written in the node circle) of $lca(w_{i+1}, w_{i+2})$, denoted by $t(lca(w_{i+1}, w_{i+2}))$, is larger than the mutation time of $root$, denoted by $t(root)$, the edge $\overline{lca(w_{i+1}, w_{i+2})root}$ is inserted into E' and reset $lca(w_{i+1}, w_{i+2})$ as the new $root$. Otherwise, if $t(lca(w_{i+1}, w_{i+2}))$ is smaller than the mutation time of $lca(w_i, w_{i+1})$, denoted by $t(lca(w_i, w_{i+1}))$, the edge $\overline{lca(w_i, w_{i+1})w_{i+1}}$ is removed from E' and the edges $\overline{lca(w_{i+1}, w_{i+2})w_{i+1}}$ and $\overline{lca(w_i, w_{i+1})lca(w_{i+1}, w_{i+2})}$ are inserted into E'. Otherwise, let $x = w_{i+1}$ and repeat do $x = father(x)$ until $t(x) < t(lca(w_{i+1}, w_{i+2})) < t(father(x))$, where $father(x)$ is the node y such that $\overline{yx} \in E'$. Then the edge $\overline{father(x)x}$ is removed from E' and the edges $\overline{lca(w_{i+1}, w_{i+2})x}$ and $\overline{father(x)lca(w_{i+1}, w_{i+2})}$ are inserted into E'.

Example 1. *An example of constructing the subtree T_2' with respect to $leaf(v_2) = \{A, B, G, H\}$ is illustrated in Figure 3. Initially, the node set V' is $\{A, B, lca(A, B)\}$ and the edge set E' includes the incident edges of the three nodes in T_2. As node A is processed, two nodes $lca(B, G)$ and G are inserted into V', and two edges $\overline{lca(A, B)lca(B, G)}$ and $\overline{lca(B, G)G}$ are inserted into E'. Later, when node B is processed, two nodes $lca(G, H)$ and H are inserted into V' and two edges $\overline{lca(B, G)lca(G, H)}$ and $\overline{lca(G, H)H}$ are inserted into E'. Meanwhile, the edge $\overline{lca(B, G)G}$ is removed from E' and the edge $\overline{lca(G, H)G}$ is inserted into E', because the mutation time of $lca(B, G)$ is larger than the time of the $lca(G, H)$.*

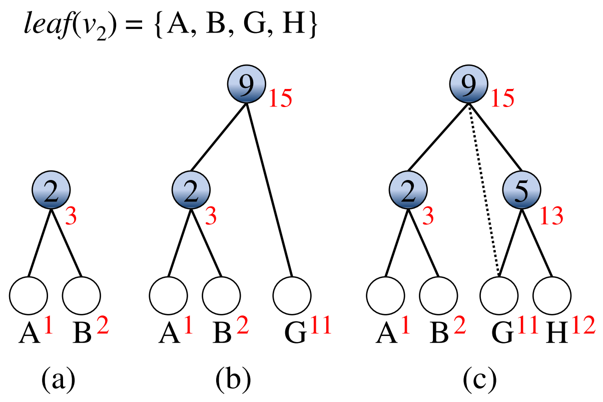

Figure 3. An example of constructing the subtree T_2' with respect to $leaf(v_2)$ in Figure 2. (**a**) The initialization of T_2'. (**b**) The intermediate of T_2' as node A is processed. (**c**) The complete subtree T_2' as node B is processed. As the mutation time of *lca*(B, G) is larger than the time of *lca*(G, H), the dotted line incident to G is removed and the other incident edge of G is inserted.

After the subtree T_2' with respect to currently processed node v is constructed, stage 3 of the algorithm performs lines 5–18 of Algorithm MIXTUREDISTANCE to compute the "partial" mixture distance between T_2' and the subtree rooted by v (only computes the distances of some nodes pairs, for which LCA is equal to v). At the end of the algorithm, $\mathcal{D}$ indicates the mixture distance between T_1 and T_2.

Theorem 2. *The improved algorithm takes $O(nh_1h_2)$ computational time and $O(n)$ preprocessing time, where n denotes the number of leaves of the mixture trees and $h_i = height(T_i)$ for $i = 1, 2$.*

Proof. The algorithm contains three main stages. The first stage ranks the leaves in T_1 and T_2, which takes $O(n)$ time.

In the second stage, a minimal subtree T_2' of T_2 involved in colored leaves with respect to each node v in T_1 is constructed. For each node v, a leaf list $leaf(v)$ is obtained from the leaf lists of its two children, which is achieved in $O(t)$ time by using the two-way merging algorithm [23] performed on the leaf list of v's children, where t is the size of $leaf(v)$. The $O(1)$-time algorithm with $O(n)$-time processing [21] is employed to compute the LCA of any pair of nodes in T_2. Constructing T_2' takes $O(th')$ time, where h' is the height of T_2' due to the "repeat" step. The last stage computes the mixture distance between v and each internal node in T_2' by performing lines 5–18 of Algorithm MIXTUREDISTANCE, which takes $O(t)$ time. Stages 2 and 3 take $O(n)$ iterations in total. However, each iteration deals with different t nodes. Note that for all internal nodes which are in the same level of T_1, the sum of t (for each node) is n. Therefore, stages 2 and 3 totally take $O(nh_1h') = O(h_1h_2)$ time, where h_1 is the height of T_1 (note that $h' \leq h_2 = height(T_2)$). Hence, the algorithm requires $O(nh_1h_2)$ computational time with $O(n)$ preprocessing time. □

4. Conclusions

In this paper, we provide a novel metric named the mixture distance metric to measure the similarity between two mixture trees. It uniquely considers the estimated evolutionary time in the trees. Two algorithms were developed to compute the mixture distance between mixture trees. One requires $O(n^2)$ computational time and the other requires $O(nh_1h_2)$ computational time with $O(n)$ preprocessing time, respectively. Note that when T_1 and T_2 are complete binary trees, h_1 and h_2 will be $O(\log n)$ and the time complexity of our algorithm will be $(n \log^2 n)$.

Now, we compare our metric with some previous methods which measure phylogenetic differences in consideration of the branch length, when we consider a mixture tree as a weighted tree (recall that the weight of an edge in a mixture tree is defined as the difference of time parameters between its two endpoints). For the geodesic tree distance, the implementation is quite complex and requires heavy computation [19], although a heuristic fast version exists [18]. The definition of the weighted path difference distance [14] is almost the same as the mixture distance. Actually, the weighted path difference distance between two mixture trees T_1 and T_2 is equal to $2d_m(T_1, T_2)$. However, it requires $O(n^2)$ computational time. The mixture distance seems to be similar to the weighted RF distance [12], but the calculation performance will vary when we consider the distance between two different extents of similar mixture trees. We give an example as follows.

Example 2. *Four mixture trees with the same lineages A, B and C are illustrated in Figure 4; the time parameters are listed in the vertices, and the associated edge weights are labeled beside each edge. All pairs of these four trees have been compared using the methods outlined in [12] and this paper. The tables of the weighted RF (wRF) and mixture distances (d_m) are given in Tables 1 and 2, respectively. From these two tables, one can find something interesting. (1) d_m seems maintain the order relationship in wRF: When wRF thinks that two trees are similar, then d_m also gets a smaller value between these two trees: $wRF(T_1, T_3) > wRF(T_2, T_3) > wRF(T_2, T_4) > wRF(T_1, T_2)$ and $d_m(T_1, T_3) > d_m(T_2, T_3) > d_m(T_2, T_4) > d_m(T_1, T_2)$. (2) When wRF thinks that the distances between two pairs of trees are the same, then d_m also thinks they are in the same: $wRF(T_1, T_2) = wRF(T_3, T_4)$, $wRF(T_1, T_4) = wRF(T_2, T_3)$ and $d_m(T_1, T_2) = d_m(T_3, T_4)$, $d_m(T_1, T_4) = d_m(T_2, T_3)$. However, there are still differences between these two metrics in the details: (3) When wRF thinks two distances between two pairs of trees are very different, sometimes d_m may not think that: $wRF(T_1, T_3) - wRF(T_2, T_3) = 1$, $wRF(T_1, T_4) - wRF(T_2, T_4) = 3$, but $d_m(T_1, T_3) - d_m(T_2, T_3) = d_m(T_1, T_4) - d_m(T_2, T_4) = 1$.*

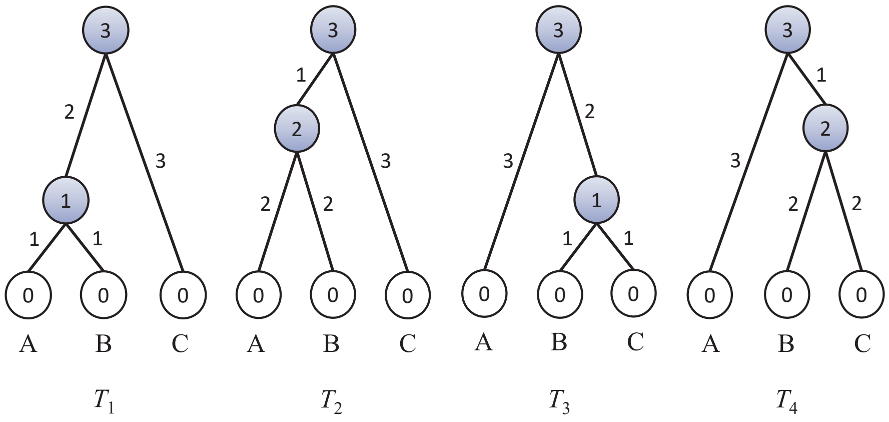

Figure 4. Four weighted trees with the same lineages A, B and C.

Table 1. The weighted RF distances wRF among T_1, T_2, T_3 and T_4.

wRF	T_1	T_2	T_3	T_4
T_1		3	8	7
T_2			7	4
T_3				3

Table 2. The mixture distances d_m among T_1, T_2, T_3 and T_4.

d_m	T_1	T_2	T_3	T_4
T_1		1	4	3
T_2			3	2
T_3				1

Therefore, it can be said that the performance of the mixture distance in calculating the similarity of two weighted trees is as good as the performance of the weighted RF distance, while the time complexity of the mixture distance is better. In addition, we compared our approaches with the methods performed with the nodal distance metric [11], geodesic tree distance [19], weighted path difference metric [14] and weighted RF distance [12], and the results are shown in Table 3. Our proposed approaches performed better than all of the previous methods when discussing the distance between two mixture trees.

Table 3. Comparison of metrics for binary trees.

Metric	**Consideration**	**Time Complexity**	
		Full Binary Trees	**Complete Binary Trees**
Mixture distance	Structure and mutation time	$O(nh_1h_2)$	$O(n \log^2 n)$
Nodal distance	Structure	$O(n^3)$	$O(n^2 \log n)$
Geodesic tree distance	Structure and mutation number	$O(n^4)$	$O(n^4)$
Weighted path difference distance	Structure and mutation number	$O(n^2)$	$O(n^2)$
Weighted RF distance	Structure and mutation number	$O(n^2)$	$O(n^2)$

Author Contributions: Investigation, Y.-C.C. and C.-H.L.; methodology, J.S.-T.J. and S.-C.C. All authors have read and agreed to the published version of the manuscript.

Funding: The first author was supported in part by the Ministry of Science and Technology of the Republic of China under Contract No. MOST100-2221-E-260-024- and MOST109-2115-M-260-001.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Saitou, N.; Nei, M. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **1987**, *4*, 406–425. [PubMed]
2. Lesperance, M.L.; Kalbfleisch, J.D. An algorithm for computing the nonparametric MLE of a mixing distribution. *J. Am. Stat. Assoc.* **1992**, *87*, 120–126. [CrossRef]
3. Chen, S.C.; Lindsay, B.G. Building mixture trees from binary sequence data. *Biometrika* **2006**, *93*, 843–860. [CrossRef]
4. Chen, S.C.; Rosenberg, M.; Lindsay, B.G. MixtureTree: A program for constructing phylogeny. *BMC Bioinform.* **2011**, *12*, 111–114. [CrossRef] [PubMed]
5. Griffiths, R.C.; Tavare, S. Ancestral inference in population genetics. *Statist. Sci.* **1994**, *9*, 307–319. [CrossRef]
6. Ward, R.H.; Frazier, B.L.; Dew-Jager, K.; Paabo, S. Extensive mitochondrial diversity within a single amerindian tribe. *Proc. Nat. Acad. Sci. USA* **1991**, *88*, 6720–6724. [CrossRef] [PubMed]
7. Steel, M.A. The maximum likelihood point for a phylogenetic tree is not unique. *Syst. Biol.* **1994**, *43*, 560–564. [CrossRef]
8. Robinson, D.F.; Foulds, L.R. Comparison of phylogenetic trees. *Biosciences* **1981**, *53*, 131–147. [CrossRef]
9. Estabrook, G.F.; McMorris, F.R.; Meacham, C.A. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Zool.* **1985**, *34*, 193–200. [CrossRef]
10. Dasgupta, B.; He, X.; Jiang, T.; Li, M.; Tromp, J.; Zhang, L. On computing the nearest neighbor interchange distance. In Proceedings of the Discrete Mathematical Problems with Medical Applications: DIMACS Workshop on Discrete Problems with Medical Applications, Piscataway, NJ, USA, 8–10 December 1999; DIMACS Series in Discrete Mathematics and Theoretical Computer Science; American Mathematical Society: Washington, DC, USA, 2000; Volume 55, pp. 125–143.

11. Bluis, J.; Shin, D. Nodal distance algorithm: calculating a phylogenetic tree comparison metric. In Proceedings of the Third IEEE Symposium on BioInformatics and BioEngineering, Bethesda, MD, USA, 12 March 2003; pp. 87–94.
12. Robinson, D.F.; Foulds, L.R. Comparison of weighted labelled trees. In *Combinatorial Mathematics VI*; Springer: Berlin/Heidelberg, Germany, 1979; pp. 119–126.
13. Billera, L.J.; Holmes, S.P.; Vogtmann, K. Geometry of the space of phylogenetic trees. *Adv. Appl. Math.* **2001**, *27*, 733–767. [CrossRef]
14. Steel, M.A.; Penny, D. Distributions of tree comparison metrics—Some new results. *Syst. Biol.* **1993**, *42*, 126–141.
15. Day, W.H. Optimal algorithms for comparing trees with labeled leaves. *J. Classif.* **1985**, *2*, 7–28. [CrossRef]
16. Pattengale, N.D.; Gottlieb, E.J.; Moret, B.M. Efficiently computing the Robinson-Foulds metric. *J. Comput. Biol.* **2007**, *14*, 724–735. [CrossRef] [PubMed]
17. Battagliero, S.; Puglia, G.; Vicario, S.; Rubino, F.; Scioscia, G.; Leo, P. An efficient algorithm for approximating geodesic distances in tree space. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2010**, *8*, 1196–1207. [CrossRef] [PubMed]
18. Amenta, N.; Godwin, M.; Postarnakevich, N.; John, K.S. Approximating geodesic tree distance. *Inf. Process. Lett.* **2007**, *103*, 61–65. [CrossRef]
19. Owen, M.; Provan, J.S. A fast algorithm for computing geodesic distances in tree space. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2010**, *8*, 2–13. [CrossRef] [PubMed]
20. Felsenstein, J. *Inferring Phylogenies*; Sinauer Associates: Sunderland, MA, USA, 2004.
21. Bender, M.A.; Farach-Colton, M. The LCA problem revisited. *Lat. Am. Theor. Inform.* **2000**, *1776*, 88–94.
22. Brodal, G.S.; Fagerberg, R.; Pedersen, C.N.S. Computing the quartet distance between evolutionary trees in time $O(n \log n)$. *Algorithmica* **2003**, *38*, 377–395. [CrossRef]
23. Lee, R.C.T.; Chang, R.C.; Tseng, S.S.; Tsai, Y.T. *Introduction to the Design and Analysis of Algorithms*; McGraw-Hill Education: Berkshire, UK, 2005.