The Mutated Distance between Two Mixture Trees

Wan Chian Li, Justie Su-Tzu Juan*, Yi-Chun Wang, and Shu-Chuan Chen

Abstract—The evolutionary tree is an important topic in bioinformatics. In 2006, Chen and Lindsay proposed a new method to build the mixture tree from DNA sequences. Mixture tree is a new type evolutionary tree, and it has two additional information besides the information of ordinary evolutionary tree. One of the information is time parameter, and the other is the set of mutated sites. In 2008, Lin and Juan proposed an algorithm to compute the distance between two mixture trees. Their algorithm computes the distance with only considering the time parameter between two mixture trees. In this paper, we proposes a method to measure the similarity of two mixture trees with considering the set of mutated sites and develops two algorithm to compute the distance between two mixture trees. The time complexity of these two proposed algorithms are \( O(n^2 \times \max\{h(T_1), h(T_2)\}) \) and \( O(n^2) \), respectively.

Keywords—evolutionary tree, mixture tree, mutated site, distance.

I. INTRODUCTION

The phylogenetic trees or evolutionary trees are described in the relationship of species. Using species information to build phylogenetic trees is a popular problem. The species information is including species external, species frame and DNA sequence, etc. There are many methods to build trees, like neighbor-joining [1], maximum likelihood [2], and so on. In this topic, to propose a method for building trees must do bootstrapping. Different trees could be built by a data set, even if using the same method [3].

In this paper, we give a new definition of distance, called mutated distance, between two mixture trees by considering the set of mutated sites. And we also give a corresponding algorithm to compute the mutated distance between two mixture trees. Then, we also give an improved algorithm, such that the time complexity of this algorithm is \( O(n^2) \). We use the path difference metric [12] concept to define the distance and design our algorithm by using the concept of Lin and Juan’s algorithm [11]. Hence, it is easy to combine our algorithm with Lin and Juan’s algorithm [11].

Path difference metric [5] - It was mentioned by Penny and Hendy in 1985. Let \( d_{ij}(T) \) denote the number of edges in the path which join two leaves that labeled by \( i \) and \( j \) in \( T \), and let \( d(T) \) be the associate vector obtained by fixed ordering of the pairs \((i, j)\). \( d_p(T_1, T_2) \) denotes the Euclidean distance between the two vector \( d(T_1) \) and \( d(T_2) \). That is, \( d_p(T_1, T_2) \) is the square root of the sum of the squares of the difference \( d_{ij}(T_1) - d_{ij}(T_2) \). The distance between two phylogenetic trees \( T_1 \) and \( T_2 \) is defined as \( Distance(T_1, T_2) = d_p(T_1, T_2) = \|d(T_1) - d(T_2)\|_2 \). Williams and Clifford [13] defined a similar dissimilarity measure on trees, except using an \( L^1 \)-norm rather than \( L^2 \)-norm. That is, \( Distance(T_1, T_2) = d_p(T_1, T_2) = \|d(T_1) - d(T_2)\|_1 \).

The mixture distance [11] - In 2008, Lin and Juan proposed mixture distance denoted by \( d_m \), as the sum of the difference of \( P_{T_1}(x, y) \) for any two leaves \( x, y \). That is, the mixture distance between two mixture trees \( T_1, T_2 \) is defined as \( d_m(T_1, T_2) = \sum_{(x, y) \in V'} |P_{T_1}(x, y) - P_{T_2}(x, y)| \), where \( V' \) is the set of leaves of \( T_1 \) (equals to the set of leaves of \( T_2 \)) and \( P_{T_1}(x, y) \) denote the time parameter of the least
common ancestor of two leaves \( x, y \) in tree \( T_i \) for \( i = 1, 2 \).

Their corresponding algorithm, called the mixture distance algorithm, only compares the least common ancestor of two leaves in two trees. For an internal node in \( T_1 \), the mixture distance algorithm finds all pairs of leaves which the least common ancestor is this internal node. Then, this algorithm finds the least common ancestors of those leaves in \( T_2 \), and calculates the distance. In order to implement this approach, similar to [12], they used two colors to color leaves of \( T_2 \) according to \( T_1 \).

**Definition 1.** [14] There are many topological spaces in which the topology is derived from a notion of distance. A metric for a set \( X \) is a function \( d \) on the cartesian product \( X \times X \) to the non-negative reals such that for all points \( x, y \) and \( z \) of \( X \),

(a) \( d(x, y) = d(y, x) \),

(b) (triangle inequality) \( d(x, y) + d(y, z) \geq d(x, z) \),

(c) \( d(x, y) = 0 \) if \( x = y \), and

(d) \( x = y \) if \( d(x, y) = 0 \).

The last one of these conditions is inessential for many purposes. A function \( d \) which satisfies only (a), (b) and (c) is called a pseudo-metric.

In Section II, we define a new metric, the mutated distance, to measure the distance between two mixture trees, and we also show that this metric is a pseudo-metric. In Section III, an algorithm for the mutated distance is proposed. Section IV will propose an improved algorithm for the mutated distance.

II. THE METRIC: MUTATED DISTANCE

Throughout this paper, we only discuss the full binary tree. A fully resolved tree is a tree in which every node bifurcates [15], and it also is called a full binary tree. The full binary tree is a tree \( T = (V, E) \) with \( n \) nodes, and each node \( v_i \) has either two children or no child. The node without child is called a leaf, which is associated with a species. Because we discuss mixture trees, every node \( v_i \) will be associated with a set \( MSF_i(v_i) \), mutated sites set, that records the set of all sites of a species mutation occurring from its father. Fig. 2 shows the data tree of the mixture tree in Fig. 1.

![Fig. 2. A data tree for the associated mixture tree \( M_1 \).](image)

In a tree \( T = (V, E) \), let \( V'(T) \) be the leaves vertex set of \( T \). Let \( LCA(T, x, y) \) be the least common ancestors of \( x, y \in V'(T) \) in \( T \). Let \( V_T(x, y) \)-path be the vertex set of \( (x, LCA(T, x, y))-path - LCA(T, x, y) \) in \( T \).

The notation \( \{ x, y \} \) is symmetric difference of two sets. Let \( LCA(T, x, y) \) be the least common ancestors of \( x, y \in V'(T) \) in \( T \). Let \( V_T(x, y) \)-path be the vertex set of \( (x, LCA(T, x, y))-path - LCA(T, x, y) \) in \( T \). Let \( V_T(x, y) \)-path = \( \{ v_1, v_2, ..., v_n = LCA(T, x, y) \} \), and \( S_T(x, y) \) be the set of \( MSF(v_1) \cup MSF(v_2) \cup ... \cup MSF(v_n - 1) \).

Define \( d'(T_1, T_2) \) be the mutated distance between two mixture trees, \( T_1 \) and \( T_2 \), by \( d'(T_1, T_2) = \sum_{x, y \in V} (|S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)|) \) where \( V' = V'(T_1) = V'(T_2) \).

From following Theorems 1, 2, 3 and Example 1, we prove that our metric is a pseudo-metric.

**Theorem 1.** The mutated distance \( d' \) satisfies d'(A, B) for any two trees A, B.

**Proof.** Because \( |S_A(x, y) \triangle S_B(x, y)| = |S_B(x, y) \triangle S_A(x, y)| \) and \( |S_A(x, y) \triangle S_B(x, y)| = |S_B(x, y) \triangle S_A(x, y)| \) for any two leaves \( x, y \), \( d'(A, B) = \sum_{x, y \in V} (|S_A(x, y) \triangle S_B(x, y)| + |S_B(x, y) \triangle S_A(x, y)|) \).

Our goal is to prove \( d'(T_1, T_2) + d'(T_2, T_3) \geq d'(T_1, T_3) \).

Since the distance is the sum of two terms of symmetric difference operations. So, if we prove that one of these two terms satisfies triangle inequality, the whole inequality will hold

Let \( S_T(x, y) \) = \( |S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)| \geq 0 \), \( S_T(x, y) \) = \( |S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)| \).

We have \( S_T(x, y) \) = \( |S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)| \) for any two leaves x and y in \( V' \). We have \( S_T(x, y) \) = \( |S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)| \).

**Theorem 2.** The mutated distance \( d' \) satisfies the triangle inequality.

**Proof.** Let \( T_1, T_2 \) and \( T_3 \) are three mixture trees with the set of leaves \( V' \). By the definition,

\[ d'(T_1, T_2) = \sum_{x, y \in V} (|S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)|) \] and \( d'(T_2, T_3) = \sum_{x, y \in V} (|S_T(x, y) \triangle S_T(x, y)| + |S_T(x, y) \triangle S_T(x, y)|) \).

Our goal is to prove \( d'(T_1, T_2) + d'(T_2, T_3) \geq d'(T_1, T_3) \).
There exist two mixture trees

\[ |S_{r_1}(x, y) \cap S_{r_2}(x, y) \cap \{S_{r_1}(x, y) \cap S_{r_2}(x, y)\}| = 2(|S_{r_1}(x, y) \cap \{S_{r_1}(x, y) \cap S_{r_2}(x, y)\}| - |\{S_{r_1}(x, y) \cap S_{r_2}(x, y)\}|).
\]

Since \( |S_{r_1}(x, y) \cap \{S_{r_1}(x, y) \cap S_{r_2}(x, y)\}| \geq |S_{r_1}(x, y) \cap S_{r_2}(x, y)\|0\| \geq 0 \), we have 
\[ \sum_{x, y \in V'} |S_{r_1}(x, y) \cap S_{r_2}(x, y)| = \sum_{x, y \in V'} S_{r_1}(x, y) + S_{r_2}(x, y) \geq 0. \]

Theorem 3. If tree A is equal to tree B, the mutated distance \( d'(A, B) \) is zero.

Proof. If tree A is equal to tree B, then \( S_{r_1}(x, y) = S_{r_2}(x, y) \) and \( S_{r_3}(x, y) = S_{r_4}(x, y) \) for any two leaves \( x, y \). Hence, \( |S_{r_1}(x, y) \cap S_{r_2}(x, y)| = 0 \) and \( |S_{r_3}(x, y) \cap S_{r_4}(x, y)| = 0 \) for any two leaves \( x, y \). That implies \( \sum_{x, y \in V'} |S_{r_1}(x, y) \cap S_{r_2}(x, y)| + |S_{r_3}(x, y) \cap S_{r_4}(x, y)| = 0. \)

Example 1 shows that the mutated distance does not satisfy \( d(x, y) = 0 \) if \( d'(x, y) \).

Example 1. There exist two mixture trees A and B in Fig. 3, such that the mutated distance of A and B, \( d'(A, B) \), is zero, but tree A do not equal to tree B.

III. THE ALGORITHM FOR MUTATED DISTANCE

Firstly, we design an algorithm for mutated distance in Section III-A. Then, we give an example in Section III-B. Section III-C is analysis of this algorithm.

A. The Algorithm

For finding the mutated distance \( d' \) of any two mixture trees, \( T_1 \) and \( T_2 \), we need find mutated site set data of a path \( S_{r_1}(x, y) \) for \( T = T_1 \) and \( T_2 \) at first. In 2008, Lin and Juan proposed an algorithm to compute the distance of time parameter between two mixture trees [11]. Their algorithm use color information to find all the least common ancestors of any two leaves in each of two trees. That will reduce the complexity for finding for any two leaves \( x \) and \( y \) in \( V' = V'(T_1) = V'(T_2) \). We will use this idea, too. Before introducing the algorithm, we have to understand some notations which are used in the algorithm.

- \( T_{1,j} u_j \) denotes a node \( u_j \) in \( T_1 \), where \( j \) is in the order of BFS, \( T_{2,j} v_j \) denotes a node \( v_j \) in \( T_2 \), where \( j \) is in the order of BFS. Note that \( T_{1,u_i} = T_{2,v_j} \) for some \( j \) for any leaf \( u_i \) of \( T_1 \) such that \( T_{1,u_i} (= T_{2,v_j}) \) has the same sequence name with \( v_j \) in \( T_2 \).
- \( color_{i} \) of \( v_j \) denotes the color information of the subtree that rooted by \( v_j \) in \( T_2 \). The \( color_{i} \) contains two integer: \( color_{i}.Red \) is the amount of leaves that are colored by red, and \( color_{i}.Green \) is the amount of leaves that are colored by green. For example, \( A, B, C \) is three nodes in a tree. Let \( B, C \) be two children of \( A \), then \( color_{A} = color_{B} + color_{C} \). That means, these two values \( color_{A} \). Red = \( color_{B}.Red \) + \( color_{C}.Red \); \( color_{A}.Green = color_{B}.Green + color_{C}.Green \). Green.
- \( sLeafTable \) is the leaves data of \( T_i \) for \( i = 1, 2 \). The data include sequence name, BFS number and color information. The size of this table is \( n \times 3 \), where a row represents one leaf. The sequence name represents the sequence title of this leaf. The BFS number is the order of this leaf in the order of BFS. The color information is the color of this leaf, which will be green, red or null.
- \( d.v_k.Red \)\( (d.v_k.Green) \) of \( T_{2,v_k} \) for \( T_{1,v_j} \) and \( T_{3,v_j} \) denotes the sum of symmetric difference between \( S_{r_1}(v_j, v_k) \) and \( S_{r_2}(v_j, v_k) \) for \( v_k \) in \( (v_j, v_k) \)-path of \( T_2 \) for any leaf \( v_j \) when we fix \( i \). And when we fix \( v_j \), if the color of \( v_j \) is red (green, respectively), this value will be stored in \( d.v_k.Red \)\( (d.v_k.Green) \), respectively. After computing all leaves \( v_j \), \( d.v_k.Red \)\( (d.v_k.Green) \), respectively is the sum of \( S_{r_2}(v_j, v_k) \) \( \Delta S_{r_1}(v_j, v_k) \) for all leave \( v_j \) in the subtree that rooted by \( v_k \) which colored by red (green, respectively).
- \( D \) is the record of the mutated distance of \( T_1 \) and \( T_2 \).
- \( S_{r_1}(v_j, v_k) \) is a temporary for calculating \( S_{r_2}(v_j, v_k) \) for any \( v_k \) in \( T_2 \).
- \( T_{1,v_j,l} \) denotes the left child of \( T_{1,v_j} \), \( T_{1,v_j,l,r} \) denotes the right child of \( T_{1,v_j} \).

The algorithm of mutated distance is presented as follows.

**Input:** Two trees \( T_1 \) and \( T_2 \) with the same \( n \) leaves.

**Output:** The mutated distance between \( T_1 \) and \( T_2 \).

1. **Step 1** Traversal \( T_1 \) and \( T_2 \), and give all nodes an order by BFS, respectively.
2. **Step 2** Find \( sLeafTable_1 \) and \( sLeafTable_2 \), and sort \( sLeafTable_1 \) and \( sLeafTable_2 \) by sequence name.
3. **Step 3** For each internal node \( u_i \) in \( T_1 \) do **Step 4** to **Step 13**.
4. **Step 4** For the subtree which rooted by \( T_{1,u_i} \), color all leaves of its left subtree by red, and color all leaves of its right subtree by green. And color all leaves in \( T_2 \) by the same color this leaf be colored in \( T_1 \).
5. **Step 5** For each be colored leaf \( v_j \) in \( T_2 \) do **Step 6** to **Step 12**.
6. **Step 6** Use \( sLeafTable \) to find leaf \( T_{1,v_k} \) with the
same sequence name of $T_2$ and find $S_{T_2}(v_k, u_i)$.
  - Step 7 For any node $v_1$ in $(T_2, v_1)$, root of $(T_2)$-path do Step 8 to Step 12.
  - Step 8 $S_{T_2}(v_1)$ is the symmetric difference of $S_{T_2}(v_j)$ and MS$(T_2, v_1)$.
  - Step 9 If $T_2, v_1$ is red do Step 10.
  - Step 10 Compute $d, v_1$.Red and $\text{color}_{v_1}.Red(T_2, v_1)$, $d, v_1$.Red add the number of element of the symmetric difference of $S_{T_1}(v_k, u_i)$ and $S_{T_2}(v_j)$. And color$_{v_1}.Red(T_2, v_1)$ add 1.
  - Step 11 If $T_2, v_1$ is green to do Step 12.
  - Step 12 Compute $d, v_1$.Green and color$_{v_1}.Green(T_2, v_1)$, $d, v_1$.Green add the number of element of the symmetric difference of $S_{T_1}(v_k, u_i)$ and $S_{T_2}(v_j)$. And color$_{v_1}.Green(T_2, v_1)$ add 1.
  - Step 13 For any internal node $v_1$ in $T_2$, compute mutated distance $D$. $D$ add the sum of $\text{color}_{v_1}.Green(v_j, l)$ multiplied by $d(v_j, r)$.Red, $\text{color}_{v_1}.Red(v_j, r)$ multiplied by $d(v_j, l)$.Green, $\text{color}_{v_1}.Green(v_j, l)$ multiplied by $d(v_j, l)$.Red and $\text{color}_{v_1}.Red(v_j, l)$ multiplied by $d(v_j, r)$.Green.

B. An Example of the Algorithm

In Fig. 4 and Fig. 5, there are two trees $T_1$ and $T_2$, and the mutated site set of each node. First give $T_1$ and $T_2$ the BFS numbers $u_1, u_2, u_13$ and $v_1, v_2, u_13$. The $sLeafTable_1$ is the leaf table of $T_1$ in Fig. 4. The $sLeafTable_2$ is the leaf table of $T_2$ in Fig. 5. Table I shows the $sLeafTable_1$ and $sLeafTable_2$ sorted by sequence name. This algorithm uses the $sLeafTable$ to find two leaves, which two leaves have the same sequence name of $T_1$ and $T_2$. Then, this algorithm uses the color to compute the color of the mutated distance between $T_1$ and $T_2$. When we fix $u_1$ as the subroot which be computing currently in $T_1$, see Fig. 4. The leaves of the left subtree of $u_1$ are $\{A, B, C, F\}$ that are colored by red, and the leaves of the right subtree of $u_1$ are $\{D, E, G\}$ that are colored by green. The leaves in $T_2$ are colored by the same color. Table II shows after computing all leaves of $T_2$, the values of $d, v_k$.Red and $d, v_k$.Green, when the algorithm color all leaves of the left subtree of the subtree, which rooted by $T_1, u_1$, by red; and color all leaves of its right subtree by green. Table III shows $\text{color}_{v_1}$ values in the algorithm.

We use these two tables to compute the mutated distance between two mixture trees. The mutated distance of $(A, E)$-path, $(B, E)$-path and $(C, E)$-path $= 4 \times 3 + 12 \times 1 = 24$ were computed in $T_2, v_4$. The mutated distance of $(D, F)$-path $= 4 \times 1 + 4 \times 1 = 8$ was computed in $T_2, v_3$. The mutated distance of $(A, G)$-path, $(B, G)$-path and $(C, G)$-path $= 3 \times 3 + 9 \times 1 = 18$ were computed in $T_2, v_2$. The mutated distances of $(A, D)$-path, $(B, D)$-path, $(C, D)$-path, $(F, G)$-path and $(E, E)$-path $= 4 \times 3 + 12 \times 1 + 4 \times 2 + 8 \times 1 = 40$ were computed in $T_2, v_1$. When the subtrees which rooted by $T_1, u_1$ round finish to compute mutated distance, the mutated distance $= 24 + 8 + 18 + 40 = 90$.

Next round the algorithm will fix the node $v_2$, and consider the subtree which rooted by $T_1, v_2$. Then, it will color all leaves of its left subtree by red, and color all leaves of its right subtree by green, and compute mutated distance until each internal node of $T_1$ has been fixed. The mutated distance between two mixture trees will be computed. The mutated distance between $T_1$ and $T_2$ is 146. Table IV shows the complete data when computing the mutated distance between $T_1$ and $T_2$.

![Fig. 4. T_1 colored according to u_1 of T_1.](image1)

![Fig. 5. T_2 colored according to u_1 of T_1.](image2)

| $sLeafTable$ sorted by sequence name with colored by $T_1, u_1$. |
|---------------------------------|---------------------------------|---------------------------------|---------------------|
| $\text{sequence name}$ | $\text{BFS order}$ | $\text{color}_{T_1}$ | $\text{sequence name}$ | $\text{BFS order}$ | $\text{color}_{T_2}$ |
| $A$ | 8 | red | $A$ | 12 | red |
| $B$ | 11 | red | $B$ | 13 | red |
| $C$ | 10 | red | $C$ | 11 | red |
| $D$ | 12 | green | $D$ | 6 | green |
| $E$ | 13 | green | $E$ | 9 | green |
| $F$ | 9 | red | $F$ | 7 | red |
| $G$ | 6 | green | $G$ | 5 | green |
The total time complexity is \(O(n)\) for complete binary trees. The time complexity while finds distance in Section IV-A. Then, we give an example in Section IV-B. Section IV-C is analysis of this algorithm.

### IV. THE IMPROVED ALGORITHM FOR MUTATED DISTANCE

Firstly, we design an improved algorithm for mutated distance in Section IV-A. Then, we give an example in Section IV-B. Section IV-C is analysis of this algorithm.

#### A. The Improved Algorithm

This algorithm improves the time complexity of modified algorithm, it transforms the data MS to TMS. The TMS of vertex represents the difference between the root of \(T_1\) and this vertex. This algorithm uses TMS and \(\text{color}_i\) of vertex to compute the mutated distance between two mixture trees.

Before introducing the algorithm, we have to understand some notations which are used in the algorithm.

- \(T_1,u_j\) denotes a node \(u_j\) in \(T_1\), where \(j\) is in the order of BFS, \(T_2,v_j\) denotes a node \(v_j\) in \(T_2\), where \(j\) is in the order of BFS. Note that \(T_1,u_1 = T_2,v_1\) for some \(j\) for any leaf \(u_i\) of \(T_1\) such that \(T_1,u_i(= T_2,v_j)\) has the same sequence name with \(v_j\) in \(T_2\).

- \(\text{color}_i\) of \(v_i\) denotes the color information of the subtree that rooted by \(v_i\) in \(T_2\). The \(\text{color}_i\) contains two integer: \(\text{color}_i\).\(\text{Red}\) is the amount of leaves that are colored by red, and \(\text{color}_i\).\(\text{Green}\) is the amount of leaves that are colored by green. For example, \(A,B,C\) is three nodes in a tree. Let \(B,C\) be two children of \(A\), then \(\text{color}_i(A) = \text{color}_i(B) + \text{color}_i(C)\). That means, these two values \(\text{color}_i(A)\).\(\text{Red}\) = \(\text{color}_i(B)\).\(\text{Red}\) + \(\text{color}_i(C)\).\(\text{Red}\); \(\text{color}_i(A)\).\(\text{Green}\) = \(\text{color}_i(B)\).\(\text{Green}\) + \(\text{color}_i(C)\).\(\text{Green}\).

- \(\text{sLeafTable}_i\) is the leaves data of \(T_i\) for \(i = 1,2\). The data include sequence name, BFS number, color information. The size of this table is \(n \times 3\), where a row represents one leaf. And each row includes three items: the sequence name represents the sequence title of this leaf, the BFS number is the order of this leaf in the order of BFS, the color information is the color of this leaf, which will be green, red or null.

- \(D\) is the record of the mutated distance of \(T_1\) and \(T_2\).

- \(S_{T_2}(v_j)\) is a temporary for calculating \(S_{T_2}(v_j, v_k)\) for any \(v_k\) in \(T_2\).

- \(T_1,v_j.l\) denotes the left child of \(T_1,v_j\), \(T_1,v_j.r\) denotes the right child of \(T_1,v_j\).

- \(\text{Path}_\text{number}\) of \(v_i\) in \(T_2\) denotes an integer that is the inner product of color information of the two children of the subtree that rooted by \(v_i\). For example, let \(B,C\) be two children of \(A\), then the path number of \(A\) is equal to \(\text{color}_i(B)\).\(\text{Red}\times \text{color}_i(C)\).\(\text{Green}\) + \(\text{color}_i(B)\).\(\text{Green}\times \text{color}_i(C)\).\(\text{Red}\).

- TMS of \(v_i\) denotes a set, which represents the difference mutated sites between root of \(T_1\) and \(v_i\). Moreover, this set reveals the distance between root of \(T_1\) and \(v_i\).

The improved algorithm of mutated distance is presented as follows.

**Input:** Two trees \(T_1\) and \(T_2\) with the same \(n\) leaves.

**Output:** The mutated distance between \(T_1\) and \(T_2\).

1. **Step 1** Traversal \(T_1\) and \(T_2\), and give all nodes an order by BFS, separately.
2. **Step 2** Find \(\text{sLeafTable}_1\) and \(\text{sLeafTable}_2\), and sort \(\text{sLeafTable}_1\) and \(\text{sLeafTable}_2\) by sequence name.
3. **Step 3** Transform \(T_1\), transformed set of mutated sites TMS of root in \(T_1\) is null. For other node \(u_i\) of \(T_1\), compute TMS of node from \(u_2\) to \(u_{2n-1}\); TMS of \(u_i\) is the symmetric difference between TMS of the father of \(u_i\) and MS(\(u_i\)).
4. **Step 4** Transform \(T_2\), the TMS(\(v_j\)) of leaves of \(T_2\) is the same with the TMS(\(u_i\)) of \(T_1\) where \(v_j\) and \(u_i\) has the same sequence name. For any internal node \(v_j\) of \(T_2\), compute TMS(\(v_j\)) from leaf to root, TMS of \(v_j\) is the symmetric difference between TMS(\(v_j.l\)) and MS(\(v_j\)) (\(=\text{TMS}(v_j.r)\) and MS(\(v_j\))).
5. **Step 5** For each internal node \(u_i\) of \(T_1\), do Step 6

**TABLE II**

The \(d.v_k\).\(\text{Green}\) and \(d.v_k\).\(\text{Red}\) table of \(T_2\) according to \(u_1\) of \(T_1\).

<table>
<thead>
<tr>
<th>(k)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d.v_k).(\text{Green})</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(d.v_k).(\text{Red})</td>
<td>12</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

**TABLE III**

The \(\text{color}_i(v_k)\) table of \(T_2\) according to \(u_1\) of \(T_1\).

<table>
<thead>
<tr>
<th>(k)</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{color}_i(v_k)).(\text{Green})</td>
<td>-2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\text{color}_i(v_k)).(\text{Red})</td>
<td>-3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
to Step 8.

- **Step 6** For the subtree which rooted by \( T_1.u_1 \), color all leaves of its left subtree by red, and color all leaves of its right subtree by green. And color all leaves in \( T_2 \) by the same color this leaf be colored in \( T_1 \).
- **Step 7** For each internal node \( v_j \) in \( T_2 \) do **Step 8**
- **Step 8** Compute \( \text{color}_j.\text{Red}(v_j) \), \( \text{color}_j.\text{Green}(v_j) \).
- **Path-number** and mutated distance \( D \) from \( v_{2n-1} \) to \( v_1 \): \( \text{color}_j.\text{Red}(v_j) \) is the sum of \( \text{color}_i.\text{Red}(v_j) \) and \( \text{color}_i.\text{Red}(v_j.r) \); \( \text{color}_j.\text{Green}(v_j) \) is the sum of \( \text{color}_i.\text{Green}(v_j.l) \) and \( \text{color}_i.\text{Green}(v_j.r) \); \( \text{Path-number} \) is the sum of \( \text{color}_i.\text{Green}(v_j.r) \) multiplied by \( \text{color}_i.\text{Red}(v_j,r) \) and \( \text{color}_i.\text{Green}(v_j.l) \) multiplied by \( \text{color}_i.\text{Red}(v_j,l) \); \( D \) is \( \text{Path-number} \) multiplied by two times of the number of element of the symmetric difference between \( \text{TMS}(u_i) \) and \( \text{TMS}(v_j) \).

### TABLE IV

**The distance table in the example of the algorithm.**

<table>
<thead>
<tr>
<th>( k )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>( d.v_k.\text{Green} )</td>
<td>( - )</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( d.v_k.\text{Red} )</td>
<td>( - )</td>
<td>12</td>
<td>4</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>12</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>( \text{color}_j(\text{v}_k).\text{Green} )</td>
<td>( - )</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( \text{color}_j(\text{v}_k).\text{Red} )</td>
<td>( - )</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The subtree rooted by \( T_1,v_1 \)

\[ D = (4 \times 3 + 12 \times 1) + (4 \times 1 + 4 \times 1) + (3 \times 1 + 9 \times 1) + (4 \times 2 + 8 \times 1) + (4 \times 3 + 12 \times 1) = 90 \]

The subtree rooted by \( T_1,v_2 \)

\[ D = 90 + (3 \times 1 + 3 \times 1) + (3 \times 1 + 3 \times 1) + (4 \times 2 + 8 \times 1) = 118 \]

The subtree rooted by \( T_1,v_3 \)

\[ D = 118 + (1 \times 1 + 1 \times 1) + (2 \times 1 + 2 \times 1) = 124 \]

The subtree rooted by \( T_1,v_4 \)

\[ D = 124 + (6 \times 1 + 6 \times 1) = 136 \]

The subtree rooted by \( T_1,v_5 \)

\[ D = 136 + (3 \times 1 + 3 \times 1) = 142 \]

The subtree rooted by \( T_1,v_7 \)

\[ D = 142 + (2 \times 1 + 2 \times 1) = 146 \]

The subtree rooted by \( T_1,v_{13} \)

\[ D = 146 + (6 \times 1 + 6 \times 1) = 152 \]

B. An Example of the Improved Algorithm

We use the same example as previous section in Fig. 4 and Fig. 5 to present how does this algorithm work. There are two trees \( T_1 \) and \( T_2 \), and the mutated site set of each node. First transform Fig. 4 to Fig. 6 and transform Fig. 5 to Fig. 7. Then give \( T_1 \) and \( T_2 \) the BFS numbers \( u_1, u_2, . . . , u_{13} \) and \( v_1, v_2, . . . , v_{13} \). The \( \text{sLeafTable}_1 \) is the leaf table of \( T_1 \) in Fig. 6. The \( \text{sLeafTable}_2 \) is the leaf table of \( T_2 \) in Fig. 7. Table I shows \( \text{sLeafTable}_1 \) and \( \text{sLeafTable}_2 \) sorted by sequence name. This algorithm uses the \( \text{sLeafTable} \) to find two leaves, which two leaves have the same sequence name of \( T_1 \) and \( T_2 \). Then this algorithm using the color to compute the mutated distance between \( T_1 \) and \( T_2 \). When we fix \( u_1 \) as the subroot which be computing currently in \( T_1 \), see Fig. 6. The leaves of the left subtree of \( u_1 \) are \( \{A,B,C,F\} \) that are colored by red, and the leaves of the right subtree of \( u_1 \) are \( \{D,E,G\} \) that are colored by green. The leaves in \( T_2 \) are colored by the same color.

Table III shows \( \text{color}_j \) values in the algorithm. We use these table to compute the mutated distance between two mixture trees. The mutated distance of \( \langle A, E \rangle \)-path, \( \langle B, E \rangle \)-path and
(C, E)-path = (0 × 0 + 3 × 1) × 2 × ({{} ∆ {9, 16, 12, 15}}) = 3 × 2 × 4 = 24 were computed in $T_2, v_1$. The mutated distance of (D, F)-path = (1 × 1 + 0 × 0) × 2 × ({{} ∆ {9, 16, 3, 4}}) = 1 × 2 × 4 = 8 was computed in $T_2, v_3$. The mutated distance of (A, G)-path, (B, G)-path and (C, G)-path = (1 × 0 + 3 × 1) × 2 × ({{} ∆ {9, 16, 2}}) = 3 × 2 × 3 = 18 were computed in $T_2, v_2$. The mutated distances of (A, D)-path, (B, D)-path, (C, D)-path, (F, G)-path and (F, E)-path = (2 × 1 + 3 × 1) × 2 × ({{} ∆ {9, 16, 3, 4}}) = 5 × 2 × 4 = 40 were computed in $T_2, v_1$. When finish the round of computing the mutated distance of the subtree which rooted by $T_1, u_1$, the mutated distance = 24 + 8 + 18 + 40 = 90.

Next round the algorithm will fix the node $u_2$, and consider the subtree which rooted by $T_1, u_2$. Then, it will color all leaves of its left subtree by red, and color all leaves of its right subtree by green, and compute mutated distance until each internal node of $T_1$ has been fixed. The mutated distance between two mixture trees will be computed. The mutated distance between $T_1$ and $T_2$ is 146. Table V shows the complete data when computing the mutated distance between $T_1$ and $T_2$.

C. Analysis

The time complexity of BFS order is $O(n)$ in this algorithm. The time complexity of this algorithm which finds $sLaftTable_i$ is $O(n)$. The time complexity of this algorithm while sorts $sLaftTable_i$ is $O(n\log n)$. The time complexity of this algorithm while transforms $T_1$ and $T_2$ is in time $O(n)$. For each internal node of $T_1$, we compute color $j$ and $D$ between two mixture trees of each node of $T_2$ in time $O(n \times \max\{h(T_1), h(T_2)\} \times s)$, where $s$ means the sequence length of the DNA sequence of species. The total time complexity is $O(n^2 \times s)$. Since the sequence length $s$ is a constant value, the total time complexity is $O(n^2)$. When $T_1$ and $T_2$ are complete binary trees, the height of a tree is $\log n$.

Hence, the time complexity of our algorithm is $O(n\log n)$ for complete binary trees.

V. Conclusion

In this work, we define a metric, the mutated distance, and propose two algorithms to compute the distance with considering the set of mutated sites between two mixture trees. Considering our algorithms and Lin and Juan’s algorithms [16], these algorithms all calculate the distance between two mixture trees. In [16], Lin and Juan also proposed two algorithms, and these two algorithms focus on the time parameter of mixture trees. Table VI shows the time complexity of these two algorithms and our two algorithms.

Hence, the two information of mixture trees are considered by our algorithms and Lin and Juan’s algorithms [16]. One can get a compound-distance $D_c$ for two mixture trees $T_1$ and $T_2$ by our mutated distance $d'$ and mixture distance (or mixture-matching distance) $d_m$ [16]. That means, let $D_c(T_1, T_2) = k_1d' + k_2d_m$ for any two real number $k_1$ and $k_2$, these two real number can be defined according to his (or her) requirement. When one choose $d'$ be mutated distance and $d_m$ be mixture distance, the time complexity of the proposed algorithm for this compound-distance $D_c$ will be $O(n^2)$. In the future, we hope to find other metric for computing the distance with considering these two information, time parameter and set of mutated sites, between two mixture trees and it can satisfy not only pseudo-metric, but also the metric conditions.

TABLE VI

<table>
<thead>
<tr>
<th></th>
<th>Mixture Distance [16]</th>
<th>Mutated Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modified Algorithm</td>
<td>$O(n^2)$</td>
<td>$O(n^2 \times \max{h(T_1), h(T_2)})$</td>
</tr>
<tr>
<td>Improved Algorithm</td>
<td>$O(n\log n)$</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>

ACKNOWLEDGMENT

This research was supported in part by the National Science Council of the Republic of China under grant NSC 100-2221-E-260-024-1.

REFERENCES


### Table V
The distance table in the example of the improved algorithm.

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{v_k}$.Green</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$c_{v_k}$.Red</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{v_k}$.Green</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>$c_{v_k}$.Red</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{v_k}$.Green</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$c_{v_k}$.Red</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>k</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{v_k}$.Green</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$c_{v_k}$.Red</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


Wan Chian Li received her B.S. degree in Information science, National Taipei College of Business in 2008. Her M.S. degrees in Department of computer science and information engineering, National Chi Nan University in 2010. She is a engineer at project development Department of Innovative Center for Cultural and Creative Industries, Tamkang University.

Justie Su-Tzu Juan received her B.S. degree in applied mathematics from Department of Mathematics, Fu Jen Catholic University in 1993, her M.S. and Ph.D. degrees in applied mathematics from National Chiao Tung University, R.O.C. in 1996 and 2000, respectively. She is currently a professor with the Department of computer science and information engineering, National Chi Nan University, R.O.C. Her research interests include graph theory, information security, cryptography, algorithms, and combinatorial mathematics.

Yi-Chun Wang received her B.S. and M.S. degree in Department of computer science and information engineering, National Chi Nan University in 2005 and 2007, respectively. She is currently pursung his PhD degree in Computer Science and Information Engineering at National Chi Nan University, Nantou County, Taiwan. Her research interests include graph theory, algorithm and secret sharing and image sharing.

Shu-Chuan Chen received her B.S. degree in Applied Mathematics from National ChungHsin University in 1994, her M. S. degree in Applied Mathematics from National Donghwa University in 1996, and her PhD in Statistics and Operations Research from Penn State University in 2003. She is currently an Assistant Professor of Statistics in School of Mathematical and Statistical Sciences, Arizona State University, US. Her current research interests include data mining, pattern recognition, and statistical methods in genetic data analysis.