Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

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 bioinformation. 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.

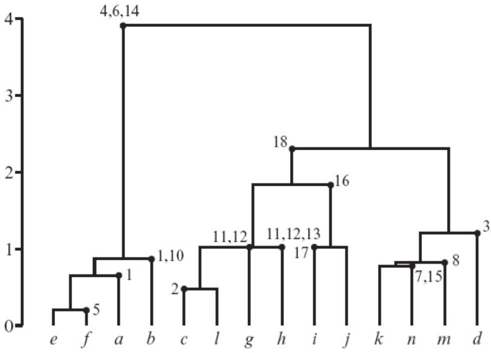


Fig. 1. A mixture tree M_1 (source: [10]).

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]. Besides, the comparison of phylogenetic trees is necessary when we execute phylogenetic queries on databases of phylogenetic trees [4]. Thus, this is an important problem that how to measure distance between two trees for tree comparison. It is difficult to compare two trees. Unlike the comparison of two numbers or points in space [5], there does not have obvious or natural way to measure the distance between two trees. Many tree comparison metrics have been proposed before, including the partition metric [6], the quartet metric [7], the nearest neighborhood interchange metric [8], the metric from the nodal distance algorithm [9], etc. In 2006, Chen and Lindsay proposed a new method to build the mixture tree from DNA sequences [10]. Mixture tree is a type of evolutionary tree. Mixture tree has two information. One of the information is time parameter, and the other is the set of mutated sites. Fig.1 shows a mixture tree.

In 2008, Lin and Juan gave a definition, called *mixture distance*, and the corresponding algorithm to compute distance between two mixture trees [11]. However, their algorithm only considers the time parameter for computing the distance between two mixture trees. Moreover, the time complexity of this algorithm is $O(nlogn)$. 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 computes 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 $P_{T_i}(x,y)$ for any two leaves x, y. That is, the mixture distance between two mixture trees T_1, T_2 is defined as $d_m(T_1,T_2) = \Sigma_{x,y \in V'} |P_{T_1}(x,y) - P_{T_2}(x,y)|$, where V' is the set of leaves of T_1 (equals to the set of leaves of T_2) and $P_{T_i}(x,y)$ denote the time parameter of the least

Wan Chian Li, Justie Su-Tzu Juan and Yi-Chun Wang are with Department of Computer Science and Information Engineering National Chi Nan University Puli, Nantou, 54561, Taiwan, R.O.C.

Shu-Chuan Chen is with School of Mathematical and Statistical Sciences Arizona State University, Tempe, AZ 85287, USA

*Corresponding author. Email: jsjuan@ncnu.edu.tw.

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

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, a algorithm for the mutated distance is proposed. Section IV will proposes 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 V nodes and n leaves, 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 $MS_T(v_i)$, *mutated sites set*, that records the set of all sites of a species mutation occuring from its father. Fig. 2 shows the data tree of the mixture tree in Fig. 1.

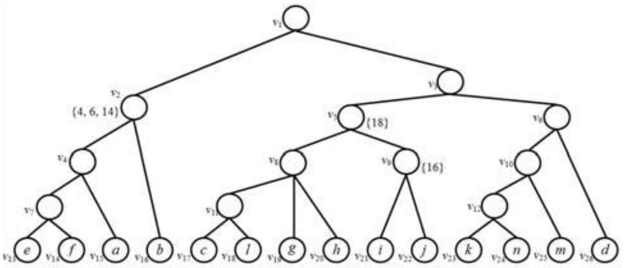


Fig. 2. A data tree for the associated mixture tree M_1.

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 $(x, LCAT(x, y))$-path$-LCAT(x, y)$ in T.

The notation $\triangle$ is symmetric difference of two sets. Let $\mathrm{LCA}_T(x, y)$ be the least common ancestors of $x, y \in V'(T)$ in T. Let $V_{T_i}(x, y)$-*path* be the vertex set of $(x,\mathrm{LCA}_T(x, y))$-*path*$-\mathrm{LCA}_T(x, y)$ in T. Let $V_{T_i}(x, y)$-*path* $= \{v_1 = x, v_2, \ldots, v_t = \mathrm{LCA}_T(x, y)\}$, and $S_T(x, y)$ be the set of $\mathrm{MS}_T(v_1) \triangle \mathrm{MS}_T(v_2) \triangle \ldots \triangle \mathrm{MS}_T(v_t - 1)$. Define $d'(T_1, T_2)$ be the *mutated distance* between two mixture trees, T_1 and T_2, by $d'(T_1, T_2) = \sum_{x,y \in V'} (|S_{T_1}(x, y) \triangle S_{T_2}(x, y)| + |S_{T_1}(y, x) \triangle S_{T_2}(y, x)|)$ 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 $d'(A, B) = d'(B, A)$ 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 $|S_A(y, x) \triangle S_B(y, x)| = |S_B(y, x) \triangle S_A(y, x)|$, for any two leaves x, y. $d'(A, B) = \sum_{x,y \in V'} |S_A(x, y) \triangle S_B(x, y)| + |S_A(y, x) \triangle S_B(y, x)| = \sum_{x,y \in V'} |S_B(x, y) \triangle S_A(x, y)| + |S_B(y, x) \triangle S_A(y, x)|$ $=d'(B, A)$

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 same set of leaves V'. By the definition,
$d'(T_1, T_2) = \sum_{x,y \in V'} (|S_{T_1}(x, y) \triangle S_{T_2}(x, y)| + |S_{T_1}(y, x) \triangle S_{T_2}(y, x)|)$, $d'(T_2, T_3) = \sum_{x,y \in V'} |S_{T_2}(x, y) \triangle S_{T_3}(x, y)| + |S_{T_2}(y, x) \triangle S_{T_3}(y, x)|$ and $d'(T_3, T_1) = \sum_{x,y \in V'} |S_{T_3}(x, y) \triangle S_{T_1}(x, y)| + |S_{T_3}(y, x) \triangle S_{T_1}(y, x)|$
Our goal is to prove $d'(T_1, T_2) + d'(T_2, T_3) \geq d'(T_3, T_1)$. 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 $S_1(x, y) = |S_{T_1}(x, y) \triangle S_{T_2}(x, y)| + |S_{T_2}(x, y) \triangle S_{T_3}(x, y)| - |S_{T_3}(x, y) \triangle S_{T_1}(x, y)| \geq 0$, $S_2(x, y) = |S_{T_1}(y, x) \triangle S_{T_2}(y, x)| + |S_{T_2}(y, x) \triangle S_{T_3}(y, x)| - |S_{T_3}(y, x) \triangle S_{T_1}(y, x)| \geq 0$ for any two leaves x and y in V'. We have $S_1(x, y) = |S_{T_1}(x, y) \triangle S_{T_2}(x, y)| + |S_{T_2}(x, y) \triangle S_{T_3}(x, y)| - |S_{T_3}(x, y) \triangle S_{T_1}(x, y)| = |S_{T_1}(x, y) \cup S_{T_2}(x, y)| - |S_{T_1}(x, y) \cap S_{T_2}(x, y)| + |S_{T_2}(x, y) \cup S_{T_3}(x, y)| - |S_{T_2}(x, y) \cap S_{T_3}(x, y)| - |S_{T_3}(x, y) \cup S_{T_1}(x, y)| + |S_{T_3}(x, y) \cap S_{T_1}(x, y)|$
Since $|S_{T_i}(x, y) \cup S_{T_j}(x, y)| = |S_{T_i}(x, y)| + |S_{T_j}(x, y)| - |S_{T_i}(x, y) \cap S_{T_j}(x, y)|$, $S_1(x, y) = \{|S_{T_1}(x, y)| + |S_{T_2}(x, y)| + |S_{T_2}(x, y)| + |S_{T_3}(x, y)| - 2|S_{T_1}(x, y) \cap S_{T_2}(x, y)| - 2|S_{T_2}(x, y) \cap S_{T_3}(x, y)| - |S_{T_1}(x, y)| - |S_{T_2}(x, y)| + 2|S_{T_1}(x, y) \cap S_{T_3}(x, y)|\} = \{2|S_{T_2}(x, y)| - 2|S_{T_1}(x, y) \cap S_{T_2}(x, y)| - 2|S_{T_2}(x, y) \cap S_{T_3}(x, y)| + 2|S_{T_1}(x, y) \cap S_{T_3}(x, y)|\} = 2\{|S_{T_2}(x, y)| - |S_{T_1}(x, y) \cap S_{T_2}(x, y)| - |S_{T_2}(x, y) \cap S_{T_3}(x, y)| + |S_{T_1}(x, y) \cap S_{T_3}(x, y)|\} = 2\{|S_{T_2}(x, y) \cup (S_{T_1}(x, y) \cap S_{T_3}(x, y))| + |S_{T_2}(x, y) \cap (S_{T_1}(x, y) \cap S_{T_3}(x, y))| - |(S_{T_1}(x, y) \cap S_{T_2}(x, y)) \cup (S_{T_2}(x, y) \cap S_{T_3}(x, y))| - |(S_{T_1}(x, y) \cap S_{T_2}(x, y)) \cap (S_{T_2}(x, y) \cap S_{T_3}(x, y))|\} = 2\{|S_{T_2}(x, y) \cup (S_{T_1}(x, y) \cap S_{T_3}(x, y))| +$

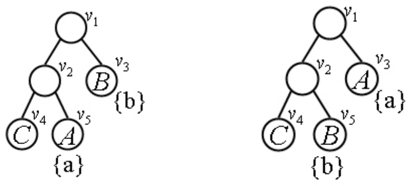
$|S_{T_2}(x,y)\cap S_{T_1}(x,y)\cap S_{T_3}(x,y)|-|(S_{T_1}(x,y)\cap S_{T_2}(x,y))\cup (S_{T_2}(x,y)\cap S_{T_3}(x,y))|-|S_{T_1}(x,y)\cap S_{T_2}(x,y)\cap S_{T_3}(x,y)|\} = 2\{|S_{T_2}(x,y) \cup (S_{T_1}(x,y) \cap S_{T_3}(x,y))| - |(S_{T_1}(x,y) \cap S_{T_2}(x,y)) \cup (S_{T_2}(x,y) \cap S_{T_3}(x,y))|\}$.
Since $|S_{T_2}(x,y) \cup (S_{T_2}(x,y) \cap S_{T_3}(x,y))| \geq |S_{T_2}(x,y)| \geq |S_{T_2}(x,y) \cap (S_{T_2}(x,y) \cup S_{T_3}(x,y))|$, we have $S_1(x,y) = 2\{|S_{T_2}(x,y) \cup (S_{T_1}(x,y) \cap S_{T_3}(x,y))| - |(S_{T_1}(x,y) \cap S_{T_2}(x,y)) \cup (S_{T_2}(x,y) \cap S_{T_3}(x,y))|\} \geq 0$. In the same way, we can also prove that $S_2 \geq 0$. Hence, adding these two terms for any leaves x and y, we have $d'(T_1,T_2) + d'(T_2,T_3) - d'(T_3,T_1) = \sum_{x,y\in V'} \{S_1(x,y) + S_2(x,y)\} \geq 0$.

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 $S_A(x,y) = S_B(x,y)$ and $S_A(y,x) = S_B(y,x)$ for any two leaves x, y. Hence, $|S_A(x,y) \triangle S_B(x,y)| = 0$ and $|S_A(y,x) \triangle S_B(y,x)| = 0$ for any two leaves x, y. That implies $\sum_{x,y\in V'} |S_A(x,y) \triangle S_B(x,y)| + |S_A(y,x) \triangle S_B(y,x)| = 0$.

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

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 dose not equal to tree B.*



(a) T_1

(b) T_2

Fig. 3. T_1 and T_2 of a counterexample.

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

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 $S_T(x,y)$ 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.

- $\boldsymbol{T_1.u_j}$ denotes a node u_j in T_1, where j is in the order of BFS, $\boldsymbol{T_2.v_j}$ denotes a node v_j in T_2, where j is in the order of BFS. Note that $T_1.u_i = T_2.v_j$ 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.
- $\boldsymbol{color\_i}$ of v_i denotes the color information of the subtree that rooted by v_i 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 $color\_i(A) = color\_i(B) + color\_i(C)$. That means, these two values $color\_i(A).\ Red = color\_i(B).Red + color\_i(C).Red$; $color\_i(A).Green = color\_i(B).Green + color\_i(C).\ Green$.
- $\boldsymbol{sLeafTable_i}$ 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.
- $\boldsymbol{d.v_k.Red}$ $(\boldsymbol{d.v_k.Green})$ of $T_2.v_k$ for $T_1.v_j$ and $T_2.v_j$ denotes the sum of symmetric difference between $S_{T_1}(v_j, v_i)$ and $S_{T_2}(v_j, v_k)$ for v_k in (v_1, v_j)-path of T_2 for any leaf v_j when we fix i. And when we fix v_i, 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 $S_{T_2}(v_j, v_k) \triangle S_{T_1}(v_j, v_i)$ for all leave v_j in the subtree that rooted by v_k which colored by red (green, respectively).
- $\boldsymbol{D}$ is the record of the mutated distance of T_1 and T_2.
- $\boldsymbol{S_{T_2}(v_j)}$ is a temporary for calculating $S_{T_2}(v_j, v_k)$ for any v_k in T_2.
- $\boldsymbol{T_i.v_j.l}$ denotes the left child of $T_i.v_j$, $\boldsymbol{T_i.v_j.l.r}$ denotes the right child of $T_i.v_j$.

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.

- Step 1 Traversal T_1 and T_2, and give all nodes an order by BFS, respectively.
- Step 2 Find $sLeafTable_1$ and $sLeafTable_2$, and sort $sLeafTable_1$ and $sLeaf\ Table_2$ by sequence name.
- Step 3 For each internal node u_i in T_1 do Step 4 to Step 13.
- 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.
- Step 5 For each be colored leaf v_j in T_2 do Step 6 to Step 12.
- Step 6 Use $sLeafTable$ to find leaf $T_1.v_k$ with the

same sequence name of $T_2.v_j$, and find $S_{T_1}(v_k, u_i)$.

- `Step 7` For any node v_l in ($T_2.v_j$, root of T_2)-path do `Step 8 to Step 12`.
- `Step 8` $S_{T_2}(v_j)$ is the symmetric difference of $S_{T_2}(v_j)$ and MS($T_2.v_l$).
- `Step 9` If $T_2.v_j$ is red to do `Step 10`.
- `Step 10` Compute $d.v_l.Red$ and *color_i*$.Red(T_2.v_l)$, $d.v_l.Red$ 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_i*$.Red(T_2.v_l)$ add 1.
- `Step 11` If $T_2.v_j$ is green to do `Step 12`.
- `Step 12` Compute $d.v_l.Green$ and *color_i*$.Green(T_2.v_l)$, $d.v_l.Green$ 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_ i*$.Green(T_2.v_l)$ add 1.
- `Step 13` For any internal node v_j in T_2, compute mutated distance D. D add the sum of *color_i*$.Green(v_j.l)$ multiplied by $d.(v_j.r).Red$, *color_i*$.Red(v_j.r)$ multiplied by $d.(v_j.l).Green$, *color_i*$.Green(v_j.r)$ multiplied by $d.(v_j.l).Red$ and *color_i*$.Red(v_j.l)$ 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 $v_1, v_2, , v_{13}$. 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 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 *color_i* 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 (F, E)-path $= 4 \times 3 + 12 \times 1 + 4 \times 2 + 8 \times 1 = 40$ were computed in $T_2.v_1$. When the subtree 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 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 IV shows the complete data when computing the mutated distance between T_1 and T_2.

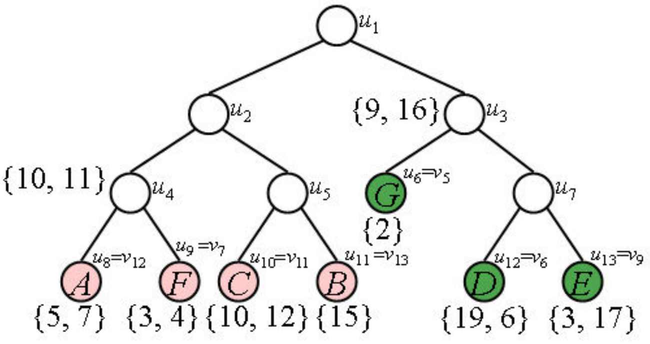


Fig. 4. T_1 colored according to u_1 of T_1.

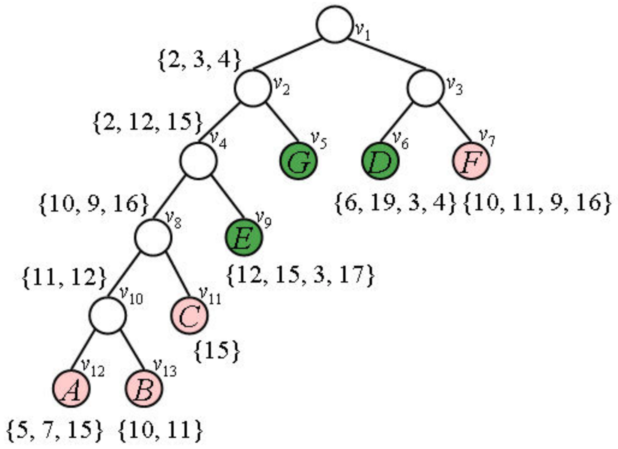


Fig. 5. T_2 colored according to u_1 of T_1.

TABLE I
$sLeafTable$ SORTED BY SEQUENCE NAME WITH COLORED BY $T_1.u_1$.

T_1			T_2		
sequence name	BFS order	*color_i*	sequence name	BFS order	*color_i*
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

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

TABLE II
THE $d.v_k.Green$ AND $d.v_k.Red$ TABLE OF T_2 ACCORDING TO u_1 OF T_1.

$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	8	4	3	3	4	0	0	4	0	0	0	0
$\boldsymbol{d.v_k.Red}$	-	12	4	9	0	0	4	12	0	6	3	3	3

TABLE III
THE *color_i*(v_k) TABLE OF T_2 ACCORDING TO u_1 OF T_1.

$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
color_i$\boldsymbol{(v_k).Green}$	-	2	1	1	1	1	0	0	1	0	0	0	0
color_i$\boldsymbol{(v_k).Red}$	-	3	1	3	0	0	1	3	0	2	1	1	1

C. Analysis

The time complexity of BFS order is $O(n)$ in this algorithm. The time complexity while finds $sLaftTable_i$ is $O(n)$. The time complexity of thise algorithm while sorts $sLaftTable_i$ is $O(n\log n)$. For each internal node of T_1 we compute *color_i*, $d.v_k.Red$, $d.v_k.Green$ 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 $O(n^2 \times max\{h(T_1), h(T_2)\} \times s)$. Since the sequence length s is a constant value, the total time complexity is $O(n^2 \times max\{h(T_1), h(T_2)\})$. 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 $O(n^2\log n)$ for complete binary trees.

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 sites between the root of T_1 and this vertex. This algorithm uses TMS and *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.

- $\boldsymbol{T_1.u_j}$ denotes a node u_j in T_1, where j is in the order of BFS, $\boldsymbol{T_2.v_j}$ denotes a node v_j in T_2, where j is in the order of BFS. Note that $T_1.u_i = T_2.v_j$ 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_i denotes the color information of the subtree that rooted by v_i 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_i*$(A) =$ *color_i*$(B) +$ *color_i*(C). That means, these two values *color_i*(A). $Red =$ *color_i*$(B).Red +$ *color_i*$(C).Red$; *color_ i*$(A).Green =$ *color_i*$(B).Green +$ *color_i*$(C).Green$.
- $\boldsymbol{sLeafTable_i}$ 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.
- $\boldsymbol{D}$ is the record of the mutated distance of T_1 and T_2.
- $\boldsymbol{S_{T_2}(v_j)}$ is a temporary for calculating $S_{T_2}(v_j, v_k)$ for any v_k in T_2.
- $\boldsymbol{T_i.v_j.l}$ denotes the left child of $T_i.v_j$, $\boldsymbol{T_i.v_j.r}$ denotes the right child of $T_i.v_j$.
- ***Path_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 *color_i*$(B).Red \times$ *color_i*$(C).Green +$ *color_i*$(B).Green \times$ *color_i*$(C).Red$.
- **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.

- Step 1 Traversal T_1 and T_2, and give all nodes an order by BFS, seperatedly.
- Step 2 Find $sLeafTable_1$ and $sLeafTable_2$, and sort $sLeafTable_1$ and $sLeaf\ Table_2$ by sequence name.
- 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).
- Step 4 Transform T_2, the TMS(v_j) of leaves in 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) (=TMS$(v_j.r)$ and MS(v_j)).
- Step 5 For each internal node u_i in T_1, do Step 6

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

TABLE IV
THE DISTANCE TABLE IN THE EXAMPLE OF THE ALGORITHEM.

the subtree rooted by $T_1.u_1$	$D = (4 \times 3 + 12 \times 1) + (4 \times 1 + 4 \times 1) + (3 \times 3 + 9\times 1) + 4(4 \times 2 + 8 \times 1) + (4 \times 3 + 12 \times 1) = 90$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	8	4	3	3	4	0	0	4	0	0	0	0
$\boldsymbol{d.v_k.Red}$	-	12	4	9	0	0	4	12	0	6	3	3	3
$\boldsymbol{color\_i(v_k).Green}$	-	2	1	1	1	1	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	3	1	3	0	0	1	3	0	2	1	1	1
the subtree rooted by $T_1.u_2$	$D = 90 + (3 \times 1 + 3 \times 1) + (3 \times 1 + 3 \times 1) + (4 \times 2+ 8 \times 1) = 118$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	8	0	6	0	0	0	8	0	3	3	0	3
$\boldsymbol{d.v_k.Red}$	-	4	4	3	0	0	4	4	0	3	0	3	0
$\boldsymbol{color\_i(v_k).Green}$	-	2	0	2	0	0	0	2	0	1	1	0	1
$\boldsymbol{color\_i(v_k).Red}$	-	1	1	1	0	0	1	1	0	1	0	1	0
the subtree rooted by $T_1.u_3$	$D = 118 + (1 \times 1 + 1 \times 1) + (2 \times 1 + 2 \times 1) = 124$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	2	2	1	0	2	0	0	2	0	0	0	0
$\boldsymbol{d.v_k.Red}$	-	2	0	0	1	0	0	0	0	0	0	0	0
$\boldsymbol{color\_i(v_k).Green}$	-	1	1	1	0	1	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	0	1	0	0	0	0	0	0	0	0
the subtree rooted by $T_1.u_4$	$D = 124 + (6 \times 1 + 6 \times 1) = 136$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	0	6	0	0	0	6	0	0	0	0	0	0
$\boldsymbol{d.v_k.Red}$	-	6	0	5	0	0	0	6	0	3	0	1	0
$\boldsymbol{color\_i(v_k).Green}$	-	0	1	0	0	0	1	0	0	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	1	0	0	0	1	0	1	0	1	0
the subtree rooted by $T_1.u_5$	$D = 136 + (3 \times 1 + 3 \times 1) = 142$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	4	0	3	0	0	0	4	0	3	0	0	3
$\boldsymbol{d.v_k.Red}$	-	4	0	3	0	0	0	4	0	0	3	0	0
$\boldsymbol{color\_i(v_k).Green}$	-	1	0	1	0	0	0	1	0	1	0	0	1
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	1	0	0	0	1	0	0	1	0	0
the subtree rooted by $T_1.u_7$	$D = 142 + (2 \times 1 + 2 \times 1) = 146$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{d.v_k.Green}$	-	2	0	1	0	0	0	0	2	0	0	0	0
$\boldsymbol{d.v_k.Red}$	-	0	2	0	0	2	0	0	0	0	0	0	0
$\boldsymbol{color\_i(v_k).Green}$	-	1	0	1	0	0	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	0	1	0	0	1	0	0	0	0	0	0	0

`to Step 8.`

- `Step 6` 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.
- `Step 7` For each internal node v_j in T_2 do `Step 8`
- `Step 8` Compute $color\_i.Red(v_j)$, $color\_i.Green(v_j)$, $Path\text{-}number$ and mutated distance D from v_{2n-1} to v_1; $color\_i.Red(v_j)$ is the sum of $color\_i.Red(v_j.l)$ and $color\_i.Red(v_j.r)$; $color\_i.Green(v_j)$ is the sum of $color\_i.Green(v_j.l)$ and $color\_i.Green(v_j.r)$; $Path\text{-}number$ is the sum of $color\_i.Green(v_j.l)$ multiplied by $color\_i.Red(v_j.r)$ and $color\_i.Green(v_j.r)$ multiplied by $color\_i.Red(v_j.l)$; D is $Path\text{-}number$ multiplied by two times of the number of element of the symmetric difference between TMS(u_i) and TMS(v_j).

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 $sLeafTable_1$ is the leaf table of T_1 in Fig. 6. The $sLeafTable_2$ is the leaf table of T_2 in Fig. 7. Table I shows the $sLeafTable_1$ and $sLeafTable_2$ sorted by sequence name. This algorithm uas the $sLeafTable$ 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 $color\_i$ values in the algorithm. We use these table to compute the mutated distance between two mixture trees. The mutated distance of (A, E)-path, (B, E)-path and

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

(C, E)-path= $(0 \times 0 + 3 \times 1) \times 2 \times (|\{\} \triangle \{9, 16, 12, 15\}|) = 3 \times 2 \times 4 = 24$ were computed in $T_2.v_4$. The mutated distance of (D, F)-path $= (1 \times 1 + 0 \times 0) \times 2 \times (|\{\} \triangle \{9, 16, 3, 4\}|) = 1 \times 2 \times 4 = 8$ was computed in $T_2.v_3$. The mutated distance of (A, G)-path, (B, G)-path and (C, G)-path $= (1 \times 0 + 3 \times 1) \times 2 \times (|\{\} \triangle \{9, 16, 2\}|) = 3 \times 2 \times 3 = 18$ 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 $= (2 \times 1 + 3 \times 1) \times 2 \times (|\{\} \triangle \{9, 16, 3, 4\}|) = 5 \times 2 \times 4 = 40$ 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 $= 24 + 8 + 18 + 40 = 90$.

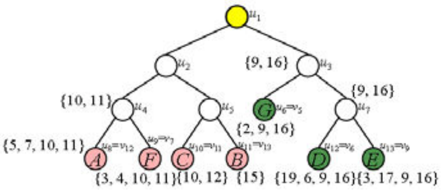


Fig. 6. The transform T_1 colored according to u_1 of T_1.

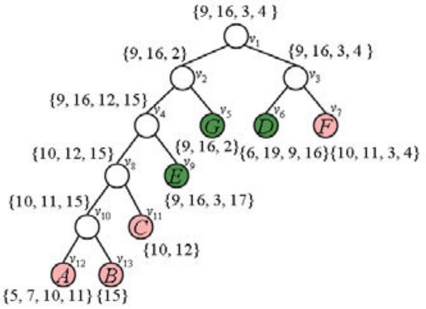


Fig. 7. The transform T_2 colored according to u_1 of T_1.

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 $color\_i$ 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 paremeter 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 Juans 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 $D_c(T_1, T_2) = k_1 d' + k_2 d_m$ 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
The comparison of the algorithms of our work.

	Mixture Distance [16]	**Mutated Distance**
Modified Algorithm	$O(n^2)$	$O(n^2 \times max\{h(T_1), h(T_2)\})$
Improved Algorithm	$O(nlogn)$	$O(n^2)$

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- .

References

[1] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees," *Mol Biol Evol*, vol. 4, no. 4, pp. 406–425, Jul 1987.

[2] M. L. Lesperance and J. D. Kalbeisch, "An algorithm for computing the nonparametric mle of a mixing distribution," *Journal of the American Statistical Association*, vol. 87, no. 417, pp. 120–126, Mar. 1992.

[3] M. A. Steel, "The maximum likelihood point for a phylogenetic tree is not unique," *Systematic Biology*, vol. 43, pp. 560–564, 1994.

[4] G. Valiente, "A fast algorithmic technique for comparing large phylogenetic trees," *SPIRE*, pp. 370–375, 2005.

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf

TABLE V
THE DISTANCE TABLE IN THE EXAMPLE OF THE IMPROVED ALGORITHM.

the subtree rooted by $T_1.u_1$	$D=(1\times 0+1\times 0)\times 2(\vert\{10,11,15\}\triangle\{\}\vert)+(1\times 0+2\times 0)\times 2(\vert\{10,12,15\}\triangle\{\}\vert)+(3\times 1+0\times 0)\times 2(\vert\{9,16,12,15\}\triangle\{\}\vert)+(1\times 1+0\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)+(3\times 1+0\times 1)\times 2(\vert\{9,16,2\}\triangle\{\}\vert)+(3\times 1+2\times 1)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)=24+8+18+40=90$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	2	1	1	1	1	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	3	1	3	0	0	1	3	0	2	1	1	1
the subtree rooted by $T_1.u_2$	$D=D+(1\times 1+0\times 0)\times 2(\vert\{10,11,15\}\triangle\{\}\vert)+(1\times 1+0\times 1)\times 2(\vert\{10,12,15\}\triangle\{\}\vert)+(1\times 0+0\times 2)\times 2(\vert\{9,16,12,15\}\triangle\{\}\vert)+(0\times 0+1\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)+(1\times 0+0\times 2)\times 2(\vert\{9,16,2\}\triangle\{\}\vert)+(1\times 0+1\times 2)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)=90+6+6+0+0+0+16=118$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	2	0	2	0	0	0	2	0	1	1	0	1
$\boldsymbol{color\_i(v_k).Red}$	-	1	1	1	0	0	1	1	0	1	0	1	0
the subtree rooted by $T_1.u_3$	$D=D+(0\times 0+0\times 0)\times 2(\vert\{10,11,15\}\triangle\{9,16\}\vert)+(0\times 0+0\times 0)\times 2(\vert\{10,12,15\}\triangle\{9,16\}\vert)+(0\times 1)+0\times 0\times 2(\vert\{9,16,12,15\}\triangle\{9,16\}\vert)+(0\times 0+0\times 1)\times 2(\vert\{9,16,3,4\}\triangle\{9,16\}\vert)+(0\times 0+1\times 1)\times 2(\vert\{9,16,2\}\triangle\{9,16\}\vert)+(1\times 1+1\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{9,16\}\vert)=118+2+4=124$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	1	1	1	0	1	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	0	1	0	0	0	0	0	0	0	0
the subtree rooted by $T_1.u_4$	$D=D+(1\times 0+0\times 0)\times 2(\vert\{10,11,15\}\triangle\{10,11\}\vert)+(1\times 0+0\times 0)\times 2(\vert\{10,12,15\}\triangle\{10,11\}\vert)+(1\times 0+0\times 0)\times 2(\vert\{9,16,12,15\}\triangle\{10,11\}\vert)+(0\times 1+0\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{10,11\}\vert)+(1\times 0+0\times 0)\times 2(\vert\{9,16,2\}\triangle\{10,11\}\vert)+(1\times 1+0\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{10,11\}\vert)=124+12=136$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	0	1	0	0	0	1	0	0	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	1	0	0	0	1	0	1	0	1	0
the subtree rooted by $T_1.u_5$	$D=D+(0\times 1+0\times 0)\times 2(\vert\{10,11,15\}\triangle\{\}\vert)+(0\times 0+1\times 1)\times 2(\vert\{10,12,15\}\triangle\{\}\vert)+(1\times 0+0\times 1)\times 2(\vert\{9,16,12,15\}\triangle\{\}\vert)+(0\times 0+0\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)+(1\times 0+0\times 1)\times 2(\vert\{9,16,2\}\triangle\{\}\vert)+(1\times 0+0\times 1)\times 2(\vert\{9,16,3,4\}\triangle\{\}\vert)=136+6=142$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	1	0	1	0	0	0	1	0	1	0	0	1
$\boldsymbol{color\_i(v_k).Red}$	-	1	0	1	0	0	0	1	0	0	1	0	0
the subtree rooted by $T_1.u_7$	$D=D+(0\times 0+0\times 0)\times 2(\vert\{10,11,15\}\triangle\{9,16\}\vert)+(0\times 0+0\times 0)\times 2(\vert\{10,12,15\}\triangle\{9,16\}\vert)+(0\times 1+0\times 0)\times 2(\vert\{9,16,12,15\}\triangle\{9,16\}\vert)+(1\times 0+0\times 0)\times 2(\vert\{9,16,3,4\}\triangle\{9,16\}\vert)+(0\times 0+0\times 1)\times 2(\vert\{9,16,2\}\triangle\{9,16\}\vert)+(0\times 0+1\times 1)\times 2(\vert\{9,16,3,4\}\triangle\{9,16\}\vert)=142+4=146$												
$\boldsymbol{k}$	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
$\boldsymbol{color\_i(v_k).Green}$	-	1	0	1	0	0	0	0	1	0	0	0	0
$\boldsymbol{color\_i(v_k).Red}$	-	0	1	0	0	1	0	0	0	0	0	0	0

[5] M. A. Steel and D. Penny, "Distributions of tree comparison metrics-some new results," *Systematic Biology*, vol. 42, no. 2, pp. 126–141, 1993.

[6] D. F. Robinson and L. R. Foulds, "Comparison of phylogenetic trees," *Mathematical Biosciences*, vol. 53, no. 1–2, pp. 131–147, February 1981.

[7] C. A. Meacham, G. F. Estabrook, and F. R. McMorris, "Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units," *Systematic Zoology*, vol. 34, no. 2, pp. 193–200, 1985.

[8] B. Dasgupta, X. He, T. Jiang, M. Li, J. Tromp, and L. Zhang, "Proceedings of the dimacs workshop on discrete problems with medical applications, dimacs series in discrete mathematics and theoretical computer science," *American Mathematical Society*, vol. 55, pp. 125–143, 2000.

[9] J. Bluis, D. Shin, and J. Bluis, "Nodal distance algorithm: calculating a phylogenetic tree comparison metric," *Proc. Third IEEE Symposium on Bioinformatics and Bioengineering (D. Shin, ed.)*, pp. 87–94, 2003.

[10] S.-C. Chen and B. G. Lindsay, "Building mixture trees from binary sequence data," *Biometrika*, vol. 93, no. 4, pp. 843–860, 2006.

[11] C.-H. Lin and J. S.-Z. Juan, "Computing the mixture distance between mixture tree," *Proceedings of the 2008 international conference on bioinformatice & computational biology*, vol. I, pp. 98–103, 2008.

[12] G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen, "Computing the quartet distance between evolutionary trees in time $o(nlogn)$," *Algorithmica*, vol. 38, no. 2, pp. 377–395, 2003.

[13] W. T. Williams and H. T. Clifford, "On the comparison of two classifications of the same set of elements," *Taxon*, vol. 20, no. 4, pp. 519–522, Aug. 1971.

[14] J. L. Kelley, *General Topology I: Basic Concepts and Constructions Dimension Theory*. Encyclopaedia of Mathematical Sciences, 1990.

[15] M. J. Fortin, M. R. T. Dale, and J. V. Hoef, "Encyclopedia of environmetrics," vol. 4, ch. Spatial analysis in ecology, pp. 2051–2058, John Wiley & Sons, Ltd, 2002.

[16] C.-H. Lin, "A study on measuring distance between two mixture trees," *In Partial Fulfillment of the Requirements for the Degree of Master of Science, Department of Computer Science and Information Engineering National Chi Nan University, Puli, Nantou Hsien, Taiwan, Republice of China*, Juan 2008.

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 pursuing 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.

Open Science Index, Mathematical and Computational Sciences Vol:6, No:5, 2012 publications.waset.org/14336.pdf