An improved phylogenetic tree comparison method

Suneel Potiny

A thesis submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Master of Science in the Department of Biomedical Engineering

Chapel Hill

2010

Approved By:

Dr. Shawn Gomez

Dr. Oleg Favorov

Dr. Glenn Walker

Abstract

Suneel Potiny

An improved phylogenetic tree comparison method

(Under the direction of Dr. Shawn Gomez)

The comparison of phylogenetic trees is an evolving area of study. The need to analyze the similarities among trees found in phylogenetic databases is growing, and while several tree comparison methods exist, further work in this area can be beneficial. This research presents a novel, distance-based tree comparison method (MDS-Procrustes). Our approach begins with two trees and represents the patristic distances between nodes within each tree in separate distance matrices. Then we use classical multi-dimensional scaling to represent these distances as Euclidean structures in a high-dimensional space while maintaining the original distance information. Finally, Procrustes analysis is used to superimpose one Euclidean structure onto the other to generate a similarity score between the two trees. In this work, we compare the effectiveness of the proposed method to existing approaches found in the literature in terms of both similarity score distribution and computational performance.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Phylogenetic Trees

Understanding the evolutionary history of genes and species is central to the field of evolutionary biology. Evolutionary (Phylogenetic) trees are the basic structures that contain data fundamental to visualizing and statistically analyzing this historical information (Felenstein 2003). These trees are designed to reveal evolutionary relationships among DNA or protein sequences across species. Comparing these structures can be an effective way of predicting evolutionary events and may even assist in detecting random evolutionary occurrences such as horizontal gene transfer (Choi and Gomez 2009). The advent of sequencing technologies have increased the production of phylogenetic trees and further popularized a branch of science known as molecular phylogeny. Several different methods currently exist to provide useful metrics for comparing one or more phylogenetic trees (Brown and Day 1984, Bryant et al. 2000, Keselman and Amir 1994, Robinson and Foulds 1980, Wang et al. 2005), however due to the increasing number of phylogenetic trees being created and made publicly available, the need to develop an effective means of quickly searching and comparing trees for similarity is considerable. Since it can be theorized that two species with similar phylogenetic tree structures would contain genes or proteins that co-evolve with one another, these comparisons can be a useful method for predicting such phenomena and continued work in this area can be informative.

1.2 Types of Phylogenetic Trees

In a phylogenetic tree, each internal node with descendant nodes represents the most recent common ancestor of the descendants. Each node in the tree is called a taxanomic unit and internal nodes are commonly known as Hypothetical Taxonomic Units(HTUs) since they often cannot be directly observed. Phylogenetic trees can either be rooted or unrooted. A rooted tree refers to a tree in which all leaves in a tree share a common ancestry to a “root” sequence or node. An unrooted tree is a tree that makes no inferences about the ancestry of the leaves contained therein and can be useful to visualize related sequences contained in a particular organism(despite the ancestry being unknown).   Figure 1 shows both a rooted and an unrooted tree.

Any rooted tree can be made unrooted by simply removing the root (as seen in Figure 1), however, an unrooted tree cannot be made into a rooted tree without making an inference as to the ancestry. This is an important distinction to make, as it is often difficult to detect a common ancestry between species, and so it is often necessary for evolutionary biologists to analyze both rooted and unrooted trees.

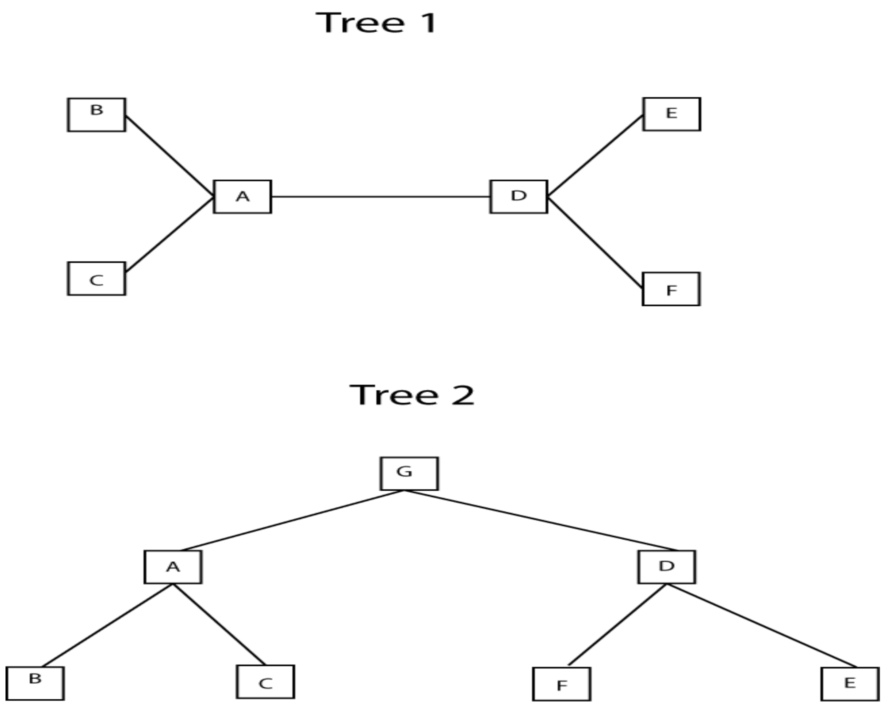


Figure 1 – Rooted and Unrooted Trees

Tree 1 shows an unrooted tree. Tree 2 shows the same tree as in Tree 1, except it becomes rooted with a common ancestor (G). It can be seen from this figure that a rooted tree can become unrooted by removing the root node (ie removing Node G from Tree 2 to transform it into Tree 1), however the converse is not true.

Additionally, both rooted and unrooted trees can be either bi-furcating or multi-furcating. A bifurcating tree is a tree in which each internal node has exactly two immediate decendants. A multi-furcating tree can have nodes with more than two descendants. Trees can also be either labeled or unlabeled. A labeled tree has values assigned to its nodes, while unlabeled trees only define a topology of relationships between sequences/events without containing information about the nodes themselves.

Also, some phylogenetic trees can assign values to the branch lengths between nodes. These are known as weighted trees. This branch length can often be considered a measure of evolutionary (genetic) distance between two nodes. Usually, this measure will represent a genetic divergence between two sequences or species. Often times it is useful to think of this as a measure of time in which the node was changed from its ancestor. The ability to obtain this information can be difficult and while there are existing hypotheses (such as the molecular clock that uses fossil constraints and rates of molecular change to compute this time), they often contain a great deal of error and cannot always be used with much confidence in a scientific study. However, if this information exists, it can obviously be extremely useful in deducing similarities among taxa between species, as it gives more definition to the topology of a particular tree.

One commonly used technique for obtaining these distance measures is by looking at the similarity between two or more biological sequences across species (generally DNA, RNA, or protein sequence data obtained for a particular set of species). In order for this measure to generate a biologically relevant tree, the sequences being analyzed should be *homologs* (known to derive from a common ancestor). There are many models available to analyze the similarity between two homologous sequences and determine an evolutionary distance between the sequences. One such model is the Kimura substitution model. This DNA substitution model assumes that changes between the two base types (pyrimidines and purines) occur at different rates than changes within the base types (thus changes from A->G and T->C occur at one rate and changes across base types occur at a different rate). To illustrate this process, we will look at the following two homologous, aligned input DNA sequences from two different species (1 and 2):

1 - AGAATAGTTAG

2 - CGATGA –T – AG

We then define a similarity matrix (S_{ij}) according to the described Kimura model as follows:

	A	G	T	C
A	0	1	0.5	0.5
G	1	0	0.5	0.5
T	0.5	0.5	0	1
C	0.5	0.5	1	0

We will also assign a gap penalty of 2 in this example (to correspond to a deletion event in a sequence). Thus, based on the above given information, for this particular example we get the following output:

S(A,C) + S(G,G) + S(A,A) + S(A,T) + S(T,G) + S(A,A) + (gap penalty) + S(T,T) + (gap penalty) + S(A,A) + S(G,G) = (0.5) + (0) + (0) + (0.5) + (0.5) + (0) + (2) + (0) + (2) + (0) + (0) = 4.5

So, the calculated evolutionary distance between the sample sequences would be 4.5. In this form of analysis, the greater the evolutionary distance, the longer the time it would take for one sequence to evolve into the other.

The above example was a simple pairwise distance measurement, but of greater significance to studies on phylogenetic trees are alignments of multiple sequences (MSAs). There are several methods available find the optimal alignments (alignments that minimize the distance measure) between multiple sequences, including ClustalW and T-coffee. These programs essentially break down the multiple sequences into a series of pairwise alignments and use that in conjunction with a guide tree to generate a distance value. Once an evolutionary distance measure between sequences is obtained, this value can be used in a distance-matrix tree construction model that will be detailed further in the proceeding section.

1.3 Construction of Phylogenetic Trees

As previously stated, the development of sequencing technology has benefited the study of evolutionary relationships among organisms. The data from a sequence alignment (along with other types of data) is often used in phylogenetic tree reconstruction. Construction of phylogenetic trees is generally based on *cladistics* (the systematic classification of organisms on the basis of shared characteristics deriving from a common ancestor). The different construction methods can loosely be categorized into three different types: Distance-based methods, character based methods, and probabilistic methods.

Distance-matrix methods (such as Neighbor-Joining, Fitch-Margoliash, or Unweighted Pair Group Method (UPGMA)) generally require a multiple sequence alignment (MSA) as input. The example in the previous section briefly illustrated how this distance can be calculated and provided this distance value, evolutionary origin can be inferred. Distance-matrix methods generally function by clustering the nodes together based on the input distance information resulting from a sequence alignment. Due to the complexities of the input into these methods, the algorithms to produce this are computationally complex and consequently heuristic approaches are often used to help simplify the problem.

Character based methods treat differences between the input sequences as discrete characters with each character considered to be in a particular state (e.g. present vs not present). The most common character based method is maximum parsimony. In this approach, a phylogenetic tree is generated and deemed optimal based on the fewest number of changes between the states of the informative characters in a particular data sequence. Probabilistic approaches are generally similar to character based approaches (and use a similar generalization of discretized characters representing particular traits). An example of a probabilistic approach is the maximum-likelihood method. This uses a *substitution model* (the assigning of particular traits/characteristics to a character) to define the probability of a particular mutation and then uses standard statistical analysis to infer a probability distribution from this data and generate a particular phylogenetic tree.

However, the problem of finding a suitable phylogenetic tree given even a relatively small set of nodes is computationally intensive. This occurs because the number of possible tree structures that can be created from a set of species or sequences greatly increases with the inclusion of each additional unit. Given a set of species, n, the following equation defines the number of possible formations of a rooted, bifurcating and unweighted trees:

$$\frac{(2n-3)!}{2^{n-1}(n-1)!} \tag{1}$$

This equation can easily be computed to show that the number of possible trees grows greatly with additional nodes (at n =1 there is 1 possible tree, at n =5 there are 105 possible trees and at n =10 there are 34,459,425 possible trees). Since any unrooted tree can be made a rooted tree simply by omitting the root, it is clear that the number of unrooted trees possible is less than the number of rooted trees. When adding a node to a tree in all possible positions, you would simply add it to each existing branch in the tree. And combining this with the fact that a rooted tree is made unrooted by removal of the root node, we see that the number of possible rooted, bifurcating trees with n nodes would be the number of possible unrooted (n -1) trees. Due to the high number of possible trees, optimizations techniques are needed for any of the algorithms to efficiently compute the most desired phylogenetic tree as the computational cost of going through all possible trees can be very time consuming (and even unrealistic) for trees of larger size.

1.4 An Example of Character Based Phylogeny Reconstruction

To further illustrate the process by which a tree is constructed, we will look at a simple character based example and use a maximum parsimony approach to partially evaluate the proposed phylogenetic tree's accuracy (Felsenstein 2003). Table 1 below shows a list of five species and each species' character state for 4 chosen characters (for simplicity, we will assume that the characters each have two states, 0 and 1).

	Characters			
Species	1	2	3	4
A	1	0	0	1
B	1	0	1	1
C	0	1	0	1
D	1	0	1	0
E	0	1	0	0

Table 1 – Species Character Chart

Table shows species and the corresponding state for each of 4 different characters.

From here, we will generate a random, rooted phylogenetic tree structure with 5 nodes representing each of the species. The tree that we will evaluate here is shown in figure 2.

To evaluate the proposed tree's accuracy according to the maximum parsimony criteria, we will count the number of state changes that occur along the defined edges in each of the character definitions from the table. Figures 3-6 show the phylogenetic tree from Figure 2 evaluated for each character from Table 1 (Regular lines represent a state of 1 and bold lines represent state of 0, the period where the line changes indicates a change of state along that branch).

In figure 3, from the chart we see that species A and B have the same state for character 1, so there is no change of state at the node connecting those two species. The next node is connecting this subtree to species C, which has a state different than that of species' A and B, so for this tree to be accurate according to the chart, there must be a change of state along the branch connecting the A-B subtree to C (depicted by the bold line, signaling a change in state from 1 to 0). This same convention is used throughout the figures, so the number of changes in state can easily be counted by looking at the number of times the lines go from normal to bold. In these figures, we assume the state of the root is the same as the left side of the tree at the point which it attaches to the root. Thus, if there is a change of state from the left side to the right side of the tree, we show that change of state occurring on the right side of the tree along the branch leading from the root to the D-E subtree.

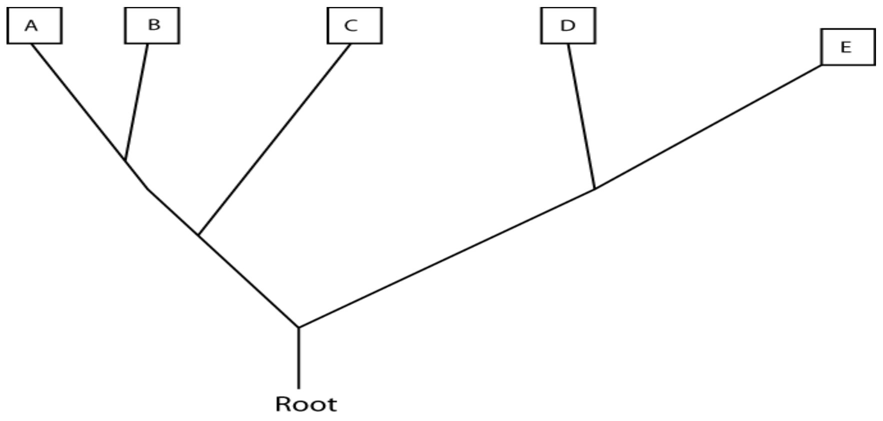


Figure 2 – Proposed Phylogenetic Tree

Proposed phylogenetic tree for evaluation with character states in Table 1.

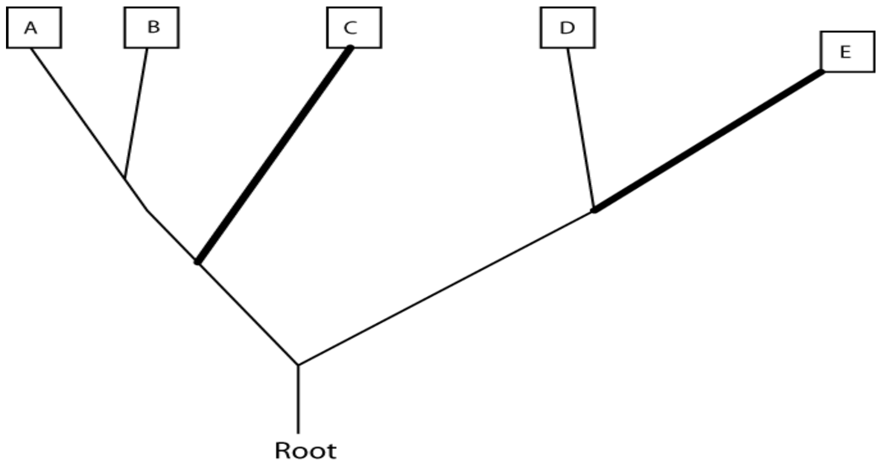


Figure 3 - Reconstruction of character 1 on proposed tree.

Character state changes from 1 to 0 along the branch extending from A-B subtree to species C and also from the Root to species E. Two total changes of state.

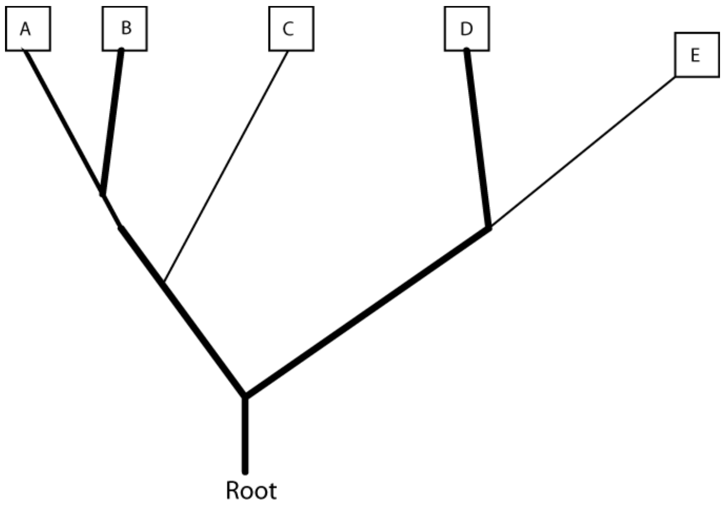


Figure 4 - Reconstruction of character 2 on proposed tree

Changes of state occur on the branch from the A-B subtree connecting to C and then from the Root to species E(exact opposite of the state changes from Figure 3). Two total changes of state.

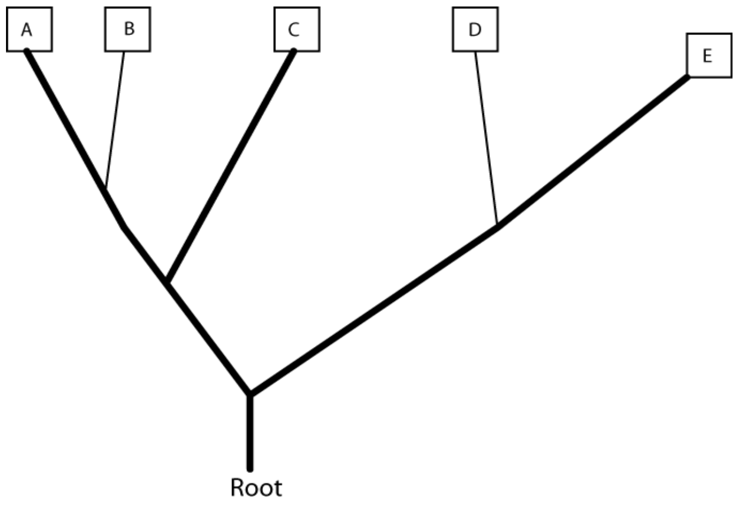


Figure 5 - Reconstruction of character 3 on proposed tree.

Changes of state from 0 to 1 occur along the branch leading to species B and the branch leading to species D. 2 total changes of state.

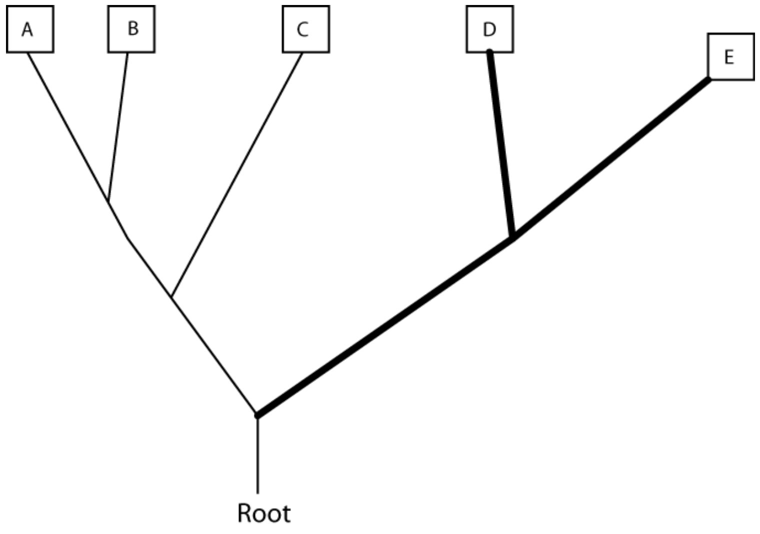


Figure 6 - Reconstruction of character 4 on proposed tree.

Change of state from 1 to 0 occurs along path leading from root to the D-E subtree. 1 change of state in tree.

We then compute the total number of state changes across each of the characters (2+2+2+1) to get a total of 7 changes for this particular tree. In a maximum parsimonious approach, this number of changes would be compared to all other possible tree constructions to determine the most desirable tree for the informative characters defined in Table 1.

1.4 Comparison of Phylogenetic Trees

The comparison of these phylogenetic tree structures can be very informative. As stated previously, since the data contained in these trees represents genetic or other biological similarities across species, much can be explained from analyzing evolutionary trees. With several databases, such as TreeBASE(Sanderson 1994), publicly available for scientists to upload and analyze their own constructed trees with other existing trees, the need for finding a suitable method for comparing these trees is increasing. While many current methods exist, the distance-based method presented in this paper is shown to be useful and in many ways superior to other existing methods analyzed in previous studies.

One specific biologically significant use of comparing trees is in the detection of non-standard genetic events, such as horizontal gene transfer (MacLeod 2005). Horizontal gene transfer (HGT) is the process by which an organism incorporates genetic material from another organism without being an offspring of that organism. This is in contrast to vertical gene transfer, through which an organism obtains genetic material directly from a parent (or species from which it evolved). This phenomena is thought to play a significant role in drug resistance (which refers to the reduction in effectiveness of a drug in curing a disease), as a bacteria can pass on these mutated and resistant genes from one cell to another. Providing an effective means of detecting these events can potentially be beneficial to studies in that area.

Also, these comparisons can be useful as a means of analyzing co-evolution between various species or molecules. This is commonly seen in host-parasite relationships. An example of this would be the relationship between the attine ant, the fungi that it cultivates, and the Escovopsis parasite (Currie 2003). It is often assumed that genes and proteins in these types of symbiotic relationships will change together, so that a change in one will correspond to a change in the other as a means of adaptation. Comparing trees for similarity can greatly aid in predicting these events.

Furthermore, outside of the aforementioned biological uses of these methods, comparing trees can be useful in other disciplines as well. Any system that attempts to predict events based on a previous history of input (such as search algorithms which attempt to suggest things to buy or view based on your previous purchases or search criteria) can often times be categorized in tree structures and so comparing them can aid in the predictive abilities of such tools. The goal of this paper is to evaluate a new method and compare it to the already existing methods in hopes that it can be a potentially useful tool for scientists and researchers/developers in these areas of interest.

In the method (referred to as MDS-Procrustes) presented here, we use classical multi-dimensional scaling to represent the distances between nodes within a tree as structures in a high dimensional space and then attempt to fit each tree's structure together through a Procrustean-related approach to determine the similarity of the two trees. To understand the approach and how it improves in solving the problem of comparing trees from databases, it is important to fully discuss the advantages/disadvantages of existing methods (which will be explained in the proceeding sections) and the ways in which our method differs from them to attain an improved searching function.

1.5 Existing Comparison Methods

Existing methods for comparing phylogenetic trees can generally be classified into two categories, methods that compare trees based on topological features and distance-based methods. In Choi and Gomez (2009), a similar approach to MDS-Procrustes was compared the tol-mirrortree approach presented by Pazos and Valencia (2001) in the prediction of protein-protein interaction networks. It was shown there that this technique offered an improvement to some distance based methods and here we will extend that work to show how it also outperforms many of the topological methods in the specific problem of comparing phylogenetic tree structures for similarity. Topological methods generally ignore the distances between tree nodes and instead rely on comparing trees for similarity based on related topological features and relatedness between taxa (Quartet, Partition, Maximum Agreement Subtree, Updown) or on transforming one tree into another (Nearest Neighbor Interchange). While these methods can be useful in certain limited comparisons or in generating groupings among trees, we find that they either do not scale well to comparisons on large data sets of many trees with many different nodes or do not offer the distribution of similarity scores necessary to effectively rank trees by similarity score values.

To fully evaluate the quality of the proposed method, we compared MDS-Procrustes to several existing methods found in the literature by looking at computational time and similarity score distribution. The metrics we compared our method to here are Nearest Neighbor Interchage (NNI), Partition (PAR), Maximum-agreement subtree (MAST), Quartet, and Updown (USim). These metrics are among the more popular methods found in the literature and have proven to be very useful methods in previous studies (Steel and Penny 1993, Wang 2005). All of these methods are topologically based (as opposed to our distance-based metric) and thus rely on direct relationships among nodes to determine similarity. In the proceeding sections, we discuss and illustrate each of these methods before offering a full analysis of our proposed MDS-Procrustes approach.

1.6 Nearest Neighbor Interchange Metric

The NNI metric refers to the number of near neighbor interchange operations necessary to transform the query tree into the data tree (we use the term query tree to refer to the tree for which we are searching through a data set of data trees to find a nearest match). It essentially involves the swapping of nodes around a common edge of the query tree until the data tree is fully realized. In a labeled binary tree, the end points of each interior edge will be adjacent to two distinct subtrees. A nearest neighbor interchange simply involves the interchanging of these two subtrees, one adjacent to each endpoint. Figure 7 illustrates this process and shows a tree and the two possible nni operations along a particular interior edge. In the figure, the internal edge e induces two separate subtrees, A-B and D-C. The two possible nearest neighbor interchanges on that edge that would induce different subtrees from that edge involve swapping B and D (to induce subtrees A-D and B-C) or swapping C and B (to induce subtrees A-C and D-B).

A binary subtree with n nodes will have n -3 interior edges. Conversely it can be transformed by at most $2n$ -6 nni operations (Day 1983). Per the metric, the fewer the number of nni operations necessary to transform the query tree into the data tree, the more similar the trees. This has been a well-studied algorithm, and while its simplicity and topological sensitivity is an advantage, the problem of computing this distance has proven to be an NP-complete problem with regards to the size of the tree (DasGupta et al. 2000). Also, while it has a natural extension to the comparison of weighted trees (Hon and Lam 2000), NNI does require that the trees have the same labels in order to produce an accurate comparison of two trees (Wang et al. 2005a).

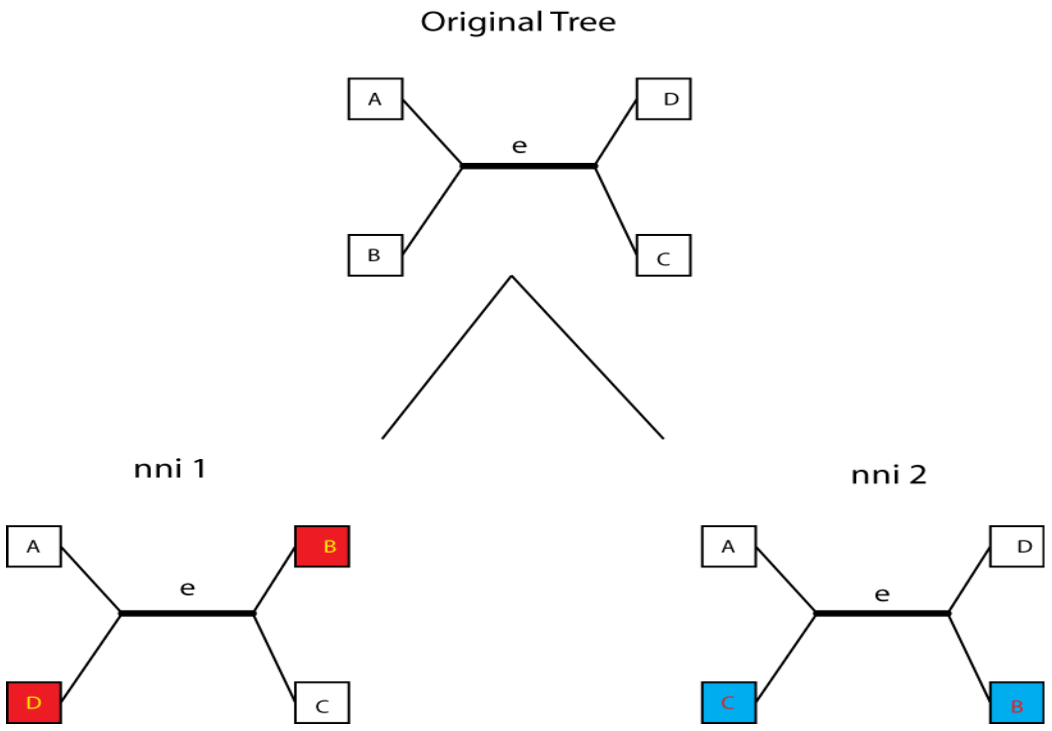


Figure 7 – NNI diagram

Shows the two possible Nearest Neighbor operations along internal edge(e). The highlighted nodes in the two trees below the original tree shows how the tree transformed in each case by using 1 NNI operation along edge e.

1.7 Partition Metric

The PAR metric (also known as symmetric difference or Robinson-Foulds) is one of the older methods found in the literature. PAR was originally presented by Bourque (Felsenstein 2003) in 1978 and has been cited in several studies since (Wang 2005a, Steel and Penny 1993). In this method, a partition refers to the two sets of structures created upon the deletion of an edge of a tree. In order to divide the tree, every branch or edge splits the tree into two new trees. Thus, each edge indicates the presence of two separate partitions of the original tree. The symmetric difference between two trees consists of the sum of the different edges between two trees, and so for two trees that are identical, they would have a symmetric difference of zero since they would have no different partitions. Figure 8 shows two trees and details the different Robinson-Foulds partitions that exist between them. In the figure, it is easily seen that the only difference between the trees is that the positions of nodes E and C are transposed. When computing the symmetric difference, we look at the partitions created upon the removal of each internal edge of the tree. When doing that in this example, it is seen that the only times that separate partitions are created occur when the highlighted edges are removed. In tree 1, when we remove the highlighted edge, the partitions created are {(A,B,E),(D,C)}, while in tree 2, removal of the highlighted edge generates the partitions {(A,B,C),(D,E)}. Removal of the other internal edges in either trees leads to the same partitions and thus for these two trees, the symmetric difference is equal to 2.

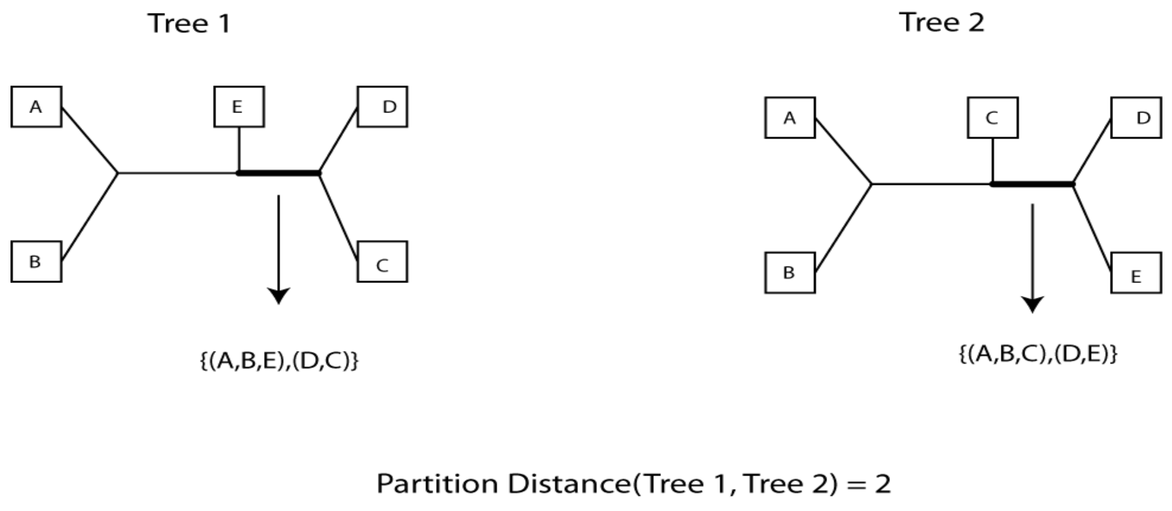


Figure 8 – Partition Diagram

Shows two trees and the partition distance between them. The highlighted edges show the different edges between the two trees and the partitions that result from a deletion of that edge. The partition distance between these two trees is 2.

In terms of computational time, this method scales well with respect to the number of nodes in each tree and so the symmetric difference can be computed quickly even when dealing with large trees. However, it can be very sensitive to the location of certain nodes. For example, moving the root of the tree to a different spot (and leaving all other nodes the same) can lead many different partitions between the two trees, thus indicating that the trees are extremely different, even though they only differ in the placement of one node. Also, the distribution scores that it can provide are limited to the number of branches in a tree and thus it is not able to provide a good enough distribution to be able to effectively rank trees according to similarity.

1.8 Maximum Agreement Subtree(MAST) metric

The maximum agreement subtree (MAST) method is another relatively straight-forward topological metric used for comparing two trees. In this method, introduced by Gordon (1979), the objective is to choose a subtree from the query tree for which there is an equivalent match in the data tree. A subtree of a tree *T* is a tree consisting of a node in *T* and all of its descendants in *T*. The maximum agreement subtree is the subtree with the highest number of nodes from the original tree. The metric used in this paper is equal to the number of leaves removed from the data tree to obtain this maximum agreement subtree. Figure 9 shows two trees and the maximum agreement subtree between them.

This method is convenient for visualizing the similarity between two trees and for searching for a ”consensus” tree within a large group of trees that is close to all given trees (Keselman 1994). This “consensus” tree may contain valuable information regarding common ancestry among the species/taxa and so it can be extremely useful for a researcher, but like the partition metric, its distribution scores are limited to the number of nodes in the tree and so it cannot be used to rank trees according to similarity from a large data set.

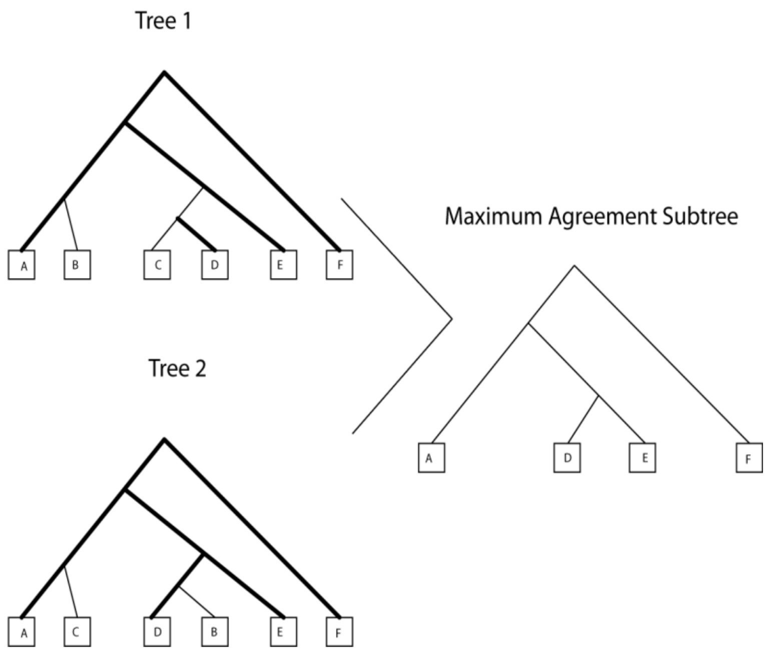


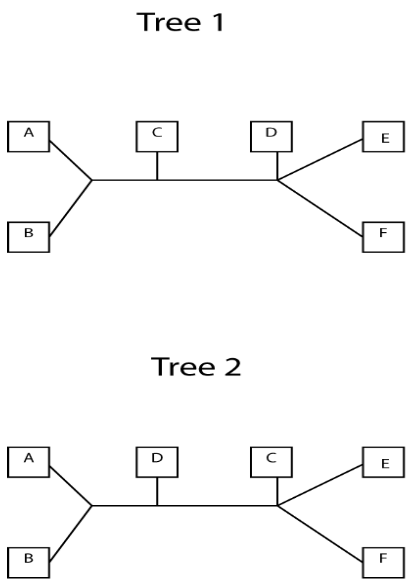
Figure 9 – MAST Diagram

Shows two trees and their Maximum Agreement Subtree(MAST). The highlighted branches show the branches that exist in both trees to form the MAST structure.

1.9 Quartet Metric

The quartet metric compares two trees based on the configurations of quartets of nodes in each tree. Any unrooted tree will induce a quartet topology on any four nodes within the tree. An unrooted tree with n nodes containts Q = $n(n-1)\ (n-2)(n-3)/24$ quartets. The set of all of these quartet topologies is theoretically unique to each tree (Eastbrook 1985). The metric used here refers to the number of quartet topology differences between the query tree and data tree (the less quartets that are different, the more similar the trees). In Figure 10 and the adjoining table, we show the different topologies induced in two trees by evaluating the quartets. In the table, we list all of the possible quartet topologies ($n = 6$, so there are 15 possible quartets) and describe how each is resolved in tree 1 and tree 2. We mark the differences with arrows, and since four of the quartets are resolved differently in the two trees, we say the quartet distance is equal to 4.

Since this method observes four node topologies, it is very stable for trees of very large sizes (when the number of possible quartets is large) and can be an excellent means for obtaining an effective similarity ranking among a number of trees in those cases.   Studies have also been done to show how to improve the run-time for finding the quartet distance (Stissing et al. 2005), as finding all of the possible topologies is computationally inefficient (albeit not NP-complete).



	Quartet	Tree 1	Tree2
	ABCD	AB|CD	AB|CD
	ABCE	AB|CE	AB|CE
	ABCF	AB|CF	AB|CF
	ABDE	AB|DE	AB|DE
	ABDF	AB|DF	AB|DF
	ABEF	AB|EF	AB|EF
→	ACDE	AC|DE	AD|CE
→	ACDF	AC|DF	AD|CF
	ACEF	AC|EF	AC|EF
	ADEF	AD|EF	AD|EF
→	BCDE	BC|DE	BD|CE
→	BCDF	BC|DF	BD|CF
	BCEF	BC|EF	BC|EF
	BDEF	BD|EF	BD|EF
	CDEF	CD|EF	CD|EF

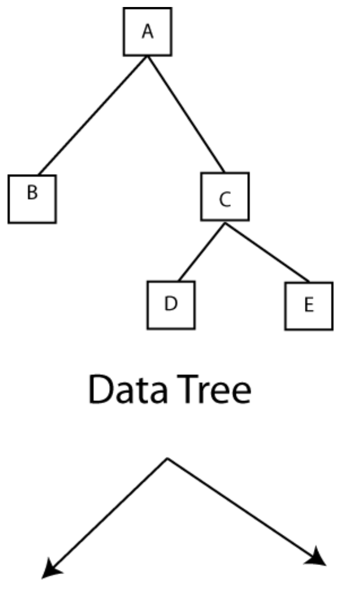
Figure 10 – Quartet Diagram

Shows two trees and the quartets induced by each of them. The chart shows all possible quartet configurations and how they are resolved in each tree. The arrows represent the quartets that are resolved and different in each tree. Thus, these two trees would have a quartet distance of 4 (Page 2001).

1.10 Updown Distance Metric

A potential improvement to these existing methods was proposed by Wang et al. (2005a). Their method is referred to as the Updown method (or USim) and is a topological approach based on the number of up and down operations between nodes in a tree. Up and down operations between two nodes refer to the direction that you move along a tree to get from one node to the other (a down operation refers to going from a parent node to its child (or descendant) node and an up operation refers to moving in the opposite direction). In this method, this convention is used to compute an Up matrix and Down matrix for each tree in a comparison. Figure 11 illustrates the process of generating both the Up and Down matrices from a given phylogenetic tree. Looking at the Up matrix first, we can see that since A is the root of the tree, it has no “up” operations to get to any of the other nodes. However, looking at node “C”, we can see that it takes one “up” operation for it to get to node A or node B (since when traversing the tree to get from C-B, we have to go to A and then to B). Conversely, there are no “up” operations required to go from C to D or E since those nodes are descendants of C. Looking at C again in construction of its row in the “down” matrix, we see that there is one down operation to go from C to D and from C to E since those are child nodes of C and there is also 1 down operation required to go from C to B (since the path is from C->A->B and B is a child node of A). Since C is a descendant of A, there are no down operations required to go from C to A. It can be seen that the up matrix can be generated from the down matrix and vice-versa with the down matrix being equal to the transpose of the up matrix.

Provided these matrices for both trees in a comparison, the Updown distance between the two trees can be calculated. Since we have seen from above that the Up matrix and Down matrix are related to each other with one being the transpose of the other, only the up matrix (U) is used in the equations for computing the Updown distance. To compute this distance, for the comparison of a query tree Q to a data tree D, a few definitions are needed. First, let V_Q be the



$$\begin{array}{c|ccccc|}
 & A & B & C & D & E \\
\hline
A & 0 & 0 & 0 & 0 & 0 \\
B & 1 & 0 & 1 & 1 & 1 \\
C & 1 & 1 & 0 & 0 & 0 \\
D & 2 & 2 & 1 & 0 & 1 \\
E & 2 & 2 & 1 & 1 & 0 \\
\hline
\end{array}$$

Up Matrix

$$\begin{array}{c|ccccc|}
 & A & B & C & D & E \\
\hline
A & 0 & 1 & 1 & 2 & 2 \\
B & 0 & 0 & 1 & 2 & 2 \\
C & 0 & 1 & 0 & 1 & 1 \\
D & 0 & 1 & 0 & 0 & 1 \\
E & 0 & 1 & 0 & 1 & 0 \\
\hline
\end{array}$$

Down Matrix

Figure 11 – Updown Diagram

Shows a phylogenetic tree and its corresponding Up and Down matrices. The updown distance(USIM value) between two trees is computed using these updown distance matrices from each tree in a comparison (Wang et al. 2005a).

set of labeled nodes in Q and V_d the set of labeled nodes in D. In calculating the Updown distance from Q to D, the only nodes of interest from D are those that match nodes from Q, so the set I will denote the intersection of matching nodes between Q and D and then J = $V_Q - V_D$(the nodes in Q that are not in D). Given these definitions, the equation for the updown distance from Q to D can be written as follows:

$$\text{Updown distance(Q,D)} = \sum_{u\varepsilon I}\sum_{v\varepsilon I} | U_Q[u,v] - U_D[u,v]| + \sum_{u\varepsilon J}\sum_{v\varepsilon J} U_Q[u,v] \qquad \textbf{(2)}$$

Then, in order to compute the similarity score between Q and D, denoted USim (the metric actually used in this paper in future sections), the following equation is given:

$$\text{USim(Q,D)} = \left(1 - \frac{Updown\_distance(Q,D)}{\sum_{u\varepsilon Vq}\sum_{v\varepsilon Vq} U_Q[u,v]} \right) \text{ x } 100\% \qquad \textbf{(3)}$$

Thus, if two trees are identical (or if the data tree is an exact substructure of the query tree) then USim = 100%. Since this distance considers nodes found only in the query tree in the computation, it is not symmetrical (thus the updown distance of Q->D is not necessarily the same as the updown distance from D->Q). This method was shown to be a significant improvement in terms of score distribution resolution and also through the use of a tree-reduction technique, proved to be more time-efficient than most of the previously described methods. The reduction technique works by finding the intersection of the common nodes between two trees. Then, the matching nodes are "marked" along with the least common ancestor node. The "marked" nodes then form a reduced data tree which is compared to the query tree to generate a similarity score. This reduction technique allowed for the removal of extraneous nodes from the computation of the updown matrices which resulted in the aforementioned improvement in computation time for the method. To further test the effectiveness of their algorithm, a function was implemented in a web-based search of the TreeBase (Sanderson 1994) database of trees, and proved to be quite effective for searching for similar structures within that database.

1.11 Possible Areas for Improvement on Existing Methods

While these methods have been extensively studied and are effective in certain cases, they can also be somewhat limited, especially in regard to the specific problem of searching large databases to find an effective ranking of phylogenies presented therein. PAR, while computationally efficient with respect to tree size, does not offer a good distribution of similarity scores and is highly sensitive to the topology of the tree. Thus, while two trees might differ with only respect to a small number of nodes, the partitions created by these nodes can be very different and thus the two trees could be considered very distant (Bryant et al. 2000). MAST offers a simple means of visualizing the comparison of two trees, but it is not time efficient and does not offer the resolution necessary to implement as a search with a large database of trees. Quartet and NNI offer a relatively high resolution of similarity measures when comparing large trees, however both can be computationally intensive and in the case of NNI, it has proven to be an NP-hard problem (DasGupta 2000), and while studies have been done to try to improve and optimize an algorithm for accurately approximating this distance (Hon and Lam 2000), it is still computationally difficult and may not be effective for searching large databases for similar tree structures. The Updown distance metric is an improvement on these methods, but from the analysis that we will show, the implementation of the method does not scale well with regards to the size of the tree.

The MDS-Procrustes method presented here not only provides a high resolution similar to or better than the existing metrics, but also is able scale well with time when comparing reasonably sized trees, which is a necessity when searching large databases. Furthermore, MDS-Procrustes has the ability to compare trees of different size or with different nodes and also is able to handle weighted trees without effecting the computational time. Also, many of the topological techniques contain no natural extension to deal with weighted trees. Since weighted trees account for the edge lengths (evolutionary distance) between nodes, omitting trees containing this key evolutionary information can limit the effective use of any comparison method.

In Chapter 2, we will discuss our method in detail and then compare it to existing methods through the use of electronically available implementations of the above described approaches.

Chapter 2

Methods and Results

2.1 Introduction to MDS-Procrustes

Our MDS-Procrustes approach begins by generating a distance matrix directly from a given phylogenetic tree's alignment information. We then generate a high-dimensional Euclidean structure that represents the nodes as points in a high-dimensional space while still closely approximating their distance relationships via classical multi-dimensional scaling (MDS). For a comparison between two phylogenetic trees, this Euclidean structure will be generated for both trees and then we will superimpose one structure onto the other with a Procrustean approach. The degree of similarity between two trees is represented by a least-squares sum of deviations between corresponding point pairs. This distance approach should be an improvement to the topological approaches since it will not be as sensitive to the existence of particular branches in a tree when determining similarities.

In the proceeding sections, we will describe the steps involved with the method and use it on synthetically generated data to validate its effectiveness when the nearest tree within a set is known. We will then compare the distribution of similarity values and its time-efficiency to other available methods to show how it can be seen as an improvement to those existing approaches.

2.2 Classical Multidimensional Scaling

Consider two trees T_1 and T_2. T_1 contains a set of nodes N = $\{n_1,n_2,\ldots..,n_n\}$ and T_2 contains a set of nodes M = $\{m_1,m_2,\ldots.,m_m\}$. We generate distance matrices D_1(nxn) and D_2(mxm) such that d_{ij} for all i,j $\in N, M$ represents the patristic distance between node n_i -> n_j and m_i -> m_j respectively. This distance is computed following the paths of the branches connecting the nodes, and thus for weighted trees, this distance will include any weights assigned to an edge length. In order to be able to compare trees of different sizes and labels, at this point, we need to reduce and align the distance matrices D_1 and D_2 so that only the distances between nodes common to both trees are preserved. Figure 12 shows two trees and their corresponding distance matrices that result after removing nodes not common to both trees (These reduced matrices are referred to as R_1 and R_2).

After generating the distances between the nodes in each tree, we take these reduced distance matrices and represent them in a Euclidean space via classical MDS. To do this for each tree, we take the reduced distance matrix R and find the contrast matrix B such that B = $-\frac{1}{2}\, J\Delta^2 J$ where J is the centering matrix $I - n^{-1}\, 11'$(n is the number of nodes and 1 is a row vector of ones) and Δ is equal to the reduced distance matrix R. We then perform eigendecomposition of B = $Q\Lambda Q'$ to get the coordinate matrix (Borg and Groenen 2005). The solution is given by:

$$\mathbf{X} = \boldsymbol{Q_p \Lambda_p}^{\mathbf{1/2}} \qquad \textbf{(4)}$$

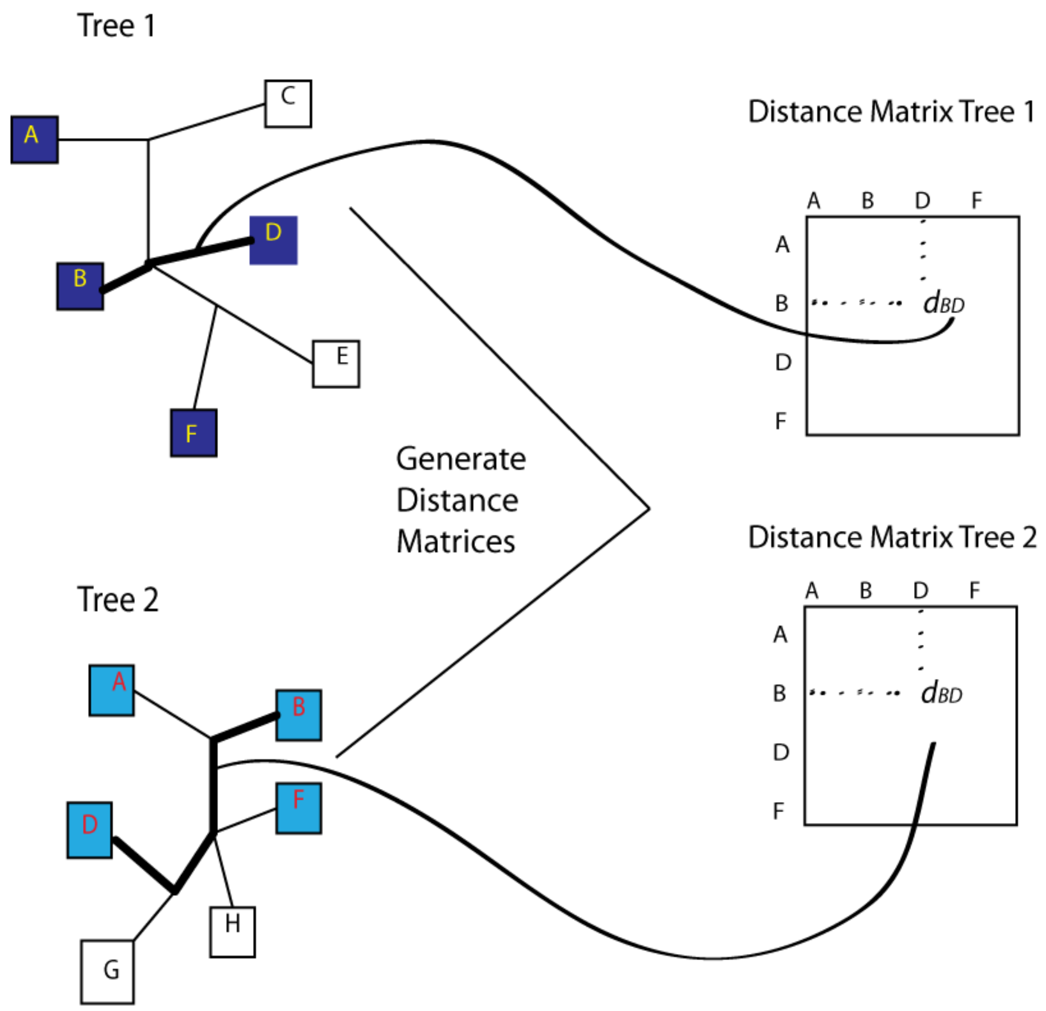


Figure 12 - Generating Distance Matrices

Generate distance values between each pair of nodes within a tree by following along the paths on tree. The matrices R_1 and R_2 are found by looking at only the nodes common to both trees in a comparison.

where Λ_p contains all positive eigenvalues p in Λ and Q_p contains the first p columns of Q. The negative eigenvalues can be removed since Δ is a Euclidean distance matrix and we can ignore the negative values as error. X is then a coordinate matrix that represents the distances between nodes in p dimensional space.

2.3 Procrustes Superimposition of Two Euclidean Structures

After obtaining the coordinate matrices X_1 and X_2 for each of the trees in a comparison, we perform a Procrustean superimposition on the matrices to determine their degree of similarity. This superimposition can be performed on two sets of points of the same size (prior to this, if X_1 and X_2 are not of the same dimensionality, we add columns of 0s to the appropriate matrix to align them) Y_1 and Y_2. Procrustes will compute the optimal linear transformation by minimizing $S = \|Y_1 T - Y_2\|$ where T is the transformation matrix. Here, Y_1 is superimposed onto Y_2 by means of applying C (translation), R (rotation and reflection) and B (scaling) to Y_2. This will be minimized when the following conditions are true:

$$\boldsymbol{R = UV', B = \frac{tr({Y_2}' Y_1 T R)}{tr({Y_1}' T Y_1)}\ , C = \frac{1(Y_1 - BY_2R)}{11'}} \qquad \textbf{(5)}$$

Where U and V are orthogonal matrices derived from the singular value decomposition of $T = U\Sigma V'$ and Σ is the diagonal matrix of singular values. Under these conditions, S represents the similarity score between trees T_1 and T_2 (Gower and Dijksterhuis 2004). Figure 13 shows a generic illustration of the process of taking a distance matrix from a phylogenetic tree and representing it in Euclidean space and then “fitting” these structures together via Procrustes superimposition.

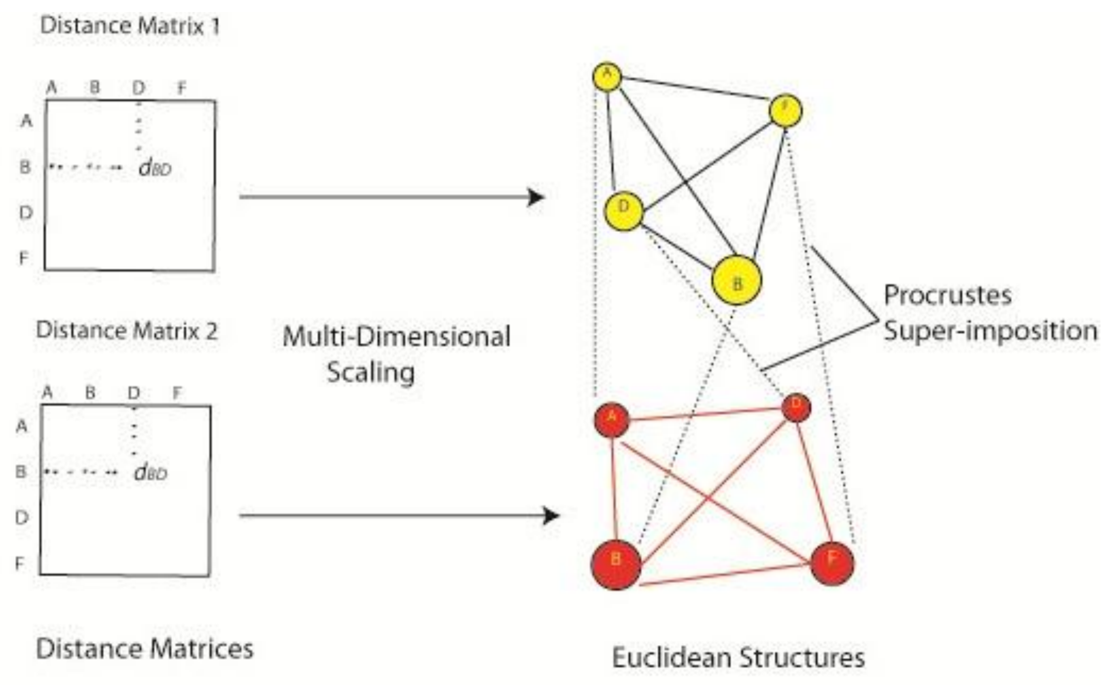


Figure 13 - Procrustes with Euclidean Structures

Take distance matrices from each tree and represent the nodes as points on a high-dimensional Euclidean structure which maintains the distance relationships provided in matrix. Then, use Procrustes super-imposition to align the two structures to one another to find a similarity score.

2.4 Implementation and Initial Testing

A program was written in Matlab to implement the above described MDS-Procrustes method. This program takes in a query and data file of phylogenetic trees in Newick format and outputs the MDS-Procrustes similarity scores from each query tree in the query tree file to each data tree in the data tree file. In order to test the effectiveness of our method, we initially compared a set of data trees with a set of query trees whose nearest matches to the data set are known.

To accomplish this, we generated a set of 40 unweighted, unrooted data trees $D = \{d_1,\ldots..,d_{40}\}$ with $N_{data} = \{n_1,\ldots..,n_{20}\}$ nodes. The trees were generated using the rtree function from the Ape package in R. We then selected a random $N_{query}=\{n_m,\ldots,n_{m+4}\}$ with $m \leq 16$ for each d in D to comprise our set of query trees $Q = \{q_1,\ldots..,q_{40}\}$. Thus, each query tree would always be a random 5 node subtree of a data tree. We then compared each $q \varepsilon Q$ to each $d \varepsilon D$ to test the ability of our method to correctly detect similarity between a tree and its subtree as a query structure.

In this scenario, MDS-Procrustes was able to correctly identify the most similar data tree as the structure from which the query tree was derived each time (with a distance equal to 0 on each instance). This is expected since we reduce the input distance matrices to include only distances between common nodes prior to MDS and Procrustes analysis, but it is important to show that the method will work for the trivial case and that the fitting will not yield any unforeseen results. We show the distribution of results of this random test in Fig 14. As seen in the figure, there were 40 queries that resulted in a distance of approximately 0 from the query tree to the data tree (each of these scores occurred in a comparison between the data tree and its subtree). The rest of the similarity score distribution was fairly evenly distributed and allowed for an effective ranking in this particular test scenario.

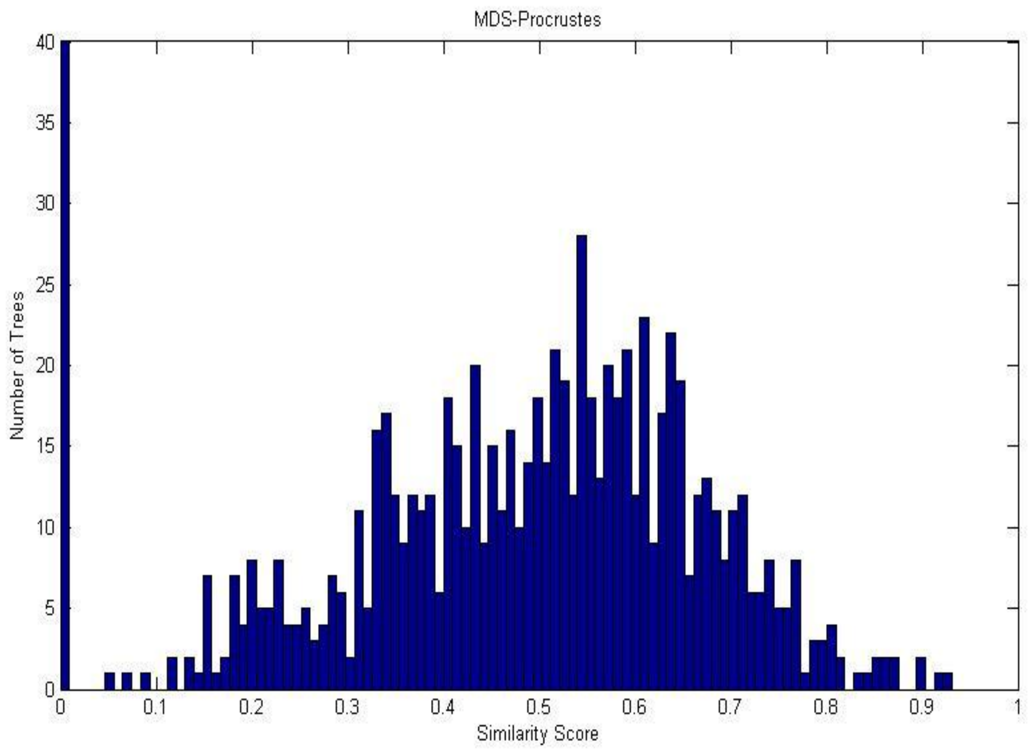


Figure 14 - MDS-Procrustes distribution for validation dataset

Similarity score distribution for random data set of 40 weighted trees with 20 nodes. Query tree used was a random 5 node subtree of each given data tree in the comparison. The 40 100% similarity matches represent the situation when the query tree was compared to the data tree from which it originated.

We further analyze the method by looking at a specific iteration of the program. Figures 15-19 show the results of a particular query in this test. Figure 15 shows the original five node query tree that was compared to all of the trees in the data set. Next, we show the top two ranked trees in terms of similarity from the data set to this query tree. Figure 16 shows the full tree of the top ranker and then figure 17 shows the subtree of common leaf nodes that was used as a basis for comparison between this tree and the query tree. As previously noted, the top ranked tree was the original tree that the five node subtree originated from and it can be observed visually from looking at the subtrees in Figures15 and 17 that these trees matched identically in this particular query. Figures 18 and 19 show the second ranked full tree and its subtree used in the comparison with the query tree from figure 15. The similarity score for this comparison was 0.1796, a relatively high degree of similarity. As illustrated in the figures, while not identical in structure, these two trees are actually highly related in terms of the distances between their common nodes. This is an important distinction to make with regards to a fundamental difference between topological searches and distance-based searches, as topological searches may not be able to resolve the fact that these two trees are similar since the nodes were in different positions in the two trees. However, here, we measure similarity as a function of the distance between the nodes, so the position of the node in the original tree is not of specific consequence, only its position relative to the other nodes is relevant. Being overly sensitive to the original location of particular taxa in a given phylogenetic tree can cause many trees of similarity to be considered dissimilar and is a somewhat limiting aspect to many topologically based tree comparison methods.

In the following sections, we further our analysis by showing how MDS-Procrustes compares to the other existing, topological tree comparison methods. To do this, we will use two separate synthetically created data sets and run the different methods on this data to compare the similarity score distributions.

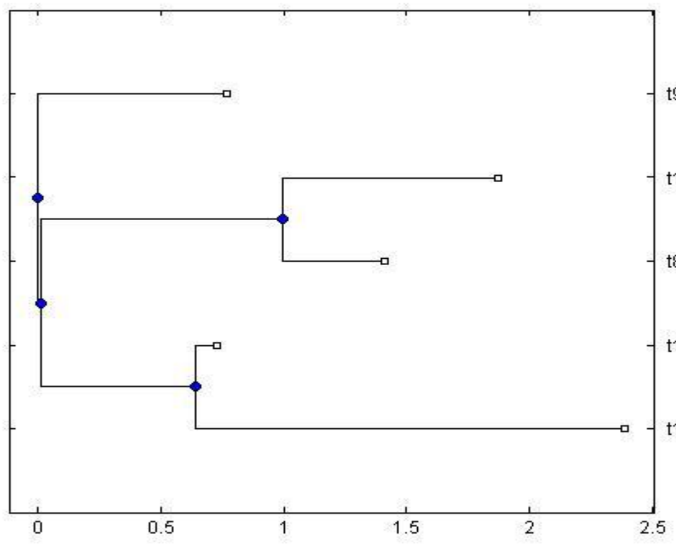


Figure 15 – Random Search Query

Query Tree used in random tree search and compared to 40 different randomly generated trees.

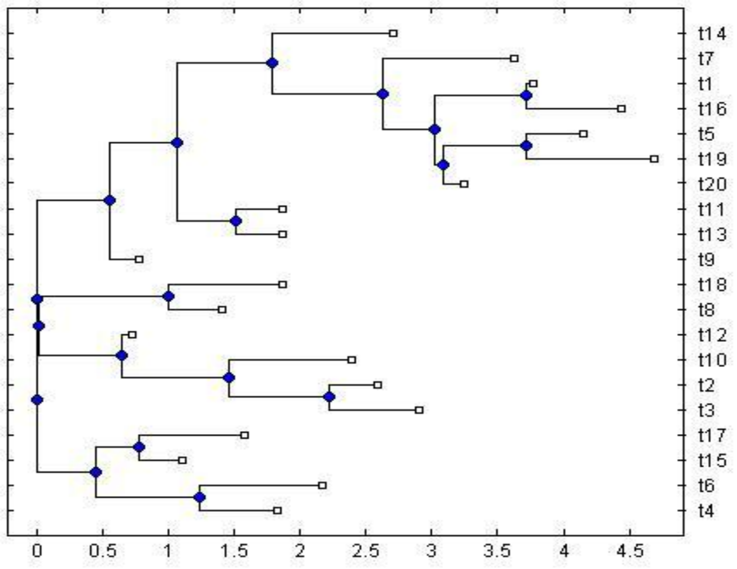


Figure 16 – Random Search Top Ranker

Most similar data tree to query tree from Fig. 15 using MDS-Procrustes in comparison to 40 different trees.

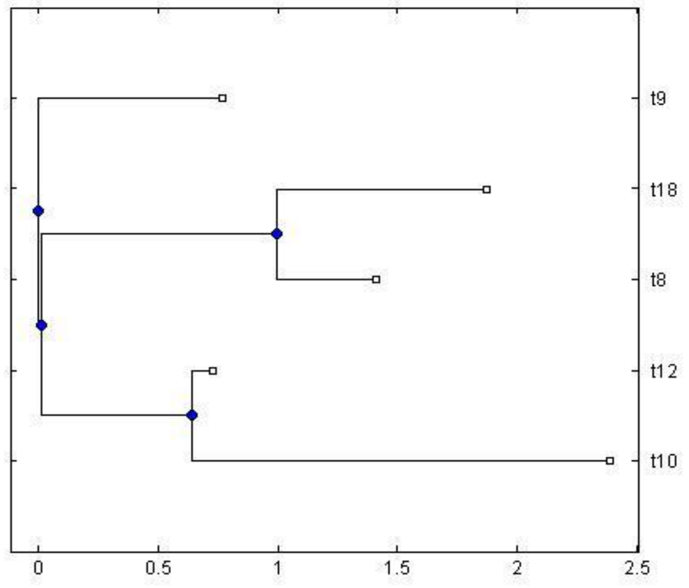


Figure 17 – Random Search Top Rank Subtree

Subtree used in MDS-Procrustes comparison between trees from Fig 15 and Fig 16. It can be easily seen that this subtree matches up exactly with Fig 15, thus the 100% similarity.

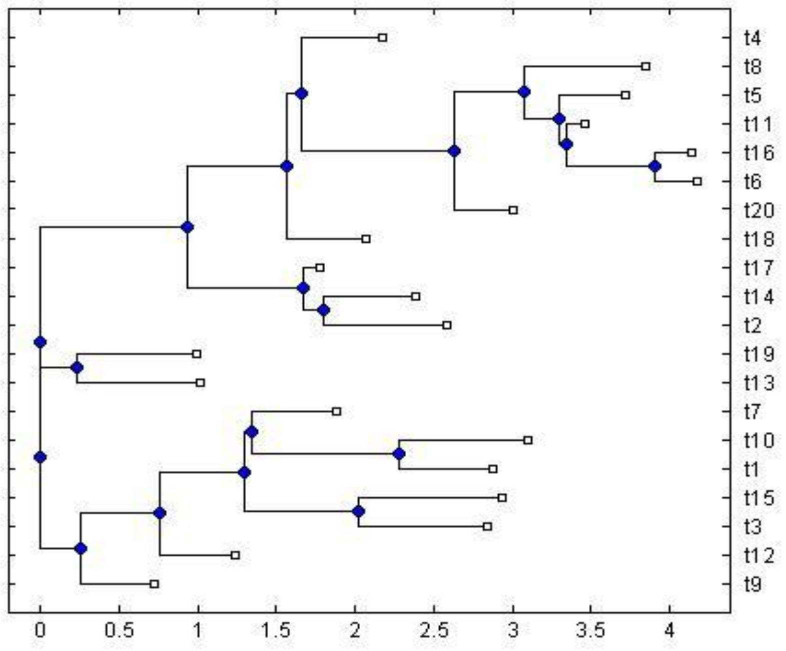


Figure 18 – Random Search Second Ranker

Second most similar data tree to query tree from Fig 1 using MDS-Procrustes in comparison to 40 different random trees.

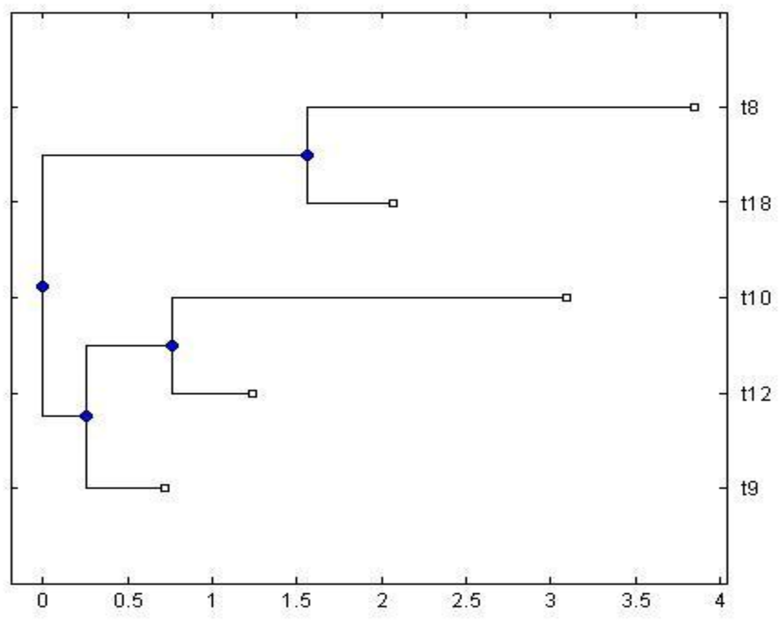


Figure 19 – Random Search Second Ranker Subtree

Subtree used in MDS-Procrustes between trees from Fig 1 and Fig 5. The MDS-Procrustes distance between these trees was found to be 0.176 and this similarity can be observed from the pictures by looking at the similarity in the distances between the sets of nodes.

2.5 Data Sets

In our analysis, we will take a similar approach to the work performed by Wang et al.(2005a) and show how MDS-Procrustes compares to the previously described methods. Initially, we will use synthetically created data to compare the distribution of similarity scores between the methods and then also look at the efficiency with respect to tree size for each method to compute the similarity scores for a given set of data trees. Here, we use the Component tool (Page 2001) to implement the NNI, PAR, MAST and Quartet metrics. The USim (UpDown) metric was implemented using a Java program from Wang et al. (2005b) and was slightly modified to be able to completely analyze the same data set that we use as a basis for comparison of the different methods. As indicated in the previous section, we use Matlab code to perform all comparisons with MDS-Procrustes.

Our method-comparison experiment used two separate sets of data trees to comprise sample databases of trees. In the initial set, the database consists of 300 unrooted, labeled trees with 8 nodes. We compared the distances/similarities among all of the trees in the given set and output the results into histograms to view the similarity score distribution of each method. A wide distribution of scores is needed to be able to effectively rank trees, otherwise several tree comparisons can generate the same score(s) and a method will not be able to distinguish between them in terms of similarity. The second database of trees consisted of 300 unrooted, labeled trees with 20 nodes. As previously noted, several methods can offer good score distribution when comparing trees with a large number of nodes, and thus we compared the similarity score outputs on trees of differing sizes to analyze this further (as searching large databases of trees effectively would optimally require resolution for both large trees and small trees).

2.6 Method Metric Distributions for Initial Data Set

Figures 20-25 show the distribution of similarity scores (metrics) for each of the described methods for the initial set of data. As theorized previously, PAR and MAST offer very low resolutions in this situation, each method only able to bin all of the comparisons into six separate bins. This would make it ineffective to use these metrics as a basis for ranking trees in terms of their similarity to a given query tree since a large number of trees would contain the same degree of similarity. NNI and Quartet offer a better distribution of scores in this case, but both distributions are not as good as those provided by the Updown and MDS-Procrustes methods shown. This is an important piece, because while we expect all methods to generate a wider distribution of scores with datasets containing larger trees (since more node/distance information is known), it shows that even with small trees, MDS-Procrustes is at least as effective as any of the tested metrics (and in many cases, it offers significantly improved score distribution).

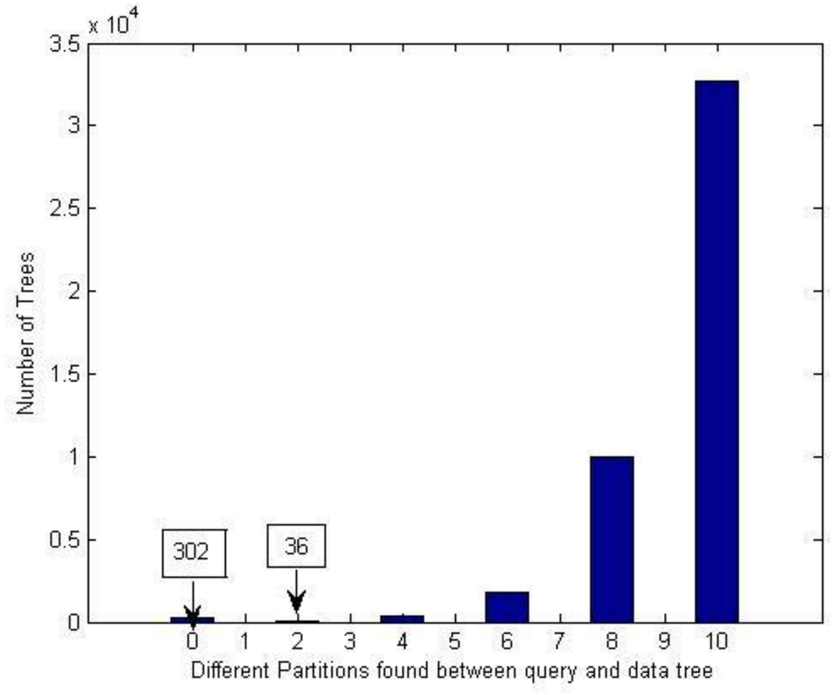


Figure 20 – PAR distribution for initial dataset

Distribution of Partition metric values for comparison of tree distances among 300 8 node trees.

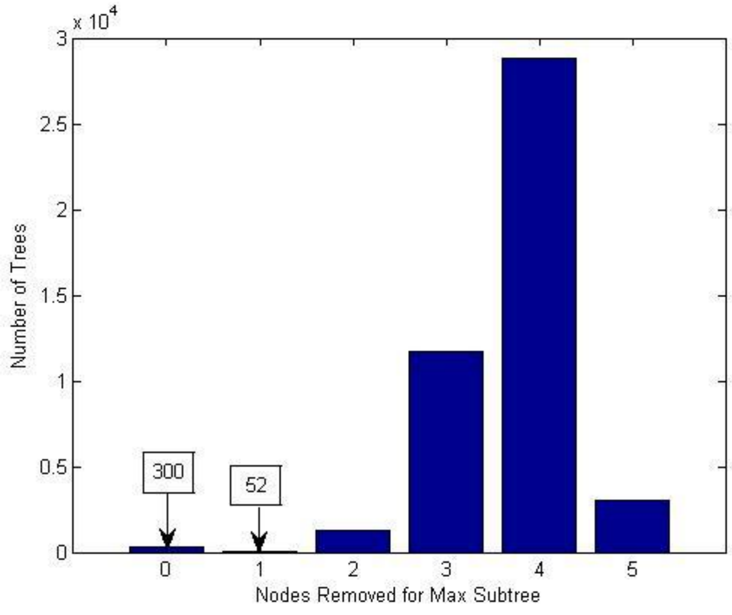


Figure 21 – MAST distribution for initial dataset

Distribution of MAST metric values for comparison of tree distances among 300 8 node trees.

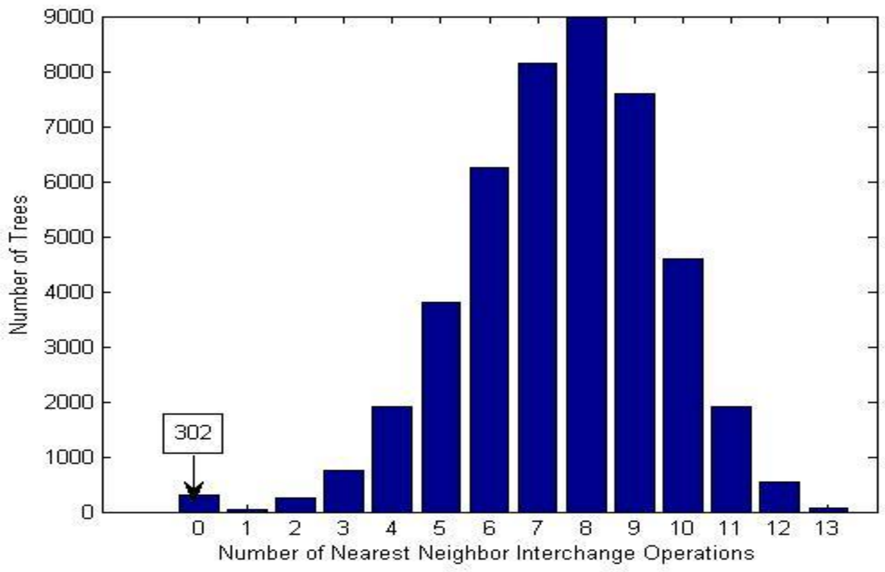


Figure 22 – NNI Distribution for initial dataset

Distribution of NNI metric values for comparison of tree distances among 300 8 node trees.

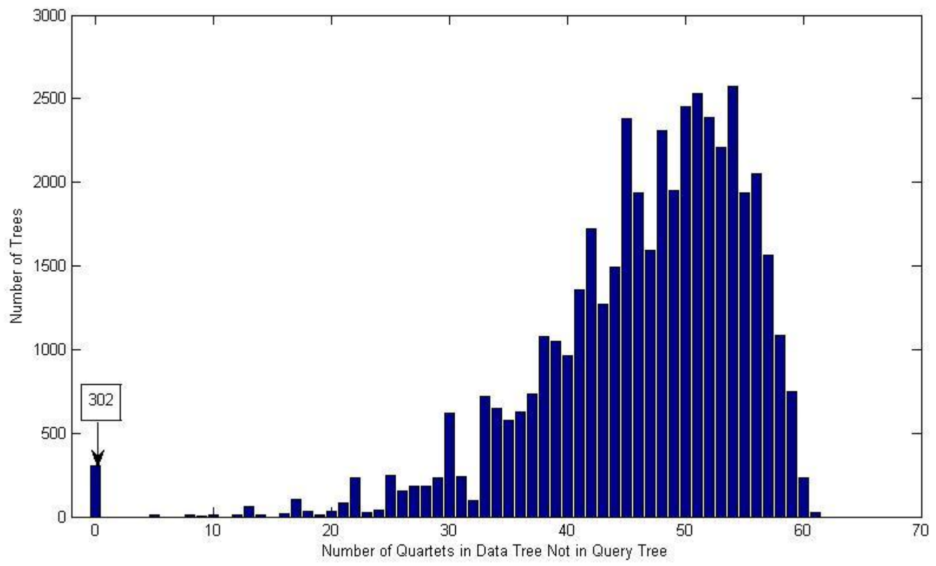


Figure 23 – Quartet distribution for initial dataset

Distribution of Quartet metric values for comparison of tree distances among 300 8 node trees.

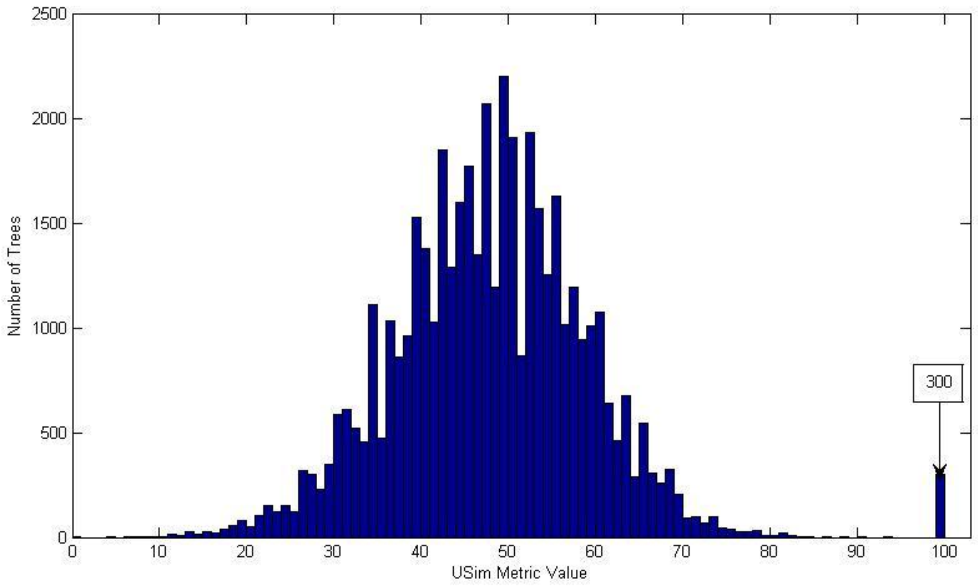


Figure 24 – Usim distribution for initial dataset

Distribution of Updown Method metric values for comparison of tree distances among 300 8 node trees.

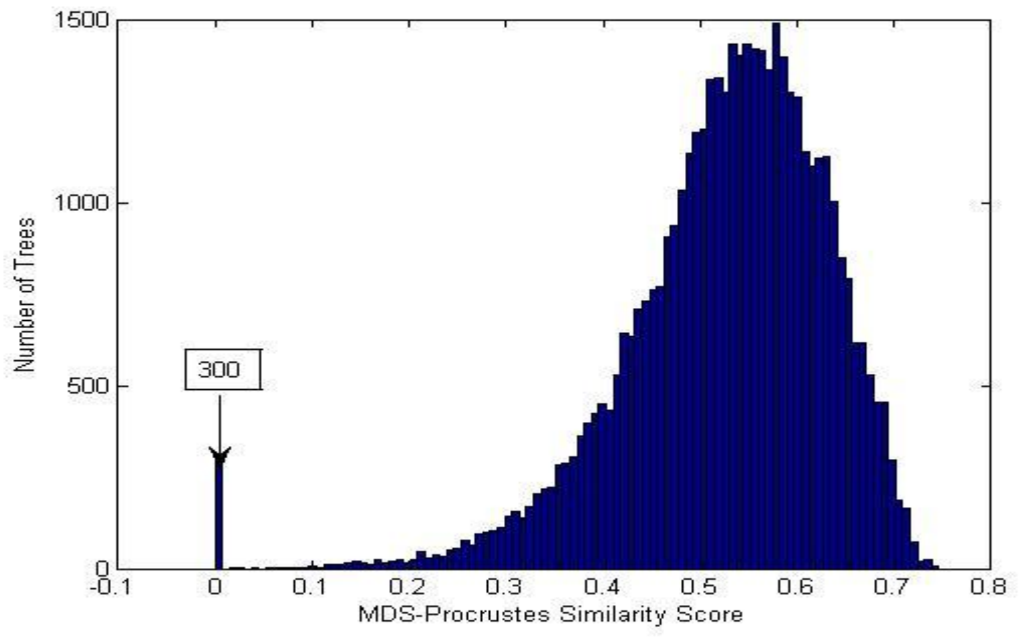


Figure 25 – MDS-Procrustes distribution for initial dataset.

Distribution of MDS-Procrustes metric values for comparison of tree distances among 300 8 node trees.

Another aspect to consider in measuring the effectiveness of each method is how often the methods find an exact match in similarity to the data tree. Since our dataset had 300 different randomly generated trees, we would expect to see only 300 exact matches (only when the query tree was compared to itself out of the dataset). However, this was not the case in NNI, Quartet, or PAR. Each of those methods found 302 exact matches, indicating that there were two sets of trees that were found to be 100% similar even though they were not exactly the same tree.

To further illustrate this error, we show two trees (Trees 72 and 177 shown in Figures 26 and 27 on next page) that were found to be exactly similar according to the NNI, Quartet, and PAR metrics. MDS-Procrustes viewed the trees as different with a similarity score of 0.2521. For Tree 72, this represented the 7th most similar tree out of the entire dataset of 300 trees and for Tree 177, this represented the 9th most similar tree. Thus, while MDS-Procrustes does detect a relatively high degree of similarity between the trees (as would be expected), it does not detect them as 100% similar. Updown and MAST also detect the trees as different, with varying degrees of similarity.

In this particular example, the figures show that the trees actually have the same sets of nodes directly connected to one another, however the relation of these sets to one another is different and thus the trees are not the exact same. NNI, Quartet and PAR are unable to resolve these differences since each of those methods essentially look only at relationships at interior edges and do not look at the relation of each node individually to the entire tree. So, while the presence of this type of error decreases as the trees get larger, it is clear that there are some minor resolution issues with comparing trees of relatively small size with those methods. This also further demonstrates the effectiveness of MDS-Procrustes in terms of ranking trees according to similarity as it is able to resolve minor differences between trees as good or better than any of the existing topological methods.

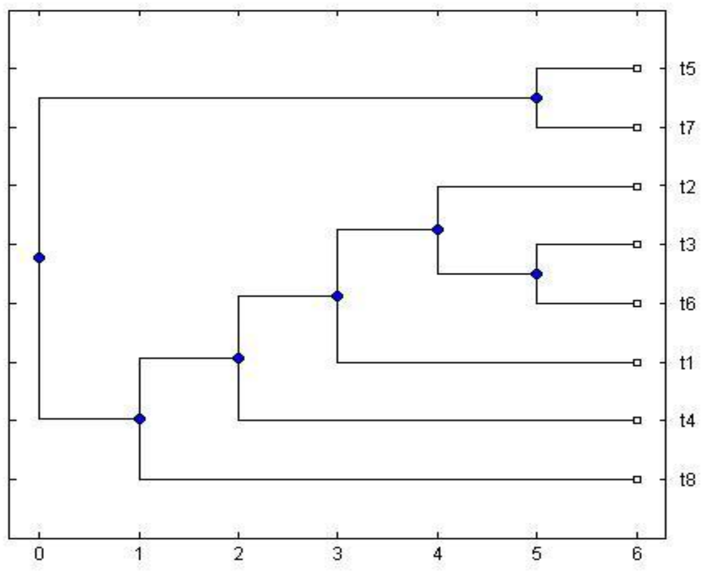


Figure 26 – Exact Match Tree 1.

Tree diagram for Tree that is exact match to Figure 21according to NNI, Quartet, and PAR metrics but not according to MDS-Procrustes.

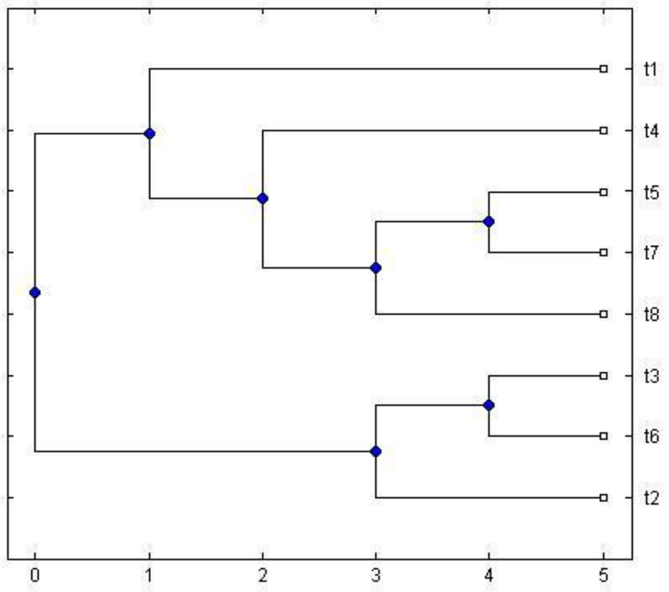


Figure 27 – Exact Match Tree 2.

Tree diagram for Tree that is exact match to Figure 20 according to NNI, Quartet, and PAR metrics but not according to MDS-Procrustes.

2.7 Method Metric Distributions for Second Data Set

Figures 28-33 show the distribution of scores for the second set of data. The MAST and PAR metrics showed no significant improvement in score distribution with the increase in size, while both NNI and Quartet showed a considerable improvement in this regard. This is expected in both cases as the number of possible interchanges/subsets greatly increases with the addition of nodes and thus there are many more possible comparisons between the trees. In this test case, Quartet, Updown, and MDS-Procrustes each generate a good enough score distribution to be able to rank trees by degree of similarity from this data set.

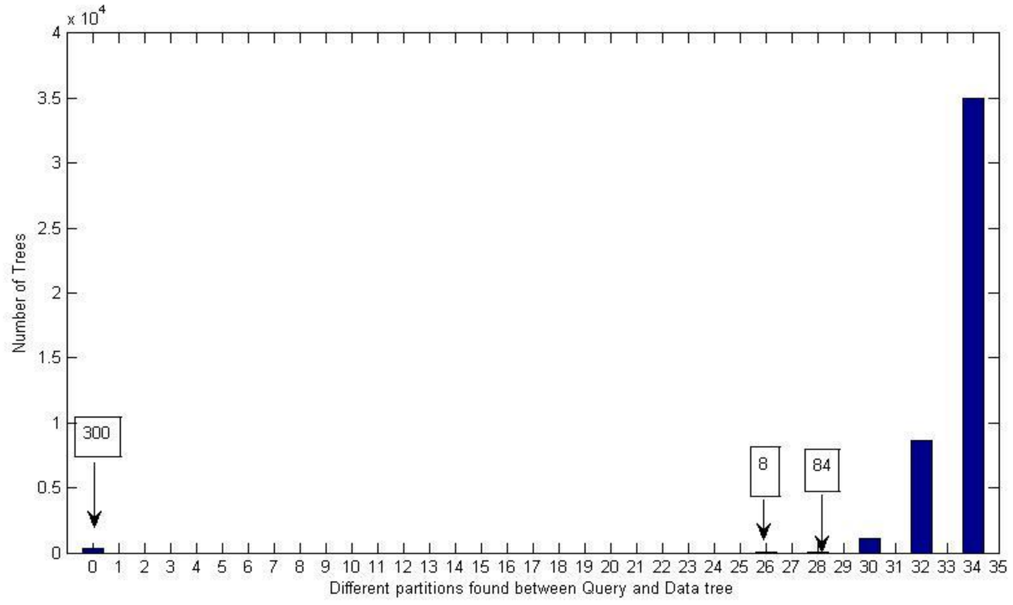


Figure 28 – Par Distribution for second dataset

Distribution of Partition metric values for comparison of tree distances among 300 20 node trees.

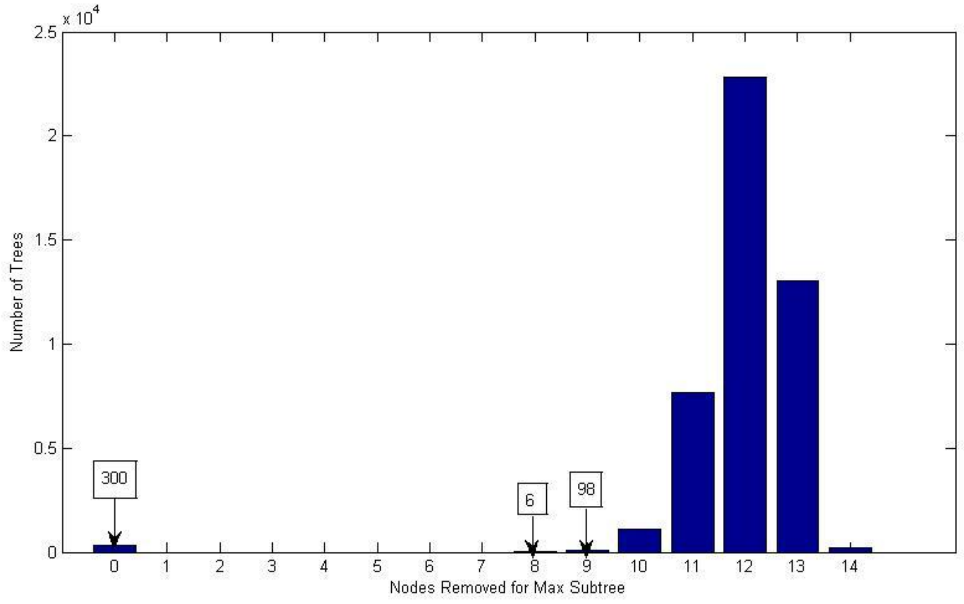


Figure 29 – MAST distribution for second dataset

Distribution of MAST metric values for comparison of tree distances among 300 20 node trees.

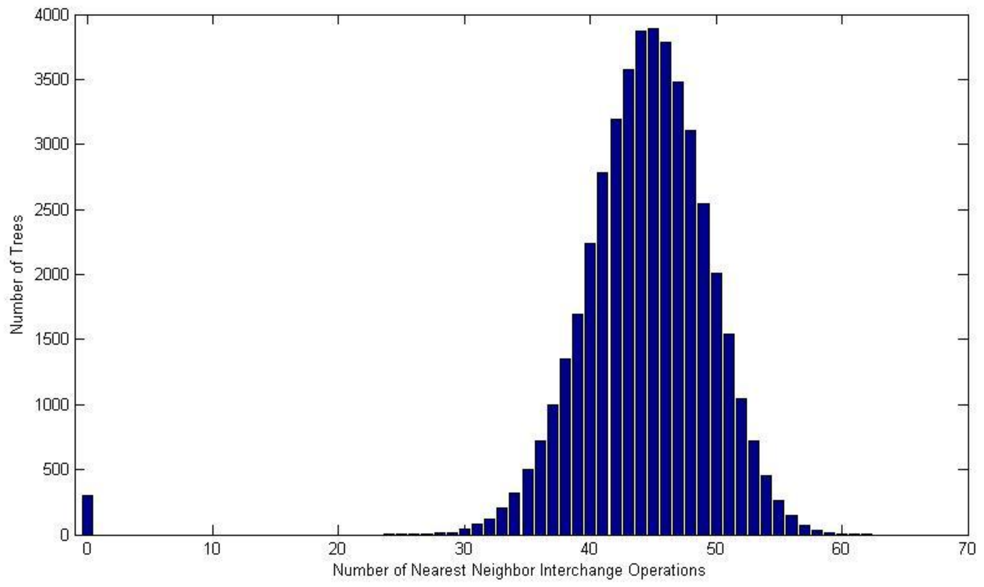


Figure 30 – NNI distribution for second dataset

Distribution of NNI metric values for comparison of tree distances among 300 20 node trees.

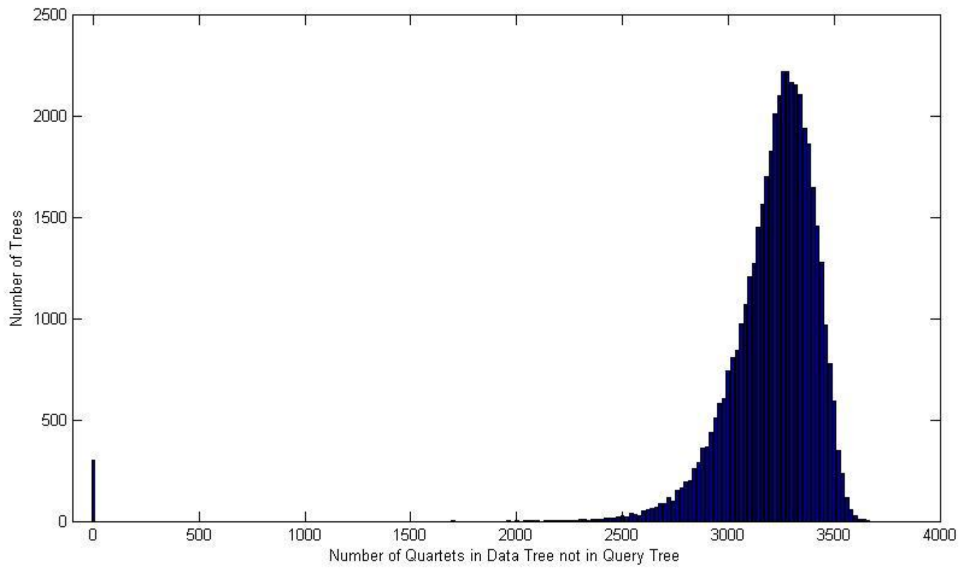


Figure 31 – Quartet distribution for second dataset

Distribution of Quartet metric values for comparison of tree distances among 300 20 node trees.

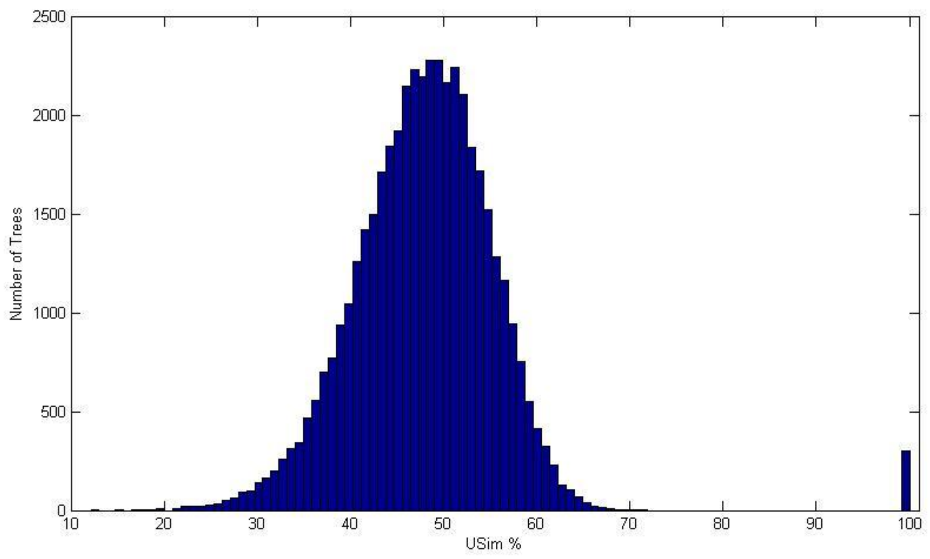


Figure 32 – Updown Distribution for second dataset

Distribution of Updown method metric (USim) scores for comparison of tree distances among 300 20 node trees.

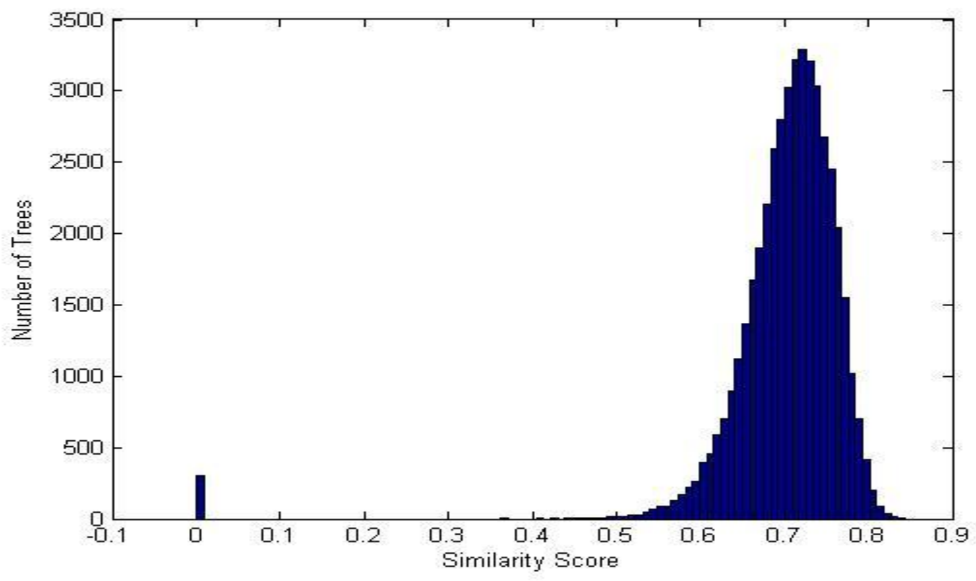


Figure 33 – MDS-Procrustes distribution for second dataset

Distribution of MDS-Procrustes metric values for comparison of tree distances among 300 20 node trees.

Furthermore, it can be seen that with a greater number of nodes in the trees, the occurrence of a tree finding another tree with 100% similarity to itself is eliminated in all methods, as the only comparisons resulting in trees being 100% similar occur when the tree is compared to itself . In combining the results here with the results from the initial data set, it is clear that only MDS-Procrustes, USim, and Quartet offer good enough resolution to effectively rank similar trees in both the small and larger tree scenarios.

2.8 Method Computation Performance with Varying Tree Size

Since tree databases can often contain trees very large in size, time scaling with respect to the size of the tree is an important aspect of any comparison method. With many topological searches, the increase in time with respect to the size of the tree is large, as the presence of additional nodes/branches presents many more possible solutions and thus requires a great deal of computational time to process (which can offset the resolution improvement provided by the inclusion of more nodes). We will initially provide the reported time complexity for each of the methods used and then perform runtime tests with trees of varying sizes to see which method(s) perform best with regards to time when varying the tree size.

The partition metric is computationally fast and as implemented here, the time complexity is O(n) (Critchlow 1996). This is to be expected since the method simply lists all partitions found in both trees and then compares them together to determine the degree of similarity between trees. However, this comes at a cost as the resolution of distribution scores for this metric was particularly poor. Another method that had a poor distribution of similarity values was the MAST method. However, unlike the par metric, the MAST approximation used here does not scale linearly with regards to tree size. The time complexity of the method used here was O($n^{\log 2\ n}$), which is considerably higher than that of the partition metric (Kubicka 1995).

The Quartet method used here can compute the quartet distance in O(n^3) time, which is the same computational complexity of the proposed MDS-Procrustes algorithm. Also, as previously stated, the NNI distance between two trees cannot be computed in polynomial time, however the approximation used here computes this distance by rooting each tree at each node and transforming one tree into the other beginning at each of these points and then determining the minimum nni distance by taking the minimum of these values. This approximation requires O($n \log_{10} n$) time to compute for rooted trees and O($n^2 \log_{10} n$) for unrooted trees (Brown and Day 1984). The computational complexity for the Updown distance metric is O(n^2+ m) where n is the number of nodes in the query tree and m is the number of nodes in the data tree (thus, in all comparisons in this paper since we used trees with the same nodes, the complexity would be n^2 + n).

Beyond looking at the computational complexity of each algorithm, we now want further analyze each method's runtime as a function of tree size here. In this test, we looked at 100 unrooted and unweighted trees, ranging in size from five nodes to fifty nodes. For each run, we compared each tree to every other tree in the set and calculated the run-time in seconds of the total search of 4950 comparisons. All programs were run on a Windows Lenovo ThinkCentre workstation with an Intel Core 2 2.13 GHz CPU.

Figures 34 and 35 show the results of these tests. Figure 34 shows the raw time in seconds with respect to the tree size of each method, while figure 35 shows a log-log scaled plot of the same data. It is easily seen from figure 34 that only two of the methods scaled well with regards to the tree sizes tested here (PAR and MDS-Procrustes) and we observed an immediate and near exponential increase in the Quartet, MAST, and NNI methods. This is to be expected since the number of operations needed to calculate these distances increases greatly with the size of the tree. Updown performed slightly better than MAST, Quartet and NNI, but not as well as MDS-Procrustes. The PAR method saw no significant change in computational time when comparing varying sizes of trees, but its limitations with respect to distribution scores make it highly limited in its effectiveness in a comparison among large sets of trees. MDS-Procrustes performed well in both providing a highly-resolved distribution of similarity scores while also maintaining a relatively fast runtime with respect to the tree size, a highly desirable quality when searching through tree databases with large number of trees of varying nodes/sizes. It should be noted, that we would expect the MDS-Procrustes method time complexity to grow similar to the other topological methods at some point in time (when the size reaches a certain point), since their computational complexities are of the same or similar order. However, due to initial bias in the implementations of the methods, MDS-Procrustes did not begin displaying that time increase at the size that we reached here and while tests can be run on trees of greater size it is important to see how this method performed on moderately sized trees relative to the other methods.

Figure 35 further illustrates the time scaling of the different methods tested. The slope of a log-based plot should show the degree of polynomial increase and so we would expect that the method with the greater slope in this graph would have the higher scaling factor in time. From looking at the figure, we can see that the slopes of the lines do mostly correspond with what we would expect from the given time-scaling factors reported in the literature with the exception of MDS-Procrustes (and again, we expect that once we get to large enough trees, the slope of the log-based line would closely approximate that of the Quartet method).

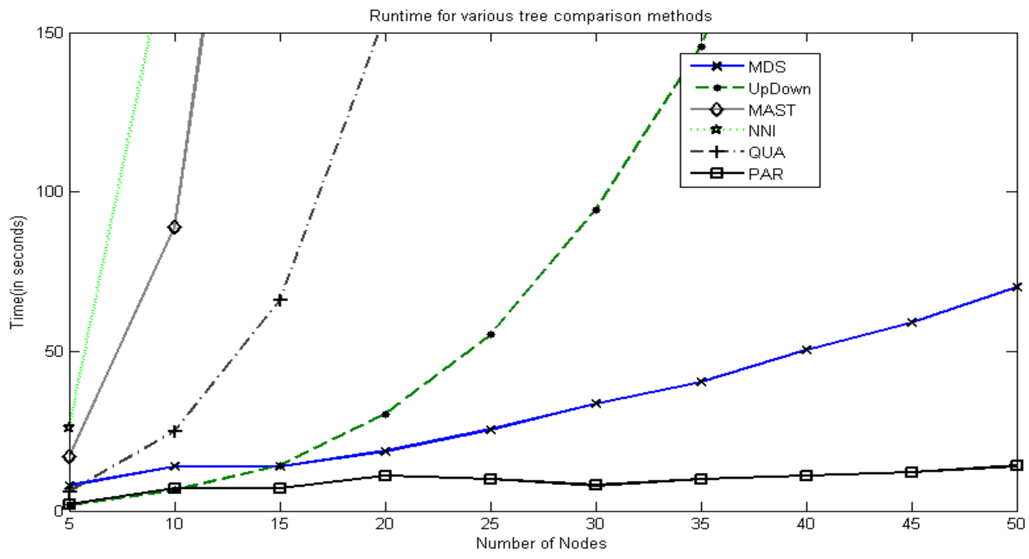


Figure 34 – Runtime Comparison

Run times for each described method in comparison of similarity among 100 unweighted and unrooted trees. The tree size varied from 5-50 nodes.

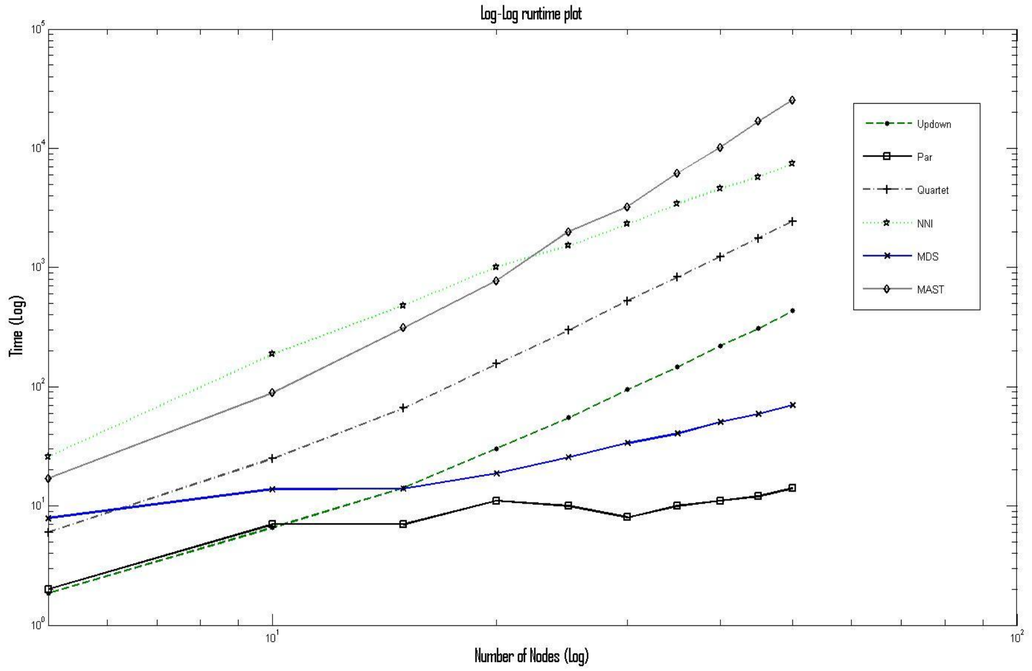


Figure 35 – Log Run Time Comparison

Log based scale of Figure 34.

Chapter 3

Conclusion and Future Work

3.1 Comparison Method Conclusion

In this work, we have introduced a novel method for comparing phylogenetic trees by viewing them as Euclidean structures in a high dimensional space and fitting them with a Procrustean approach. We have shown through the use of several sets of synthetic data that the application of this method produces a wide similarity score distribution making it effective in resolving tree similarities among different trees in a database. While other methods exist to achieve this same goal, we have seen that MDS-Procrustes offers a good distribution of similarity scores and time performance that is unmatched by several existing methods found in the literature.

In addition to the high resolution and good computational performance of our MDS-Procrustes implementation, another consideration is its ability to compare trees of different sizes. In our method, we are able to compare trees with different nodes and of different sizes effectively provided they have at least a certain number of nodes in common (if less than four common nodes exist, the comparison is not effective and the trees are considered not similar). This is an improvement over many of the existing methods described here as several of them (PAR, NNI, Quartet) are unable compare trees if they do not have the same labeled nodes (Wang 2005a). The Updown and MAST methods are able to effectively compare trees which contain different nodes.   This is a desirable feature to have in a phylogenetic tree comparison algorithm as tree databases will have thousands of trees with different species/nodes and being limited to searching only those that contain the same sets of leaves would result in a limited application of any proposed method.

Also, here we offer a method that contains an inherent extension for the comparison of weighted trees, an area in which many existing methods fail to address. Two of the methods discussed, NNI and Updown, contain extensions for including weighted trees. For NNI, an operation is charged a cost equal to the weight of the branch associated with it and while this has been proven effective, it adds to the computational time of a particular comparison (Hon and Lam 1999). Wang et al. presented an extension for supporting weighted trees in their Updown approach; however this was not made available in their Java implementation for further study and evaluation here. PAR, MAST, and Quartet do not have natural extensions towards the support of weighted trees and thus cannot utilize any evolutionary distance information to refine their comparisons. Since we provide a method that generates a distance matrix directly from the node information provided, MDS-Procrustes inherently computes weights with no added computational time to process the comparison (and we’ve already shown that our implementation is time efficient to other methods even in the unweighted case).   Given the number of different algorithms available to generate these types of phylogenetic trees, this added functionality is another feature that allows for the proposed method to be useful in the comparison of many varied trees.

It is also important to note that there are general advantages to using some of the other tree comparison metrics over MDS-Procrustes. For example, finding the consensus tree (MAST) between sets of trees can be very informative and so while MAST might not be useful in the case of comparing against a large set of trees for similarity, in a strict one to one comparison, this comparison generates a very specific meaning for its similarity measure (as the presence of a common subtree can have biological significance). MDS-Procrustes generates a similarity score based on the distances between nodes and how similarly those distances are resolved in both trees, but that distance does not contain a specific meaning, so in a one-to-one comparison, the MDS-Procrustes distance is not particularly useful other than to give a general indication of similarity. So, while MDS-Procrustes has been shown here to be effective in determining similarity when comparing a tree to a large set of trees and ranking the trees from that set by degree of similarity, some of the other topological methods (specifically MAST, Quartet and Partition) might be more useful when comparing one tree to another to determine exactly how the trees are similar and derive meaning from that similarity.

3.2 Future Work

The work here has shown MDS-Procrustes to potentially be very useful for the purposes of comparing a phylogenetic tree to a database of phylogenetic trees and being able to rank the results by degree of similarity. Here we used various forms of synthetically created data to create artificial databases and determine the effectiveness of the method(s), and since MDS-Procrustes performed well in comparison to other methods in this case, the next step would be to actually implement the method on one of the many phylogenetic tree databases in existence today (such as TreeBase).

The likely approach would be to build a website that allows for a user to input a phylogenetic tree for which a similar tree is desired. The site would then connect to a database and use the method outlined here to rank trees by degree of similarity to the input tree. There would have to be some thought put into exactly how to visualize the results, but implementing the method in this manner could potentially provide a useful resource to be used by other researchers in their studies.

Also, as detailed previously, while we studied only phylogenetic trees in this paper, this methodology could prove to be useful in more generic forms of database searching. Any method that is able to quickly and effectively rank results with a high degree of resolution given even a relatively small number of nodes (comparison data points), can prove to be very useful in many types of database search and predict algorithms. Many forms of data (Chemical elements, favorite movies of a particular user, etc.) can be represented in the same form as phylogenetic trees are here (high dimensional Euclidean structures) and thus this method might prove to be useful in searching databases containing that information and ranking results by degree of similarity. So, while this research focused on the area of phylogenetic tree comparison, it is clear that MDS-Procrustes can potentially benefit not only researchers in that area, but in many others as well.

References

[1] Borg I and Groenen P. 2005. *Modern Multidimensional Scaling Theory and Applications, 2nd edition*. Springer Science+Business Media, Inc. New York, NY.

[2] Brown E and Day W. 1984. A computationally efficient approximation to the nearest neighbor interchange metric. *Journal of Classification,* 1(1), 93-124.

[3] Bryant D, Tsang J, Kearney P and Li M. 2000. Computing the quartet distance between evolutionary trees. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms*.

[4] Choi K. and Gomez S. 2009. Comparison of phylogenetic trees through alignment of embedded evolutionary distance. *BMC Bioinformatics 2009,* 10:423.

[5] Crichlow D, Pearl D, Qian C 1996. The triples distance for rooted bifurcating phylogenetic trees. *Syst. Biol*., 45(3), 323-334.

[6] Currie C, Wong B, Stuart A, Schultz T, Rehner S, Mueller U, Sung G, Spatafora J, Straus N 2003. Ancient tripartite coevolution in the attine ant-microbe symbiosis. *Science* 2003, 299,386-388.

[7] DasGupta B, He X, Jiang T, Li M, Tromp J. 2000. On computing the nearest neighbor interchange distance. In *Proc. DIMACS Workshop on Discrete Problems with Medical Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* Am. Math. Soc., 55, 125-143.

[8] Day W. 1983. Properties of the Nearest Neighbor Interchange Metric for Trees of Small Size. *J. Theor Biol.*, 101, 275-288.

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

[10] Felsenstein J. 2003. *Inferring Phylogenies*. Sinauer Associates, Inc., Publishers, Sunderland, MA.

[11] Gordon A. 1979. A measure of the agreement between rankings. *Biometrika*(66), 7-15.

[12] Gower J and Dijksterhuis G. 2004. *Procrustes Problems*. Oxford University Press Inc.,Oxford, NY

[13] Hon W and Lam T. 1999. Approximating the Nearest Neighbor Interchange Distance for Evolutionary Trees with Non-Uniform Degrees. *COCOON '99*, 1627, 61-70.

[14] Keselman D and Amir A. 1994. Maximum agreement subtree in a set of evolutionary trees: Metrics and efficient algorithms. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, 758-769.

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

[16] Lee C, Hung L, Chang M, Shen, C, Tang C. 2005. An improved algorithm for the maximum agreement subtree problem. *Information Processing Letters*, 94(5), 211-216.

[17] MacLeod D, Charlebois R, Doolittle F, Bapteste E 2005. Deduction of probable events of lateral gene transfer through comparison of phylogenetic trees by recursive consolidation and rearrangement. *BMC Evolutionary Biology* 2005, 5:27.

[18] Page, R. 2001. http://taxonomy.zoology.gla.ac.uk/rod/cpw.html.

[19] Pazos F. and Valencia A. 2001. Similarity of phylogenetic trees as indicator of protein-protein interaction. *Protein Engineering,* 14(9), 609-614.

[20] Robinson D and Foulds L. 1980. Comparison of Phylogenetic Trees. *Mathematical Bioscience*, 53, 131-147.

[21] Saitou N and Nei M. 1987. The neighbor-joining method: A new method for reconstructing phylogenetic trees. Molecular Biology Evol, 4(4), 406-425.

[22] Sanderson M, Donoghue M, Piel W and Eriksson T. 1994. TreeBASE: A prototype database of phylogenetic analyses and an interactive tool for browsing the phylogeny of life. *American Journal of Botany*, 81(6), 183.

[23] Steel M and Penny D. 1993. Distributions of tree comparison metrics – some new results. *Systemic Biology*, 42(2), 126-141.

[24] Stissing M, Pedersen C, Mailund T, Brodal G and Fagerberg R. 2006. Computing the quartet distance between evolutionary trees of bounded degree. In *Proceedings of the 5th Asia-Pacific Bioinformatics Conference*, 101-110.

[25] St John K, Warnow T, Moret B and Vawter L. 2003. Performance Study of Phylogenetic Methods: (Unweighted) Quartet Methods and Neighbor-Joining. *Journal of Algorithms*, 48(1), 173-193.

[26] Wang J, Shan H., Shasha D., Piel W. 2005a. Fast Structural Search in Phylogenetic Databases. *Evolutionary Bioinformatics Online 2005,*1, 37-46.

[27] Wang J, Regula S, Shasha D, Hua L and Patel R. 2005b. http://web.njit.edu/~wangj/updown.htm.