CALCULATION, VISUALIZATION, AND MANIPULATION OF MASTS (MAXIMUM AGREEMENT SUBTREES)

by

SHIMING DONG

(Under the Direction of Eileen Kraemer)

ABSTRACT

Phylogenetic trees are used to represent the evolutionary history of a set of species. Comparison of multiple phylogenetic trees can help researchers find the common classification of a tree group, compare tree construction inferences or obtain distances between trees. We present TreeAnalyzer, a freely available package for phylogenetic tree comparison. A MAST (Maximum Agreement Subtree) algorithm is implemented to compare the trees. Additional features of this software include tree comparison, visualization, manipulation, labeling, and printing.

INDEX WORDS: phylogenetic tree, MAST, metric

CLACULATION, VISUALIZATION, AND MANIPULATION OF MASTS (MAXIMUM AGREEMENT SUBTREES)

by

SHIMING DONG

B.S., East China University of Science and Technology, P.R.China, 1998

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2004

CALCULATION, VISUALIZATION, AND MANIPULATION OF MASTS (MAXIMUM AGREEMENT SUBTREES)

by

SHIMING DONG

Major Professor: Eileen Kraemer

Committee: Robert W. Robinson
Liming Cai

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
August 2004

DEDICATION

For my family, who offered me unconditional love and support throughout the course of this thesis.

ACKNOWLEDGEMENTS

First, I want to thank Dr. Eileen Kraemer, as my major professor and designer of the software, she always gives me advice and direction. And I also want to thank Dr. Greg Siragusa and Dr. Kelli Hiett in shaping the goals and requirements of this software.

And finally, a special word of thanks goes to my husband Minrong Song for his constant support and encouragement.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

CHAPTER 1

INTRODUCTION

1.1 Introduction

1.1.1 Definition of Phylogenetic Tree

A phylogenetic tree is a structure frequently used to represent the evolutionary history of a set of species. According to the formal definition in [1], a phylogenetic tree *(or an evolutionary tree)* for a species set *A* is a rooted tree in which the leaves (or taxa) are uniquely labeled by the species in *A*, and the internal vertices represent ancestors. In [2], the author defined the unrooted phylogenetic tree as an unrooted tree with labeled leaves in which no interior vertex has just two incident edges; the labels associated with the tree's leaves identify organisms or evolutionary units in a study collection. In a rooted phylogenetic tree, the root is the ancestor of all the nodes that compose the tree. An unrooted tree lacks a root, and does not allow the determination of ancestors and descendants. The branch length usually represents the number of changes that have occurred in that branch and is proportional to time or to the number of fixed novel mutations that have accumulated [3].

Constructing a phylogenetic tree is a fundamental problem in the field of bioinformatics, because it provides a great deal of information to the scientists. For instance, the evolutionary relationships among species highlighted by these trees may provide information about the biochemical machinery of the organisms [4]. Studies of evolutionary trees are also helpful for research in gene evolution, population subdivision, analysis of mating systems and paternity testing, as well as studies of individual relatedness, geographic variation, and species boundaries.

We have developed **TreeAnalyzer**, a phylogenetic tree visualization and analysis software package developed to meet the needs of research microbiologists at the Russell Research Center of the U.S. Department of Agriculture (USDA)[1]. TreeAnalyzer reads an unrooted phylogenetic tree from a NEXUS [5] or TreeAnalyzer format file and visualizes it in a tree structure. In TreeAnalyzer, the MAST (Maximum Agreement Subtree) metric is computed for tree comparison. The detailed description of MAST metric is provided in section 1.2.5.. The MAST metric is used to compare multiple trees and the resultant MASTs are visualized by mapping them onto the original compared trees in TreeAnalyzer. The MASTs are highlighted so as to directly differentiate the common area from the dissimilar area of the compared trees. A normalized number is computed to represent the proportion of common parts of those trees. The compared trees may have different numbers of leaves or different leaf names. Theoretically, there is no limitation on the number of trees compared and the number of leaves they may have. Good performance has been observed for sample trees of real data with fewer than 120 leaves. It may take more than 10 minutes to do the multiple tree comparison for 130-leaf trees, which is a little slow for the users. In practice, the target users typically deal with trees containing 50 to 100 leaves. Users may interact with the visualized phylogenetic trees and the created MASTs to perform manipulations such as modifying leaf names, swapping the subtrees of a specific internal node, or changing the font of leaf labels. This software was developed using Java [6] and runs on any platform that supports Java.

[1] We wish to acknowledge the many contributions of Dr. Greg Siragusa and Dr. Kelli Hiett in shaping the goals and requirements of this software.

1.1.2 The Application of Phylogenetic Tree Comparison

1.1.2.1 Why do we have multiple phylogenetic trees?

Constructing a phylogenetic tree is an estimation procedure [7]. Generally, we do not have direct information about the past and can only access contemporary data. Thus, the procedure of constructing a phylogenetic tree is an "estimate" of an evolutionary history, made based on incomplete information contained in contemporary data.

The phylogenetic tree construction methods that have been developed vary in many ways. Some methods choose the tree, from among all the trees possible, that either maximizes or minimizes some optimality criterion (criterion-based methods), while other methods merely follow an algorithm to produce a single tree (purely algorithmic methods). For the criterion-based methods, a model must exist that distinguishes what makes one tree better than another. Many tree optimality criteria have been proposed, such as the parsimony criterion [7], maximum likelihood criterion [7] and pairwise distance criterion [7]. *Parsimony* is a method for scoring a tree that assigns the highest score to a tree with the least number of mutations. In parsimony analysis, the tree that accounts for a sequence set's phylogeny with the fewest character state changes (the shortest overall tree length) is considered the best [8]. ML(Maximum Likelihood) methods use a statistical model to calculate the estimated number of actual changes. The tree with the highest likelihood (the conditional probability of the observed data given the tree) is the best hypothesis [7]. Pairwise distance methods calculate distances between pairs of organisms from the original data using the same sorts of models employed in maximum likelihood analysis. Some distance methods use Minimum Evolution as a criterion: the tree with the smallest sum of branch lengths that explains the data is selected as the best. Some researchers, such as Cavalli-

Sforza, A. W. Edwards and David M. Hillis view pairwise distance methods as less desirable approximations to a full maximum likelihood approach [9, 7].

The "perfect" phylogenetic tree is the one that reflects the true evolutionary relationship among species in the real world. Phylogenetic relationships inferred from a specific set of sequences should, in theory, be congruent if the sequences have the same overall history [7]. However, different tree construction methods based on different criteria with the same set of species may result in different trees. No method exists that is ideal for all performance criteria [10]. Furthermore, the optimality criteria mentioned above turn out to be NP-hard to optimize [11, 12]. In some cases, even a single phylogenetic tree construction method, such as the parsimony method, may produce hundreds of trees on a given set of data. Thus, phylogenetic tree comparison, which is able to provide similarity and dissimilarity information for the tree structures, has been investigated extensively in the field of bioinformatics.

1.2.1.2 Phylogenetic tree comparison applications

Phylogenetic tree comparison has multiple applications in bioinformatics. For instance, it can be used to investigate the stability of classifications. By comparing the result of a classification procedure with the result of the procedure on slightly perturbed data, the investigators may develop insight into the stability of the classification [13].

In addition, tree comparison can also be used to evaluate phylogenetic tree construction methods. When an accepted "accurate" tree exists for a given set of data, and a new method produces different results, comparison between the standard tree and the new result can determine how well the new method approximates the standard.

Another application of tree comparison is to calculate the distance between two rival trees. The result can be used to analyze the results from phylogenetic tree building algorithms and to compare conflicting evolutionary hypotheses. It can also allow a quantitative assessment of the similarity or dissimilarity of trees constructed from the same data sets.

Sometimes, biologists want to combine the information from rival phylogenetic trees into a new, hopefully more accurate, phylogenetic tree [13]. This new tree is helpful for summarizing those relationships common to all the alternative trees: through comparing a given set of trees, one is able to find the common hierarchy information and the relationships among the observed data.

1.2 Tree Comparison Metrics

A number of tree comparison metrics have been proposed. Examples include the partition metric [14, 15], the quartet metric [16, 2], the NNI metric [17], the consensus tree metric [18] and the maximum agreement subtree metric [19].

1.2.1 Partition Metric

1.2.1.1 Definition

In [20], the *partition metric* is defined as follows: When an edge of a tree is deleted, the taxa are partitioned into two sets. The *partition* is usually called a split or bipartition, and the partition metric between two trees, d_s, measures how many splits exist in one tree but not in the other. More formally, $d_s(T_1, T_2)$ is the size of the symmetric difference of the sets of splits induced by two trees T_1 and T_2. Equivalently,

$$d_s(T_1, T_2) = i(T_1) + i(T_2) - 2v_s(T_1, T_2),$$

where $i(T)$ denotes the number of edges of T that are internal and $v_s(T_1, T_2)$ denotes the number of pairs of identical splits of the leaf set induced by deleting an internal edge from each of T_1 and T_2.

In other words, the difference between two trees is defined as the number of edges for which there is no equivalent edge in the other tree [21].

1.2.1.2 Pros and Cons

The partition metric is easy to calculate, and Day described a linear time algorithm in [22]. Also, the range of this metric is well known; the maximum value of d_s across all pairs of trees with n leaves is $2n-6$ [20]. Further, the partition metric can be used for all labeled trees (binary or nonbinary, see section 2.2.2.1), and it has a known probability distribution [23]. For obtaining more reliable trees and measuring the reliability of the trees, the asymmetric distribution of the partition metric is also an advantage. Compared to the NNI metric, the partition metric is more general.

One problem with the partition metric is that it is not well regulated. That is, differences in the positioning of a small number of leaves may lead to a large difference in the value of the partition metric.

1.2.2 Quartet Metric

1.2.2.1 Definition

Any subset of four leaf labels is called a *quartet* [2]. A *subtree* of a tree is any tree formed by pruning one or more leaves from that tree (and removing any internal nodes that have

degree two as a result of pruning a leaf). If the subtree is not binary, it is called *unresolved*; otherwise, it is *resolved*. The two possible topologies for a quartet are shown in Figure 1:

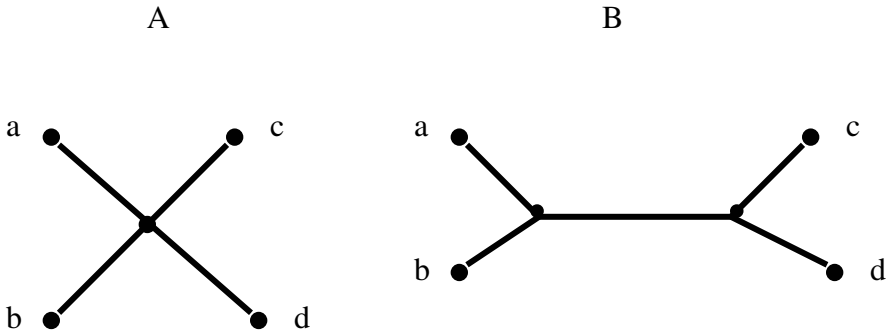


Figure 1 The two possible quartet topologies. Type *A* is unresolved and type *B* is resolved. Type *A* topologies are only found in nonbinary (unresolved) trees.

An unrooted tree with n leaves contains

$$Q = n\,(n-1)\,(n-2)\,(n-3) / 24 \qquad (1)$$

quartets, where *n* is the number of leaves. Each quartet (a, b, c, d) will be one of the four possible types: abcd, ab|cd, ac|bd, and ad|bc, as shown in Figure 2 below.

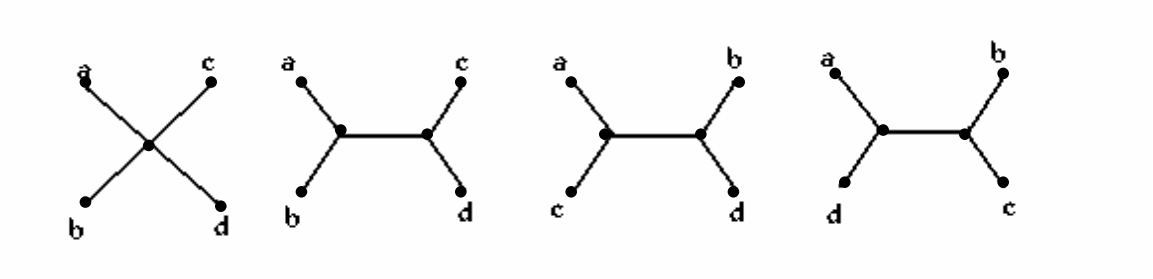


Figure 2 The Four Possible Quartets.

Each quartet of a phylogenetic tree is a subtree providing basic information about branching relationships among the quartet's organisms in the phylogenetic tree. It is well-known that the complete set of quartet topologies is unique for a given tree and the tree can be uniquely recovered from its set of quartet topologies in polynomial time [24]. Consequently, quartets can be used to measure resolution of, and to construct measures of, similarity and dissimilarity between such trees.

1.2.2.2 Pros and Cons

The quartet metric has many favorable features for a general tree comparison metric. It has a much larger range than the partition metric and so is more discriminating. Metrics that are based on the number of split differences are unstable with respect to the placement of a few leaves. The quartet metric is more well regulated, especially when *n* is large.

However, the calculation of the quartet metric is more time consuming than the partition metric. So far, the fastest algorithm for quartet metric has the time complexity of $O(n \log^2 n)$ [25]. But partition metric has its linear algorithm.

1.2.3 NNI metric

1.2.3.1 Definitions

Moore, Goodman & Barnabas in [26] introduced the concept of "nearest neighbor 1-step change" for the first time, which we refer to as "nearest neighbor interchange" (NNI). To give a careful definition of "nearest neighbor interchange", the *partition* of the binary tree is a useful concept. In [27], the author defined partition as follows: Removal of each interior edge *i* in the binary tree *T* yields a *partition* $\pi = \{A, B\}$ of the leaves of *T*. As shown in Figure 3, removal of

the end vertices (and their incident edges) corresponding to i yields a *separation* $v = \{\{A_1, A_2\}, \{B_1, B_2\}\}$ of the partition π, where $A_1 \cap A_2 = \emptyset$, $A_1 \cup A_2 = A$, $B_1 \cap B_2 = \emptyset$, $B_1 \cup B_2 = B$. Waterman and Smith (1978) proved that the set of *n-3* partitions of T, associated with the *n-3* internal edges, characterizes the binary tree T.

Let $v_0 = \{\{A_1, A_2\}, \{B_1, B_2\}\}$ be the separation associated with internal edge i of binary tree T. A *nearest neighbor interchange (NNI)* of T with respect to edge i produces other binary trees with separation $v_1 = \{\{A_1, B_1\},\{A_2, B_2\}\}$, or $v_2 = \{\{A_1, B_2\},\{A_2. B_1\}\}$ for edge i. All other edge separations agree with those of T. Figure 4 shows the two trees T_1 and T_2, resulting from a nearest neighbor interchange with respect to edge i of the tree T. T_1 has a separation v_1 and T_2 has a separation v_2 with respect to edge i.

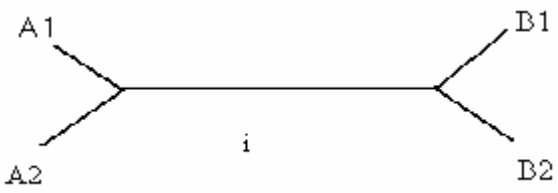
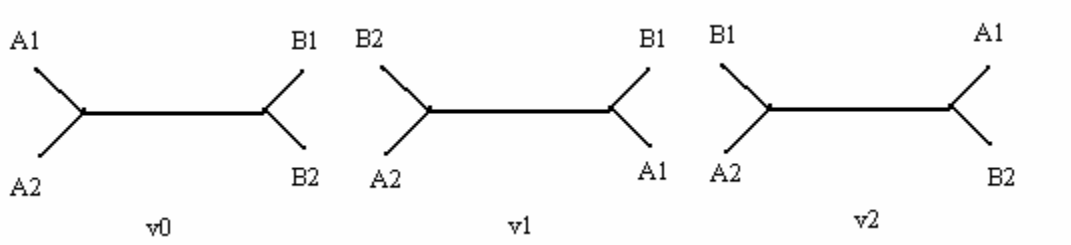


Figure 3 Separation of $v = \{\{A_1, A_2\}, \{B_1, B_2\}\}$ after Removing Internal Branch i.

Thus it can be seen that a nearest neighbor interchange from v_0 results in one of two possible trees: v_1 or v_2. There are two equivalent ways of viewing the process to create v_1 or v_2. One is to consider moving the branch associated with A_i ($i \in \{1,2\}$) onto the branch associated with B_1 or B_2. We may also generate the separation by interchanging the position of A_i and B_j ($i, j \in \{1,2\}$).



T T_1 T_2

Figure 4 The three nearest neighbor trees: *T* with separation v_0, T_1 with separation v_1 and T_2 with separation v_2.

The *Nearest Neighbor Interchange Metric* counts the minimum number of nearest neighbor interchanges required to change one tree to another. Here we assume the interchange weights are equal. Now, given *T* and *S,* two trees with the same set *{1, 2, …, n}* of terminal vertices, we define :

$\{T \rightarrow S\} = \{ \tau_k\tau_{k-1}\ldots \tau_1 : \tau_k(\tau_{k-1}(\ldots\tau_1(T)\ldots)) = S\}$,

τ_i: is the result in a nearest neighbor interchange , $1 \le i \le k$. Also it can be shown that $\{T \rightarrow S\} \neq \emptyset$ if $T \neq S$.

1.2.3.2 Pros and Cons

Maddison defined an *island* of trees as a set of trees "all less than a specified length, each tree connected to every other tree in the island through a series of trees, and each one differing from the next by a single rearrangement of branches." [28]. The nearest neighbor interchange is practical and useful for finding the islands of trees that are helpful in leading to a global optimal solution of tree construction.

The disadvantage of the NNI metric is its complexity of computation. In [29] M. Li, J. Tromp and L. Zhang reported that whether computing the nearest neighbor interchange metric for unlabeled or labeled binary trees is NP-complete or not is still an open question and that no

efficient exact algorithm has been found to solve the problem. Brown and Day [30] described an efficient approximation algorithm that produces an upper bound of d_{NNI}, but they did not give any approximation ratio analysis. M. Li, J. Tromp and L. Zhang [29] present a polynomial algorithm that approximates the NNI distance within a logarithmic factor, but the question of whether NNI distance can be approximated in polynomial time within a constant factor is still open [29].

1.2.4 Consensus Metric

1.2.4.1 Definitions

The consensus of two or more trees is a tree representing only that information that is shared by all of the trees [13]. Consensus methods are used to construct a tree that best represents common substructures in the compared trees and also identifies areas of conflict in the input trees. If the compared trees are very dissimilar, the consensus tree contains little information; if the compared trees are very similar, the consensus tree contains much information. The first consensus method was presented by Edward N. Adams, whose solution became well known as the *Adam consensus tree*. A variety of consensus methods for both rooted and unrooted tree groups have since been developed, such as the *strict consensus method*, *loose consensus method* and *cluster height methods* [31].

Strict consensus is the simplest of the all consensus methods [31]. Before introducing the basic idea of the strict consensus method, some preliminaries are given. In [31], a *group* is defined as a subset of the set of taxa. A group is *monophyletic* on a tree if and only if it contains all the descendents of its most recent common ancestor. The monophyletic groups of a tree *T* are called *cluster*s of T. If *A* is the group of taxa on one side of the branch and *B* is the group of taxa

on the other, then *A|B* is said to be the *split* corresponding to that branch. Given a collection of unrooted trees, the strict consensus tree contains exactly those splits common in the collection. With regard to the rooted tree group, the strict consensus tree contains the clusters common to the collection.

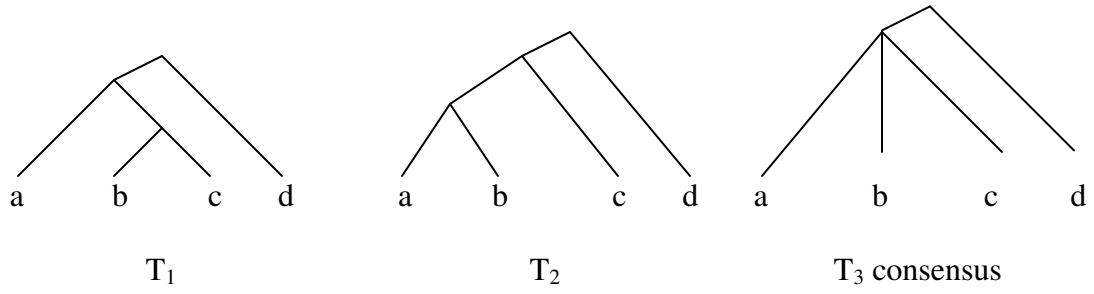
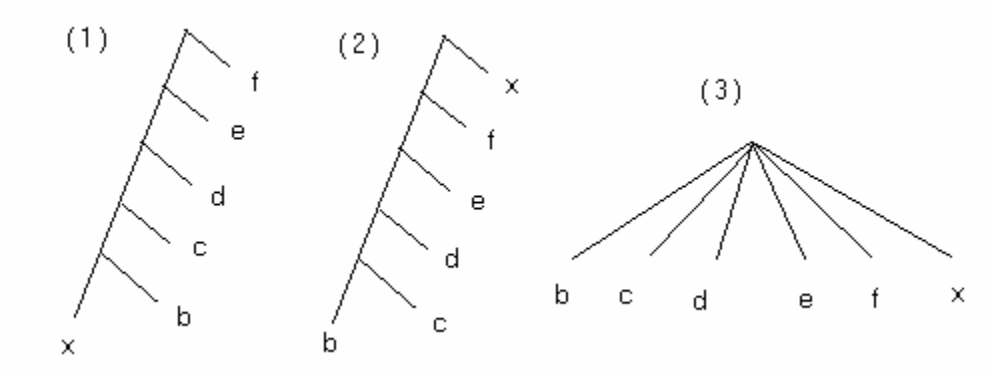


Figure 5 Two Rival Trees and Their Consensus.

Here we use a simple example quoted from [31] to explain the strict consensus tree. As in Figure 5, there are two trees T_1 and T_2, with topologies *((a, (b, c)), d)* and *(((a, b), c), d)*. T_1 has the cluster set *{{a}, {b}, {c}, {d}, {b, c}, {a, b, c}, {a, b, c, d}}* and T_2 has a cluster set *{{a}, {b}, {c}, {d}, {a, b}, {a, b, c}, {a, b, c, d}}*. Clusters *{a, b, c, d}* and *{a, b, c}* are common to the two trees, thus the strict consensus tree *is ((a, b, c), d*).

1.2.4.2 Pros and Cons

Consensus methods can be used to show which parts of the trees in a group are similar. In some cases, researchers try to ascertain the stability of a tree by comparing it with trees resulting from perturbed data. The consensus tree helps to provide adequate descriptions of the stability information in such cases. Consensus methods are a useful tool for phylogenetic inference when used in conjunction with a model or paradigm. However, sometimes the results from consensus methods are misleading. For example, consider the following trees (1) and (2) in figure 6:



T_1 T_2 T_3

Figure 6 T_1 and T_2 are the Compared Rooted Trees, T_3 is the Strict Consensus Tree of T_1 and T_2.

The rooted trees (1) and (2) only differ in the leaf *x*, but the strict consensus tree (3) is unable to provide that information. For such a purpose, the Maximum Agreement Subtree (MAST) method, which may also show the identical part of a set of trees, is more informative and is preferred by most researchers when such information is needed.

1.2.5 MAST metric

1.2.4.1 Definitions

MAST is a method that combines information from rival phylogenetic trees into a new agreement subtree. The MAST approach is useful for several reasons:

- It can provide increased confidence in the quality (optimality or near-optimality) of a specific phylogenetic tree. [1]
- It can summarize the relationships common to multiple alternative trees. [1]
- It can be used to investigate the stability of a tree topology. [1]

According to [32], MAST may be defined more accurately as follows:

"Suppose we are given two rooted binary trees T_1 and T_2 with n leaves each. The leaves in each tree are labeled with the same set of labels and further, no label occurs more than once in a particular tree. Let L_1 be a subset of the leaves of T_1 and L_2 be the subset of those leaves of T_2 that have the same labels as leaves in L_1. The subtree of T_1 *induced* by L_1 is an agreement subtree of T_1 and T_2 if and only if it is *isomorphic* to the subtree of T_2 *induced* by L_2. The MAST asks for the largest agreement subtree of T_1 and T_2. Intuitively, the subtree of T *induced* by a subset L of the leaves of T is the topological subtree of T restricted to the leaves in L, with branching information relevant to L preserved. More formally, for any two leaves a, b of a tree T, let $\text{lca}_T(a, b)$ denote their lowest common ancestor in T. If $a = b$, $\text{lca}_T(a, b) = a$. The subtree U of T *induced* by a subset L of the leaves is the tree with leaf set L and interior node set $\{\text{lca}_T(a, b) \mid a, b \in L\}$ inheriting the ancestor relation from T, that is, for all $a, b \in L$, $\text{lca}_U(a, b) = \text{lca}_T(a, b)$.

Intuitively, two trees are *isomorphic* if the children of each node in one of the trees can be reordered so that the leaf labels in each tree occur in the same order and the shapes of the two trees become identical. Formally, we say two trees U_1 and U_2 with the same leaf labels are isomorphic if there is a one – one mapping μ between their nodes, mapping leaves to leaves with the same labels and such that for any two different leaves a, b of U_1, $\mu(\text{lca}_{U1}(a, b)) = \text{lca}_{U2}(\mu(a), \mu(b))$."

UMAST is the unrooted version of MAST.

1.2.4.2 Pros and Cons

Compared to the consensus tree metric, the MAST will provide more precise information on the similarity of the trees. The resultant tree shows the common relationships among a set of

trees on the same given set of data. This approach is particularly useful when only a few taxa are responsible for the incongruence among trees, thereby providing a means of identifying "unstable" taxa.

One drawback of the MAST metric is the difficulty of identifying the maximum agreement subtree [33]. The speed of finding a MAST can be a problem. Even if finding them were easy, maximum agreement subtrees are not always preferable. Sometimes the maximum agreement subtree is clearly less useful than a smaller agreement subtree in showing the common tree structure.

The goal of the microbiologists in using this software is to compare the genotypic traits of different sets of bacterial isolates in order to determine either a common source ("source tracking") or a common phylogenetic lineage ("clonality") of individual representative bacterial isolates. To accomplish this, microbiologists use many different subtyping methods. Some methods are sequence-derived, while others are restriction site (banding pattern) based. The users would like the software to compare the trees obtained by these two different methods to determine how well the results of these approaches correspond. Many algorithms to implement the MAST metric have been proposed and we choose the algorithm introduced by Farach et al. in [34] to do multiple tree comparison and the algorithm presented by W. Goddard et al. in [35] to compare two trees. The time complexity of the algorithm in [35] is $O(n^2)$, and the tree group comparison algorithm in [34] is $O(kn^4+n^{d+1})$, where d is the degree bound and k is the number of trees compared. Here, for binary trees, d is 3 and the algorithm takes $O(kn^4+n^4)$. Both of these algorithms provide exact solutions, their speeds are acceptable, and the algorithms are relatively straightforward to implement.

CHAPTER 2

ALGORITHMS FOR MAST PROBLEM

2.1 Metric and Method

Briefly speaking, a phylogenetic tree comparison *metric* is the criterion used to measure how similar a set of trees is. A number of tree comparison metrics have been proposed to meet with different requirements of researchers. A specific metric may be implemented by using various *method*s, which may have different complexity or speed but the same results. For instance, with the emergence of the NNI metric introduced by Robinson [36], many NNI methods have been proposed, such as the Interior Vertex algorithm proposed by Waterman and Smith [17], an approximation algorithm to the NNI metric provided by Brown and Day [30], and the approximation algorithm of NNI distance within a logarithmic factor introduced by Li, Tromp and Zhang [29].

2.2 Algorithms

2.2.1 Previous Work on MAST Algorithms

During the past twenty years, many researchers have worked on the MAST problem. Finden *et al*. proposed a pruning algorithm in 1985 [1]. This is an approximation algorithm and its time complexity is $O(n^5)$. In 1992, Kubicka and Kubicki provided an exact algorithm with time complexity $O(n^{\log n})$ [19]. The SW (Steel and Warnow) algorithm is the first $O(n^2)$ algorithm for the MAST problem [37]. It is based on dynamic programming. Since 1993, several algorithms have been developed based on the SW algorithm. Farach and Thorup in their paper

[38] suggest an efficient method to simplify the SW algorithm. They derived an algorithm with time complexity $O(nc^{\sqrt{\log n}})$ for the binary unrooted tree, and $O(n^2c^{\sqrt{\log n}})$ for the unbounded unrooted case, where *c* denotes a sufficiently long constant. In 1995, Farach and Thorup developed an algorithm that improved the time complexity of the previous polynomial algorithms. They gave an $O(n \log^3 n)$ time algorithm for the MAST for the binary trees. The fastest algorithm thus far is that provided by Cole *et al*. in [32]. That algorithm is based on Farach and Thorup's work and makes some improvements, presenting an O(n log n) algorithm.

2.2.2 Algorithm for Two Tree Comparison

The two tree comparison implemented in this thesis work is based on the MAST algorithm presented by Goddard et al. presented in 1993, which takes $O(n^2)$ time [35]. Although it is not the fastest algorithm, its speed is acceptable, it produces an exact answer, and its implementation is straightforward. In their paper, the authors provide an exact polynomial-time algorithm for comparing two trees, based on dynamic programming, for rooted trees, and then generalize the algorithm to unrooted trees. In addition, the authors in [35] presented another algorithm to compute the tree distance - the sum of distances between the pruned leaves with respect to the MAST. The tree distance provides accurate and detailed information on how dissimilar the two trees are. Furthermore, the algorithm introduced in that paper allows the compared trees to have different numbers of leaves.

2.2.2.1 Preliminaries

graph A *graph*(or *simple graph*, for emphasis) *G* is an ordered pair of disjoint sets (*V, E*) such that $V \neq \emptyset$ and *E* is a subset of the set V^2 of unordered pairs from *V*. We consider only finite

graphs, so V and E are always finite. Here V is the set of *vertices* and E is the set of *edges*. If G is a graph, then $V = V(G)$ denotes the vertex set of G, and $E = E(G)$ denotes the edge set. The *order* of G is $|V|$. The *size* of G is $|E|$. If E is a multiset, that is, if edges are allowed to occur more than once, then G is a *multigraph*. If the edges are ordered pairs of vertices, then the structure is a *directed graph*, or *digraph*. [39]

tree A graph containing no cycle is *acyclic*. A *tree* is an *acyclic* connected graph. [39]

rooted tree A *rooted tree* is a tree with a special vertex, called the *root*. [39]

binary tree A *binary tree* is the tree whose leaves are labeled without repetition and all internal nodes are unlabeled with degree 3. [35]

size of a binary tree The *size* of a binary tree is the number of leaves that the tree has. [35]

prune The operation of *pruning* on a tree T is the removal of a subset of leaves from T and suppression of all inner vertices of degree 2 which are formed by this deletion. [35]

agreement subtree A *greatest common pruned tree* of two trees T and U, which we simply call an *agreement subtree*, is a tree that can be obtained from T as well as from U by pruning the fewest number of leaves from the two trees. [35]

A(T, U) *A(T, U)* is the set of all agreement subtrees for two binary trees T and U. [35]

#(T, U) *#(T, U)* is the *size* of *A(T, U)*. [35]

T_a T_a is the subtree of tree T rooted at a vertex a. [35]

MAST MAST is the maximum agreement subtree problem. [40]

UMAST *UMAST* is the unrooted maximum agreement subtree Problem. [41]

2.2.2.2 Goddard *et al.*'s Algorithm and Pseudo Code

The following lemma is the basis of this algorithm:

Lemma 1

Let T_a be a tree rooted at a vertex a with children b and c, and let U_w be a tree rooted at w with children x and y. Then $\#(T_a, U_w)$ is the maximum of the following six numbers [35]:

- (T_b, U_x) and (T_c, U_y)
- (T_b, U_y) and (T_c, U_x)
- (T_a, U_x)
- (T_a, U_y)
- (T_b, U_w)
- (T_c, U_w).

First we consider only rooted trees. According to the Lemma 1, what can be solved by the dynamic programming method. Suppose *mast* is the maximum agreement subtree between one branch of T and one branch of U. In order to get the MAST for rooted trees T and U, the *mast*s between all branches of tree T and all branches of tree U must first be computed. Suppose tree T has size m and tree U has size n. A table of size $(2m-1)*(2n-1)$ is built to store all of the intermediate *mast*s. Each element C_{ij} $(1 \leq i \leq m, 1 \leq j \leq n)$ in the table represents *mast*s between a specific subtree of T and a specific subtree of U. Thus, suppose the elements in the i^{th} row of the table represent the *mast*s between T_b and all of the subtrees of tree U, and the elements in the j^{th} column represent the *mast*s of U_y and all of the subtrees of tree T. Then C_{ij}, the i,j^{th} element in the table, represents the *mast*s of T_b and U_y. The table is filled from left to right and from top to bottom according to a post-fix order traversal of trees T and U. An initial condition is set as follows: when we try to get *mast*s of T_a and U_w, assume a is a leaf, if U_w also contains a leaf

labeled a, then #(T_a, U_w) = a, otherwise #(T_a, U_w) = Ø; if U_w is a leaf, vice versa. If neither T_a nor U_w is a leaf, search the previously computed *mast*s in the table and compute the specific *mast* according to Lemma 1. Thus, each element in the table can be computed through dynamic programming. Finally we get MAST for trees T and U, which is the $(2m-2)*(2n-2)^{th}$ element in the table. This algorithm not only retains the MAST tree structures but also keeps the similarity information, the size of the MAST trees, of the compared trees.

Here is the pseudo code for computing the MAST of rooted trees:

```
PROCEDURE MAST(Tree T, Tree U)
(input: T and U are two compared rooted trees)
1       a = the left most node of tree T
2       w = the left most node of tree U
3       posT = 0;
4       WHILE  a is not null
5           posU = 0
6           WHILE w is not null
7               IF a or w is leaf
8                   IF Ta's leaf set intersect with Uw's leaf set
9                       TAB(posT, posU) = common leaf of Ta and Uw.
10                  ELSE
11                      TAB(posT, posU) = null
12              ELSE
13                  TAB(posT, posU) = compute maximum agreement
```

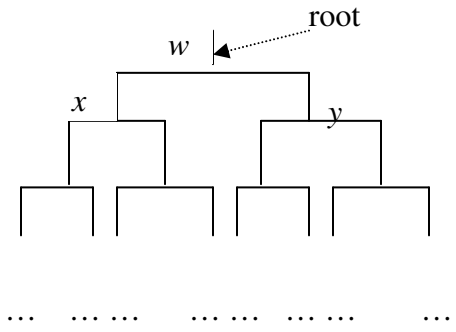
```
14                                  subtrees according to Lemma 1
15                  posU ++
16                  w = next node of w to be processed in tree U according to post-fix
                        order
17              END OF WHILE
18              posT++
19              a = next node of a to be processed in tree T according to post-fix order
20          END OF WHILE
21          RETURN TAB
```

Figure 7. Pseudo Code for MAST Procedure of Rooted Trees *T* and *U*

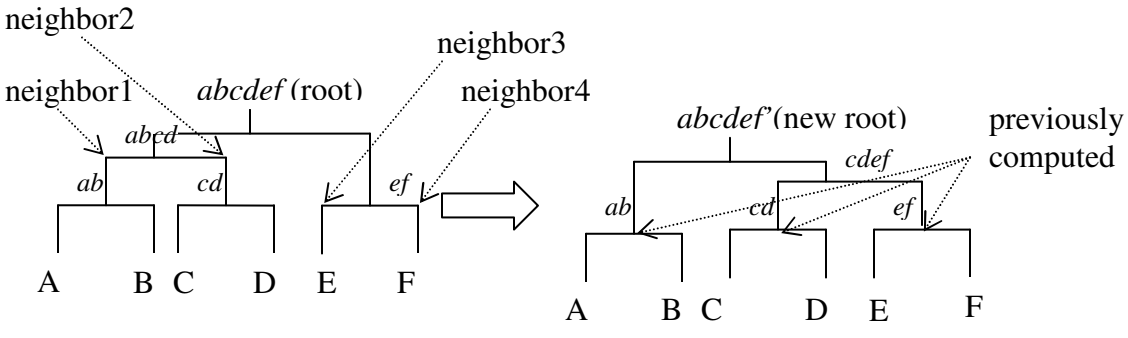
The rooted tree comparison procedure may be viewed as a sub-procedure of the *UMAST* – the unrooted tree comparison version. In order to compare two unrooted trees, fix the root for the tree *T* as *T'* and try all the possible locations of root for the second tree *U*. The comparison between any possible rooted trees *T'* and *U'* uses the rooted tree comparison algorithm mentioned above. The final solution of *T* and *U* is the collection of all *T'* and *U'* pairs: $UMAST = max(\Sigma\, MAST(T', U'))$. Suppose we start from a randomly selected rooted version *T'* and *U'*, after each comparison between *T'* and *U',* move the root of *U'* to its neighbor until all the possible roots of *U* have been tried. During that procedure, re-constructing the whole table after each re-root is not necessary. The table of the previous comparison can be used and only two more rows of the new table have to be computed.

Figure 8 illustrates how to reduce the computation time of the root moving procedure. Assume each internal node is labeled as the concatenation of the gene names of its descendant

leaves. Assume two unrooted trees *T* and *U* are being compared and both *T'* and *T"* are the rooted versions of *T*. *T"* is created by moving the root of *T'* to the *neighbor1*. Suppose MAST(*T'*, *U*) is already computed. In order to get MAST(*T"*, *U*), six elements have to be computed first: $mast(U_x, T"_{ab}) + mast(U_y, T"_{cdef})$, $mast(U_y, T"_{ab}) + mast(U_x, T"_{ab})$, $mast(U, T"_{ab})$, $mast(U, T"_{cdef})$, $mast(U_x, T")$ and $mast(U_y, T")$. All the elements are available from the previous table except $mast(U, T"_{cdef})$, which can also be computed directly from the existing data. Thus in order to get MAST(*T"*, *U*), only $mast(U, T"_{cdef})$ and MAST(*U*, *T"*) have to be computed. The other three neighbors may also computed in this way. Making full use the previously computed data improves the speed.



tree topology of *U*



Original tree topology *T'* New tree topology *T"*

Figure 8 Re-root Phylogenetic Tree *T*.

Pseudo code to find Unrooted Maximum Agreement Tree(UMAST)

```
PROCEDURE UMAST(Tree T, Tree U)
(input: T and U are two compared unrooted trees)
1       T' = a randomly selected rooted version for T
2       U' = a randomly selected rooted version for U
3       umast_set = ø
4       FOR each possible T'
5           MAST(T', U')
6           put the resultant unrepeated tree set into umast_set
7           move root of T' to its left and right neighbors to form new T'
8       END FOR
9       RETURN umast_set
```

Figure 9 Pseudo Code for UMAST for Unrooted Trees *T* and *U*

For rooted trees *T* and *U*, a *(2m-1)*(2n-1)* table has to be constructed, where *m* is the size of the leaf set of *T* and *n* is the size of the leaf set of *U*. The time complexity is *O(mn)*. Generalizing the rooted version to the unrooted version requires moving the root of one tree *O(m)* times. The computation takes *O(m)* for new rooted *T* and *U*. Thus, the total time complexity is *O(mn)*. If tree *T* and *U* have the same number of leaves, it is $O(n^2)$.

2.2.2.3 Algorithm for tree distance

In some tree comparison algorithms, for the two unrooted trees *T* and *U* with the same leaf set, the tree distance can be $d(T, U) = n - \#(T, U)$, where *n* is the size of *T* and *U*. According to that definition, the tree distance is the number of pruned leaves. However, this metric does not take into account the distance of pruned leaves and therefore can possibly obscure information about the intuitive "closeness" of two trees [34]. Thus it is unable to describe the detailed dissimilarity information. Here is an example:

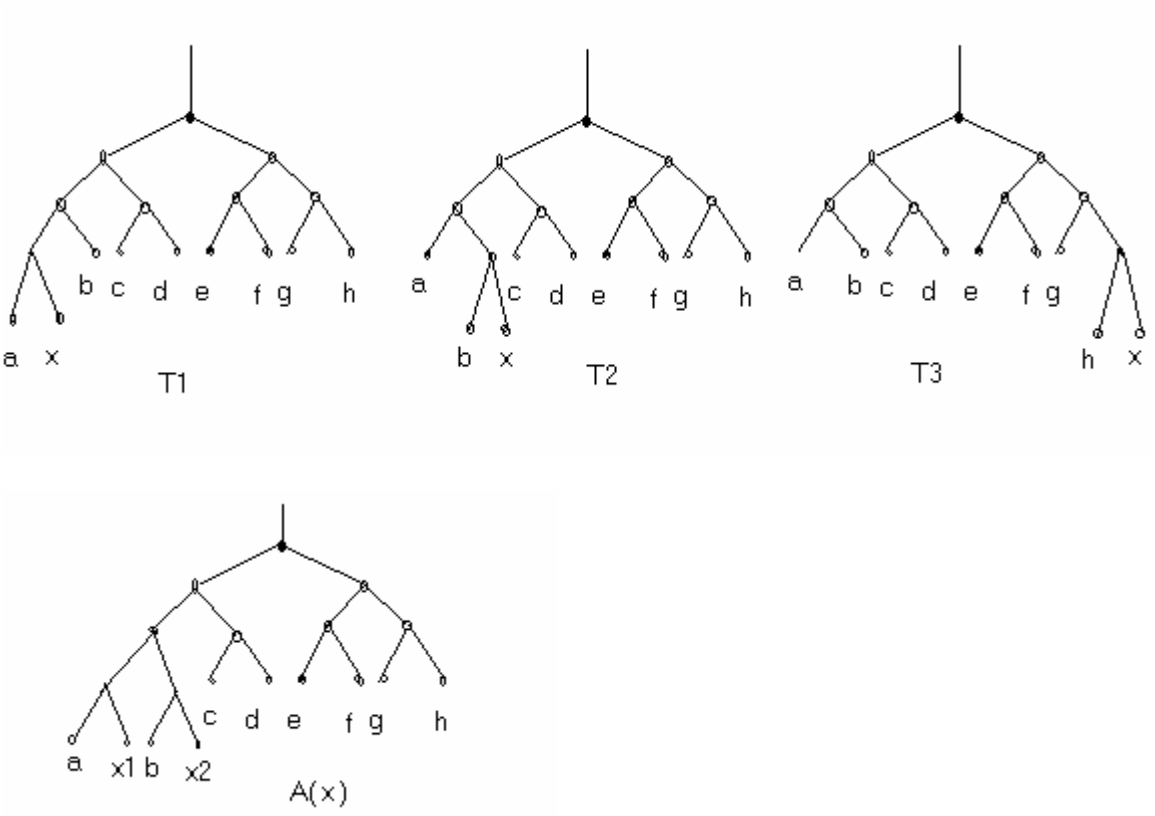


Figure 10 Tree T_1, T_2, T_3 and MAST

T_1, T_2 and T_3 have the same leaf set and the $MAST(T_1, T_2) = MAST(T_2, T_3) = MAST(T_1, T_3) = \{a, b, c, d, e, f, g, h\}$ (Leaf *x* is pruned). And the distance $d_1(T_1, T_2) = d_1(T_1, T_3) = d1(T_2, T_3) = 1$. Obviously, T_1 and T_2 look to be closer than T_1 and T_3 or than T_2 and T_3. But

this tree distance is unable to provide that information. Thus in the paper, the author modifies the tree distance by adding a fractional part, which reflects how far apart, with respect to a maximum agreement subtree, the pruned leaves are.

Suppose we have tree *T* and *U* with size *n*, *A* is a maximum agreement subtree of *T* and *U*. Let *S* denote the set of leaves pruned from *T* and *U* to form *A*. For each element *x* in *S*, build tree *A(x)* such that *A(x)* is formed from *A* pruning leaf *x*. Insert leaf x_t and x_u into *A(x)*, so that $A_t(x)$, which is *A(x)* without leaf x_u is isomorphic to *T*, and $A_u(x)$, which is *A(x)* without leaf x_t is isomorphic to *U*, are created. Compute the distance *l* between x_t and x_u. Get the sum *L* of all the *l*s for each leaf in *S*. Thus, the fractional part is the minimum of *L* for all the agreement trees of *T* and *U*. The new distance metric can be described as:

$$d(T,U) = d_1(T,U) + \frac{1}{nd_1(T,U)} L(T,U)$$

For trees with different leaf sets, leaves pruned on each tree are different. $d_1(T, U)$ should be the size of the union of the unrepeated pruned leaves from two trees. And the pruned leaves that do not belong to the common leaf set of the two trees are not considered in the computation of *L*. According to the modified algorithm in [35], the time complexity of computing tree distance is also polynomial.

Pseudo code:

PROCEDURE TREEDIS(tree T, tree U, tree []umaset)

(input: T and U are two compared unrooted trees, umast is the set of unrooted maximum agreement subtrees between T and U)

1 d = size of a umast tree

2 S = leaf set of T

```
3       MINL = d + 1
4       FOR each tree A in the umast set
5           L = 0
6           FOR each leaf x in the set of S
7                   prune x from A
8                   insert xt and xu into A
9                   get path length l from xt to xu
10                  L = L + l
11          END OF FOR
12          IF MINL > L
13                  MINL = L
14      END OF FOR
15      d = d + 1/(n*d)*MINL
16      RETURN d
```

Figure 11 Pseudo Code for Tree Distance

2.2.3 Algorithm for multi - tree comparison

The algorithm in [35] provided a fast and optimal solution for the MAST problem of two tree comparison; however, its application on multiple tree comparison is not as efficient as that on two trees. In order to get the common tree structure for all the trees compared, almost all the intermediate results have to be retained until the MAST size is computed. For a tree group with more than five trees and each of which has the size of fifty leaves, which is a typical tree size in biology, the computation takes too long if the brute-force algorithm is used. Even for a branch-

and-bound algorithm, which is faster, speed is still a big problem. Thus for multiple tree comparison, we choose the MAST method from [34].

2.2.3.1 Preliminaries

Farach's algorithm is also a dynamic programming procedure. Before describing the algorithm, some definitions must be introduced. All of the terminology is defined in [34].

> **"topological restriction of *T* to *B*** Given a leaf labeled tree T on set A, and given B $\subseteq$ A, then the topological restriction of *T* to *B*, written *T*|*B*, is the tree with vertex set $\{lca^T(a, b) | (a, b) \in B^2\}$, where the arcs are defined such that for all $(a, b) \in B^2$, $lca^{T|B}(a, b) = lca^T(a, b)$
>
> **lca^T- tuples** - Suppose we fix a tuple $T = (T_1, \ldots, T_k)$ of leaf labeled trees over a set *A* on size *n*, and let *d* be the degree bound of T_1. An *agreement set* is any set $B \subseteq A$, such that $T_1|B, \ldots, T_k|B$ are isomorphic. So MAST is the problem of finding a maximum cardinality agreement set. For any pair $(a, b) \in A^2$, let $lca^T(a, b) = (lca^{T1}(a, b), \ldots, lca^{Tk}(a,b))$. We will refer to the *k*-tuples generated in this way as lca^T- *tuples*.
>
> **agreement tree** – For any agreement set *B* of *T*, the agreement tree *T*|*B*, is the tree with vertex set $\{lca^T(a, b) | (a, b) \in B^2\}$, where the arcs are defined such that for all $(a, b) \in B^2$, $lca^T(a, b)$ is the least common ancestor of $lca^T(a, a)$ and $lca^T(b, b)$. The size of *T*|*B* is the cardinality of *B*.
>
> **mast (v)** – *mast (v)* denotes the maximum size of an agreement tree with root *v*, here *v* is an lca^T- tuple.

dominate – Suppose v and w are lca^T- tuples, let v *dominate* w, written $v \succ w$, if $v[i]$ is a strict ancestor of $w[i]$, for all $i \leq k$.

direction – Let domination $v \succ w$ have *direction* $\delta = (\delta_1, \ldots, \delta_k)$ if for $i \leq k$, $w[i]$ descends from, or is equal to, the δ_ith child of v. In this case, denote δ by $(v \ni w)$, and call δ an *active direction* from v.

compatible – Two directions δ_1, δ_2 are *compatible*, denoted $\delta_1 \perp \delta_2$, if they differ in all coordinates.

properly dominate – Let v properly dominate w, denoted $v > w$, if $v \succ w$ and there exists a $u \prec v$ such that $(v \supset u) \perp (v \supset w)$. In this case, call $(v \supset w)$ a proper direction from v, and denote the set of such *proper directions* from v by $D(v)$.

compatibility graph of v: $G(v) = (D(v), E(v))$, where $E(v) = \{\{\sigma_1, \sigma_2\} | \sigma_1 \perp \sigma_2\}$.

Lemma 2.6 For each internal *lcat* - tuple v, let $G(v) = (D(v), E(v))$ be its compatibility graph. For each $\sigma \in D(v)$, let $M[v, \sigma] := max\{mast(w) \mid v > w \text{ and } (v \supset w) = \sigma\}$ be the weight of σ in $G(v)$.

Then $mast(v) = \begin{cases} l & \text{If } v \text{ is a leaf,} \\ MWC(G(v)) & \text{otherwise.} \end{cases}$

Where MWC(G(v)) is the weight of a *maximum weight clique* in $G(v)$. ”

2.3.4.2 Algorithm

This algorithm is also based on a dynamic programming method. It is applied to the comparison of multiple d-degree trees. Since a mast set uniquely defines an agreement subtree given a set of trees, this algorithm aims to finding the maximum mast set of the tree group instead of the mast tree topology. With regard to the rooted trees, according to Lemma 2.6 in the

paper, the problem is reduced to find the clique with a maximum weight in *G(v)* for the least common vertex vector *v*, where *v* is the root for all the trees. Assume there is a tree group *T* and its leaf set *S*. The first step for this algorithm is to list all the lca^T- tuples of *T*, and then find the dominant relationship among all those produced vectors. For each lca^T- tuple *v*, classify its children according to their proper directions with *v*. For each *v*, compute the weight of each edge and then each clique in *G(v)*, and select the ones with the greatest size, which is the weight of the maximum weight clique of the compatibility graph of *G(v)*. Finally, the *mast(v)* is the size of the maximum agreement subtree of *T* where *v* is the root of the trees, and at the same time, the corresponding mast sets are also retained. This is a dynamic programming procedure and I implemented it by a recursive method. The algorithm can also be generalized to the unrooted version by enumerating each leaf in the leaf set as a root for the tree group and choosing the agreement subtrees with the maximum size.

Peudo code for M. Farach, T. M. Przytycka and M. Thorup's algorithm:

1. Compute all the *lcat(a, b)*, where $a, b \in S$.
2. For all labels *a*, set *mast(lcat(a, a))* = 1.
3. For all internal *lcat*-tuples *v* in topological sort order in $\prec$ do:

a. For all $\delta \in D(v)$:

i. $M[v, \delta] := \max\{\text{mast}(w) | w \in W(v, \delta)\}$.

b. Set $\text{mast}(v) = \text{MWC}(G(v)) = \max_{c \in C(v)}\{\Sigma\, M[v, \delta_i] \mid c = (\delta_1 \ldots, \delta_k)\}$.

4. Return $\max_v\{\text{mast}(v)\}$.

Figure 12 Peudo Code for M. Farach, T. M. Przytycka and M. Thorup's Algorithm.

As the authors mentioned, the steps from 1 to 3.a are computed in time $O(n^3)$, and the step 3.b takes $O(n^d)$. Since my software deals only with binary trees, the time complexity for the whole procedure is $O(n^3)$ for the rooted trees, and for unrooted tree group, it takes $O(n^4)$.

CHAPTER 3

DESIGN AND IMPLEMENTATION

3.1 Requirements Analysis

The first stage of system design is to define the requirements of users. This software is developed to meet the requirements of the research group of microbiologists, Dr. Greg Siragusa and Dr. Kelli Hiett, in the Russell Research Center, the University of Georgia. After I met with them over several times, I obtained and refined their requirements. This software should have the following features:

1. The tool should be able to read and visualize a phylogenetic tree from a tree file specified with NEXUS format.
2. The tool should be able to make a tree comparison between two or among multiple trees. The compared trees may have different leaf set.
3. The compared trees may be loaded from multiple data sets, or a single NEXUS file which contains multiple tree topologies.
4. The tool should produce and visualize Maximum Agreement Subtrees (MASTs) to show the common tree structure of the compared trees. The tool should also calculate and display the normalized similarity index and the tree distance.
5. The visualized MASTs should be mapped to the compared trees and highlighted. The users may select any two trees in the tree group to match them and show the common structure between them.

6. Users should be able to interact with both the loaded phylogenetic trees and the created MAST by swapping the children of a specific internal node, changing the fonts, sizes, and colors of the leaf labels, or changing the header names.

7. Any tree structure on the main tab may be saved as an object or printed. The object file may also be loaded, saved, printed and interacted by users.

3.2 Usage and Features

3.2.1 Technological Infrastructure

Java [6] is a powerful, general purpose object-oriented programming language that provides packages useful in the creating of graphical displays and user interfaces. JFC, the Java Foundation Classes, encompass a group of features to help people build graphical user interfaces. Its Swing package provides many standard GUI components that can be combined to create a program's GUI. For these reasons, as well as Java's ability to execute on multiple platforms, TreeAnalyzer is implemented in Java.

Many phylogenetic applications provide tree comparison functions but only few of them produce graphical displays that the user may interact with. Such visual displays are quite useful for understanding and exploring Maximum Agreement Trees. A primary strength of this software is that it not only creates the visualized Maximum Agreement Tree but also permits interaction between the users and the resultant trees. This software runs on any platform supporting the Java language (J2SDK 1.4.2 or higher version must be installed on the machine).

3.2.2 Input Format

The input file format for TreeAnalyzer is the NEXUS format [5], which is used by a number of popular phylogeny program such as PAUP [42], Phylip [43], and MacClade. The NEXUS format file consists of the "#NEXUS" directive at the beginning of the file, the translation table and the tree blocks. The translation table starts with the label "TRANSLATE". It is used to map the sequence names in the real world into the internal leaf labels. Figure 13 depicts a sample NEXUS file. The numbers on the left side "1", "2", …, "7" are the internal leaf labels, and the strings of "K@riboprint1@00000002", "K@riboprint1@00000001", ..., "K@riboprint1@00000009" are the real world sequence names. To simplify the representation of the tree structure, the internal leaf labels, instead of sequence names, are used to describe the tree topology. Trees are described using a standard parenthetical notation. Each cluster in the tree is enclosed by a pair of parentheses ("()").In such a cluster (1:3.55, 2:3.55), the string to the left of the colon is a node's leaf label and the floating point number to the right of the colon is the branch length between that node and its ancestor. In the example shown, each line that starts with "UTREE PAUP" is a tree topology. A detailed definition of the NEXUS format is provided in [5].

```
#NEXUS
BEGIN TREES;
        TRANSLATE
                1       K@riboprint1@00000002,
                2       K@riboprint1@00000001,
                3       K@riboprint1@00000005,
                4       K@riboprint1@00000007,
                5       K@riboprint1@00000006,
                6       K@riboprint1@00000003,
```

```
                7       K@riboprint1@00000009,
UTREE PAUP_1=((((1:3.55,2:3.55):2.89,3:6.44):6.03,(4:5.73,5:5.73):6.73):1.95,(6:0.35,7:0.35):3.7);
UTREE PAUP_2=((((7:1.3,3:1.3):2.4,(5:0.9,4:0.9):1.5):1.3,2:2.3):3.5,(1:0.8,6:0.8):1.4);
        ....
UTREE PAUP_n=(7:0.9,((3:1.6,(5:3.4,4:3.4):1.5):1.4,(2:0.9,(1:0.4,6:0.4):1.7):1.7):1.6);
ENDBLOCK;
```

Figure 13 Nexus Format Sample

3.2.3 Functions

Through the TreeAnalyzer user interface (Figure 14) users may perform file operations (open, close, save and print) and phylogenetic tree operations (subtree swap, label change, MAST computation, label head change). To view a phylogenetic tree or perform operations, the user must first read the binary tree topologies and leaf sets from files. The tree structures labeled with branch lengths are then displayed as seen on the left side of the panel. Information about the corresponding leaf names, ribotypes and comments are provided as seen on the right side of the panel. An image available for riboprints can also be loaded beside the corresponding leaf, seen in the center panel. Each tree's display is accessible on a separate tabbed pane.

Users may interact with the visualized trees to swap subtrees, change leaf names, modify header names and change the color and fonts of leaf names. To do so, the user first chooses a mode by clicking on an icon in the toolbar. The icons, from left to right, support rotation at nodes, editing of labels, editing of headers, creation of a MAST, and printing. Created or modified trees can be saved as an object. The print function permits the trees to be printed out or sent to image files, such as tiff and bmp format files, which may then be included in papers, posters, slides, etc.

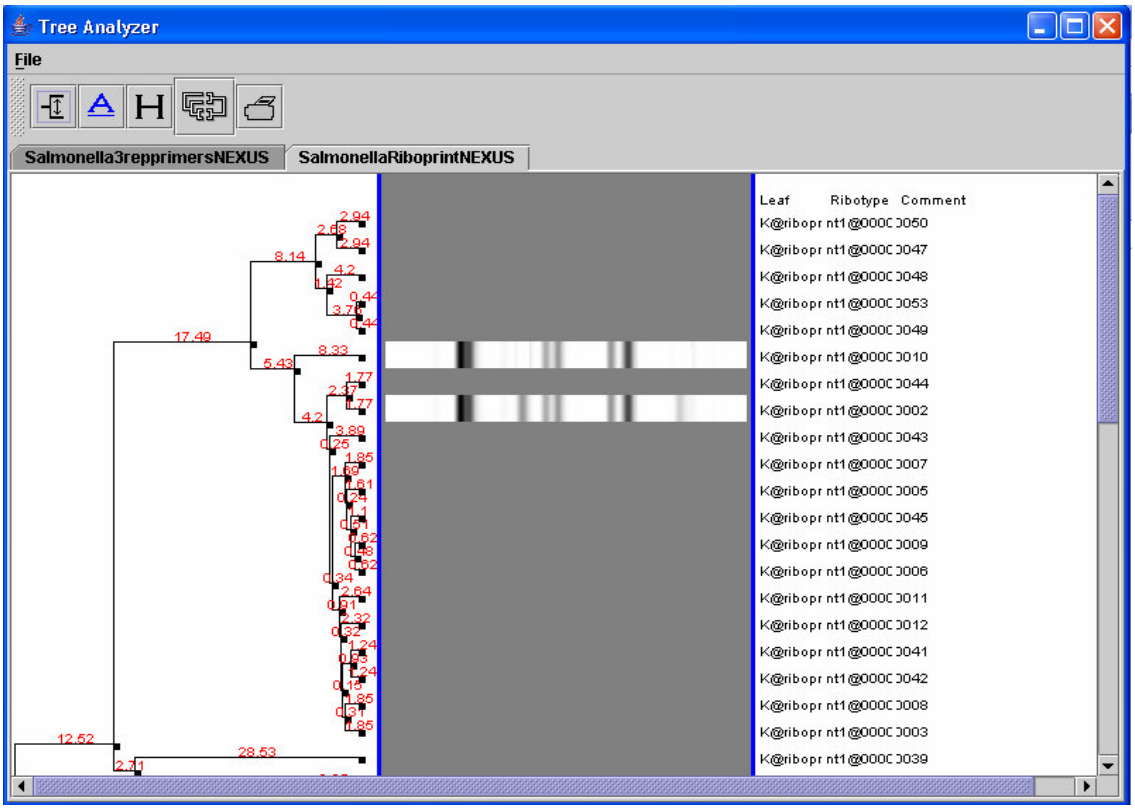


Figure 14 Data Input and Display Tree. The tree name is displayed as the panel name. The tree topology with branch lengths is displayed on the left side; the corresponding sequence name, type and comments are shown on the right side; the images of some sequences are displayed in the center.

As seen in figure 15, MASTs are visualized as highlighted areas on a pair of trees selected from the compared tree group, using both color and line thickness. Users may select other tree pairs for visualization. Corresponding leaves are connected by red lines; a "matching" procedure is applied to minimize line crossings. The tree distance and the *similarity index* are computed and displayed. The similarity index is a metric *x/y*, where *x* is the proportion of leaves that remain in the MAST from the original leaf set and *y* is the proportion of leaves pruned to produce the MAST. The MAST shows a direct picture of which parts of the compared trees are common, and the similarity index provides a quantitative measure of how similar or dissimilar the compared trees are.

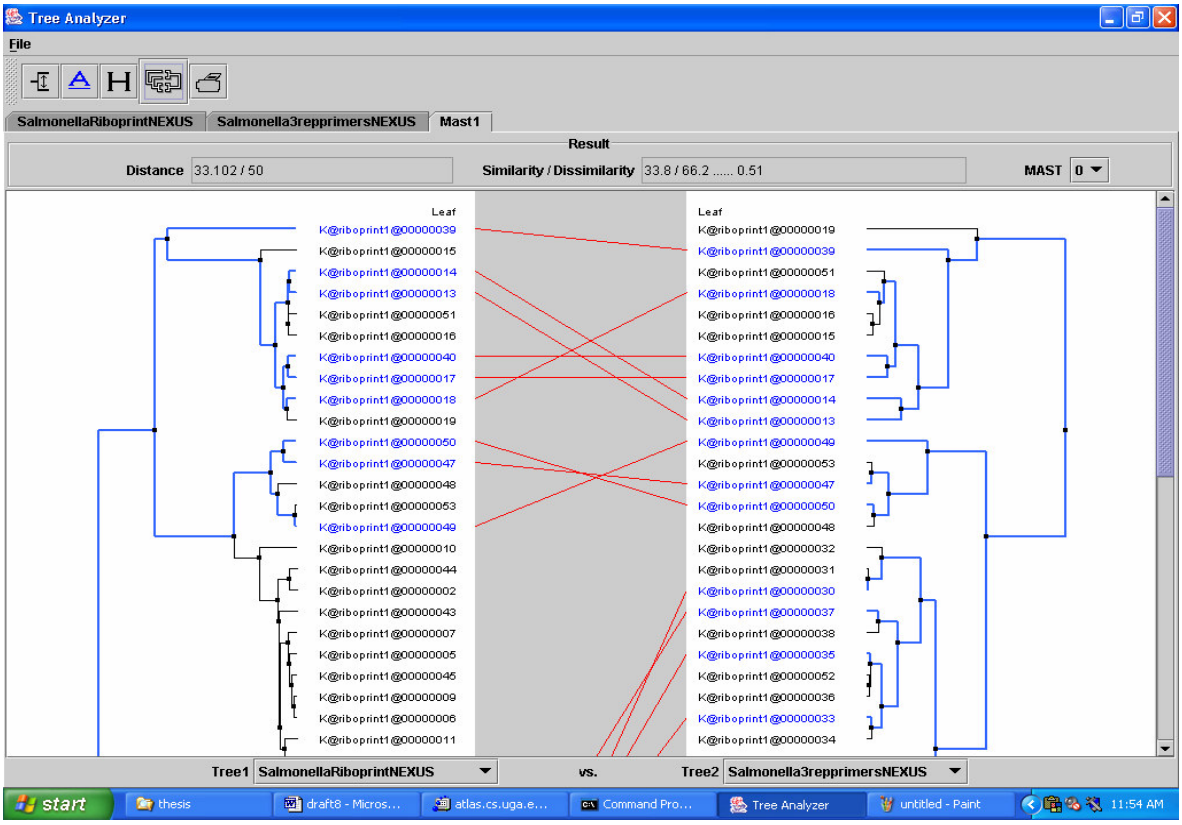


Figure 15 Visualized MASTs. Two trees from the compared group are selected to display the results; users may select other tree pairs. If multiple MASTs exist, the user may select other MASTs. Leaves and branches that belong to the selected MAST are highlighted. The leaves are also connected by red lines to show the common parts of the two trees. The tree distance (the number of leaves pruned to get MASTs) and the similarity index are computed as well.

The users may perform the same operation on MASTs as on the original trees: swapping the subtrees of a specific node or changing the fonts and color of leaves. MASTs may be saved or printed. MASTs are displayed one at a time. Trees may be loaded from multiple NEXUS files or a tree group may be loaded "en masse" from a single NEXUS file. The software supports multiple tree comparison among unrooted binary trees in which differences exist in the leaf sets. We use the algorithm described in [35] for two tree comparison and that introduced in [34] for multiple tree comparison.

3.3 Performance Numbers on Trees of Various Sizes

We used LumberJack [44], a phylogenetic inference tool, and sequence data files provided with the package to create groups of phylogenetic trees of various sizes. Each tree group is based on the same dataset and contains at least three trees. The results from two-tree groups are shown in table 1 and figure 16, and the results from three-tree groups are shown in

Table 1 – Average Running Time for MASTs on Two-Tree Groups of Indicated Size

Tree Size	10	30	48	70	90	100
Avg. time (sec)	0.104	0.411	0.761	1.573	2.281	2.877

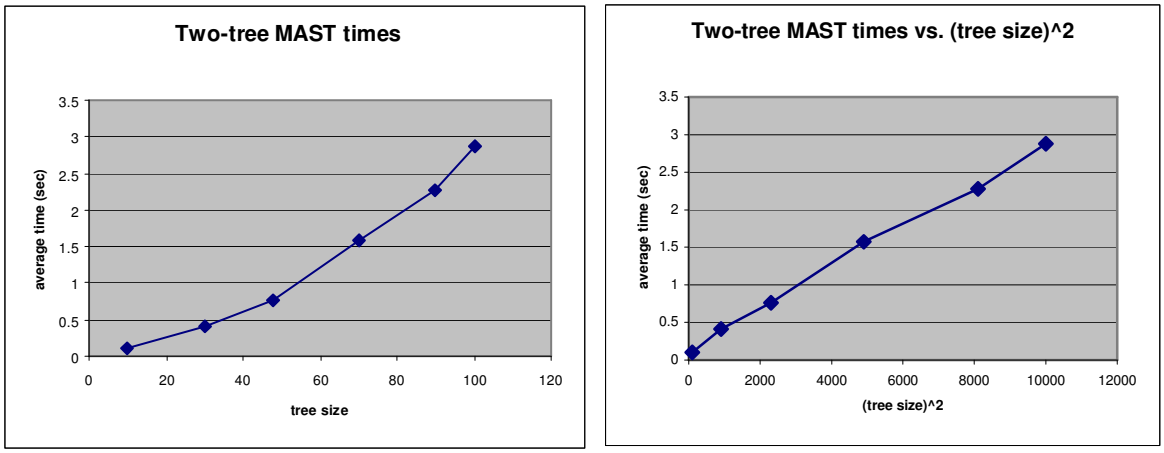


Figure 16 Average Running Time for MASTs of Two-Tree Groups, Plotted versus Tree Size, Indicating $O(n^2)$ Behavior.

Table 2 – Average Running Time for MASTs on Three-Tree Groups of Indicated Size

Tree Size	30	50	70	80	90	100	110
Avg. time (sec)	2.73	15.42	47.02	77.56	120.99	187.10	260.13

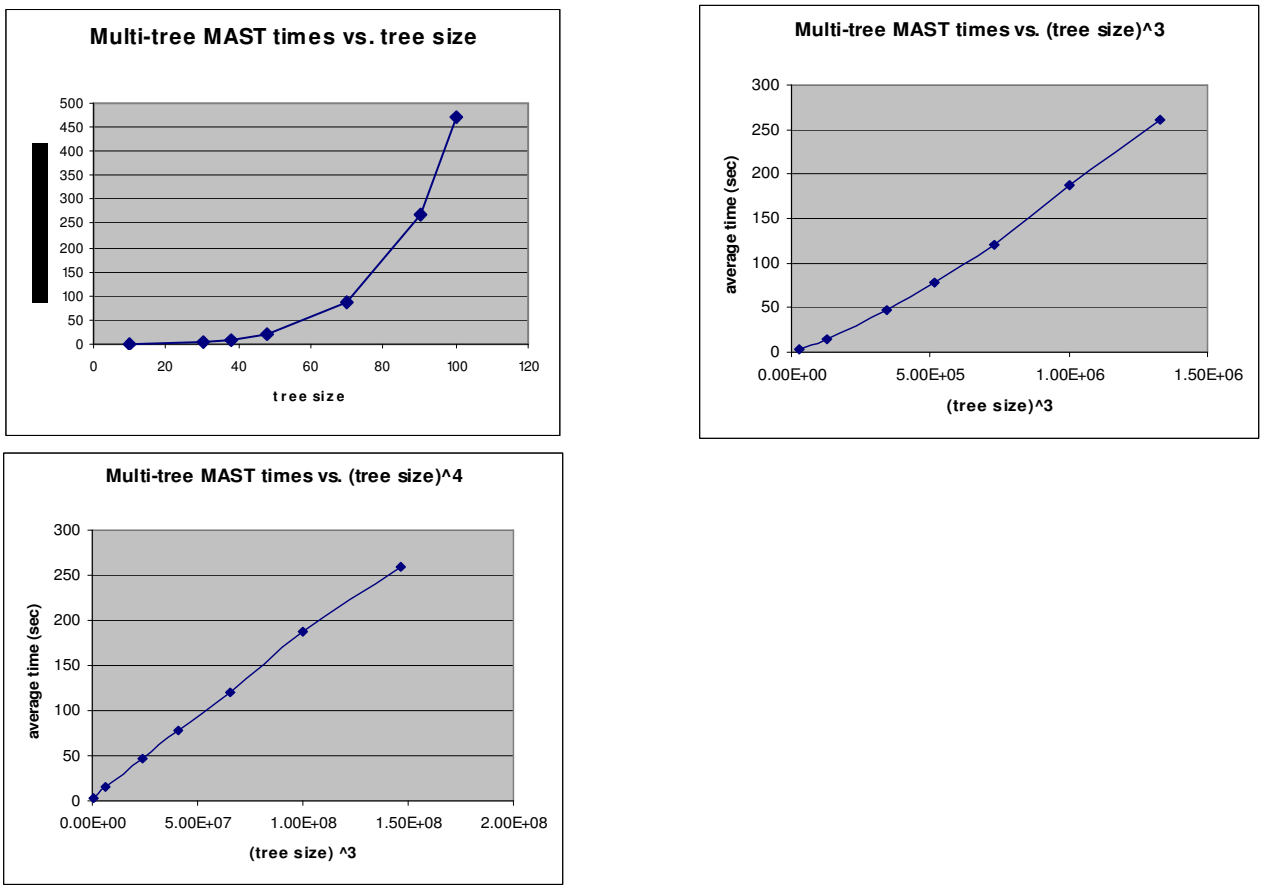


Figure 17 Average Running Time for MASTs of Three-Tree Groups, Plotted versus Tree Size, Indicating $O(n^4)$ Behavior.

table 2 and figure 17. We can verify from the plots that the two-tree algorithm exhibits $O(n^2)$ behavior. The multi-tree algorithm exhibits $O(n^4)$ behavior.

Table 3 Average Running Time for MASTs on 50-leaf Tree Groups with Indicated Group Sizes.

Tree Group Size	3	5	6	7	9
Avg. time (sec)	16.11	20.02	21.74	23.31	27.16

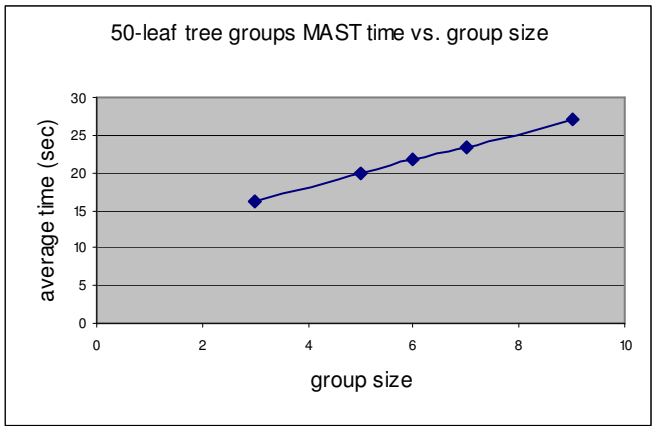


Figure 18 Average Running Time for MASTs of 50-leaf Tree Groups, Plotted versus Group Size.

We then constructed 5 sets of 50-leaf trees. The sets were of size 2, 3, 5, 6, 7 and 9. MASTs were constructed for each group and times were as reported in table 3. The two-tree group used the Goddard algorithm [35], while the Farach algorithm [34] was applied to all other groups. It can be seen from figure 18 that running time varies roughly linearly with group size in groups of size 3 or larger.

CHAPTER 4

CONCLUSION AND FUTURE WORK

4.1 Related Work

Many applications of phylogenetic tree analysis and comparison have been developed, such as PAUP [42], Phylip [43] and REDCON [45]. Each of these packages has its own advantages and limitations. PAUP is one of the most widely used phylogenetic tree processing software packages. It provides multiple tree construction algorithms, tree manipulations and tree display methods. It works on Unix, Windows and Macintosh platforms. However, in the windows version of PAUP most of the operations are implemented through a command-line interface. PAUP does not support the printing or saving of trees in its Unix, windows or DOS version, which is an important feature for our users. Phylip is also a powerful software package for phylogenetic tree construction and manipulation. It can also be used to compute the tree distance and obtain the consensus trees among a tree group. However, it does not provide a user interface and all of the functions must be run through the DOS command line. Also, the users must run multiple different executable programs in the package in order to implement a series of operations, which is inconvenient. Ntsyspc [46] is a software package composed of two programs for phylogenetic tree text file editing, tree construction and manipulation. It provides the consensus metric instead of the MAST metric for tree comparison but neither the displayed phylogenetic tree nor the resulting consensus trees can be manipulated or interacted with. COMPONENT [47] is a free phylogenetic tree processing software on the windows platform. It provides multiple tree comparison methods, such as the quartet metric, partition metric,

consensus metric and the MAST metric. However, in the case of the existence of multiple equivalent MASTs, it produces only one MAST. The solution is incomplete and not representative. In addition, the algorithm used is slow, and it does not work for comparing trees with more that fifty leaves. REDCON, developed by Mark Wilkinson in the department of Zoology of the Natural History Museum, is a phylogenetic tree comparison software. It provides multiple consensus methods, such as the strict consensus metric and the majority consensus metric. However, it is a DOS command based software and it has a severe limitation on both the number of trees compared and the number of leaves the trees contains. All of its tree comparison methods are unable to deal with trees with more than eighty leaves, which is not an unusual phylogenetic tree size.

4.2 Conclusion and Future work

In this paper, we introduced TreeAnalyzer, a phylogenetic tree comparison and visualization tool. TreeAnalyzer helps researchers to determine common phylogenetic structure and a quantitative index of closeness of trees. The similarity of compared phylogenetic trees indicates the relationships between two sequences and can provide insight into how those organisms have evolved.

Compatibility is one of the features of TreeAnalyzer. TreeAnalyzer allows efficient access to the generally used PAUP NEXUS format of tree data and provides MAST output that can either be saved as an object file or exported to an image file.

Neither of the two algorithms implemented takes into account the branch lengths. Algorithms do exist for the maximum weighted agreement subtree problem [48], which uses branch length information. However, these require the exact integer distance between each node

in all the trees, which is not usually known. If the exact branch length is unknown, the time complexity of the trees with approximated branch length is $O(kn^{d+1}+n^{2d})$, where k is a constant, n is the number of leaves in the trees, and d is the degree bound of the trees.

The software has a sound user interface. It differs from other phylogenetic tree comparison software in several aspects. It reads input data and visualizes it with a tree structure that the users can interact with. MASTs are mapped on the original trees to highlight or emphasize the areas of the trees that differ. Users may interact with MASTs by swapping the subtrees of any internal node, modifying leaf names or replacing the trees as a view.

TreeAnalyzer is able to compare multiple trees and produce an exact solution. The similarity index, a numeric result, is also applied as a quantitative measurement of similarity and dissimilarity information for tree comparison.

However, there are still some issues that need to be addressed in the future. First, the branch length may be taken into consideration in computing MAST. Also, the visualization of MAST may be improved by using other techniques, such as 3D trees. With 3D tree-mapping techniques, the degree of similarity can be represented as the angle of the MASTs away from the plane, and multiple MASTs can be displayed at once. Finally, for some researchers, the similarity index does not provide enough information and some other phylogenetic tree comparison metrics such as the quartet metric or partition metric can be used as a complement to provide more information.

REFERENCES

[1] C. R. Finden and A. D. Gordon. Obtaining common pruned trees. *Journal of Classification*, 2: 255 – 276, 1985.

[2] M. S. Waterman and T. F. Smith. On the similarity of dendrograms. *J. theor. Biol.* 73:789-800, 1978.

[3] Joseph L. Thorley and Roderic D. M. Page. The RadCon Manual v1.1.2, 2002.

[4] Martin Farach and Mikkel Thorup. Optimal evolutionary tree comparison by sparse dynamic programming (Extended Abstract). FOCS: 770-779, 1994.

[5] Maddison, D. R., Swofford, D. L., Maddison, W. P. NEXUS: An extensible file format for systematic information. *Syst. Biol*. 46:590-621, 1997.

[6] Java 2 SDK, Standard EditionVersion 1.4.1_05. http://java.sun.com/products/archive/j2se/1.4.1_07/README.html

[7] David M. Hillis, Craig Moritz, and Barbara K. Mable. Molecular Systematics. 1996.

[8] Carolyn J. Lawrence. Research Proposal, 2000.

[9] L. L. Cavalli-Sforza and A. W. Edwards, Phylogenetic analysis: models and estimation procedures, *Am. J. Human Genetics*, 19: 233-257, 1967.

[10] Penny, D., M. D. Hendy and M. Steel. Progress with methods for constructing evolutionary trees. *Trends Ecol. Evol*. 7:73 – 79, 1992.

[11] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. *Proc. Of 19th International Colloquium on Automata Languages and Programming*, 1992.

[12] W. H. E. Day. Computational complexity of inferring phylogenies from dissimilarity matrices. *Bulletin of Mathematical Biology*, 49(4): 461-467, 1987.

[13] Edward N. Adams. Consensus techniques and the comparison of taxonomic trees. *Systematic Zoology,* 21: 390 – 397, 1972.

[14] D. F. Robinson and L. R. Foulds. Comparison of weighted labeled trees. *In Combinatorial mathematics, VI (Proc. Sixth Austral. Conf., Univ. New England, Armidale, 1978), Lecture Notes in Mathematics : 119 – 126, Springer, Berlin,* 1979.

[15] W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 1:7-28, 1985.

[16] G. F. Estabrook, F. R. McMorris and C. A. Meacham. Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Zool*. 34:193-200, 1985.

[17] M. S. Waterman and T. F. Smith. On the similarity of dendrograms. *J. theor. Biol.* 73:789-800, 1978.

[18] Roderic D. M. Page. Components 2.0 manual, 1993.

[19] E. Kubicka, G. Kubicki and F. R. McMorris. An algorithm to find agreement subtrees, *Journal of Classification*, 12:91-99, 1995.

[20] M. A. Steel and D. Penny. Distributions of tree comparison metrics - some new results. *Systematic Biology,* 42(2), 1993: 126-141.

[21] D. Penny and M.D. Hendy. The use of tree comparison metrics. *Systematic Zoology*, 34: 75-82, 1985.

[22] W. H. E. Day. Optimal algorithms for comparing trees with labeled leaves. *Journal of Classification*, 1:7-28, 1985.

[23] M. D. Hendy, C. H. C. Little, and D. Penny. Comparing trees with pendant vertices labeled. SIAM. J. Appl. Math. 44: 1054-1067, 1984.

[24] P.Buneman. The recovery of trees from measures of dissimilarity. *Mathematics in the Archeological and Historical Sciences*, 387-395, Edinburgh University Press, 1971.

[25] Gerth Stolting Brodal, Rolf Fagerberg, and Christian N. S. Pedersen. Computing the Quartet Distance Between Evolutionary Trees in Time O(n $\log^2$n). *Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC).* Springer Verlag, Lecture Notes in Computer Science, Vol. 2223, pp: 731-742, 2001.

[26] Moore GW, Goodman M and Barnabas J. An iterative approach from the standpoint of the additive hypothesis to the dendrogram problem posed by molecular data sets. *J Theor. Biol.* 38(3):423-457, 1973.

[27] J. P. Jarivs, J. K. Luedeman and D. R. Shier. Comments on computing the similarity of binary trees. *J. theor. Biol.* 100: 427-433, 1983.

[28] David R. Maddison. The discovery and importance of multiple islands of most parsimonious trees. *Systemic Zoology*, Vol. 40(3), pp. 315-328, 1991.

[29] M. Li, J. Tromp, and L. Zhang. On nni distances on evolutionary trees. *J. of Theoretical Biology* 182:463-467, 1996.

[30] Edward K. Brown and William H. E. Day. A Computationally efficient approximation to the nearest neighbor interchange metric. *Journal of Classification* 1:93 – 124, 1984.

[31] David Bryant, A classification of consensus methods for phylogenetics, F. (eds) *Bioconsensus* . DIMACS-AMS. 163—184, 2003.

[32] R. Cole, M. Farach, R. Hariharan, T. Przytycka and M. Thorup. An O(N log N) algorithm for the maximum agreement subtree problem for binary trees, *SIAM J. Comput*. 30(5): 1385-1404, 2000.

[33] D. L. Swofford. When are phylogeny estimates from molecular and morphological data incongruent? Pages 295-333 in: *Phylogenetic analysis of DNA sequences (M. M. Miyamoto and J. Cracraft, eds.). Academic Press, New York*, 1991.

[34] Martin Farach, Teresa M. Przytycka and Mikkel Thorup. On the agreement of many trees. *Information Processing Letters*, 55:297-301, 1995.

[35] W. Goddard, E. Kubicka, G. Kubicki and F. R. McMorris. The agreement metric for labeled binary trees, *Mathematical Biosciences* 123: 215-226, 1994.

[36] Robinson, D. F.. Comparison of labeled trees with valency three. *J. Comb. Theor*. 11: 105-119, 1971.

[37] M. Steel and T. Warnow. Kaikoura tree theorems: computing the maximum agreement subtree, *Information Processing Letters*, 48: 77-82, 1993.

[38] M. Farach and M. Thorup. Fast comparison of evolutionary trees (extended abstract), in *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia,* pp. 481-488, 1994.

[39] B. Bollobas. Modern Graph Theory, Springer – Verlag, New York, NY, 1998.

[40] M. Farach, T. Przytycka and M. Thorup. The maximum agreement subtree problem for binary trees. *Proc. Of 2nd ESA*, 1995.

[41] M. Farach and M. Thorup. Fast comparison of evolutionary trees (extended abstract), in *Proc. 5th Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia,* pp. 481-488, 1994.

[42] Swofford, D. L. 2003. PAUP*. Phylogenetic Analysis Using Parsimony (*and Other Methods). Version 4. Sinauer Associates, Sunderland, Massachusetts.

[43] Felsenstein, J. 1989. PHYLIP -- Phylogeny Inference Package (Version 3.2). Cladistics 5: 164-166.

[44] Lawrence, C. J., C. M. Zmasek, and R. K. Dawe, and R. L. Malmberg. LumberJack: a heuristic tool for sequence alignment exploration and phylogenetic inference. *Bioinformatics*, in press, 2004.

[45] Wilkinson, M. REDCON 3.0: software and documentation. Department of Zoology, The Natural History Museum, London.

[46] Rohlf, F. J. 2002. NTSYSpc: Numerical Taxonomy System, ver. 2.1. Exeter Publishing, Ltd.: Setauket, NY.

[47] Page, R.D.M. 1993. *COMPONENT*: Tree comparison software for Microsoft Windows, version 2.0. The Natural History Museum, London.

[48] Amihood Amir and Dmitry Keselman. Maximum agreement subtree in a set of evolutionary trees: metrics and efficient algorithms. *SIAM Journal of Computing*, Volume 26, Number 6, pp:1656 – 1669, 1997.