2010-08-04

Partition Based Phylogenetic Search

Kenneth A. Sundberg
Brigham Young University - Provo

Partition Based Phylogenetic Search

Kenneth A. Sundberg

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Mark Clement, Chair
Quinn Snell
Dan Ventura
Scott Woodfield
Parris Egbert

Department of Computer Science

Brigham Young University

December 2010

ABSTRACT

Partition Based Phylogenetic Search

Kenneth A. Sundberg

Department of Computer Science

Doctor of Philosophy

Evolutionary relationships are key to modern understanding of biological systems. Phylogenetic search is the means by which these relationships are inferred. Phylogenetic search is NP-Hard. As such it is necessary to employ heuristic methods. This work proposes new methods based on viewing the relationships between species as sets of partitions. These methods produce more parsimonious phylogenies than current methods.

Keywords: phylogenetics, cartographic projection, partition space, parsimony, tree mixing

Contents

List of Figures

Chapter 1

Introduction

Phylogenetic search is the problem of inferring evolutionary relationships between organisms. This is of interest to many subfields of biology such as biogeography [15], epidemiology [12, 49], and viral transmission [13, 26].

In one case [42] phylogenetic search was used in court to show the transmission of HIV from a dentist to several patients. As an example consider a simplified set which consists of samples from a dentist, two patients, a representative of the general Florida HIV population, and a representative from the US HIV population. With five samples there are fifteen possible relationships. Figure 1.1 shows three of these relationships. In Figure 1.1a the patients are most closely related to the dentist, implying that he was the source of infection. In Figure 1.1b the dentist is closely related to one patient, but distantly related to the second. This implies that he was the source for one infection but not the other. Finally in Figure 1.1c the dentist is distantly related to both patients, but they are closely related. This implies that the dentist is not the source of infection, though there may still be a common source. The problem of phylogenetic search then consists of finding these different configurations (Section 2.3) and determining which configuration is best (Section 2.1).

The inputs to the problem are a set of descriptions for the organisms in question, called taxa. Typically these descriptions consist of DNA sequences from each taxon, though some methods include morphological (visible features) or other characteristics.

The result of the problem is a tree, typically bifurcating and unrooted. An unrooted, bifurcating phylogenetic tree is a connected graph where every node is degree one or three.

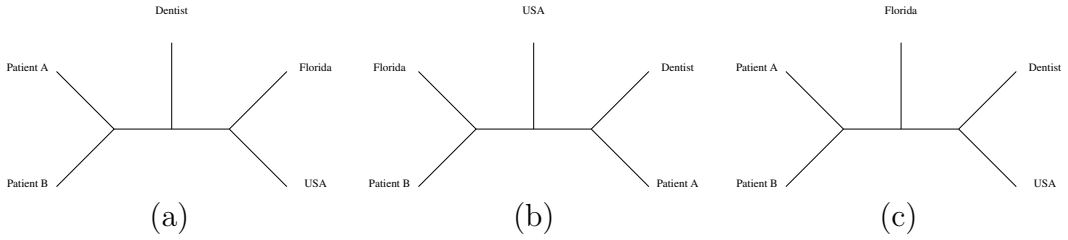


Figure 1.1: Three possible relationships between five HIV samples

Unrooted trees do not make any statements about which nodes are ancestors and which are descendants. There exist methods for rooting the resulting trees, but they are beyond the scope of this work.

The computationally difficult problem of phylogenetic search is known to be NP-Hard [11]. Due to the complexity of the problem, heuristic methods are required. This work endeavors to improve not only the methods and techniques in current use, but also the current understanding of the solution space as a whole, independent of these methods. The work introduces a new definition for the space of all possible trees and a means to visualize this space (Chapter 3). It then addresses the problem of designing a phylogenetic search method in light of this new tree space (Chapter 4). Finally it concludes by developing an improved method of finding an initial tree for phylogenetic search, again using the new tree space (Chapter 5).

Chapter 2

Related Work

2.1 Scoring Metrics

A key component of any search problem is the selection of an optimality criterion. There are two competing criteria in the area of phylogenetic search. The central issue to this competition is the acceptance of a model of evolution.

2.1.1 Maximum Parsimony

The simplest optimality criterion for a phylogeny is that of Maximum Parsimony(MP). When MP is used as the optimality criterion phylogenetic search is known to be NP-Complete [14]. Maximum Parsimony does not explicitly use any model of evolution. Rather than an explicit model, MP assumes that a tree which implies the least number of mutation events is the best. Under this criterion, the score of a tree is the number of base pair mutations required to explain the data. All changes are given an equal weight in the final score, though weighted methods exist. The tree with the least number of changes (lowest score) is the most parsimonious tree.

This score can be computed with a post-order pass on the tree. Each character is represented by a set of possible states taken from the DNA base pairs (A,C,T,G). Each ancestral sequence is computed from the character sequences of its children. The sequence of an ancestral node is the intersection of each of it's children's characters, if that intersection is not the empty set. If the intersection of the two characters is empty, then the current parsimony score for the tree is incremented and the character for the node is the union of

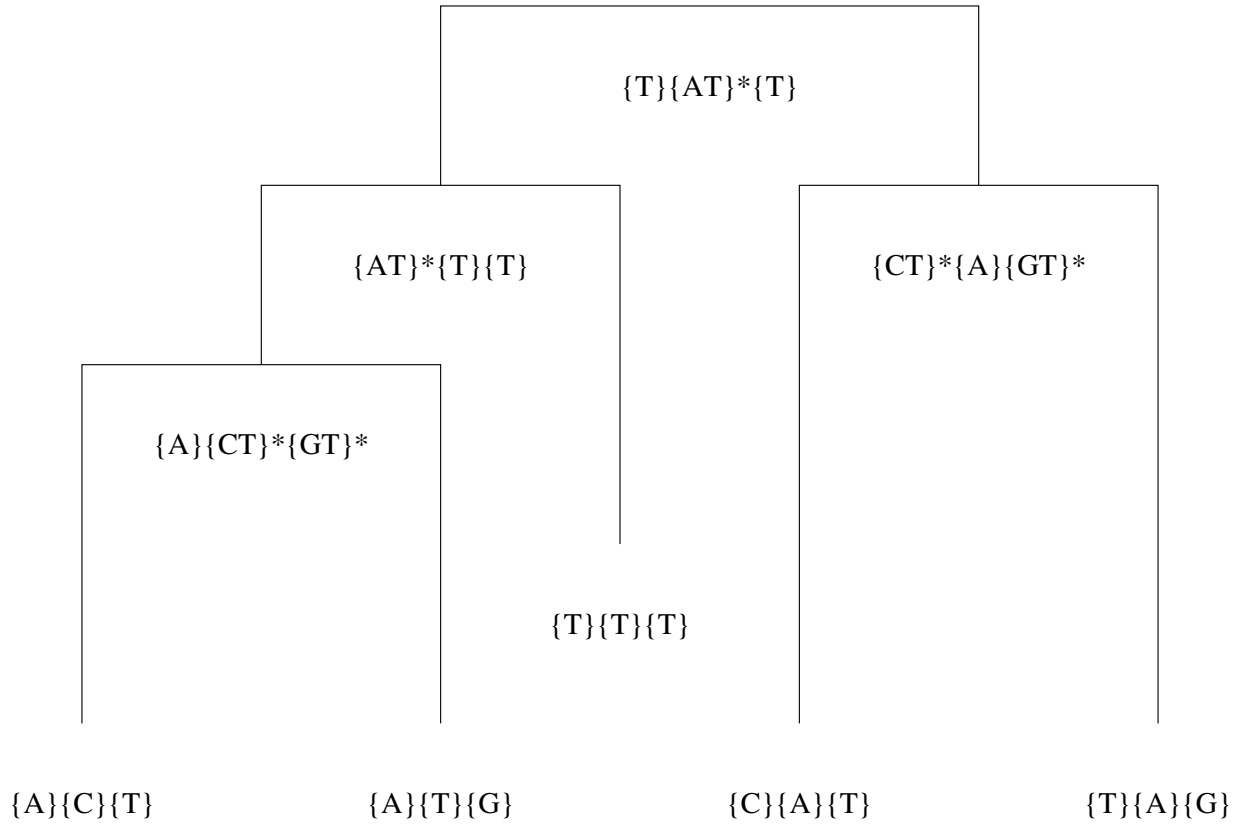


Figure 2.1: Calculating the Maximum Parsimony score for a tree. The MP score for a tree is calculated in post-order fashion. Each character of an ancestral node is the intersection of its children, unless that intersection is empty in which case the character is the union. Characters formed by union are marked with an *, and each increases the MP score by one. The final score for this tree is 6.

its children. Figure 2.1 illustrates the parsimony computation for 5 taxa with 3 characters each.

2.1.2 Maximum Likelihood

Maximum Likelihood [16] differs from Maximum Parsimony in that it requires an explicit model of evolution. This model is a 4×4 matrix containing the probabilities for each transition from one nucleotide to another, over some period of evolutionary time. Several such models exist, such as Jukes-Cantor [32], F81[17], F84 [18], HKY85 [25], and GTR [35, 46]. The most general model is GTR+Γ, which is a parameterized family of models.

The likelihood of a tree depends not only on the topology of the tree, but also on the branch lengths. These lengths are optimized using numerical methods, typically Newton-Raphson. Once the branch lengths (l) are determined, transition matrices ($\mathbf{T}$) can be computed for each branch from the model of evolution ($\mathbf{M}$) through matrix exponentiation.

$$\mathbf{T}(l) = e^{l\mathbf{M}} \tag{2.1}$$

Given the character data and the probabilities for every possible transition along each branch of the tree, the likelihood of each character given the tree can be computed. Each character site is assumed to be independent so that the final likelihood of all of the data given the tree is simply the product of the likelihoods for each character.

Like the scoring of a tree by parsimony, this process is done as a post order traversal of the tree. Figure 2.1 shows this process on the same tree used to explain MP in Figure 2.1. This example uses a simple transition matrix for all branches in the tree. Note that under this simple model, the results are very similar to those of maximum parsimony, a typical ML calculation would use a different transition matrix for each branch.

The characters at each node in the tree are represented by a likelihood for each possible value that the characters can take. This can be thought of as a $4 \times c$ matrix where c is the number of characters in the data set. To calculate the matrix of an ancestral node, each of the child matrices are premultiplied by $\mathbf{T}$. The matrix of the ancestral node is then the element-wise product of these two matrices.

Once the final ancestral node has been calculated, the character likelihoods are computed by premultiplying this node's matrix with a vector of base character frequencies. The likelihood of the tree is the product of these character likelihoods, as each character is assumed to be independent of all others.

Finally, the tree for which the data is the most likely is selected as the best tree.

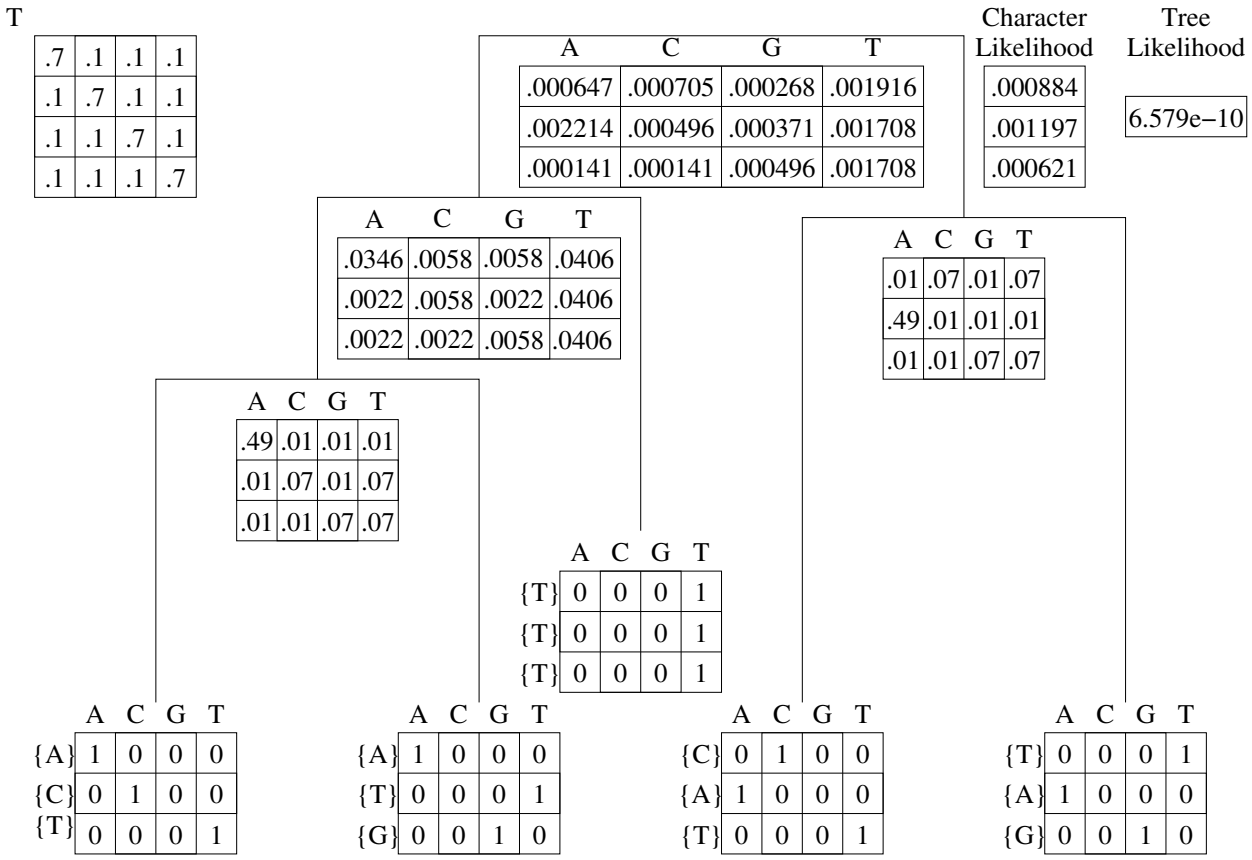


Figure 2.2: Calculating the Maximum Likelihood score for a tree. The ML score for a tree is calculated in post-order fashion. The probabilities for each leaf node, are simply the observations. The probabilities of each ancestor can be found by multiplying the vector of probabilities of each child by the transition matrix. The probability vector for the ancestor is the element-wise product of these two vectors. The final ancestor is then multiplied by a vector of base character frequencies to yield the character likelihoods. The likelihood of the tree is the product of all the character likelihoods. The final likelihood of the data given this tree is $6.57e-10$

2.1.3 Concordance of ML and MP

While there is debate over the relative merits of ML and MP as optimality criteria, it is known that methods which improve MP scores also improve ML scores [53]. This is due to a general concordance between the scores of trees under the two metrics. While it is not possible to accurately estimate one score using the other, there is often a large overlap between the best trees under both criteria. The methods discussed in this work are scoring criteria agnostic. However as MP is much easier to compute, results will be presented as the most parsimonious rather than the most likely ones.

2.2 Subtree Transfer Operations

Subtree Transfer Operations are a class of operators in the space of tree topologies. Each takes one or more subtrees and changes their positions within a tree. A variety of Subtree Transfer Operations have been used in phylogenetic search heuristics. These operations are the most commonly used method to move about in the space of all phylogenetic trees. Those used can be broadly classified into two groups. The first group, containing Tree Bisection and Reconnection(TBR), is the most commonly used. This group involves dividing a tree into separate subtrees and then reconnecting those subtrees in different ways. The second group involves conflating neighboring nodes of the tree to build unresolved trees and then considering all possible resolutions of this resulting tree.

2.2.1 Tree Rearrangements

The following Subtree Transfer Operations are those that divide the tree into subtrees. These divided subtrees are then rearranged in various ways and finally reconnected to form a new tree. Throughout this section n refers to the number of taxa represented by the tree.

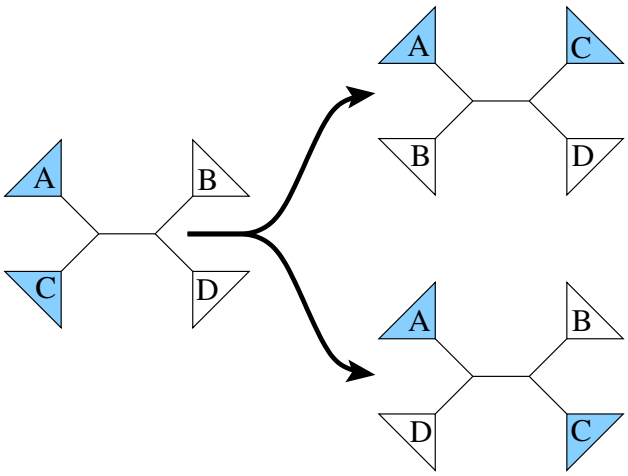


Figure 2.3: A Nearest Neighbor Interchange (NNI) rearranges a tree into two new topologies

Nearest Neighbor Interchange

Nearest Neighbor Interchange (NNI) is the simplest of the subtree transfer operations. In this operation two subtrees that are separated by only one branch are swapped.

This operation, shown in Figure 2.3, can be applied to any interior branch of a tree to yield two new topologies. Thus, every tree has $O(n)$ trees that are within 1 NNI move.

Subtree Pruning and Re-grafting

Subtree Pruning and Re-grafting (SPR) is the next order of subtree transfer operation. In SPR a subtree is selected as in Figure 2.4a. This subtree is then temporarily removed, or pruned from the tree, shown in Figure 2.4b. The subtree is then reattached to the tree in a new location (re-grafting) which yields the new topology as shown in Figure 2.4c.

Every NNI move can be accomplished by an SPR move also. This is done by selecting one of the subtrees that is to be interchanged and re-grafting it next to the sibling of the other subtree to be interchanged. Thus

$$NNI \subset SPR \tag{2.2}$$

The neighborhood of trees that can be reached by one SPR is $O(n^2)$. Each branch corresponds with two subtrees, one on each side of the branch. These subtrees can then be re-grafted to any of the remaining $O(n)$ branches, yielding the final size of the neighborhood.

TBR

Tree Bisection and Reconnection (TBR) is the most commonly used subtree transfer operation, and is central to the majority of existing phylogenetic search programs. It is also the subtree transfer operation used by this work. A TBR rearrangement proceeds in four phases. First a branch is selected. This branch is then completely removed from the tree yielding two separate subtrees. Two branches, one in each of the two subtrees, are selected. Finally these two branches are reconnected to form a new topology, as shown in Figure 2.5.

Should one of the branches selected in a subtree be the location from which the bisecting branch was removed then this operation is equivalent to SPR. Thus

$$SPR \subset TBR \tag{2.3}$$

and the entire hierarchy can be expressed as

$$NNI \subset SPR \subset TBR \tag{2.4}$$

The neighborhood of trees within one TBR rearrangement contains $O(n^3)$ trees as three branches must be selected out of the n available, each combination yielding a potentially different topology.

Search programs based on TBR typically take a given tree and consider all trees within one TBR rearrangement. The best trees found are taken and all trees within one TBR of those trees are considered, and so on until no improvements are made. One of the weaknesses of TBR is that many of the $O(n^3)$ trees considered at each step are not unique. Furthermore, many trees that are within two TBR rearrangements of a given tree are also

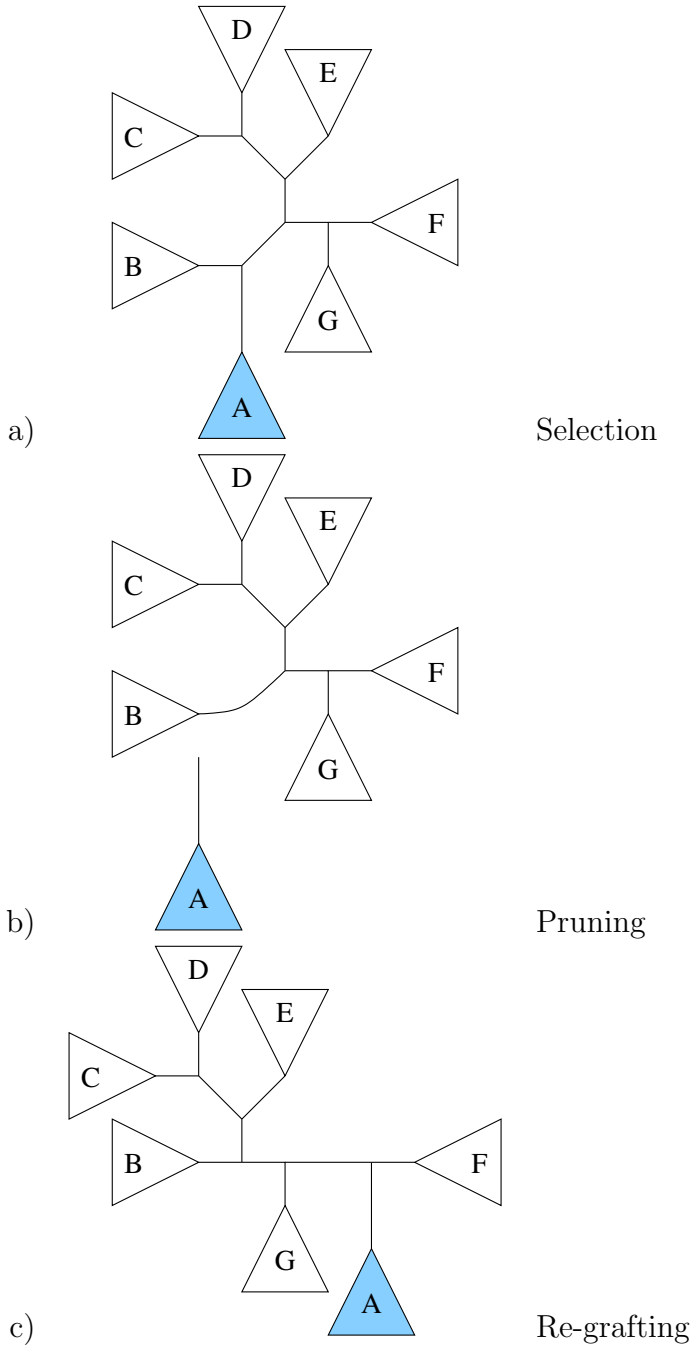


Figure 2.4: A Subtree Prune and Re-graft (SPR) rearranges a tree into a new topology through selection, pruning, and re-grafting

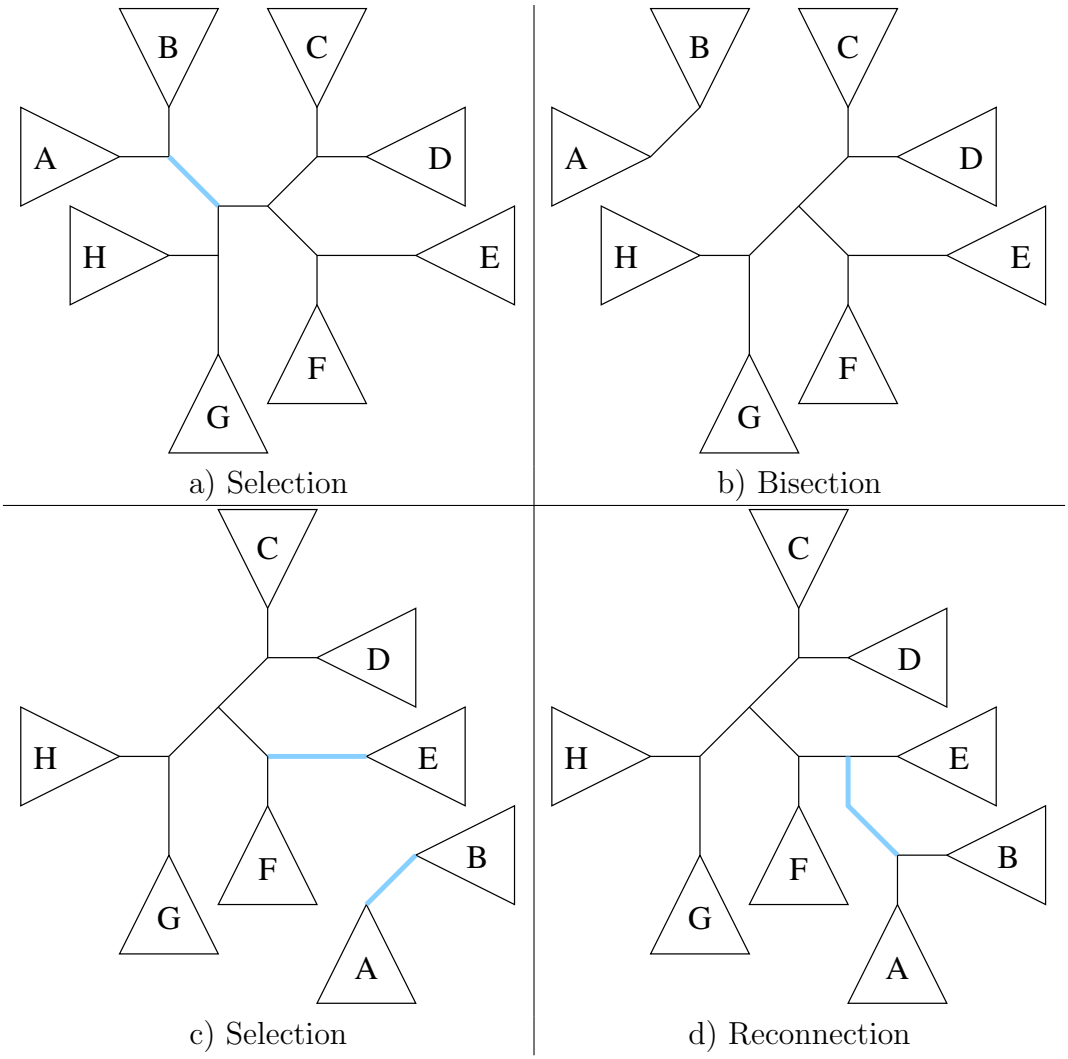


Figure 2.5: A Tree Bisection and Reconnection (TBR) rearranges a tree into a new topology by dividing it into two separate subtrees and then reconnecting these trees in novel ways.

within one TBR rearrangement. Current methods do not remember trees that have been previously considered. So, much of the effort in a TBR based search is duplicated effort. Avoiding this duplication of effort is addressed in Chapter 4.

2.2.2 p-Edge Contraction and Refinement

The second type of tree rearrangements, those that conflate and refine neighboring nodes, is composed of the family of p-Edge Contraction and Refinement (p-ECR). NNI, while a member of the first type is also equivalent to 1-ECR and so also belongs in this second type.

In p-Edge Contraction and Refinement, shown in Figure 2.6, p branches are selected in the tree. The branches are contracted until they have no length and the tree contains unresolved nodes (nodes with degree > 3). Each of these nodes is then replaced by a resolution tree for that node. The neighborhood of p-ECR trees is $O(n^p p!!)$

The family of p-ECR rearrangements forms a separate hierarchy of subtree transfer operations. At the lowest level, 1-ECR, the ECR and TBR hierarchies are equivalent as

$$NNI = \text{1-ECR} \tag{2.5}$$

However, from this point they diverge as there are $O(n)$ trees that are within one 2-ECR, but not within one TBR transition [19].

The TBR and ECR families are also qualitatively different. ECR moves leave most of the topology unchanged, as they affect only the relationships described by the contracted and refined edges. At no point is the tree disconnected, this constrains the resulting trees to bear a certain resemblance to the original tree. In this sense NNI is much more closely related to ECR than to TBR. TBR moves on the other hand are more violent, by completely dividing the tree first these rearrangements are free to affect more radical changes. There is always at least one tree within one TBR rearrangement in which any two taxa in the original tree are siblings in the new tree. Depending on the current search context either of these two properties may be more desirable than the other. This work also uses a subset of the p-ECR family.

2.3 Search Methods

A variety of phylogenetic search methods are currently in use. The first group, including distance methods and stepwise maximum parsimony are greedy heuristics. They process very quickly and produce a single tree. These trees are often the starting points for the remaining methods. Chapter 5 explores a new method for building these initial trees.

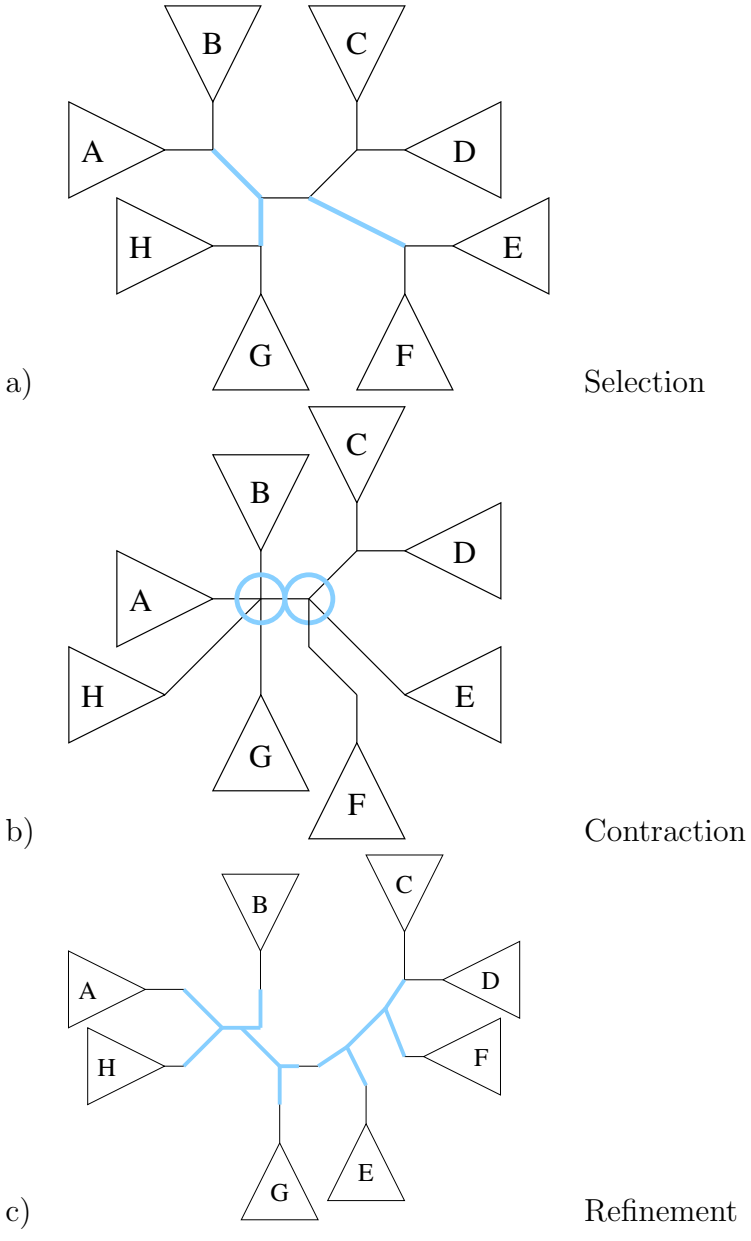


Figure 2.6: p-Edge Contraction and Refinement (p-ECR). Here a 3-ECR is made by selecting 4 branches, contracting them to form unresolved nodes, and finally replacing each unresolved node with a resolution tree.

2.3.1 Distance Methods

The simplest search methods are distance methods. These techniques require an all-to-all distance matrix for the taxa. Typically these distances come from pairwise alignments but any distance metric could be used. The nearest neighbors are joined into a subtree and the distance between this new subtree and all others is recalculated. The two most popular distance methods, neighbor joining and UPGMA (Unweighted Pair Group Method with Arithmetic mean), differ only in how this distance is calculated. This process continues until there are no more subtrees to join and a final tree is produced. Other variations on agglomerative clustering could also be tried, but are not currently used.

2.3.2 Stepwise Maximum Parsimony

Stepwise maximum parsimony begins with a random shuffle of the taxa. The first three taxa are joined into a tree. Since there is only one three taxon tree, no decisions are required at this point. Then in the random order each taxa is examined and placed at the location which yields the best MP score.

One insight into the difficulty of phylogenetic search is made apparent by stepwise maximum parsimony. There is only one three taxa tree. Therefore, at this point in the search the subtree is clearly optimal. As the fourth taxa is added, three trees are examined. There are only three possible four taxa trees, so at this point the chosen four taxa subtree is also optimal as all possibilities have been examined. By the same token the subtree containing only the fifth taxa to be added is optimal. However, the five taxa subtree formed by joining these two optimal subtrees together in an optimal fashion is not itself guaranteed to be optimal. The stepwise search will only consider five of fifteen possibilities, and data sets can be constructed for which one of the ten unexamined topologies is the optimal topology. Thus the order in which the 5 taxa are added will affect the results of the algorithm, with different orders resulting in different trees. In general, the subtrees of optimal trees may

not themselves be optimal. This unfortunate observation makes devising good divide and conquer strategies difficult.

2.3.3 Quartet Puzzling

One divide-and-conquer algorithm is quartet puzzling. A quartet is a tree of four taxa, as for any four taxa there are only three possible quartets it is easy to insure that quartets are optimal. There are $O(n^4)$ quartets for n taxa, so all quartets can be found in polynomial time. The final tree is then selected to be the one compatible with the largest number of quartets. A tree is compatible with a quartet if when all taxa which are not in the quartet are removed from the tree and likewise all taxa in the quartet that are not in the tree are removed the tree is identical to the quartet.

In general, finding this tree with optimal quartet compatibility is also NP-Hard [51]. There do, however, exist polynomial heuristic methods. Quartet Puzzling is one of these methods, and works in a fashion similar to stepwise maximum parsimony. The taxa are ordered randomly, and then added to the final tree one at a time. As each is added, the position of the taxa is chosen as the position which makes the tree compatible with as many quartets as possible.

2.3.4 Hill Climbing with Subtree Transfer Operations

After a distance method or stepwise maximum parsimony has produced a tree, this tree is typically refined through hill climbing. Subtree transfer techniques are the most common heuristics in current use [50, 38, 54, 23]. The neighborhood for the hill climbing heuristic is formed through a given subtree transfer operation, typically TBR (Tree Bisection and Reconnection, see Section 2.2.1). The current tree is rearranged using the selected subtree transfer operation into many other trees which are all scored. The best scoring tree becomes the new current tree and this process is repeated until no tree scores as well or better than the current tree. In the case of equally scored trees, they are also treated as the current

tree and all trees that are their neighbors are also examined. This method returns the set of trees sharing the best score found.

The Ratchet

TBR-based methods are prone to finding local minima and remaining in them. One very successful technique for avoiding these minima is known as the ratchet [40, 56]. Periodically the data set is randomly re-weighted with some characters being removed and others being repeated. The search then proceeds as normal under this skewed data set. After a time the data set is restored. This process continues to alternate between randomly skewing the search, and restoring the original data set until no further progress is made.

2.4 Tree Spaces

A key component of visualizing and more importantly analyzing the performance of phylogenetic search techniques is the study of tree spaces. A tree space consists of all the trees that might be considered by a search and their relationships with each other. Various tree spaces have been defined and used in phylogenetics.

2.4.1 Subtree Transfer Induced Spaces

The most common tree spaces encountered in phylogenetic search are those which are induced by the subtree transfer operations used by current search programs [1]. As these spaces merely arise from the current search techniques it is not surprising that they provide little structure or guidance to searching. Subtree transfer induced spaces have a graph structure, where every possible tree is a node in the graph. If a tree can be transformed into a second tree through a subtree transfer operation then those two trees are connected by an edge in the graph. Figure 2.7 shows the TBR induced space for all five taxa trees.

A few points of interest are to be noted. First the graph is highly, though not fully, connected. As a result TBR-based searches will often reexamine trees. Second, as the space

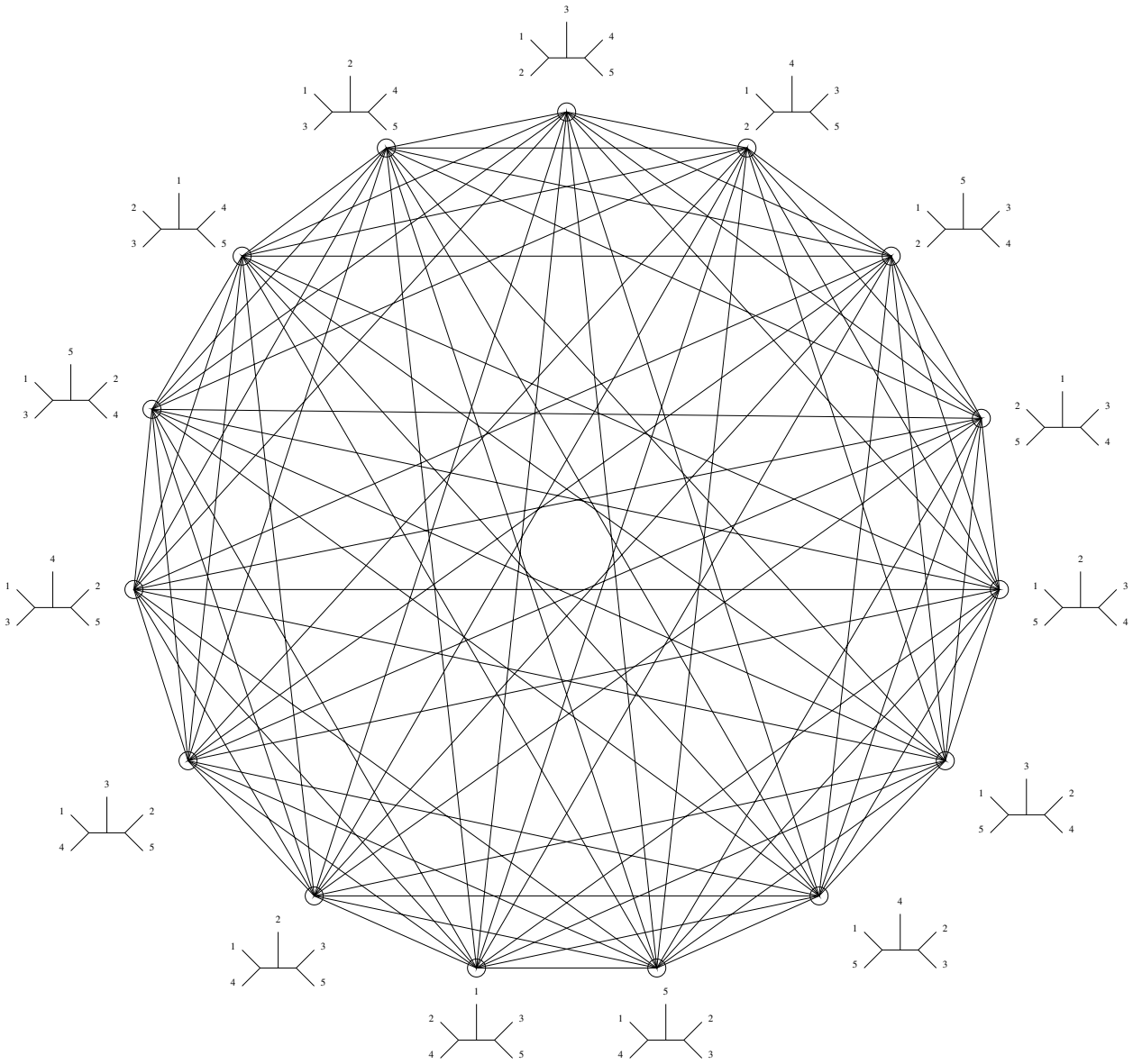


Figure 2.7: A graphical representation of the TBR induced tree space for all 5 taxa trees.

is a graph, there is no meaning to direction, and distance can be difficult to compute. This is especially true for the TBR induced space as it is known that the problem of finding the TBR distance between two trees is NP-Hard [1]. It is unfortunate that there does not appear to be any structure in this space that could be used to aid phylogenetic searches.

One extension of subtree transfer induced spaces exists. This extension, proposed by Billera *et al.* [5], is a geodesic space. Geodesic spaces are composites of simple spaces and rules for moving between them. In this specific space, each topology has an Euclidean space associated with it. In this space the dimensions correspond with the branch lengths. Along the edges and corners of this space where one or more branch lengths are zero, other topologies can be reached. By definition the distance between any two points representing the same unresolved topology is zero. Thus the connections between these spaces is the graph of all p-ECR transitions between topologies.

Distance is still very difficult to compute, though approximation methods exist. This space forms a very complete picture of all possible trees, but it has not been used to inform any search techniques. Furthermore, due to its close relationship with other subtree transfer induced spaces it is not clear how this space could be used to improve current search techniques. Figure 2.8 shows a graphical representation of such a space.

2.4.2 Multidimensional Scaling

Multidimensional Scaling (MDS) has been used to form a descriptive tree space [3, 27]. Figure 2.9 shows an example of this type of tree space. MDS takes as input a matrix containing the true all-to-all distances between data points. To date the only distance measure for phylogenetic trees used in MDS has been Robinson-Foulds distance. While topological methods exist to calculate the intrinsic dimensionality of a space defined by a metric, the process is non-trivial and beyond the scope of this work. The MDS algorithm then tries to find a mapping from the space implied by the given distance matrix into a Euclidean

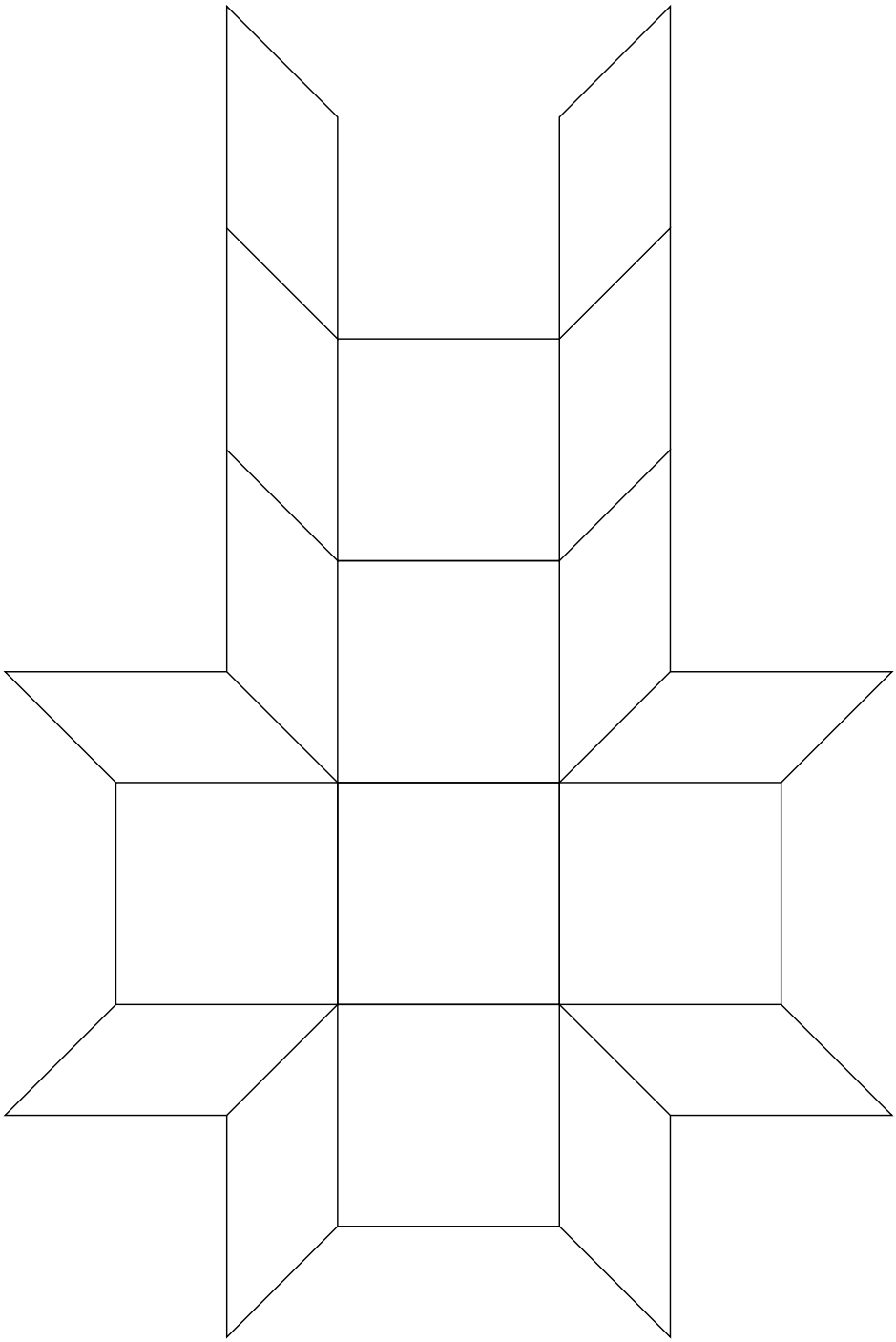


Figure 2.8: A graphical representation of the geodesic space proposed by Billera *et al.* Each of the square tiles represents a specific topology of five taxa. Within each tile, there exists a Euclidean space with coordinates which correspond to the lengths of each of the branches in the topology. Along the edges and corners where at least one of the branch lengths is zero, the tiles connect with other tiles that could be reached through a p-ECR move that contracts the zero length edges. The structure formed has been unfolded for ease of display.

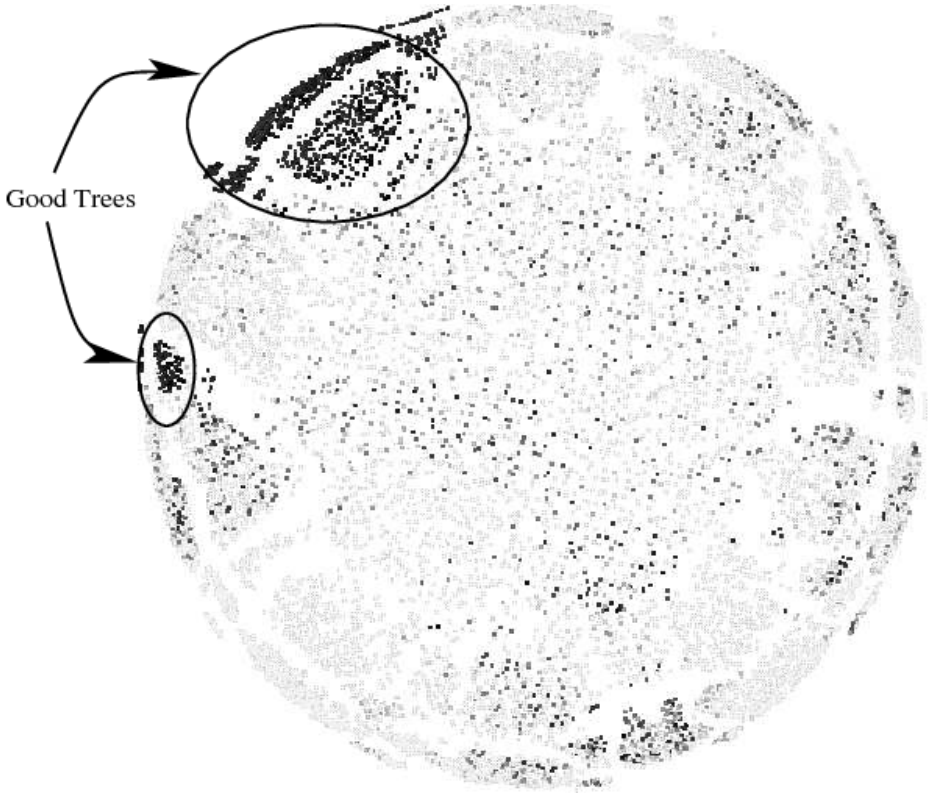


Figure 2.9: A visualization of tree space using 2-D Multi-Dimensional Scaling (MDS). The trees are colored using their parsimony scores.

space of given dimensionality which minimizes a strain function. A common function used is the RMS error between the all-to-all distances before and after the mapping.

The MDS technique which has been used in phylogenetics is strictly a post-processing step. MDS is therefore a purely descriptive and not a predictive method. There exist variations of MDS and other manifold based techniques for which this limitation does not apply, though they have yet to be applied to phylogenetic search. Further, as MDS is a highly non-linear transformation it is not clear how any structure which might be found could be exploited to further our understanding of phylogenetic search. As a descriptive method, one of the important contributions of MDS was the confirmation of 'Tree Islands' [27], groups of identically scoring trees that are all 1-TBR neighbors of each other. It is currently believed that these islands are one of the major stumbling blocks to the performance of TBR based methods.

Chapter 3

On The Use Of Cartographic Projections In Visualizing Phylogenetic Tree Space

Published: WABI 2009 and Algorithims for Molecular Biology

3.1 Abstract

Phylogenetic analysis is becoming an increasingly important tool for biological research. Applications include epidemiological studies, drug development, and evolutionary analysis. Phylogenetic search is a known NP-Hard problem. The size of the data sets which can be analyzed is limited by the exponential growth in the number of trees that must be considered as the problem size increases. A better understanding of the problem space could lead to better methods, which in turn could lead to the feasible analysis of more data sets. We present a definition of phylogenetic tree space and a visualization of this space that shows significant exploitable structure. This structure can be used to develop search methods capable of handling much larger data sets.

3.2 Background

Phylogenetic analysis has become an integral part of many biological research programs. These include such diverse areas as human epidemiology [12, 49], viral transmission [13, 26], and biogeography [15]. With the advent of new automated sequencing techniques, the ability to generate data for inferring evolutionary histories (phylogenies) for a great diversity of organisms has increased dramatically. Researchers are now commonly generating many

sequences from many individuals. However, our ability to analyze the data has not kept pace with data generation.

Phylogenetic search is a difficult problem. When parsimony is used as the optimality criterion the problem is known to be NP-complete [14]. The search problem itself, independent of scoring, is known to be NP-Hard [11]. This means that optimal phylogenetic searches on even hundreds of taxa will take years to complete and heuristic searches for near optimal trees must be used.

A variety of heuristic search methods have been used to find optimal trees within a tree space. The most common method is to search tree space using tree rearrangements [50, 38, 54, 23]. Other methods such as those based on Bayesian inference [47], or genetic algorithms [58] also exist. However all of these methods rely only on local information to guide the phylogenetic search. This limitation arises because no global exploitable structures have been previously observed in tree space.

Greater understanding of the problem space may allow more sophisticated search techniques to be applied, with a consequent improvement in the effectiveness of the search. One technique that can be used to better understand the space of phylogenetic search, and the behavior of search algorithms within this space, is visualization. This includes two separate activities: first, defining the search space of phylogenetic trees, or tree space, and second, developing methods to display tree space in a way that is exploitable in search techniques.

This visualization must have the following properties to be useful.

- Each tree should map to a single deterministic position. Otherwise the method is restricted to post-processing, and cannot be used to guide a search.
- Distance between trees should be easy to calculate. If it is not, the visualization will not be able to be used in real time to guide a search.
- The visualization should reveal exploitable structure. This is important because if a visualization shows no structure it provides no guidance for a search.

- This mapping should be reversible, meaning that there should be a method of turning a position into a tree. This is necessary, as to be useful in searching it must be possible to quickly find trees in the space suggested by the visualization.

This work presents an elegant linear projection of trees. This projection can be computed much faster than current alternatives and is better at preserving structural continuity between trees after the projection. Furthermore this projection is deterministic, allowing it to be used as an inline rather than a post-process analysis. This property coupled with the structural preservation allows the consideration of novel search strategies in the new projected space.

The Results and Discussion section presents a definition of tree space and an elegant projection of that space that has all four of these desirable properties. This projection is then used to visualize the tree space and expose structure that can be exploited to guide the searches of common, but computationally expensive methods.

3.2.1 Related Work

Tree space consists of all of the possible phylogenetic trees for a given set of taxa and their relationships with each other. This space is the domain of whatever search strategy is employed. Previous search strategies have not explicitly defined this domain, and the tree space that implicitly arises from these strategies is very cumbersome to work with. Tree spaces have also been explicitly defined without designing algorithms to take advantage of these spaces. This is primarily due to a lack of exploitable structure in these explicitly defined tree spaces. Figure 3.1 contains a visual comparison of three tree spaces that have been used previously and are discussed in the following sections.

Subtree Transfer Induced Spaces

The most common tree spaces used in phylogenetic search are the spaces implicitly defined by the subtree transfer operations, such as Tree Bisection and Reconnection (TBR) or Subtree

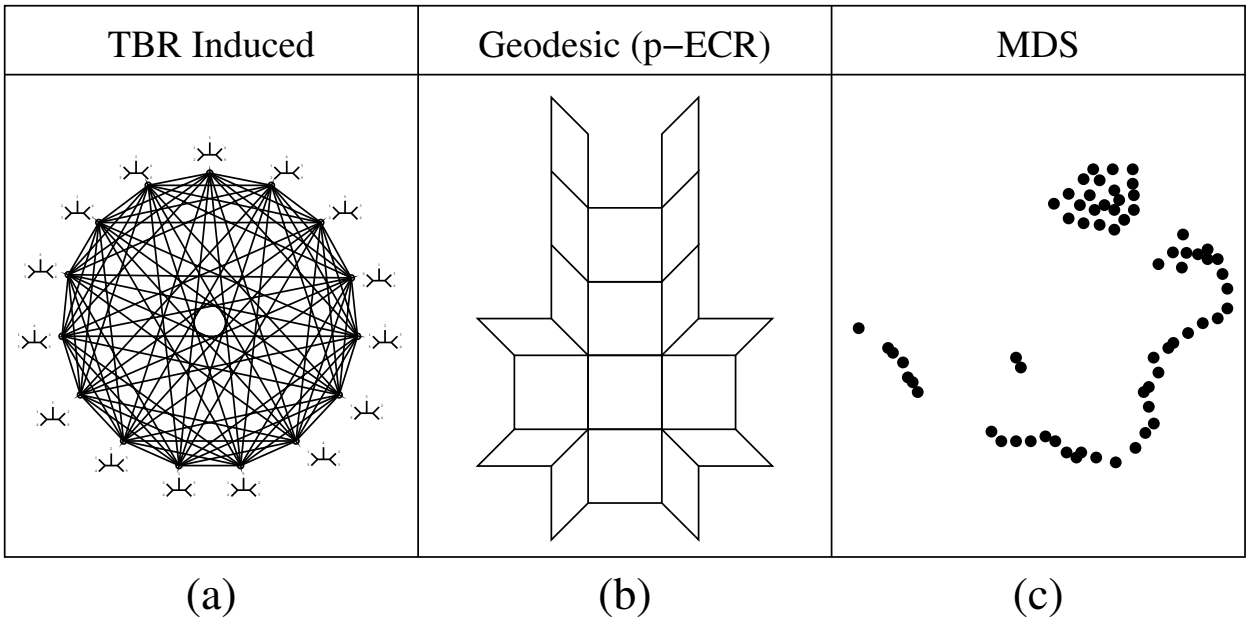


Figure 3.1: A visual comparison of three tree spaces previously used. The graph structure induced by TBR (a) moves is highly connected. The geodesic structure (b) consists of tiles of Euclidean space (orthants) each consisting of one topology with all its possible branch lengths. These tiles are joined together along their edges in accordance with valid p-ECR moves. Finally Multidimensional Scaling (MDS) (c) plots trees in locations that preserves some distance metric. A typical search is shown where a long tail of trees is followed by a larger group of topologically similar trees.

Prune and Re-graft (SPR) , used during the search. These operations in turn induce distances between trees [1]. These tree spaces take the form of graphs where each node is a specific tree. Each pair of trees that can reach each other with a single subtree transfer operation is connected with an edge of the graph.

This type of space is very amenable to hill climbing, a search strategy in which the search moves from a tree to its best neighboring tree until no neighbor trees are better than the current tree. The typical phylogenetic search begins at some node in this graph of tree space corresponding to an initial tree. This tree is typically either selected randomly, determined by the user, or is built using a heuristic. Common choices for this heuristic include UPGMA and stepwise maximum parsimony. The tree is then modified using a subtree transfer operation such as Nearest Neighbor Interchange (NNI), Subtree Prune and Re-graft (SPR), Tree Bisection and Reconnection (TBR), or p-Edge Contraction and Refinement (p-ECR) [20]. The new best node becomes the starting node and the process is repeated until convergence. This is also the space used by Keith *et al.* [33] to build their generalized Gibbs sampler.

Unfortunately, though this space has been commonly used for searching, it is not easily visualized. For example using TBR, a very popular subtree transfer operation, the graph that represents this tree space has $O(n!!)$ nodes and each node is degree $O(n^3)$. Displaying this graph is clearly not practical for any problem of significant size. Worse, as this tree space is essentially a graph, there is no significant meaning to position, violating the first two criteria for a useful visualization. Also, distance can be extremely difficult to compute. Calculating TBR distance is NP-Hard [1]. These difficulties violate the third criterion. Finally this graph structure shown in Figure 3.1a does not exhibit exploitable structure, the fourth criterion, as trees of similar score are not grouped together. As shown in Figure 3.2, the quality of trees that are within 1 TBR rearrangement of a given tree varies wildly over the range of possible scores. Furthermore, due to the graph structure of the space there is no

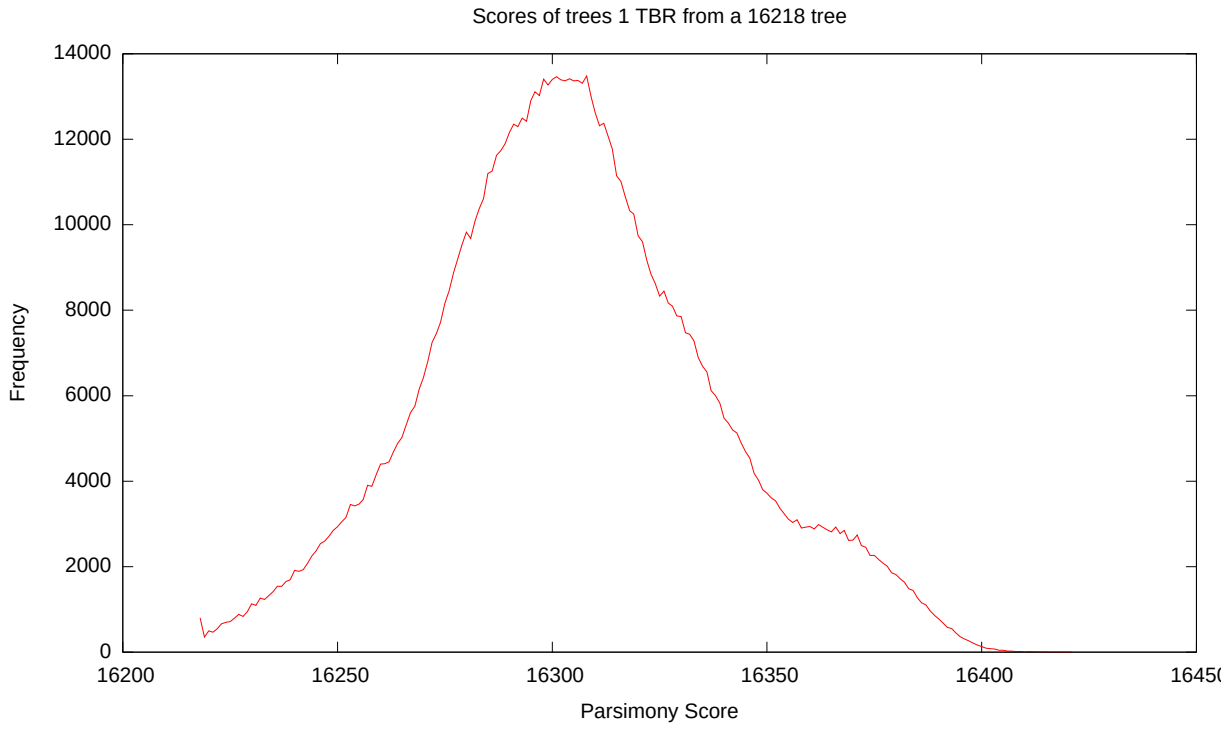


Figure 3.2: The frequency of various parsimony scores for trees found within 1 TBR rearrangement of a tree with a score of 16,218, the best known score on the Zilla data set. Note the wide spread of scores and that most neighbor trees are much worse than the initial tree.

way to distinguish one such tree from another, without performing the rearrangement and examining the resulting tree.

Geodesic Tree Space

Billera *et al.* [5] introduced a new description of tree space, which has been further refined by Hultman [29]. Under this description, each fully resolved (bifurcating) topology is given its own orthant, the higher dimensional analog of a graph quadrant. Each dimension of the orthant corresponds to one of the branches in the topology, and the value associated with that dimension is the length of that branch. Within each orthant, distance is a simple Euclidean distance. At the edges of the orthant, where at least one coordinate becomes zero, the tree becomes an unresolved (multifurcating) tree. This unresolved tree has a corresponding point on each of the orthants that represent a potential resolution of this tree. The distance

between these points on separate orthants is defined to be zero, thus forming a geodesic space. These connections between orthants are directly related to p-ECR rearrangements. The structure of this space can be seen in Figure 3.1b.

This space is unlike the tree space induced by subtree transfer operations. The branch lengths of the trees are included and this tree space is continuous. However, because it is a geodesic, it can be difficult to calculate distances, though recent work [2, 34, 41] has begun to address this issue. Unfortunately, like the subtree transfer induced spaces used during phylogenetic search, geodesic tree space is not easily visualized due both to the high dimensionality of each orthant and the complex connections between orthants. These connections are based on a subtree transfer operation, p-ECR, and so like the tree space defined by TBR there is no significant meaning to position between orthants. Thus, like the TBR induced tree space, this tree space does not meet the criteria for a good visualization. While position and trees are tightly connected, distance is difficult to compute and it is not clear that there is any exploitable structure.

Multidimensional Scaling

Multidimensional Scaling (MDS) has also been used to visualize tree space [3, 27]. This method does not directly define a tree space, rather it uses the space induced by the distance metric used for the MDS. In both the work of Hillis *et al.*[27] and the prior work by Amenta and Kilinger[3], Robinson-Foulds distance was used. This distance is a measure of how many branches are not in common between two trees. MDS is a highly non-linear projection, as it moves points around to minimize the sum of the squared differences of the distances between points before and after the projection. One difficulty is that in the presence of many points, clusters formed by MDS may not reflect topological similarity, but instead reflect the best found compromise in this strain function.

Using this method Hillis *et al.* [27] were able to show some important characteristics of phylogenetic search. The most notable characteristic visualized was the presence of plateaus, large groups of closely related trees, that tend to slow down the search.

There are however some significant limitations to their use of MDS, which may not apply to the many variants of Multidimensional Scaling[4] or to other manifold-based methods. First, MDS is strictly a post-processing step. All of the points to be projected must be known beforehand, which limits the method to analysis of a search. Secondly there is no meaning to the space between points. It is not possible under MDS to determine a tree that would map to a specific point. Third, the axes of the new space have no consistent meaning. The only thing that MDS tries to preserve is some sense of distance; direction does not have any meaning after MDS is performed. As a result of these limitations, while MDS is a good visualization technique it does not meet the criteria of this work. This is primarily due to the highly non-linear and irreversible nature of the MDS transformation. MDS can be a very descriptive visualization, but it is a poor predictive visualization.

3.3 Results and Discussion

Another tree space is one defined in terms of partitions of taxa. A projection can be defined from this space which both deterministically maps trees to single points and is reversible. These properties give us the first three criteria for a good tree space and visualization. In the results section we show that this space also displays exploitable structure.

There are several varieties of trees that can be used in phylogenetics. Since only one specific set of n taxa will be considered at any time we constrain tree space to contain only trees of exactly those n taxa. Both candidate scoring metrics (likelihood and parsimony) work with unrooted trees so the space is further constrained to contain only unrooted and fully resolved trees.

Definition 3.3.1. An n-tree is a graph in which all vertices have degree one or three, with exactly n vertices of degree one.

Every branch in an n-tree divides the taxa on the tree into two sets, one on each side of the branch. Thus every branch can be thought of as a partition of the taxa. Some of these branches, those that connect to the leaves, are common to all n-trees. These branches are not useful in discriminating between different tree topologies and so are called trivial.

Definition 3.3.2. A trivial branch is a branch that connects a leaf node with an internal node.

Given n taxa there are $2^{n-1} - n - 1$ possible nontrivial partitions of those taxa. We define a space, called *split space*, where every possible nontrivial partition is associated with a unique dimension. We denote the split space associated with trees of n taxa as $\mathbb{T}_n$.

The location of a given tree in $\mathbb{T}_n$ is a vector, where each element of the vector is 0 if the corresponding partition is not part of the tree and 1 if the partition is present in the tree. There is a one-to-one mapping between vectors in split space and n-trees.

The mapping from an n-tree to a vector in $\mathbb{T}_n$ is simple. Initially, every element of the vector is set to 0. A non-trivial branch is selected and the associated partition is created by putting all taxa on one side of the branch into the first group in the partition and all other taxa in the second. The element in the vector associated with this partition is set to 1. This process is repeated for each non-trivial branch. This mapping is one-to-one but not onto, as there are more possible vectors than n-trees. This is because there exist conflicting partitions which cannot both be in one tree; however there are vectors which would include these conflicts.

Building an n-tree from a vector in $\mathbb{T}_n$ is also possible. However given a vector that does correspond to a valid tree, that tree can be reconstructed in the following manner. This is very similar to the method proposed by Gusfield[24]. First, all of the trivial branches are added to the tree. Next, all non-trivial partitions where the smaller group contains two taxa are considered. Each of the two taxa in the smaller group are joined at a new internal node and a new branch is added to that node. Next, partitions with incrementally larger small groups are considered, and their sub-clades which have already been built are joined at new

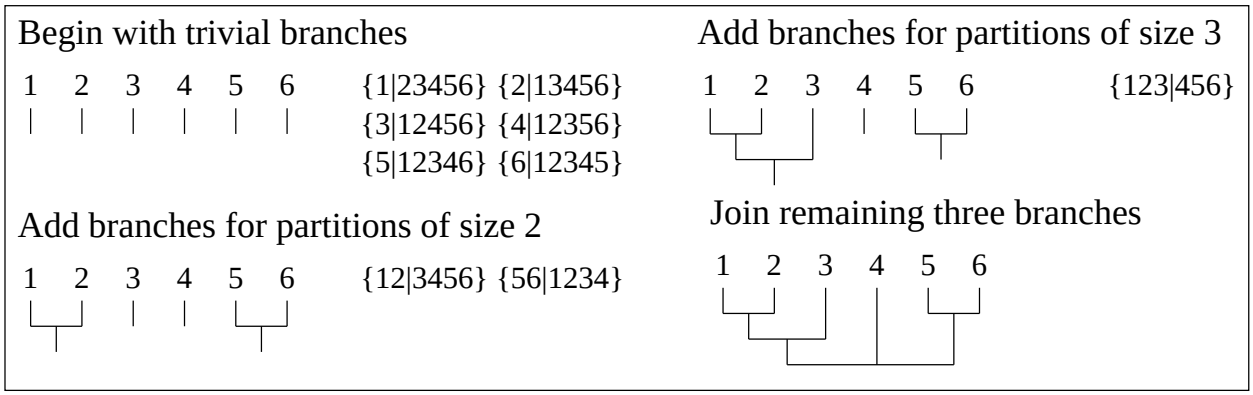


Figure 3.3: Converting a six taxon partition set to an unrooted tree structure

internal nodes. After all non-trivial partitions have been considered, there will remain three clades. These three subtrees are joined together at the final internal node and the tree has been reconstructed. Figure 3.3 graphically shows this reconstruction. As there is a mapping from an n-tree to a vector in $\mathbb{T}_n$ and the reverse mapping also exists, these trees and vectors are equivalent.

3.3.1 The Hypersphere of Trees

A hypersphere consists of the set of all points which are equidistant from a given center point. It is the higher dimensional analog of circles and spheres. The set of all vectors in $\mathbb{T}_n$ which correspond to valid n-trees has this structure as shown in Theorem 3.3.4.

Lemma 3.3.3. *All n-trees have $2n-3$ branches, and $n-3$ of which are nontrivial.*

Proof. See Waterman[57], Proposition 14.1 □

Theorem 3.3.4. *All n-trees lie on a hypersphere in $\mathbb{T}_n$.*

Proof. By Definition 3.3.1, n-trees are fully resolved. All fully resolved trees on n taxa have $n-3$ nontrivial branches by Lemma 3.3.3. As each such branch corresponds to exactly one of the possible partitions, an arbitrary n-tree in $\mathbb{T}_n$ will have exactly $n-3$ axes along which the coordinate of the tree will be 1 and all other axes will have a coordinate of 0. The Euclidean

distance to this point from the origin of $\mathbb{T}_n$ will therefore be $\sqrt{n-3}$, which is the same for all n-trees. As all n-trees are equidistant from the origin, they lie on a hypersphere. □

Projecting the Sphere

Directly visualizing the hypersphere model is clearly infeasible as the number of dimensions that would need to be included quickly exceeds the number of dimensions that we can conveniently visualize. Therefore some form of dimension reduction is needed.

Sphere to Plane Projections

Cartographic projections [7] are particularly apt at sphere to plane transformations. The basic cartographic projection takes a hypersphere in n dimensions and projects it onto a hyperplane of $n-1$ dimensions. This is done by selecting $n-1$ vectors, typically chosen from a basis set. Figure 3.4 shows how this reduction can project three dimensional data onto two dimensions. The inner product of each point on the hypersphere to be projected with each of the selected vectors is computed. These inner products become the coordinates of the projected point on a hyperplane of $n-1$ dimensions.

This cartographic projection can be extended to a new projection that reduces the dimensionality of the space more than the basic cartographic projection. Reducing the n dimensional space by one dimension when n grows as the number of possible partition sets is not significant. Therefore, rather than choosing $n-1$ vectors which results in a $n-1$ dimensional space, three vectors are used, yielding a three dimensional space. Three dimensions are used because it is well known how to display 3-D data, and the use of three dimensions preserves more structure than if the data were reduced to two dimensions.

As an example of this process, consider all trees with five taxa numbered 1-5 respectively. Every non-trivial branch has two taxa on one side and three on the other. There are ten such partitions, yielding a ten dimensional space. To project this space onto a two dimensional plane, two reference vectors are required. The vectors chosen, along with the

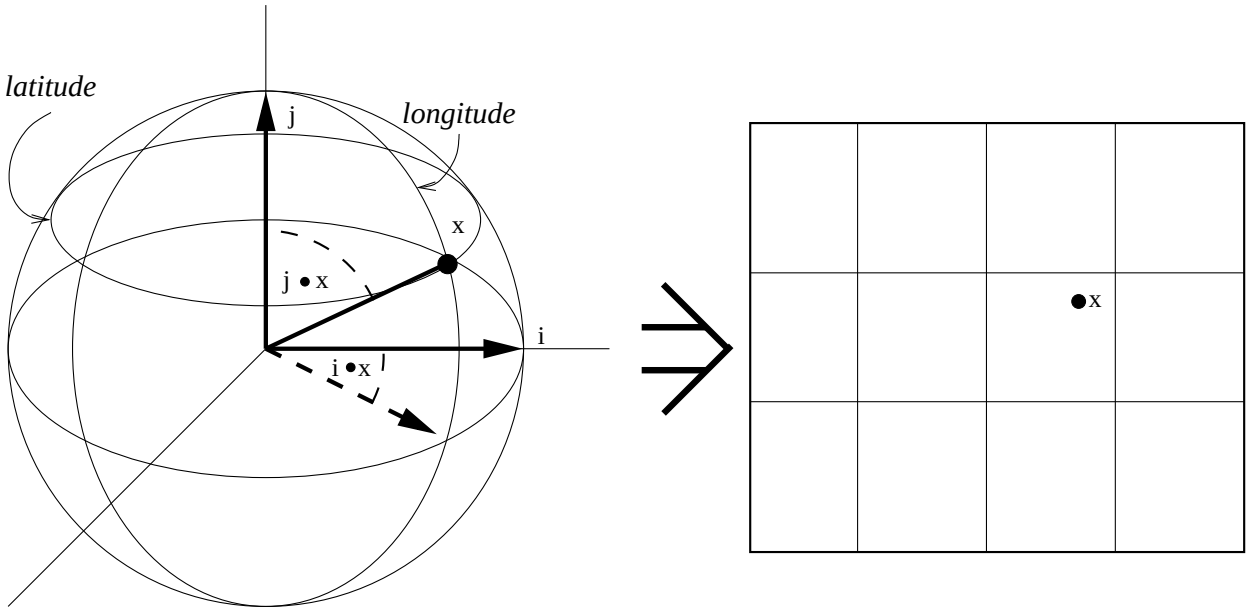
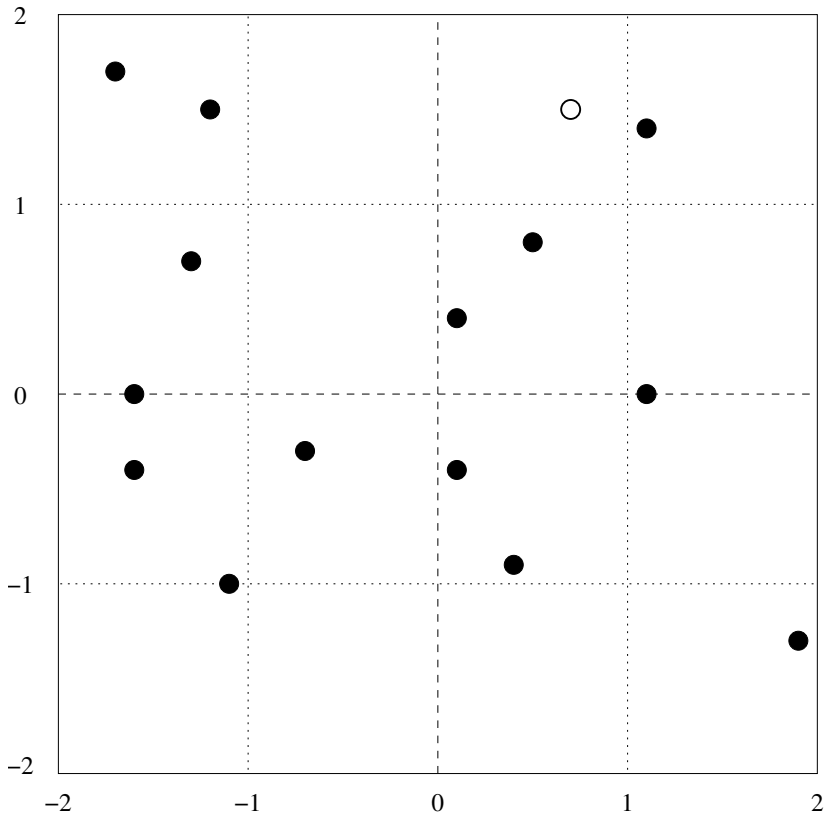


Figure 3.4: Cartographic projection of a sphere onto a plane, the most familiar of which is used in map making. Two vectors are selected, indicated as i and j. For any given point x on the sphere the inner products $i \bullet x$ and $j \bullet x$ are computed. These two quantities become the new coordinates of the point on the map.

projected positions of all five taxon trees are shown in Figure 3.5. In these examples trees are expressed in Newick format, with the taxa represented by numbers and parenthesis to indicate clades. The tree ((1,2),3,(4,5)) is mapped in the following manner. The partition (1,2) has an x value of 1.0 and a y value of 0.9. Likewise the partition (4,5) has an x value of -0.3 and a y value of 0.6. These values are added together to give the final location of the tree ((1,2),3,(4,5)) at the point (0.7,1.5).

The spherical structure of trees in $\mathbb{T}_n$ shown in Theorem 3.3.4, permits the use of cartographic projections. As this class of projections is deterministic, the position of a tree after cartographic projection is deterministic and depends only on the tree in question, thus satisfying the first visualization criterion. Furthermore the space both before and after the projection is a simple Euclidean space where distance is easily calculated, satisfying the second criterion. The results section shows the exploitable structure revealed by the



Partition	x	y	Tree
1,2	1.0	0.9	1
1,3	−0.9	0.9	0
1,4	−0.8	−0.8	0
1,5	0.9	−0.8	0
2,3	−0.8	0.4	0
2,4	1.0	−0.5	0
2,5	−0.8	0.8	0
3,4	−0.5	−0.1	0
3,5	0.1	0.5	0
4,5	−0.3	0.6	1

Figure 3.5: A 2-D cartographic projection of all 5 taxon trees, with reference vectors. The vector for the tree ((1,2),3,(4,5)) is also shown. The point corresponding to this tree is highlighted in the graph.

projection, which satisfies the third criterion. The projection is also reversible, which satisfies the final criterion.

Thus, the hypersphere structure and the use of cartographic projections allow us to represent phylogenetic search in a manner consistent with the original visualization criteria.

3.3.2 Results

The definitions of $\mathbb{T}_n$ and the cartographic projection are deterministic, reversible and have an easily calculated distance metric, fulfilling three of the four criteria for a useful visualization. The fourth criterion, exploitable structure, is the most important. The cartographic projection places similarly scored trees together in the data sets examined. This creates a gradient, an exploitable structure, which allows future work to develop a gradient descent strategy, which would be an improvement over current hill climbing techniques.

Locality of Structure

To have any exploitable structure there must be some correlation between position in the projected space and the topology of the trees near that position. Three methods will be considered: first, the method of Cartographic Projections, second, Multidimensional Scaling in two dimensions as in TreeSetVis [27], and finally Multidimensional Scaling in three dimensions to account for any effects from the extra degree of freedom. The test case will be the exhaustive set of all trees of seven taxa, with each method running 100 times as they all have random elements. Once each projection is calculated, the nearest m neighbors for every tree are found, with m ranging from 0 to 25. A majority rule consensus tree is then constructed for each of these neighborhoods. This tree contains only those partitions which are present in a majority of the trees in a neighborhood. The resolution of these trees is reported, with a value of 1 indicating that the tree was fully resolved and a value of 0 indicating that the tree was fully unresolved.

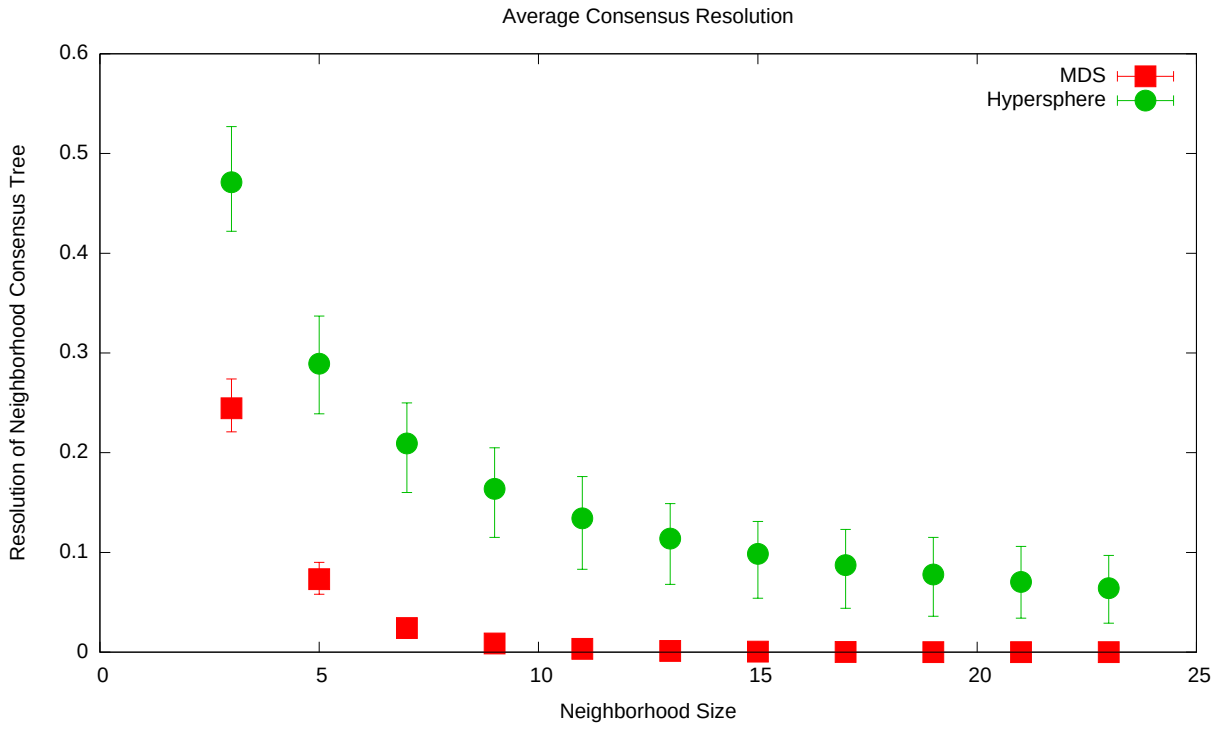


Figure 3.6: The average degree of consensus across near neighbors among all trees with 7 taxa, note that higher values are better. As both cartographic projections and MDS have a random component each point consists of 100 projections with the average, minimum and maximum values for the consensus across all neighborhoods of the given size plotted. MDS was run both in the two dimensional case as in TreeSetVis and in a three dimensional case as the chosen cartographic projection resulted in a three dimensional result.

Figure 3.6 shows the results of this test. The points are plotted with the minimum, average and maximum values for the resolution. Note that cartographic projections are superior to both two and three dimensional MDS in every case. Not only are close trees more structurally similar, but also the neighborhoods over which some degree of topological similarity is found are much larger. It is thus concluded that cartographic projections produce, in terms of topology, a smoother mapping of tree space. Further this superiority is not due to the added flexibility of projecting onto three dimensions rather than two.

Results from Nine Taxon Set Exhaustive Searches

To explore the inherent structure of the maximum parsimony problem, several nine taxon data sets out of BAliBASE[55] were fully analyzed. The data set size was selected because with only $135,135$ possible solution trees, it was very feasible to exhaustively enumerate all solutions for many different data sets of this size and to plot all of the points. Each set was exhaustively enumerated and scored using PAUP* [54]. The three reference points for the projection were chosen at random. Under this projection each of the possible trees mapped to a unique point in the new three dimensional space. The same projection was used for all of the data sets. These points were then colored according to the parsimony score of the corresponding tree, with white indicating a poor score and black indicating a good score.

In all of the data sets, there is significant exploitable structure. In some, such as that shown on the right in Figure 3.7, a clear nearly linear gradient was visible throughout the entire cloud of possible trees. While in others, such as that shown on the left in Figure 3.7, clustering of scores is clear. Even though the gradient was much more complex, it would still be possible to use gradient descent.

Visualizing an Exhaustive Search with MDS

The tool TreeSetViz was used to produce a visualization of a complete data set for comparison with our cartographic projections. Due to the very high memory requirements of multi-dimensional scaling, it was not possible to use a nine taxon data set. An eight taxon subset was used instead. The program was run overnight to allow the program adequate time to converge to the mapping shown in Figure 3.8.

A few features are noteworthy. First, the circular shape, which is a result of the hyperspherical nature of tree space. As all of the trees lie on the surface of a specific sphere, the best MDS solutions are circular. Also the MDS clustering, like the cartographic projection, has a large concentration of good trees. Unlike the cartographic projection of these 8 taxon sets (result not shown), however, the MDS formed two separate clusters and

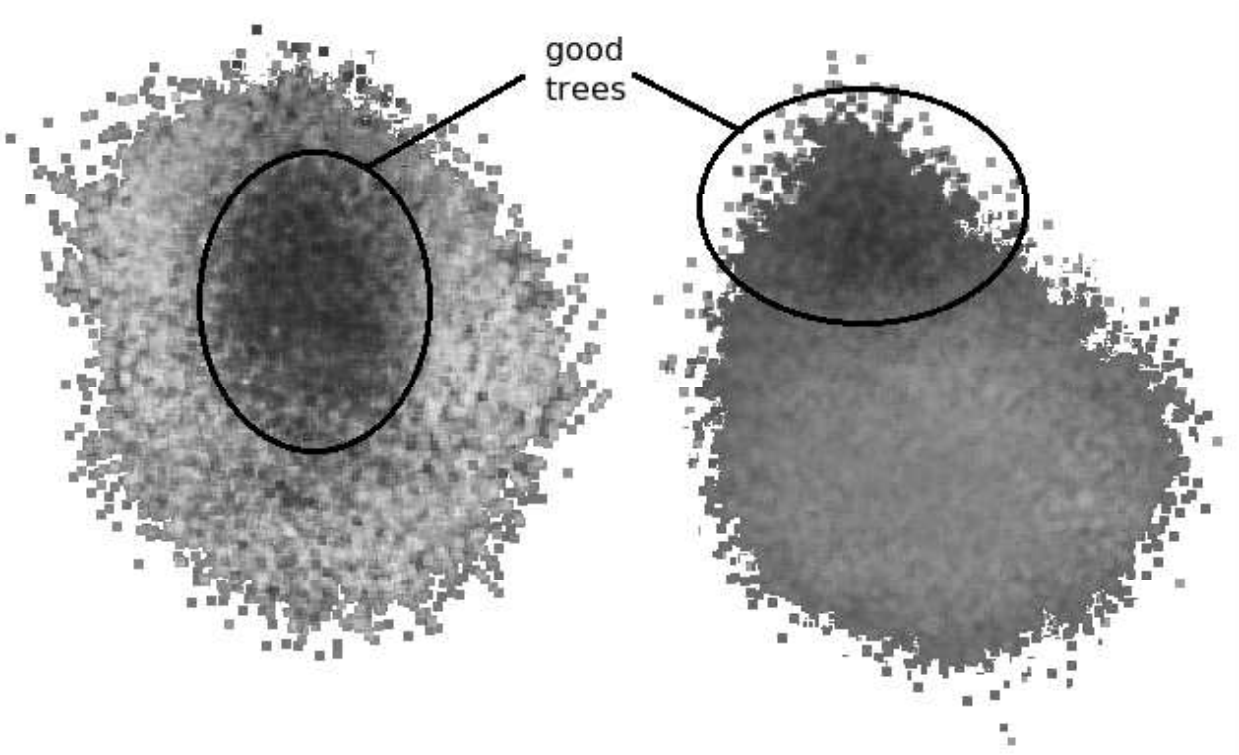


Figure 3.7:]

Two distinct 9 taxon data sets under cartographic projection. Dark points represent trees with better scores. The set on the left shows clear clustering with good trees near the center of the cloud. The set on the right shows a gradient, with good trees at the upper point of the cloud.

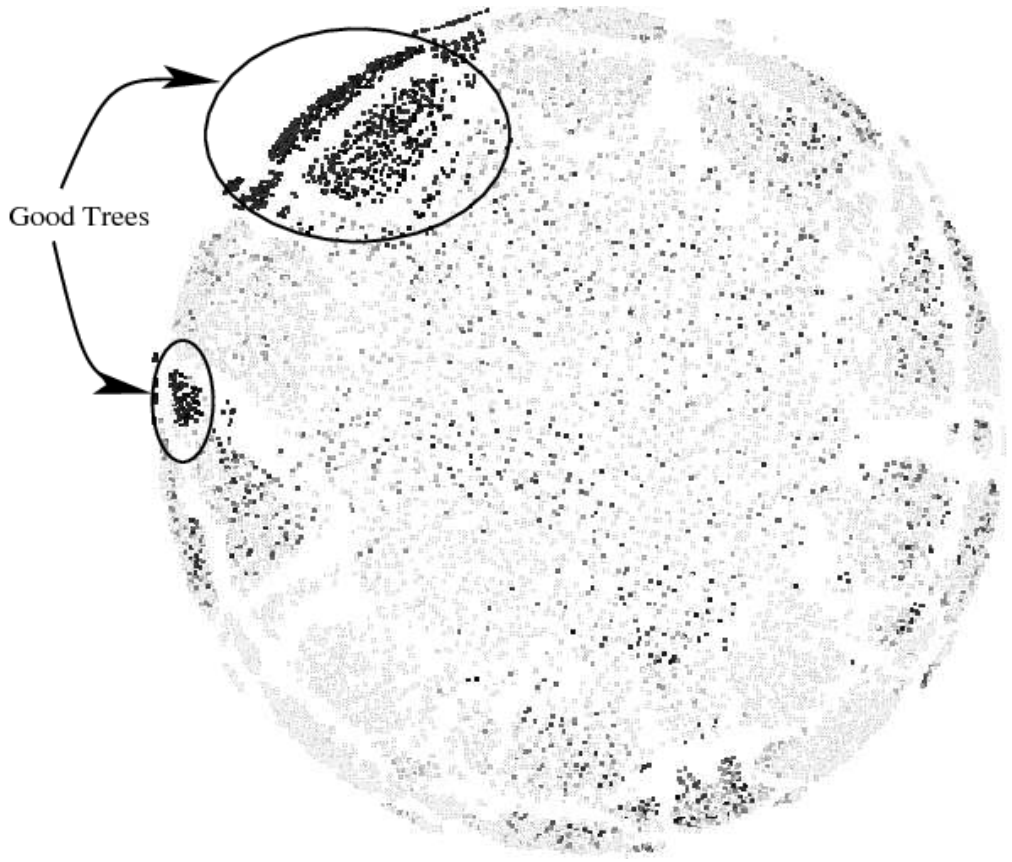


Figure 3.8: A multi-dimensional scaling (MDS) visualization of an exhaustive search of 8 taxa. Dark points represent trees with better scores. Note that there is some clustering of good trees but that they can be found throughout the visualized set.

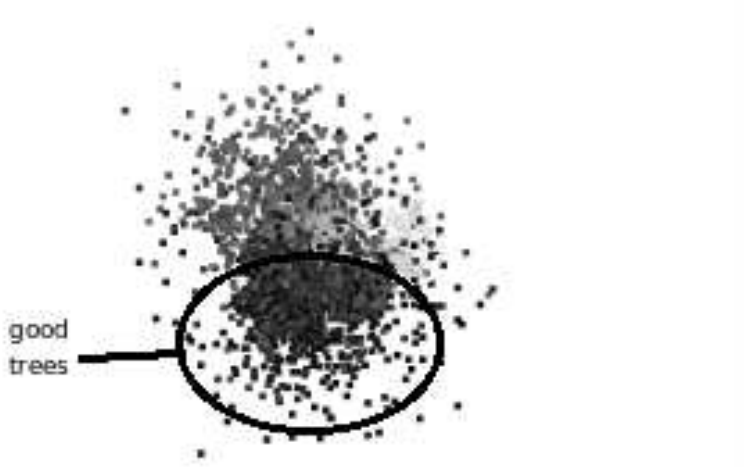


Figure 3.9: Projection of a search through the Zilla 500 taxa data set.

also has a scattering of good trees throughout a large portion of the visualization. Although it is not clear that the clustering of scores caused by MDS is inferior to that of cartographic projections, it is crucial to note that MDS is a post process step and cannot be used to guide a search. Therefore any structure is inherently not an exploitable structure.

Results from Large Data Set Searches

It is not practical to exhaustively search the tree space associated with a large data set. Instead the phylogenetic search program PSODA [8] was modified to output every tree that it was going to perform a TBR rearrangement on, and every 100th rearrangement so produced. This gives not only the path of best trees found by the search as it progressed, but also a sampling of the trees that were rejected.

Figure 3.9 shows a projection of a TBR search with the Zilla data set [10] using cartographic projections. Again, a clustering of scores is apparent among the trees considered by the search, revealing exploitable structure in this difficult data set.

3.3.3 Future Work

The cartographic projection from the hypersphere of trees has revealed significant structure to the problem of phylogenetic search. Further contributions can be made in improving our understanding of the revealed structure. More importantly new search techniques can be developed that can exploit this structure.

Axis Optimization

The current projection from split space to the 3-D visualization is based on the random selection of the points in split space. These points are guaranteed to result in linearly independent reference vectors and are very likely to result in vectors which are orthonormal as well. Although the initial random selection provides encouraging results, a more intelligent selection of basis vectors could improve the quality of the visualization.

Improved Phylogenetic Searches

There are two directions in which to take this work with respect to improving phylogenetic searches by utilizing the structure seen in the visualization. The first is to create a human guided search. As the projection from split space to the visualization is a simple linear transformation, it is possible to select a point in the visualized space and calculate the subspace of split space that corresponds to that point. A tree or trees in that subspace would then be generated and added to the list of trees used in a typical TBR based search, thereby restarting the search from the desired location. The second approach is to calculate and directly use the apparent gradient seen in the visualization to find better trees.

3.4 Conclusions

This cartographic projection from $\mathbb{T}_n$ fulfills all defined criteria for a good visualization. First the mapping from n-trees to $\mathbb{T}_n$ is one-to-one and further the cartographic projection for $\mathbb{T}_n$to $\mathbb{R}^3$ is linear. This means that each tree maps to exactly one point, and this point

is not affected by any outside influences. Also because the mapping is linear, it is reversible, which meets the second criterion. Euclidean distance in $\mathbb{T}_n$ is easy to calculate. Robinson-Foulds distance is also closely related to $\mathbb{T}_n$ as both definitions are based on the partition sets of trees. Either of these distance metrics are easily calculated and meet our third criterion.

More importantly, the use of a cartographic inspired projection has revealed significant structure to the problem of phylogenetic search. The visualization shows a general clustering of trees with similar scores, and in some data sets a clear gradient structure is observed. This promises to be useful in furthering our understanding of the problem of phylogenetic search and for informing the development of new methods in the field. These new methods will expand our ability to perform phylogenetic analysis which has implications for many biological fields.

3.5 Methods

The extremely high dimensionality of $\mathbb{T}_n$ makes explicit storage of the three reference vectors needed for the cartographic projection infeasible. Likewise, due to the size of these vectors the typical calculations used for computing inner products require infeasible amounts of time. A naïve implementation of cartographic projections is adequate for very small numbers of taxa, but more sophisticated techniques are required for most data sets.

3.5.1 Hash Table Vector Representations

The memory usage of a straightforward implementation of cartographic projections is exponential in the number of taxa. Rather than explicitly storing the very large reference vectors a hash table representation is chosen. This representation has a fixed memory size, which can be arbitrarily chosen independently of the number of taxa. A similar hash table was used by Pattengale *et al.*[43] to quickly compute RF distances. Many of the assumptions necessary for computing RF distances, such as a small incidence of collisions, are violated

in this work. However, the similarities do help to explain why the method of cartographic projections does so well at preserving RF distances.

To construct this table, a hash function and three representative vectors of a feasible dimensionality, one for each reference vector, are chosen. The hash function chosen must have a range equal to the set of nonnegative integers up to the dimensionality of the reference vectors and a domain equal to the set of natural numbers up to the dimensionality of the representative vectors.

Together these representative vectors and the hash function are used to compute the elements of the reference vectors as needed. The i^{th} element of each reference vector is defined to be the element of the corresponding representative vector with the hashed value of i as follows:

$$X_i \leftarrow X'_{h(i)}$$

This representation allows a fixed amount of memory to be adequate for data sets of any number of taxa. This bound on memory usage is critical for the visualization of large data sets.

3.5.2 Orthogonality and Normalization of the Reference Vectors

It is desirable that the three reference vectors be orthogonal to each other and also that they be normalized, so that we have an orthonormal basis for visualization. As the dimension of the three reference vectors is very large it is not practical to directly enforce either of these constraints. An additional complication is that each reference vector is not explicitly stored, but is instead implicitly defined by its representative vector and the hash function. Yet, with these constraints it is still possible to make the reference vectors mutually linearly independent and give bounds on their normality and orthogonality. These bounds and their proofs are given later in this section.

If the representative vectors are made to be orthogonal then regardless of the choice of hash function, the true reference vectors are linearly independent by Theorem 3.5.5. The

quality of the orthogonality property of the reference vectors is dependent on the quality of the hash function as shown in Theorem 3.5.6. Given the size of the representative vectors used (65,535 elements) and only 20 taxa the angle between reference vectors is $90 \pm 7.32 \times 10^{-5}$ degrees, very close to orthogonal. As the number of taxa increases this bound becomes even tighter.

Normalizing the reference vectors is more difficult. Due to the finite precision arithmetic of computers, it is not possible to normalize the reference vectors to unit length. As the vectors have a very high dimensionality, normalization tends to make each individual element too small to be represented, which in turn results in all of the reference vectors becoming the zero vector. As an alternative, each representative vector is made to have the same length as the others, without constraining this length to be one.

Again the normalization can only be performed on the representative vectors in the hash table. However Theorem 3.5.7 shows that the reference vectors are also normal if the hash function is perfectly even and gives a bound on how far off of normal the vectors can be in every other case.

The bounds given do not depend on the hash function, so any good hash function should be adequate. Bob Jenkins' *one-at-a-time* hash function [31] was used for the results in this paper.

3.5.3 Calculating the Inner Product

The naïve method of calculating an inner product grows linearly with the dimension of the two vectors involved. Unfortunately, in this case the size of those vectors grows as the combinations of taxa. This method therefore gives worse than exponential performance with respect to number of taxa. However, for any given tree of n taxa, the vector representing that tree will have exactly $n - 3$ non-zero components by Lemma 3.3.3. Furthermore, each of these will be exactly one by the definition of trees in $\mathbb{T}_n$. These two properties can be

exploited to give an algorithm that computes the needed inner products in time $O(n)$, where n is the number of taxa.

This method begins with a hash table. Each element of the hash table contains one element from each of the reference vectors. The keys into the hash table are partition sets. The mapping of a tree is accomplished with the following steps.

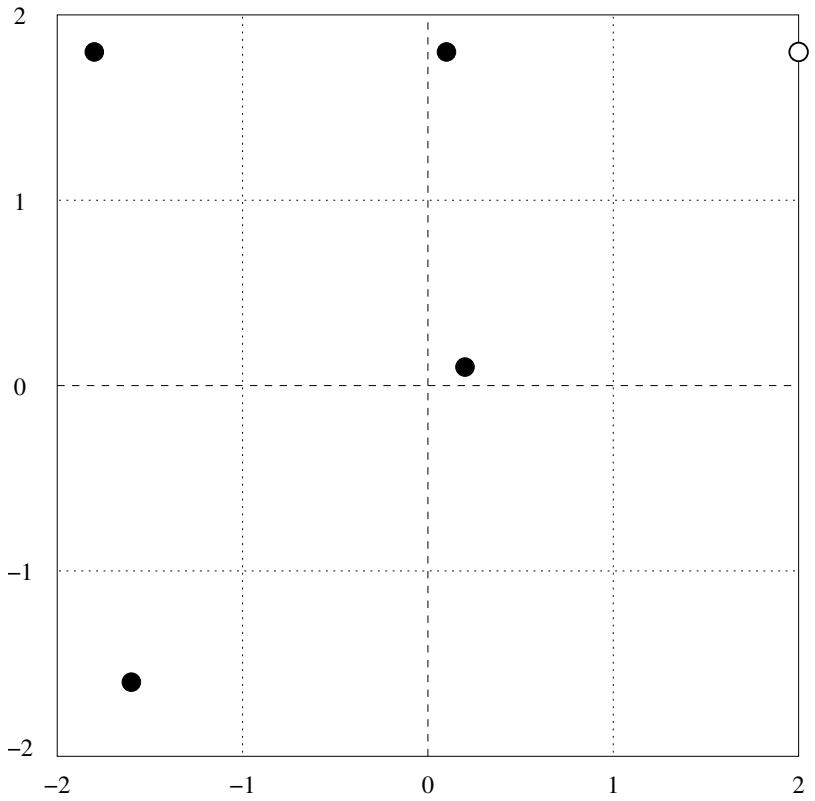
- A list of the $n-3$ partition sets is built : $O(n)$
- Each partition is used to lookup a set of x,y, and z values in the hash table : $O(1)*O(n)$
- The $n-3$ values are summed giving the final mapping : $O(n)$

These steps give an overall runtime execution of $O(n)$.

For example, consider all trees of five taxa numbered 1-5 respectively. Every non-trivial branch has two taxa on one side and three on the other. The hash function will be computed as follows; add the taxon numbers of the two taxa on one side, then divide this sum by three and take the remainder as the value of the hash function. There are three possible values for this hash function 0, 1, and 2. Figure 3.10 shows the full reference vectors. A reference vector is assigned to each of these values. The reference vectors will be axis-aligned unit vectors, the value of 0 will correspond to the vector (1.0,0.9), 1 to the vector (-0.9,0.9) and 2 to the vector (-0.8,-0.8).

This scheme gives six possible locations for each of the fifteen possible trees to map onto, one of which does not correspond to any valid trees. These locations are all shown in Figure 3.10. In this example the tree ((1,2),3,(4,5)) maps to the point (2.0,1.8). The partition (1,2) as well as the partition (4,5) both map to the vector (1.0,0.9), these results are added together to obtain the final location of the tree.

This method has two main advantages. First the time needed to compute the inner product scales with the number of taxa rather than with the dimensionality of split space. Secondly only a fixed amount of storage for the hash table is required, regardless of the



Tree	Partition	Hashed Value	x	y
1	1,2			
0	1,5	0	1.0	0.9
0	2,4			
1	4,5			
0	1,3			
0	2,5	1	−0.9	0.9
0	3,4			
0	1,4			
0	2,3	2	−0.8	−0.8
0	3,5			

Figure 3.10: A 2-D cartographic projection of all 5 taxon trees, with reference vectors represented through a simple modulo 3 hash. Under this simple hash function and small representative table size there are only 5 locations which correspond to valid trees. This projection is therefore not one-to-one. In practice this possibility is mitigated by using a larger table. The vector for the tree ((1,2),3,(4,5)) is also shown. The point corresponding to this tree is highlighted in the graph.

number of taxa in the tree. This upper bound on necessary storage makes the visualization of larger data sets feasible.

3.5.4 Proofs

As givens in all of the following proofs are two vectors X and Y, each of dimension d. These vectors are arbitrarily chosen orthogonal vectors. They are also used to construct two vectors X' and Y', each of dimension d', using a hash function h.

This paper used three vectors of dimension 65535, with elements chosen randomly with a uniform distribution from $[-1, 1]$. Using the Gram-Schmidt method these vectors were all made to be orthogonal to each other. Finally they were each modified to make their magnitudes equal to the magnitude of the first vector.

Definitions

Definition 3.5.1. h is a function with the following properties:

$$range(h) \subset \{\mathbb{N} < d\}$$
$$\{\mathbb{N} < d'\} \subset domain(h)$$

Such a function is easily constructed. One such function when $d < d'$ is $h(x) = x \bmod d$.

Definition 3.5.2. X' and Y' are two vectors constructed from X,Y, and h as follows:

$$X'_i \leftarrow X_{h(i)}$$
$$Y'_i \leftarrow Y_{h(i)}$$

Definition 3.5.3. The frequency with which h maps any number j onto a given number i is

$$f_i = \frac{\sum_{j=1}^{d'} \begin{cases} 1 & h(j) = i \\ 0 & h(j) \neq i \end{cases}}{d'}$$

Note that f_i has the following bounds:

$$\forall i, \frac{1}{d'} \leq f_i \leq \frac{d' - d + 1}{d'}$$

Definition 3.5.4. ξ, the quality of the function h, is a measure of how evenly the elements of X' and Y' are mapped by h onto X and Y.

$$\exists \xi \ s.t. \ \forall i, \frac{d}{d'} + \xi \geq f_i$$

Note that due to the bounds on all f_i, ξ has the following bounds:

$$0 \leq \xi \leq \frac{d' - 2d + 1}{d'}$$

Theorems

Theorem 3.5.5. *Given two orthogonal vectors X and Y, two arbitrarily larger vectors X' and Y' can be constructed such that they are linearly independent.*

Proof. As X and Y are orthogonal, they are also linearly independent. That is to say:

$$\forall s, sX \neq Y$$

$$\forall s \forall i \in \{\mathbb{N} < d\}, sX_i \neq Y_i$$

$$\forall k \exists j, X'_k = X_{h(j)} \quad \text{Definition 3.5.2}$$

$$\forall k \exists j, Y'_k = Y_{h(j)}$$

Thus all equations of the form

$$sX_j \neq Y_j : j \in range(h)$$

can be rewritten as

$$sX'_k \neq Y'_k : k \in domain(h)$$

In this fashion

$$\forall s \forall i \in \{\mathbb{N} < d\}, sX_i \neq Y_i$$
$$\forall s \forall j \in \{\mathbb{N} < d'\}, sX'_j \neq Y'_j$$
$$\forall s, sX' \neq Y'$$

From which it is clear that X' and Y' are linearly independent. □

Theorem 3.5.6. *Given two orthogonal vectors X and Y, two arbitrarily larger vectors X' and Y' can be constructed such that they are orthogonal within a given bound.*

Proof. Using Definition 3.5.3 the inner product $\langle X'|Y'\rangle$ can be written in terms of X and Y.

$$\begin{aligned}\langle X'|Y'\rangle &= \sum_{i=1}^{d'} X'_i Y'_i \\ &= d' \sum_{j=1}^{d} f_j X_j Y_j\end{aligned}$$

As X and Y are orthogonal, their inner product $\langle X|Y\rangle = 0$; therefore either

$$\forall i, X_i Y_i = 0$$

and clearly

$$\forall \xi, \langle X'|Y'\rangle = 0$$

or

$$\exists i, X_i Y_i > 0$$

$$\exists j, X_j Y_j < 0$$

In this second case it may not be true that X' and Y' are orthogonal. Even so there are bounds on $\langle X'|Y'\rangle$. The largest possible magnitude of $\langle X'|Y'\rangle$ occurs when h maps each member of $\{\mathbb{N} < d\}$ to one member of $\{\mathbb{N} < d'\}$ with the exception of one element of $\{\mathbb{N} < d\}$ which maps to the remaining elements of $\{\mathbb{N} < d'\}$. Furthermore, that sole exception corresponds with the largest magnitude of $X_i Y_i$. In this case the inner product is given by

$$\begin{aligned}
\langle X'|Y'\rangle &= \frac{d'-d}{d'}\sum_{i=1}^{d}\frac{1}{d'}X_iY_i + \frac{\xi}{d'}\operatorname{argmax}_j X_jY_j \\
&= \frac{d'-d}{dd'}\sum_{i=1}^{d}X_iY_i + \frac{\xi}{d'}\operatorname{argmax}_j X_jY_j \\
&= 0 + \frac{\xi}{d'}\operatorname{argmax}_j X_jY_j \\
\langle X'|Y'\rangle &\leq \frac{d'-2d+1}{{d'}^2}\operatorname{argmax}_j X_jY_j
\end{aligned}$$

The angle θ between X' and Y' is given by

$$\cos\theta = \frac{\langle X'|Y'\rangle}{|X'|\,|Y'|}$$

Applying the bound on $\langle X'|Y'\rangle$, and the bounds on the magnitudes of X' and Y' from Theorem 3.5.7

$$\cos\theta \leq \frac{\frac{d'-2d+1}{d'^2}\operatorname{argmax}_j X_jY_j}{|X|\,|Y|}$$

□

As X and Y are arbitrary but constant expressions, note that

$$\lim_{d' \to \infty} \cos\theta = 0$$

Therefore as the number of taxa increases the vectors in question approach orthogonality.

Theorem 3.5.7. *Given two vectors of equal magnitude X and Y, two arbitrarily larger vectors X' and Y' can be constructed such that they are also of equal magnitude within a given bound.*

Proof. As X and Y are of equal magnitude it is the case that

$$\sqrt{\sum_{i=1}^{d} X_i^2} = \sqrt{\sum_{i=1}^{d} Y_i^2}$$

The magnitude of X' is bounded above by

$$\begin{aligned} |X'| &= \sqrt{\sum_{i=1}^{d'} {X'}_i^2} \\ &= \sqrt{d' \sum_{i=1}^{d} f_i X_i^2} \\ &\leq \sqrt{d' \sum_{i=1}^{d} \left(\frac{d}{d'} + \xi \right) X_i^2} \\ &\leq \sqrt{d + d'\xi}\, |X| \end{aligned}$$

As the range of h is in $\{\mathbb{N} < d\}$ every element of X is also an element of X'. Therefore

$$|X| \leq |X'|$$

The magnitude of Y' is bounded in the same fashion. The ratio of the two magnitudes is bounded as follows

$$\frac{1}{\sqrt{d + d'\xi}} \leq \frac{|X'|}{|Y'|} \leq \sqrt{d + d'\xi}$$

Additionally, if $\xi = 0$ then

$$\begin{aligned} |X'| &= \sqrt{d \sum_{i=1}^{d} X_i^2} \\ &= \sqrt{d}\,|X| \\ |Y'| &= \sqrt{d \sum_{i=1}^{d} Y_i^2} \\ &= \sqrt{d}\,|Y| \end{aligned}$$

and the two vectors have equal magnitude □

Chapter 4

Partition Space Sectorial Search

4.1 Abstract

Phylogenetic search is an important tool in biology. It is known to be NP-complete and therefore requires the use of heuristic methods. Currently, the most common methods are greedy hill climbing heuristics that rely exclusively on local information. This work introduces Partition Space Sectorial Search (PSSS), an algorithm that creates a global representation of the search space. This representation allows the algorithm to keep track of searched regions and thereby avoid duplication of effort. The reduction in effort in turn leads to faster search times and better scores.

4.2 Introduction

Phylogenies have become a central bioinformatic tool for sequence analysis and testing of subsequent evolutionary hypotheses[28]. However, as phylogenetic search is NP-complete[14], it is necessary to employ heuristic search techniques to efficiently estimate phylogenies. The most commonly used heuristic methods consist of a tree permutation operator and a greedy tree selection strategy. This approach is limited in that it only considers local information, the scores of neighboring trees, at each optimization step. This difficulty is compounded by the presence of 'tree islands'[37]. These islands consist of groups of trees which are all connected by simple branch rearrangements and are equally optimal. Existing local information approaches can easily become trapped by these islands.

All phylogenetic trees can be described as sets of taxon partitions. Recent work[52] has described a method of visualizing global structures in this partition space. This work explores the direct application of partition space to phylogenetic searching, and proposes a new strategy for defining the tree search space Partition Space Sectorial Search (PSSS). The systematic approach used by PSSS is significantly different from the *ad hoc* approach used by many phylogenetic inference algorithms. PSSS improves results by seeking to maximize the number of unique, previously unexamined solutions found at each step. Additionally, the behavior of PSSS can be viewed as an estimated gradient descent search, implying the attendant property of convergence.

4.3 Related Work

The inference technique used by most phylogenetic search programs is a form of hill climbing. An initial tree is selected and then permuted to form new trees. The best of these trees is in turn permuted and the process continues until no further improvements are found.

4.3.1 Tree Bisection and Reconnection

A very common method of permuting trees during a phylogenetic search is Tree Bisection and Reconnection[1] (TBR). In this method a tree is divided into two subtrees. Then every tree which can be formed by reconnecting these subtrees is examined. A tree with n taxa, has $O(n)$ branches which can be removed to bisect the tree. Likewise, each subtree has $O(n)$ points where the subtrees could reconnect. Overall there are therefore $O(n^3)$ trees that can be reached from any given tree using TBR.

A problem with TBR is that, between two trees there are many sets of TBR operations which taken in combination result in the same tree. Thus, even though each step examines $O(n^3)$ trees, these trees are not unique and therefore only a limited amount of tree space is explored.

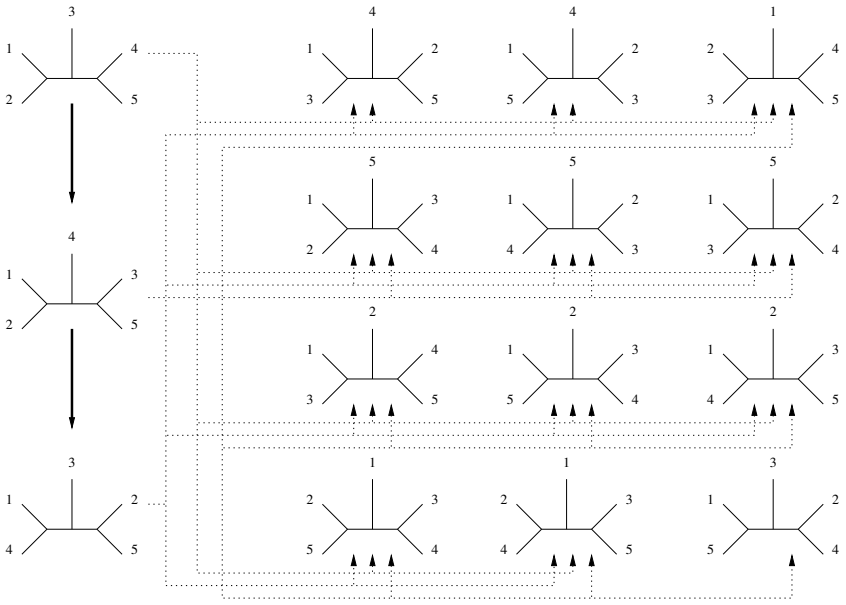


Figure 4.1: A graphical representation of a TBR based search through a 5 taxa dataset. The dark arrows represent the progress made by the search, the dotted arrows represent examinations of trees that are not selected by the search. Note that 36 examinations are made of the 15 trees.

For example, consider a TBR based search in a data set of five taxa. There are 15 possible trees. Each of these trees can be permuted through a TBR rearrangement into 12 other trees. Further suppose that the best tree and the starting tree can not be permuted into each other through a single TBR rearrangement. In the first iteration 12 new trees will be examined, one of which becomes the new starting tree. On the second iteration 12 trees are again examined, but only 2 of them are new. One of these new trees is the best tree, and is therefore selected as the new starting tree. Finally on the third iteration 12 trees are examined, none of which are new. As no further improvement is found the search terminates. At this point 36 trees have been examined, but only 15 of those 36 examinations were of previously unexamined trees. Figure 4.1 graphically shows this example search.

The lack of a global representation leads to this duplication of effort. The typical TBR based search makes no effort to remember trees which have already been examined. This directly results in an inability to make any attempt at avoiding duplicated effort.

4.3.2 The Ratchet

An important technique for avoiding local minima during a phylogenetic search with a large number of taxa is the ratchet [40, 56]. In this approach at various intervals the data set is perturbed by duplicating a random subset of the character columns. The search continues under this perturbed data set. These perturbations tend to focus the search on a subset of the taxa, which hopefully allows the search to escape from a local minima. Finally the data set is returned to its original state where the search proceeds as normal. This process can be repeated several times.

The ratchet is effective at escaping local minima. However, like the TBR search that the ratchet is based on, no attempt is made to avoid duplication of effort. Again, this is due to a lack of any form of global representation of tree space resulting in duplicate scoring of identical trees resulting in an inefficient exploration of the tree space.

4.3.3 Sectorial Search

Sectorial search[21] is a divide and conquer strategy for phylogenetic search. Like TBR, this method begins with a tree which is permuted into a set of trees. The best tree found is used in further iterations. Unlike TBR, Sectorial Search only permutes a contiguous portion of the tree, called a sector. The permutation used is TBR, with the constraint that the points of bisection, and reconnection lie within the chosen sector.

As TBR yields $O(n^3)$ neighbors, that must each be checked, this is an effective means of reducing the effort required to make improvements. However, as this method forbids movement across the sector boundary there are some TBR transitions that are never checked. This is overcome by periodic global TBR swapping.

Like the previously reviewed methods, sectorial search has not used any representation of global search information. Without such a representation, sectorial search is prone to the same duplication of work that other methods suffer from. Partition Space Sectorial Search

(PSSS) uses a representation of the phylogenetic search space to avoid duplication in sectorial search.

4.3.4 Cartographic Projections

Cartographic projections[52] have been shown to be an effective means of reducing the dimensionality of tree space, while preserving structure. The result of the mapping is a k-dimensional space which contains all possible trees. This space is the basis for the proposed PSSS technique.

First all trees are considered as sets of taxon partitions. Partitions which divide one taxon from all of the other taxa, being common to all trees, are trivial and therefore ignored. Every possible partition is associated with a dimension in tree space. Since there are many possible partitions, this space has a very high dimensionality. The coordinates of any tree in this space are 1 for dimensions associated with partitions in the tree, and 0 for all other dimensions. As all trees have the same number of partitions, it can be shown[52] that all resolved trees lie on the surface of a hypersphere centered at the origin. This allows the use of cartographic projections to reduce the dimensionality of the tree space to something more manageable. Figure 4.2 shows this space for the simple case of 4 taxon trees. In the 4 taxon case there are three possible non-trivial partitions. There are four possible trees. The first lies at the origin, and is fully unresolved containing no non-trivial partitions. There are also three resolved trees, each containing one of the three possible partitions. These trees are all of the possible solutions, and it is easily seen that they lie on the surface of a sphere centered on the origin.

Cartographic projections map a point on a hypersphere to a point on a hyperplane. The first step is to select one reference vector for each dimension of the desired hyperplane. This work uses three reference vectors, yielding a three dimensional space. Once the reference vectors are selected, the projection is simple. The coordinates of a tree after projection are

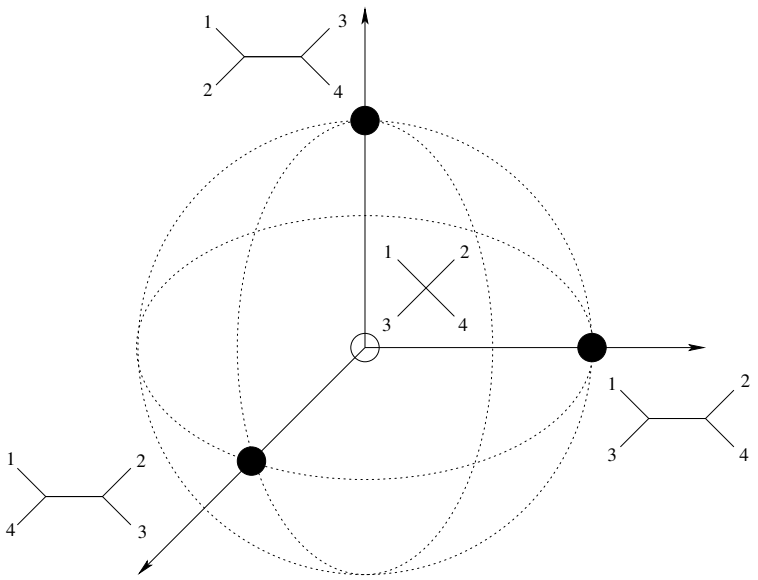


Figure 4.2: A depiction of all 4 taxon trees. Each possible partition is assigned a dimension. The three resolved 4 taxon trees, each contain only one of the possible partitions. The coordinates of a tree are 1 along the dimension associated with a partition contained in a tree, and 0 for the other dimensions. All resolved trees lie on the surface of a sphere centered at the origin, this sphere is shown in the figure.

respectively the inner products of the trees location with the reference vectors. Further detail on this method of projection can be found in our previous work on this subject[52].

As the dimensionality of the original space is very large, a hash table is used to compute these dot products. The hash table is built prior to mapping any tree. This table contains random small 3 dimensional vectors. The magnitude of all of the components to these vectors is uniformly distributed between -1.0 and 1.0. This table is used to map bipartitions of the n taxa, represented as n-bit binary numbers, onto this set of vectors. To map a tree, first the tree is decomposed into a set of bipartitions on the taxa. Each of these bipartitions corresponds with a branch in the tree. The hash table is then used to look up a vector for each of these bipartitions. Finally, the tree is mapped to the sum of these vectors.

The purpose of cartographic projections in this work is to guide the progress of a phylogenetic search. It is therefore desirable for trees that are close together to be topologically similar. One method of comparing tree topologies is the Robinson-Foulds (RF) distance[45].

This distance is the number of partitions in one tree that are not in the other. The dimension reduction technique used in this mapping is very similar to that used by Pattengale *et al.* [43] to compute RF distances. One of their results is that, provided collisions in the hash table occur infrequently, the distance between projected trees will be very close to the square root of the RF distance between those trees. This provides the desired property, trees which are topologically similar are close to each other after this projection.

This projection allows for a compact global representation of the trees examined by a search. To avoid duplicate effort, it is sufficient to focus the search of a volume of tree space which has not been examined. It is not requisite to store any specific tree, which would be prohibitive, only to mark off volumes which have been examined. The details of the method used to accomplish this are given in the next section.

4.4 Methods

As the position of a tree in cartographic tree space is closely associated with tree topology, it is possible to construct a search method which minimizes the duplication of effort common to TBR based searches. The key is that the range of an iteration of sectorial search has a distinct and finite shape. This volume, containing all possible trees resulting from the search of a sector, we call a sector search volume (SSV). To avoid duplicate effort it is sufficient to select a sector with an SSV with minimal overlap with previous SSVs (see Theorem 4.7.10).

4.4.1 The Overall Algorithm

The following summarizes the PSSS algorithm. This method is also freely available as part of the open source phylogenetics package PSODA[8], available at `http://dna.cs.byu.edu/psoda/`.

- Begin k separate searches with a maximum stepwise parsimony tree
- Repeat until convergence:

- For each search:
    - Divide current tree into non-overlapping sectors with minimal SSV overlap
    - Search each sector independently
    - Perform m iterations of global TBR
    - Randomly re-weight characters
    - Perform n iterations of global TBR
    - Reset character weights

4.4.2 The Sector Search Volume

A first step to avoiding SSV overlap is to describe the shape of these volumes. As a first approximation an SSV is bounded by a cube (see Theorem 4.7.11).

The SSV of a sector depends only on the size of the sector, and the branches which lie outside of the sector. Thus, it can be computed for a sector without performing any portion of the sectorial search it describes. This allows the search to be guided, without requiring any additional trees to be examined.

It is much easier to record the SSV than the many trees produced by the search of a sector, as there are $O(s!)$ trees in a sector. In this implementation $s \approx 100$, which is a very large number of trees that can be represented by a single vector, the center of the SSV. This compact representation makes it practical to represent areas of the global space which have been examined. At each iteration a sector can be chosen to minimize overlap of the new SSV and previous SSVs. In this fashion, duplicate effort is avoided.

4.4.3 Computing SSV Overlap

Within the cube which bounds an SSV the trees are not distributed uniformly. The trees tend to be concentrated toward the center of the SSV (see Theorem 4.7.12). Given a region of tree space which we would like to search, it is possible to compute the probability that a

tree generated by searching in a given SSV will also lie in a previously examined SSV. This probability of duplicated effort is the SSV overlap.

SSV overlap cannot be avoided completely. Each tree which is used to build a sector, other than the very first, is contained in at least two SSVs. The first is the SSV that was searched to find this tree. The second is the SSV of the sector built from this tree, as every SSV contains the tree it was built from.

We begin by finding the probability of a tree from a given SSV being mapped to any given location. The probability of an arbitrary tree from a sector search mapping to a given location is (see Theorem 4.7.12):

$$p(x, \mu, s) = \frac{1}{\sqrt{\frac{2}{3}\pi(s-1)}} e^{-\frac{3(x-\mu)^2}{2(s-1)}} \tag{4.1}$$

The value μ is derived from the topology of the tree and the vertices in a given sector.

Knowing the probability of a tree from one SSV mapping to a given location, we now find the probability of a tree from two SSVs mapping to the same location. Let μ_a be the mean of the probability function from the first SSV, and μ_b the mean from the second. Note that these values are the locations of the characteristic trees for the two sectors. Also we let s be the size of both sectors, which is true for the PSSS algorithim.

The probability of a tree from both SSVs mapping to the same location is then the product Equation 4.1 for each SSV:

$$\begin{aligned} p(x, \mu, s) &= \frac{1}{\sqrt{\frac{2}{3}\pi(s-1)}} e^{-\frac{3(x-\mu_a)^2}{2(s-1)}} \frac{1}{\sqrt{\frac{2}{3}\pi(s-1)}} e^{-\frac{3(x-\mu_b)^2}{2(s-1)}} \\ &= \frac{3}{2\pi(s-1)} e^{-\frac{3(\mu_b-\mu_a)^2}{2(s-1)}} e^{-\frac{3(x-\frac{\mu_a+\mu_b}{2})^2}{2(s-1)}} \end{aligned} \tag{4.2}$$

Finally, integrating across all possibilities, the cumulative distribution function is:

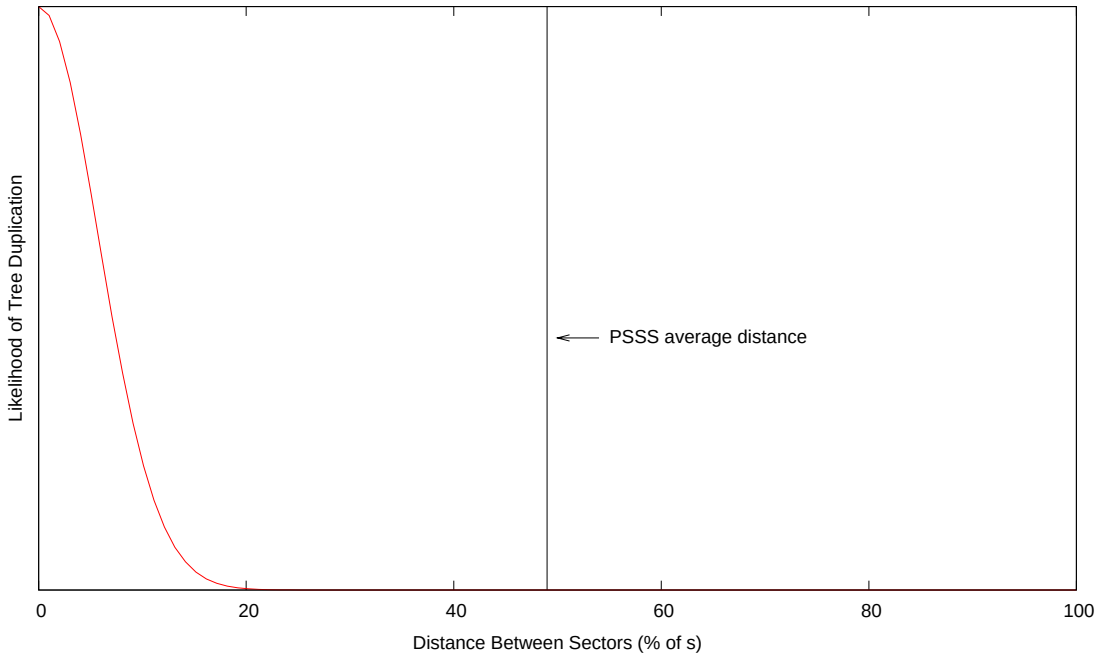


Figure 4.3: A graph of Equation 4.3. As the distance between two sectors increases the probability of a tree duplication decreases. Also plotted is the average distance between sectors achieved by PSSS.

$$\begin{aligned} cdf(\mu_a, \mu_b) &= \int_{-\infty}^{\infty} \frac{3}{2\pi(s-1)} e^{-\frac{3(\mu_b-\mu_a)^2}{2(s-1)}} e^{-\frac{3(x-\frac{\mu_a+\mu_b}{2})^2}{2(s-1)}} dx \\ &= \frac{3}{\sqrt{6\pi(s-1)}} e^{-\frac{3(x-\frac{\mu_a+\mu_b}{2})^2}{2(s-1)}} \end{aligned} \tag{4.3}$$

This value is the SSV overlap. Equation 4.3 is shown in Figure 4.3. Again, it is important that this value depends only on the two sectors to be examined, not on the results of that examination. This value can be quickly calculated without examining any trees, which allows it to be used during a search with very little overhead.

4.4.4 Selecting Sectors to Minimize Overlap

As shown in Equation 4.3, the probability of the search of one sector examining a tree that has been previously examined in the search of another sector is inversely related to

the distance between the centers of the SSVs for those two sectors. Thus, the amount of duplicate work done during a search can be minimized, by maximizing the distance between the SSV of a new sector to be examined and previously examined SSVs. Selecting a new sector of size s in an optimal fashion requires a search through the $O(s!)$ available sectors. Rather than search for this optimal sector a simple greedy heuristic is used.

This heuristic assumes that the distance between the current tree and the center of old SSVs is greater than the magnitude of any single vector in the hash table. Provided this is true, the addition of any branch with an associated vector whose dot product with the vector between the old SSV and the tree to be considered is positive, will increase the distance between the new SSV and the old SSV. That is to say that:

$$|\mathbf{x}| > |\mathbf{a}| \wedge \mathbf{x} \cdot \mathbf{a} > 0 \rightarrow |\mathbf{x} + \mathbf{a}| > |\mathbf{x}|$$

Figure 4.4, graphically shows why this is so. If this assumption is not true, it is because the search made little progress in the previous iteration. This indicates that the search is near a minima and a further search in this area may be in order. Thus if the assumption is not satisfied, either a distant sector will still be selected or a near sector that should also be searched will be selected. This forms the basis of a greedy heuristic for selecting sectors whose SSVs have small overlaps with previous searches.

The algorithm for forming a new sector from a tree is as follows.

- Find the vector $\mathbf{x}$ between the old SSV and the current tree. $O(1)$
- Find the branch, whose vector $\mathbf{a}$ has the largest dot product with $\mathbf{x}$. $O(n)$
- Put each of this branch's neighbors into a priority queue, with the priority based on the dot product with $\mathbf{x}$. $O(\log n)$
- Do s -1 iterations of: $O(s \log n)$
  - Remove a branch from the priority queue. $O(\log n)$

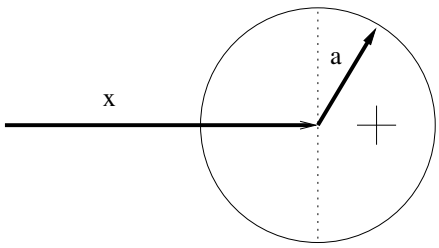


Figure 4.4: An example of the dot product heuristic for maximizing the separation of searched SSVs. The vector $\mathbf{x}$ lies between the previous sector and the current tree, $\mathbf{a}$ is the vector associated with a branch being considered for inclusion in a new sector. Provided that $||\mathbf{x}|| > ||\mathbf{a}||$ all vectors $\mathbf{a}$ where $\mathbf{x} \cdot \mathbf{a} > 0$ increase the distance between the new and old SSVs.

- Add this branch to the sector. $O(1)$
- Put each of the neighbors of this branch into the priority queue. $O(\log n)$

4.4.5 Dividing a Tree into Multiple Sectors

When possible, it is advantageous to divide a tree into multiple sectors (subgraphs). It is important that the set of branches included in all of these sectors is disjoint from all other sectors. This allows the sectors to be refined independently from one another. Also, as the sectors are disjoint, only the original tree is common to the range of these searches. Therefore the overlap of the respective SSVs is negligible.

Finding these non-overlapping sectors is very similar to the algorithm for finding one sector; however, a few additional steps are required. The branches added to a sector are marked as used. Then, before a neighboring branch is added to the priority queue this mark is checked. Only branches that have not yet been used are added.

4.4.6 Using Multiple Search Paths

It is also helpful to use multiple stepwise maximum parsimony trees to begin the search. This can be useful in avoiding local minima and is a well known technique. There is one important variation used by PSSS; each replicate search is aware of the space searched by the others. Rather than processing the searches sequentially, one iteration of the search is

processed for each replicate. This helps to prevent duplication of effort across these replicate searches as each search avoids not only the SSVs which it has searched, but also those SSVs which have been searched by any of the other replicates. The final tree is the best tree found by any of the searches.

4.5 Results

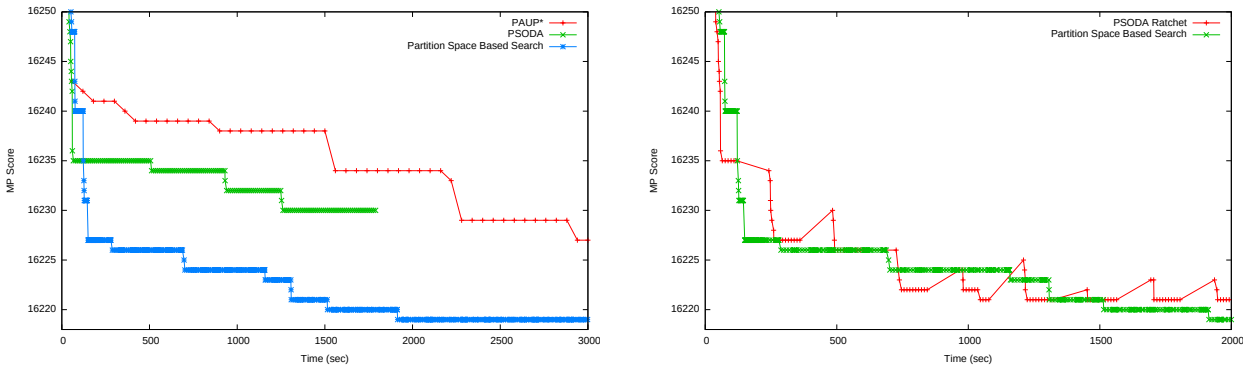
The results presented here are of two types. First, on the Zilla data set[10], a number of qualitative comparisons are made. These comparisons examine the behavior of different portions of the algorithm and the algorithm as a whole. Second, the PSSS algorithm is compared with PAUP*[54], a widely used search program, on a large set of synthetic data and a smaller selection of real data sets.

4.5.1 Results From Individual Techniques

This method relies on the combination of several techniques which together are more useful than any of the individual techniques. These techniques are: TBR, the Ratchet, and Partition Space Sectorial Search. PSSS is compared against TBR, and TBR with the Ratchet to show the additional benefit gained by the use of the new method. These detailed comparisons are shown only on a phylogenetic search through the Zilla[10] data set. A summary of results on 100 separate data sets is shown in Figure 4.9.

Figure 4.5a compares the searches of PAUP*[54], PSODA[8] using TBR, and PSODA using PSSS. The PSODA program using TBR terminates early, giving an inferior tree as its final result. However, using the PSSS algorithim, PSODA is able to outperform PAUP* in both the score of the final tree found, and the speed at which the result is obtained.

The ratchet[40] is critical to escaping local minima during the global TBR phase of the search. Figure 4.5b compares the results of using only global swapping with the ratchet and the partition based sectors. There are two advantages to the partition based method



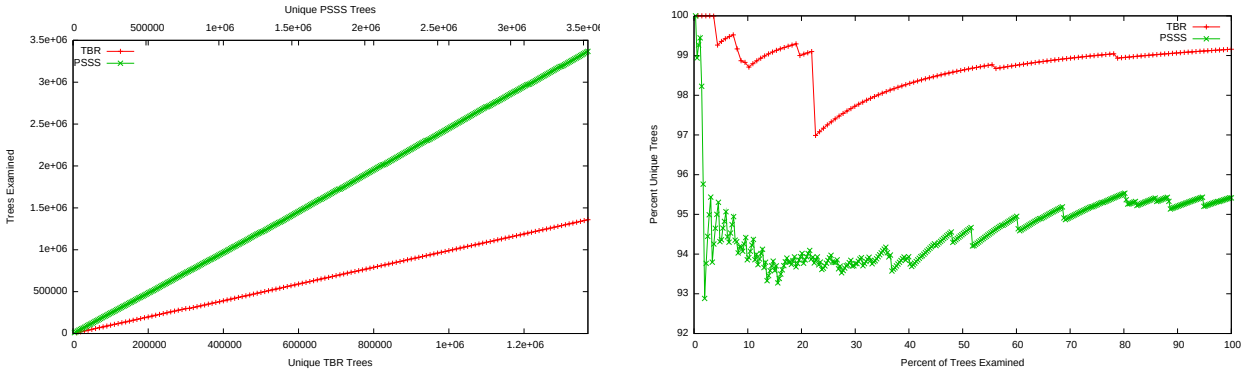
(a) (b)

Figure 4.5: A comparison of the maximum parsimony score of trees found during a phylogenetic search on the Zilla[10] data set. Figure A compares PSSS to a typical TBR search. Two implementations are shown, first the implementation of TBR in PAUP*[54] and second the implementation in PSODA[8] which is used internally by PSSS. Figure B compares the internal implementation of the ratchet[40] and the full PSSS search.

shown. First, the progress is much smoother as the scores are monotonically decreasing. Second, the partition based method finds the optimal score before the ratchet does.

4.5.2 Searching Unique Trees

As an empirical test of the ability of PSSS to examine more unique trees than traditional TBR based methods, the following experiment was performed. PSODA was altered to emit a canonical string representation of every tree examined. Then PSSS and a standard TBR search were each run for twelve hours. Figure 4.6 summarizes the results. In this time period PSSS examined 2.48 times the number of unique trees examined by the TBR search, as shown in Figure 4.6a. However, over this time period TBR does maintain a higher percentage of unique trees, shown in Figure 4.6b. This is likely due to the global swapping phase of PSSS. While this phase is necessary to permit branches to move across sector boundaries, there is no effort made to avoid duplicate effort during this phase. Furthermore, many of the trees considered during global swapping, will have already been considered during the prior sectorial searches.



(a) (b)

Figure 4.6: A comparison of the number of unique trees examined by TBR and PSSS over a twelve hour period. As shown in (a) PSSS examines a much larger number of unique trees over this time period. However, TBR examines a larger percentage of unique trees, shown in (b).

4.5.3 Search Behavior

Figure 4.7 is a representation of the behavior of the partition based search. As the search progressed, the centers of every SSV examined by the search was noted. As a post processing step, the distance to the nearest SSV center for all of tree space was found. This distance, by Equation 4.3, is directly proportional to the probability of a tree at that location having been examined during the course of the search. These distances are shown as contour lines through the space.

By Equation 4.3, trees closer to the centers of examined sectors are more likely to have been examined than trees near the edges. This figure does not account for the possibility of a tree having been examined by searching a sector that does not have the closest SSV. Therefore the actual coverage of tree space is greater than that shown by the figure.

It has been noted[52], that under cartographic projection there seems to be a clear gradient of tree scores. This gradient can be seen in Figure 4.7, in the progress made by the search through tree space over time. No particular effort is made by the search to follow this gradient structure. Rather, the search examines the area of the tree space which is close to the current tree and least likely to have already been examined. However, as long as the

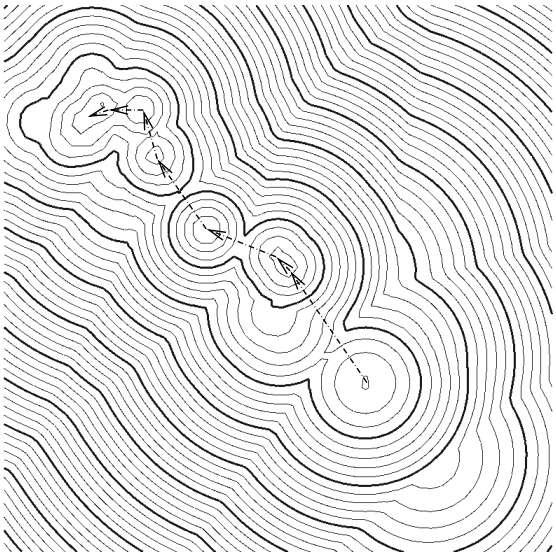


Figure 4.7: The coverage of cartographic tree space. The contours represent distance from the center of the nearest sector searched, these distances correspond to the probability that the tree at that location has been examined by the search. The progress of the search over time is marked with arrows. Of note is the path the search takes through this space over time as it implicitly follows a gradient.

search is making progress it will act like a gradient descent search. Each step of the search which improves the score can be thought of as an estimate of the local gradient. Because the search will avoid reexamining trees, it will seek an SSV which is as far away as possible from the previously examined SSVs. This will be 'downhill' with respect to the estimated gradient, leading to gradient descent behavior.

4.5.4 Results on Real Data Sets

In addition to Zilla, PSSS was run on a few real data sets of varying sizes. The results are shown in Figure 4.8. PSSS does well on the two smaller data sets, but not as well on the larger data sets. This is in large part due to the poor starting tree produced by PSODAs implementation of stepwise maximum parsimony. When a better start tree is provided, for example by PTM (see Chapter 5) better results are produced.

Dataset	Taxa	PSSS		PAUP*	
		Score	Time	Score	Time
RDPII	218	**33513**	2:04:01	33565	0:01:28
ZILLA	500	**16219**	1:03:16	16221	15:42:19
U	6722	143569	2:59:23	**93106**	20:10:42
ARB	8780	222369	6:57:39	**162906**	29:13:33

Figure 4.8: A comparison of search results between PSSS and PAUP* on several datasets. On smaller data sets PSSS does very well, though it struggles on the larger sets. In part this is due to a slower implementation of stepwise maximum parsimony in PSODA. When PSSS is used in conjunction with PTM (see Chapter 5) better results are obtained for the large data sets.

4.5.5 Results on Synthetic Benchmarks

A large set of synthetic data was produced using the DAWG[9] program. The data sets generated ranged from 200 to 2000 characters and from 100 to 1000 taxa. The number of characters was increased in increments of 200, and the number of taxa in increments of 100, yielding 100 data sets. The trees used to produce the data sets were randomly generated. Figure 4.9 summarizes the results. In this test PSSS found the same or better tree in less time than PAUP* on 82 of 100 data sets. If the sets are divided into two groups, those with at least 1000 characters and those with less than 1000 characters, another pattern emerges. On the larger data sets PSSS is faster in 96.7% of the data sets, but is faster in only 60.0% of the smaller sets. PSSS does better when the cost of duplicate scoring offsets the overhead of the PSSS method. The cost of duplicate scoring, which PSSS is designed to avoid, is proportional to the number of characters in the data set. Thus it is not unexpected that the utility of the PSSS method increases with the number of characters.

4.6 Conclusion

The PSSS algorithm shows a number of promising characteristics. First, performance is better than many popular phylogenetic search programs. Second, its behavior is not *ad hoc* like so many current techniques but is designed with respect to cartographic tree space.

	Total	PSSS		PAUP*
All	100	**82**		18
≥1000 char	60	**58**	(96.7%)	2
< 1000 char	40	**24**	(60.0%)	16

Figure 4.9: A summary of results from PSSS and PAUP* on 100 sets of synthetic data. Reported are the number of data sets on which each program was the first to find the best tree. PSSS performs much better on the data sets with more characters as the cost of duplicate scoring rises for these data sets.

Third, as a result of this design the method acts like a gradient descent search. In the future it may be possible to use this technique to design a true gradient descent search algorithm for phylogenetic trees. Finally, as the search is able to minimize duplicate effort, a larger number of unique trees are examined, lending a greater confidence to the results obtained.

4.7 Proofs and Definitions

This section contains formal definitions of terms used in this work.

Definition 4.7.1. Tree: A tree is a connected acylclic graph with no vertices of degree two. A tree is **resolved** if its vertices are only of degree one or three. The edges of this graph are also called **branches**. The vertices of degree one are called **leaves**.

Lemma 4.7.2. *All trees with n leaves have $2n - 3$ branches.*

Proof. See Waterman[57], Proposition 14.1. □

Definition 4.7.3. Sector: A sector is a connected subgraph of a resolved tree.

Definition 4.7.4. Characteristic Tree: The characteristic tree of a sector is a unique unresolved tree defined by the sector. This tree can be constructed in the following manner. Let V be the set of vertices in a tree which are not in the sector but are adjacent to a vertex which is in the sector. Remove from the tree all vertices in the sector and all edges which connect to any vertex in the sector. Add a new vertex x to the tree. Also, for every vertex v in V add the edge (v, x) to the tree. The result is the characteristic tree for the sector.

Lemma 4.7.5. *A sector with s vertices on a tree with n leaves has a characteristic tree with $2n - s - 2$ edges.*

Proof. By definition a sector is a subgraph of a resolved tree. This tree with n leaves has $2n - 3$ branches (Lemma 4.7.2). As this tree is acyclic and the sector is connected there are $s - 1$ edges which connect two vertices in the sector. The characteristic tree contains one edge for every edge connecting vertices which are not in the sector. This characteristic tree also contains one edge for every edge which connected a vertex in the sector with a vertex outside of the sector. This tree does not contain any edges corresponding to edges connecting two vertices in the sector. Thus the characteristic tree contains $2n - 3$ less $s - 1$ branches, for a total of $2n - s - 2$ branches. □

Definition 4.7.6. Resolution: A resolved tree is a resolution of an unresolved tree if the resolved tree can be iteratively constructed from the unresolved tree using the following operation. Select vertex v of at least degree four. Call the set of vertices directly connected to v, G. Remove v and all edges between v and any member of G from the graph. Add two new vertices v_1 and v_2 and the edge (v_1, v_2) to the graph. Finally, for each element g of G add either (v_1, g) or (v_2, g) such that v_1 and v_2 are at least degree 3.

Theorem 4.7.7. *Any tree found in a sectorial search is a resolution of the characteristic tree of that sector.*

Proof. Consider a resolved tree T. Also consider a sector $S = \{V, E\}$ which is a subgraph of T. Let G be the set of vertices such that $g \in G$ if $\exists v \in V \wedge (g, v) \in T$. Now, consider T' a tree found during a sectorial search on S. By definition a sectorial search only affects S, changing it to some S'.

Using Definition 4.7.4 the characteristic tree for S and S' can be built. In this construction, the sector is removed and replaced with a single vertex. As the only difference between T and T' is S and S', the characteristic tree R for both S and S' is identical. By

reversing this construction R can be resolved into T'. Thus, any tree found in a sectorial search is a resolution of the characteristic tree of that sector. □

4.7.1 Images Under Cartographic Projections

Cartographic projections are used to build a representation of the global tree space. This section covers the properties of images of various tree constructs under this projection.

Definition 4.7.8. Properties of the Cartographic Projections:

- The projection maps branches to vectors in $\mathbb{R}^n$
- The components of these vectors are uniformly distributed from $[-1, 1]$
- Trees are projected to the sum of the projections of their component branches, a point in $\mathbb{R}^n$
- All trees lie in $\mathbb{R}^n$, also referred to as global tree space

  See Chapter 3 for details.

Definition 4.7.9. Sector Search Volume: A Sector Search Volume (SSV) is a bounding volume in $\mathbb{R}^n$ which contains the image of all trees which could result from a sectorial search on a given sector.

Theorem 4.7.10. *If the SSVs of two sectorial searches do not overlap, there is no tree which will be examined by both searches.*

Proof. As SSVs are bounding volumes and contain all trees which could be examined by a search, the image of any tree which was examined by two separate searches must lie in the intersection of the SSVs. If this intersection is empty then no such tree exists. □

Theorem 4.7.11. *An SSV is a hypercube.*

Proof. By theorem 4.7.7, every tree resulting from a sectorial search is a resolution of the characteristic tree for that search. This characteristic tree has $2n - s - 2$ edges (Lemma

4.7.5), where n is the number of leaves and s is the number of vertices in the sector. A resolution of this tree has $s-1$ more branches. Each of these branches is mapped to a vector in $\mathbb{R}^n$, and the image of a tree is the sum of these vectors. A property of cartographic projections is that the components of these vectors are uniformly distributed from $[-1.0, 1.0]$ (see Definition 4.7.8). The sum of $s-1$ such components must therefore lie in the interval $[1-s, s-1]$. As this interval applies to all components of the sum of these branches, the final image of any tree resulting from a sectorial search is bound by this hypercube. By definition this hypercube is the SSV for the sectorial search. The center of this hypercube is the sum of the vectors for the branches of the characteristic tree.

□

Theorem 4.7.12. *The images of trees resulting from a sectorial search are normally distributed within the SSV of that search.*

Proof. Consider an arbitrary tree which is the result of searching some given sector. As in Theorem 4.7.11, $2n-s-2$ edges will be determined by the characteristic tree of the sectorial search, while $s-1$ will be drawn from a uniform distribution from $[-1, 1]$. The mean of a uniform distribution from $[a, b]$ is $\frac{a+b}{2}$ and the variance of such a distribution is $\frac{(b-a)^2}{12}$. Thus the mean of these $s-1$ branches is 0 while their variance is $\frac{1}{3}$. The central limit theorem states that these summed components are themselves drawn from a normal distribution. This distribution can be calculated using this theorem and is

$$p(x) = \frac{1}{\sqrt{\frac{2}{3}\pi(s-1)}} e^{-\frac{3(x-\mu)^2}{2(s-1)}} \tag{4.4}$$

Where $p(x)$ is the probability of a tree mapping to location x and μ is the image of the characteristic tree. □

Chapter 5

Partial Tree Mixing

5.1 Introduction

Phylogenetic search is an NP-Hard[11] problem. It is however important to the analysis of biological sequences and the testing of evolutionary hypothesis[28]. As such it is necessary to employ heuristic methods.

A phylogenetic search begins by using a greedy heuristic to build an initial tree. This initial tree is then improved by the full search. Unfortunately, the greedy nature of the starting trees limits the effectiveness of the full search. For this reason multiple starting trees are often used, with the hope that at least one will allow the overall search to find the global minimum.

Partial Tree Mixing (PTM) addresses this issue through the use of a global representation of partition based tree space[52]. Using this representation PTM is able to quickly begin exploring this space with a global search strategy. PTM uses a strategy focused more on exploration than exploitation. By covering more of the solution space PTM leads to an increased chance of the overall search finding a global minimum. A couple of key features of PTM allow these goals to be accomplished. PTM divides a problem into smaller, more manageable subproblems, this allows for global search methods such as TBR to be applied sooner. Also, PTM uses a global representation of all possible solutions, this allows for coordination between the subproblem search efforts.

5.2 Related Work

The most common heuristic method for phylogenetic search is a form of hill climbing. A given possible solution is permuted into several new solutions. The best of these solutions is in turn permuted until no better solutions are found.

The most common permutation operation is Tree Bisection and Reconnection (TBR)[1]. Common methods in current use for building an initial tree include distance based methods such as UPGMA (Unweighted Pair Group Method with Arithmetic Mean)[39] and neighbor joining[48] , as well as stepwise maximum parsimony. Both distance methods and stepwise maximum parsimony are $O(n^2)$ algorithms.

5.2.1 Distance Methods

Distance methods begin by computing an all-to-all distance matrix between the taxa. This is typically the hamming distance between the DNA character sequences for each taxa though some other metrics have been used[36]. The nearest taxa are joined into a clade. Then the distance from this clade to all other taxa is computed. The differences between the two most used algorithms, neighbor joiningand UPGMA, are in how this new distance is calculated. This clustering of taxa into clades continues until a complete tree has been built.

5.2.2 Stepwise Maximum Parsimony

Stepwise maximum parsimony begins by shuffling the taxa into a random order. The first three taxa are joined together into the only possible three taxon tree. In turn each taxon is inserted along every branch in the current tree. It is left in the most parsimonious position. This process continues until all the taxa have been added, resulting in a complete tree.

5.2.3 Tree Bisection and Reconnection

Tree Bisection and Reconnection (TBR) is a common means of generating new solutions during a phylogenetic search. Each iteration of TBR is an $O(n^3)$ algorithm and produces

$O(n^3)$ trees to be examined. The first step is to select a branch in the tree and remove it, producing two subtrees. A branch is then selected in each of the two subtrees. A new tree is produced by reconnecting the two subtrees at the selected branches. An iteration of TBR ends when the original tree has been split along every branch and each of those splits has been rejoined in all possible ways. If one of the new trees is better, then the search continues by performing a TBR iteration on the improved tree. If no better tree is found the search ends.

5.3 Partition Based Tree Space

Trees can be considered as collections of bipartitions of taxa. Every branch in a tree divides the taxa into two sets. Some of these bipartitions, those arising from branches connected to the leaves, are common to all trees. These trivial bipartitions are ignored. All other possible partitions are assigned a dimension in tree space. The position of a tree is a vector whose components all have the value 1 or 0. These values respectively represent the presence or absence of the associated bipartition.

In this space there is a close relationship between the Euclidean distance between two trees and the Robinson-Foulds(RF)[45] distance between those same trees. Namely the Euclidean distance is the square root of the RF distance.

5.3.1 The Hypersphere of Trees

It is well known[57] that all fully resolved trees of n taxa have $2n-3$ branches. n of these branches are trivial, and are therefore ignored. The position of any resolved tree will therefore have exactly $n-3$ elements with the value of 1, all others will have the value of 0. It is easy to see that the distance from this point, the position of an arbitrary fully resolved tree, and the origin is $\sqrt{n-3}$. As all such points are equidistant from the origin, it is the case that every fully resolved tree lies on the surface of a hypersphere.

Unresolved trees are trees which have fewer than $n-3$ non-trivial branches. Consider a tree lacking m branches, by the same argument as used for resolved trees, the distance between this tree and the origin must be $\sqrt{n-m-3}$, and all such trees lie on the surface of a smaller concentric hypersphere of radius $\sqrt{n-m-3}$.

The set of all trees, both resolved and unresolved lie upon the surfaces of a set of $n-3$ concentric hyperspheres. At the origin lies the fully unresolved tree, which possesses no branches. The next sphere out, with a radius of 1, contains all trees with 1 branch. Each succeeding sphere contains trees with one more branch in them than the last sphere, until the final sphere of radius $n-3$ is reached.

5.3.2 Cartographic Projections

The dimensionality of tree space is $O(n!)$, with respect to the number of taxa. Directly representing trees in this space quickly becomes prohibitive. One method of mitigating this explosive dimensionality is through cartographic projections[52]. A small number of reference vectors are chosen in tree space, these vectors need not correspond to a valid tree. The coordinates of a tree are then defined as the inner products of the vector representing the tree and these reference vectors. Due to the very sparse nature of a vector which represents a tree, these inner products can be computed with a single pass over the tree in $O(n)$ time. The method used to store the reference vectors is a hash table, and this has been shown[43] to preserve the relationship between Euclidean and RF distance.

5.4 Methods

Partial Tree Mixing (PTM) is intended to initialize a search through a data set with a large number of taxa. A concern with current methods is that they take $O(n^2)$ steps before any searching can occur. When n is small this is not problematic, especially as no prior methods have proposed any other solutions to initializing a TBR-based search. While PTM takes

more than $O(n^2)$ steps before handing over an initial tree to a TBR-based search, it is able to begin global searching after only $O(n \log n)$ steps.

5.4.1 Overview of Partial Tree Mixing

Partial Tree Mixing is a divide and conquer strategy for building an initial search tree. A primary goal of PTM is to use partial trees (see Definition 5.7.2), containing only a subset of the taxa to search tree space. By keeping the number of taxa small, PTM is able to search faster than traditional methods.

Unlike previous methods, PTM is not a greedy heuristic. Although it employes heuristic techniques, PTM uses a representation of the global search space to insure that a large portion of the space is explored. This global representation is based on considering trees as collections of bipartitions[52]. Each bipartition is associated with a dimension. The location of a tree in this tree space is determined by which bipartitions are in the tree. As a result topologically similar trees are close together. A thorough exploration of this space therefore leads to a thorough examination of possible topologies. This reduces the necessity of finding multiple starting trees to avoid local minima.

The PTM method is based on the idea that an unresolved tree is an approximation of all the resolutions (see Definition 5.7.3) of that tree. This is a reasonable assumption as the unresolved tree contains the information which is common to all of its resolutions. The quality of the approximation depends on the degree of resolution of the unresolved tree. The fully unresolved tree contains no information about any of its resolutions, while the fully resolved tree contains perfect information about its resolution. However, while the quality of the approximation increases as the degree of resolution increases the number trees which are represented by the approximation decreases. PTM leaves the size of partial trees, and therefore the degree of resolution, to the user. Section 5.5.1 discusses the effects of varying this parameter. The region of the global tree space which contains all of these resolutions is the image (see Definition 5.7.6) of the unresolved tree.

During tree mixing, unresolved trees are chosen which have images covering new portions of tree space. As the partial trees are kept small, many of these exploratory searches can be accomplished in a small amount of time. Although this exploratory effort is important to the success of PTM, the partial trees are constrained to only consider improvements throughout the process.

Figure 5.1 shows a graphical overview of the PTM process. First the taxa are divided into disjoint sets and the initial partial trees are built (Section 5.4.3). Then the partial trees mix together, exploring the global tree space (Section 5.4.4). Finally the partial trees are joined to build a fully resolved tree (Section 5.4.5), which can then be passed on to the usual TBR-based search.

5.4.2 Tree Images

The number of taxa in a partial tree is inversely related to the volume of its image. Conversely the larger the number of taxa, the more similar all of the resolutions of the partial tree become. In turn, this makes the partial tree a more detailed representation of the trees in its image. Thus, there is a classic exploration/exploitation trade off. At one extreme lies a fully unresolved tree, whose image covers all of tree space with no information about the trees in the image; at the other lies a fully resolved tree, whose image has zero volume (a point), but has perfect information about the tree.

PTM seeks a balance between these two extremes, partial trees which are large enough to contain useful information yet small enough to cover a large portion of tree space. As a first attempt we consider selecting partial trees such that their images tile tree space. Unfortunately, this approach becomes quickly intractable. Covering all of tree space is simple only in the degenerate case of the fully unresolved tree. Adding even a single branch to the fully unresolved tree is sufficient to make this approach impractical. Adding a branch is equivalent to selecting a bipartition of the taxa. If there are n taxa there are $O(n!)$ bipartitions to choose from. This is the case for the addition of subsequent branches also.

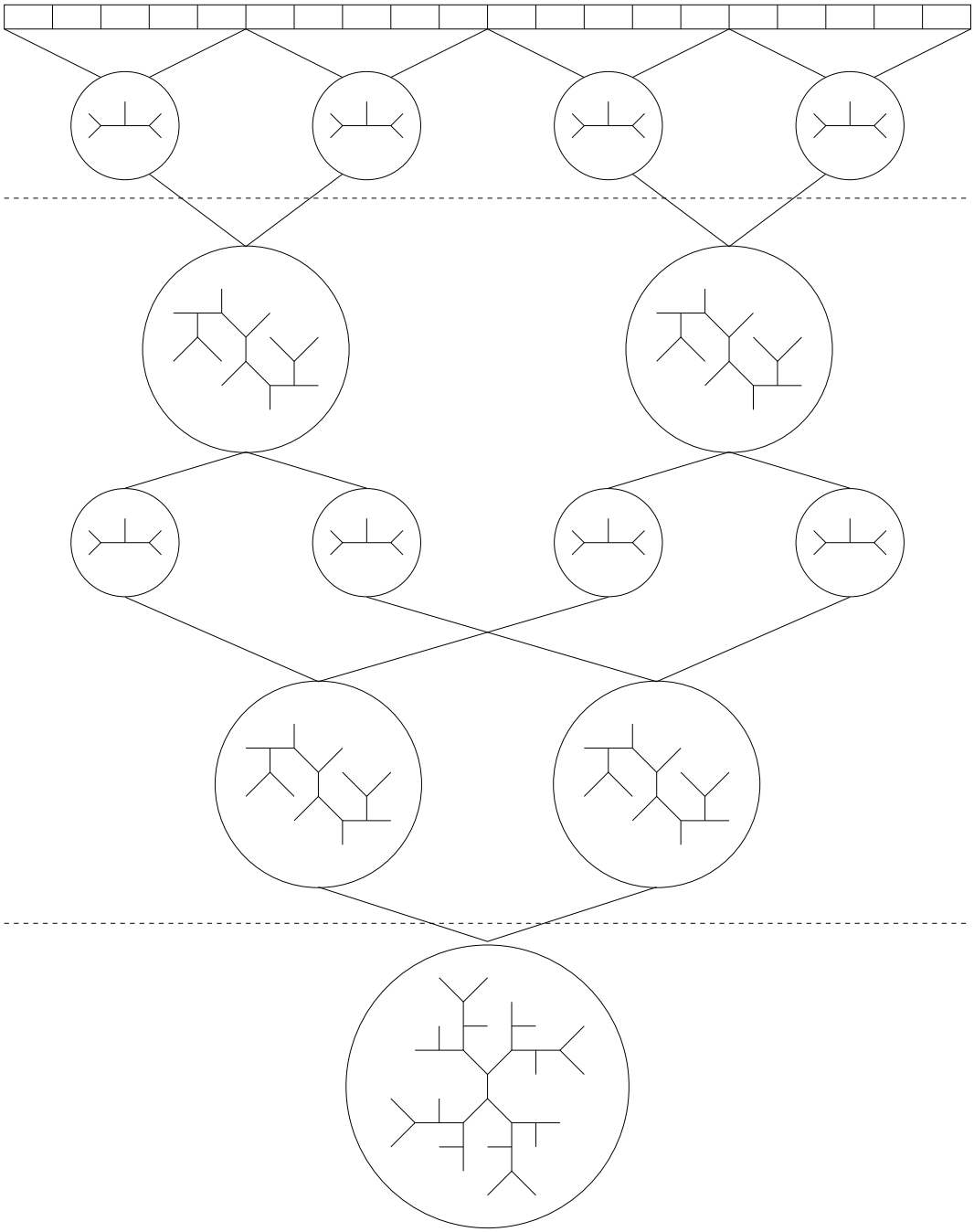


Figure 5.1: A brief overview of the PTM algorithim. In the first phase the taxa are sorted and grouped into small disjoint sets. A stepwise maximum parsimony tree is built from each of these sets. In the second phase these trees are repeatedly joined, refined, and divided. The division of trees is identical to the tree bisection portion of the TBR algorithm. Likewise, the joining of these trees is identical to the tree reconnection portion of TBR. For this joining to work, it is essential that no taxa is represented twice. To insure this, during a PTM search all leaves on all partial trees are uniquely labeled. In the final phase no division occurs. Thus, the trees continue to grow in size until a tree containing all of the taxa is produced.

This very large number of possible partial trees to choose from makes it unlikely that an analytical solution to tiling tree space with partial trees will be found.

Rather than seeking an optimal analytical solution, PTM uses heuristic methods to guide its exploration of tree space. A population of partial trees is maintained, each independently searching for a solution. This is very reminiscent of genetic algorithms. However, there are a few important differences. First, this population of partial trees is constructed such that at any time all of the partial solutions could be combined to form a complete solution. Second, each member of the population represents a partial solution not a complete solution. Third, while PTM does include exchanges between population members, the source of new solutions is not mutations and crossovers. Instead the partial solutions are refined using a hill climbing search strategy.

5.4.3 Initial Partial Trees

To begin the PTM algorithim the taxa are first divided into small disjoint subsets. An effort is made to place similar taxa into the same subset. This is done by computing a pairwise distance between an arbitrary taxon and all others. As taxa are usually given as DNA character sequences this distance is an edit distance between the two sequences. The taxa are then placed into a priority queue using this distance. Next the taxa are drawn off this queue in nearly even groups of 50-100 taxa. This $O(n \log n)$ method avoids the high costs of other distance methods. Then the initial partial trees are formed using stepwise maximum parsimony on these much smaller data sets. These initial trees are finally refined using TBR. As these partial trees have fewer than 100 taxa a local minima can usually be found in a few seconds. Thus the first searching of tree space occurs after $O(n \log n)$ steps, much sooner than under traditional methods which are $O(n^2)$.

Algorithm 1 Initial Partial Trees

Require: $\forall$ taxon, taxon $\in$ taxa
Ensure: PartialTrees $\neq \emptyset$
  taxa.sort()
  while ! taxa.empty() **do**
    PartialTrees.push(new PartialTree(taxa.getTaxaSet())
  end while

5.4.4 Tree Mixing

Once PTM has a set of disjoint locally optimal partial trees, the search progresses via tree mixing. In this process two partial trees are joined to form a new partial tree. This tree is refined with TBR to find a local minima. The optimized partial tree is then divided again into two new partial trees. These trees in turn join with others. This both keeps the size of each tree small, so that TBR is effective, and allows information to diffuse through the system.

Partial trees never join with their siblings from the previous division as this results in no progress. Beyond this constraint, they are free to join with any other partial tree. Partial trees remember where the tree they split from was located, and seek partners to join with that will place the new combined tree as far from the old combined tree as possible. The purpose of this preference is as a heuristic method to cover as much of the hypersphere of trees as possible with the images of the larger partial trees.

The image of a joined partial tree encompasses the intersection of the images of its member trees. Figure 5.2 shows this relationship graphically. The larger partial tree has a smaller image than the partial trees of which it is composed. Though smaller, this image is a more accurate representation of the quality of trees in that region of tree space. It is therefore important to cover as much of tree space as possible with these higher quality images. This is accomplished by building new partial trees as far away as possible from previous partial trees. This distance helps to encourage the exploration of tree space.

It is not necessary to remember the location of old partial trees from iterations other than the immediately proceeding iteration. While the image of a partial tree contains all

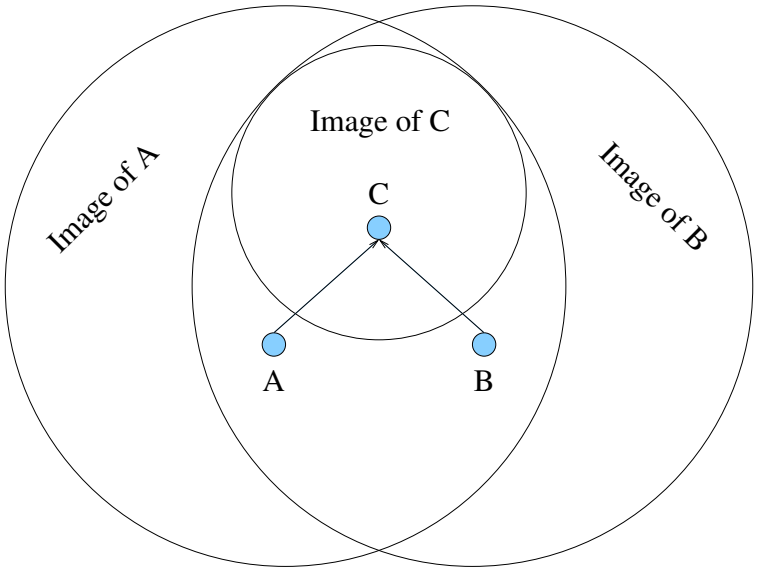


Figure 5.2: A depiction of the effects of partial tree joining on the images of the partial trees involved. Partial trees A and B are combined to form partial tree C. A and B have fewer branches than C, therefore they can be resolved into more trees and each has a larger image than C. The image of C is contained in the intersection of the image of A and B, as any resolution of C is also a resolution of A and a resolution of B. Although the image of C is smaller it is more detailed, as C is more resolved.

resolutions of that tree, it is not the case that no other trees lie within this region of tree space. It is unlikely that a partial tree whose image has a large overlap with the image of a previously considered partial tree contains no new trees. Additionally, as the search progresses the overall quality of the partial trees being used improves. It may be helpful to reexamine an area covered by an old image in light of this new information.

Figure 5.3 shows a comparison of the number of iterations used and the score found by PTM. The spike in this graph around 5 iterations occurs as PTM finds a large local minima which TBR has a difficult time escaping. With fewer iterations this minima is not found, and with more iterations it is escaped. The optimal number of iterations is data set dependent. In this work we used three iterations which worked well across the data sets tested. This small number of iterations greatly reduces any concern of duplicating effort from prior iterations.

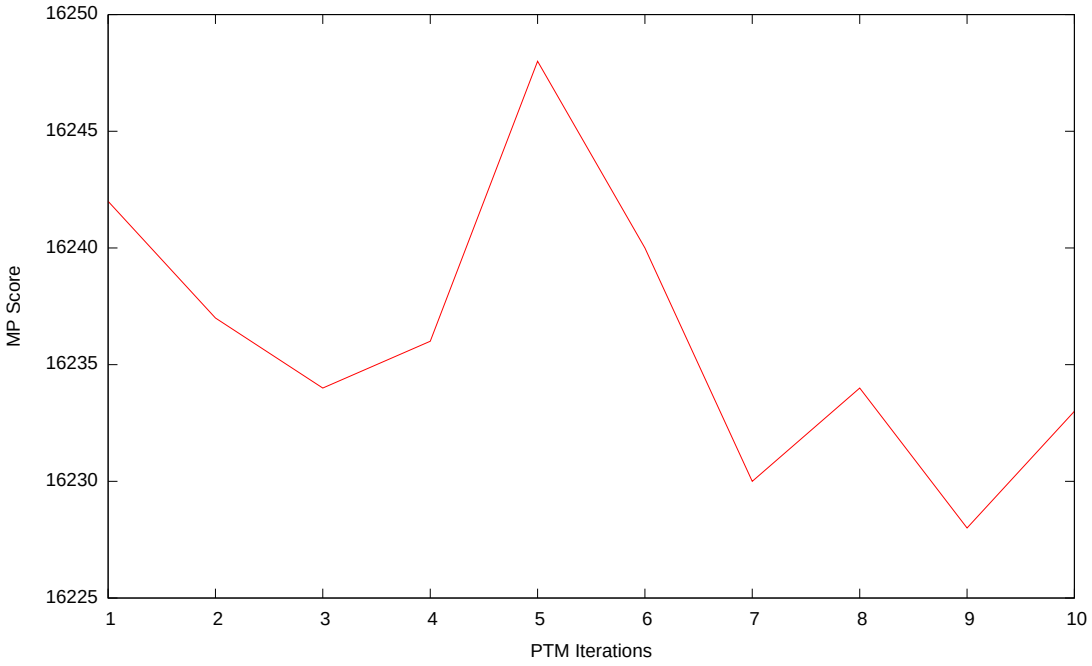


Figure 5.3: A comparison of the score of a PTM tree against the number of PTM iterations used in the search.

Partial trees are divided on their longest branch. If parsimony is the optimality criterion, this is the branch which requires the greatest number of mutation events. If likelihood is used then branch length has the usual meaning. This tends to keep taxa together during mixing that are together in an optimal tree. It also allows those taxa which are most different from others in a partial tree to migrate to a different partial tree where they can be placed more appropriately.

5.4.5 Building the Tree

After a prescribed number of tree mixing iterations, PTM begins to build a fully resolved tree. Partial trees continue to seek partners for joining as before. However no partial tree division occurs. Thus the partial trees become larger and larger until a fully resolved tree is built. During this phase PTM does progressively less exploration and progressively more exploitation. This tree is then passed on to a TBR based search or some other method as would be done with a stepwise maximum parsimony tree.

Algorithm 2 Tree Mixing

```
Require: PartialTrees ≠ ∅
Ensure: PartialTrees ≠ ∅
  for i = 0 to MaxIterations do
    for all PartialTree1 ∈ PartialTrees do
      closestDistance ← MaxDistance
      for all PartialTree2 ∈ PartialTrees do
        if PartialTree1 ≠ PartialTree2 then
          if PartialTree1.distance(PartialTree2) < closestDistance) then
            closestDistance ← PartialTree1.distance(PartialTree2)
            PartialTree1.closestTree ← PartialTree2
          end if
        end if
      end for
      PartialTree1.join(PartialTree1.closestTree)
    end for
    for all PartialTree ∈ PartialTrees do
      PartialTree.TBR()
    end for
    for all PartialTree ∈ PartialTrees do
      PartialTrees.add(PartialTree.divide())
    end for
  end for
```

5.5 Results

In this section two types of results are considered. First, the work examines the effects of the parameters available to the user on the time taken and on the quality of the trees found. Second, using default settings for these parameters the method is compared with other phylogenetic search programs. PTM followed by a standard TBR search is shown to find better trees than competing methods.

5.5.1 The Effects of Partial Tree Size

The PTM algorithm allows the user to set two parameters which affect the size of the partial trees during the search. The first is a maximum partial tree size. Two partial trees will not join together if the result would be a tree larger than the maximum size. The second is a minimum partial tree size. This is a soft limit, it does not prevent partial trees smaller than this limit. Rather, a tree which is at or below this minimum limit will not subdivide further.

Figures 5.4 and 5.5 show the effects of partial tree size on time and on the score found. A PTM search was made on the Zilla data set (500 taxa), setting the minimum and maximum size of the partial trees between 10 and 200 taxa. The time taken by the PTM search and the final score found were recorded. This time and score do not reflect the final tree found by the search, only the initial tree found with the PTM algorithm.

The time taken by the PTM algorithm increases as the size of the partial trees increases. Figure 5.4 shows this relationship. This is not unexpected as Partial Tree Mixing uses a divide and conquer strategy. There is a visible boundary between two regions of the parameter space. In one region both the minimum and maximum sizes are large and the time taken is longer. In the other region at least one of the two sizes is small.

The speed in this second region is a result of smaller tree sizes, which can be quickly optimized. As the maximum size is a hard limit it is clear how a smaller maximum size leads to smaller partial trees. It is not as obvious how a smaller minimum size leads to smaller trees. Consider a partial tree containing a small set of taxa unlike the other taxa in this

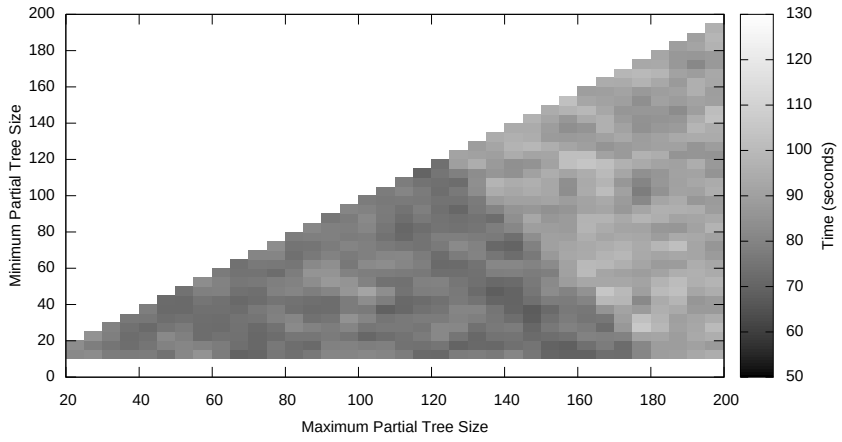
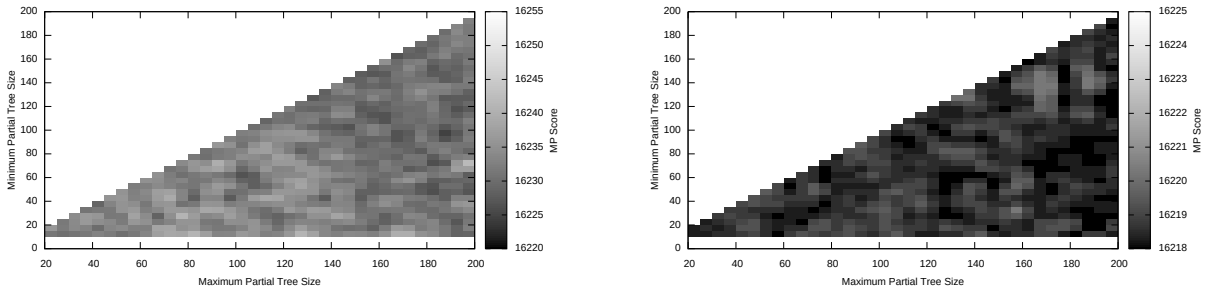


Figure 5.4: A graph of the time taken by the PTM algorithm as the size of the partial trees is varied. Two partial trees will not join if doing so would create a partial tree larger than the maximum size. A partial tree below the minimum size will not divide further. In general the PTM algorithm takes less time with smaller minimum and maximum sizes.

partial tree. After optimization these taxa will tend to group together at the end of a long branch. This long branch will be selected as the division point when forming new partial trees. The result is a tree close to the maximum size, and a small tree. The larger tree, being close to the maximum size is less likely to join with another tree in the following iteration. Small trees do not subdivide if they are below the minimum size. If the minimum size is close to the maximum size, many of these small trees will join together to form a tree within the prescribed limit. This tends to increase the average size of the partial trees. However, a small minimum size allows these smaller partial trees to form a mix without requiring that they first join together to make large trees. This in turn tends to decrease the average size of the partial trees. The reduction in average size leads to a decrease in the time spent in the PTM algorithm.

There is little variation in the score found by PTM with respect to the size of the partial trees especially after TBR refinement. However, as shown in Figure 5.5 larger partial trees tend to yield slightly more parsimonious trees after PTM only. As discussed in section



PTM only PTM + TBR

Figure 5.5: A graph of the maximum parsimony score of the tree found by the PTM algorithm as the size of the partial trees is varied. Two partial trees will not join if doing so would create a partial tree larger than the maximum size. Using larger partial trees tends to yield slightly better parsimony scores after PTM only, but near optimal scores are found by all searches after TBR refinement.

5.4.4 larger partial trees are more accurate representations of the trees in their images. Allowing some larger trees can therefore help the quality of the final tree produced. Smaller partial trees are important for exploration of the possible space. Perhaps the best solution in terms of final tree quality is to have a large to moderate maximum tree size and a small minimum tree size. This allows a variety of both larger trees for exploitation and smaller trees for exploration.

Larger partial trees lead to better scores, but longer search times. Thus, there is a tradeoff in this parameter space between the amount of time spent by PTM and the quality of the tree found. A small or moderate minimum size is desirable for both speed and accuracy. A large maximum size increases quality while decreasing speed. The best overall results occur where the maximum size is large enough to give good results, and the minimum is small enough to compensate for this maximum size in terms of execution time. The optimal parameters likely vary by data set. This implementation uses the conservative default values of 40 and 60, respectively for the minimum and maximum sizes. While these values are likely not near the optimal for most data sets, they seem unlikely to give poor performance on any.

Dataset	Taxa	PTM		PAUP*		PAUP* (multiple trees)	
		Score	Time	Score	Time	Score	Time
RDPII	218	**33534**	00:00:52	33934	<00:00:01	33855	00:00:58
ZILLA	500	**16234**	00:01:23	16414	<00:00:01	16386	00:01:40
U	6722	**92195**	09:30:48	95217	00:01:21	94922	06:30:44
ARB	8780	**162440**	21:35:32	165289	00:03:36	165149	12:18:10

Figure 5.6: A comparison of search results between PTM and stepwise maximum parsimony on several datasets. Note that in every case PTM found more parsimonious trees, but in much more time. When stepwise maximum parsimony was used to find multiple starting trees (300), PTM still found more parsimonious trees.

5.5.2 Comparison with Existing Phylogenetic Search Programs

PAUP*[54] is perhaps the most widely used program for phylogenetic inference using parsimony. For this reason, the performace of PTM was compared to PAUP* using stepwise addition and TBR. TNT[22] and DCM[30] are newer programs which implement a wide variety of heuristic methods[21, 30]. Partial Tree Mixing was implemented in the open source phylogenetics program PSODA[8]. These methods were tested on datasets ranging from 218 to 8780 taxa. PTM was compared against stepwise maximum parsimony where both were followed by a TBR based search until a minima was found. As the step which combines the two final partial trees is equivalent to a standard TBR search, the PTM algorithm was further refined using the Partition Space Sectorial Search (PSSS) algorithim (see Chapter 4).

The results are summarized in figures 5.6 and 5.7. Figure 5.6 compares the results of PTM to stepwise maximum parsimony. PTM takes significantly more time than stepwise maximum parsimony. However, PTM also yields higher quality trees. Figure 5.7 considers the effect of these higher quality trees on the overall search. This table compares the total time taken, both in PTM or stepwise maximum parsimony and in TBR. Here the value of the PTM search is made clear. The final results from PTM for all of the data sets are superior to the final results found using a stepwise tree. Furthermore, with the exception of the smallest data set, these superior trees are found in less time.

Dataset	Taxa	PTM		PAUP*		TNT		DCM	
		Score	Time	Score	Time	Score	Time	Score	Time
RDPII	218	**33515**	1:18:29	33565	0:01:28	42166	0:00:48	–	0:00:11
ZILLA	500	**16218**	2:32:03	16221	15:42:19	16219	0:00:07	16534	0:39:42
U	6722	**92195**	10:39:56	93106	20:10:42	201259	1:31:54	–	0:03:30
ARB	8780	**162438**	24:47:00	162906	29:13:33	170356	1:47:45	–	0:04:17
PROTO	25057	**810231**	23:49:40	–	–	–	–	–	–

Figure 5.7: A comparison of search results between PTM and PAUP*, TNT, and DCM on several datasets. Note that in every case PTM followed by PSSS found a more parsimonious tree than PAUP* using stepwise maximum parsimony followed by TBR. In all but the smallest case, where the overhead of PTM is more difficult to overcome, this tree was found in less time. TNT finishes much faster than PTM, but finds less parsimonious trees. DCM experienced errors in processing many of the data sets and reported no score in these cases. However, the result from the successful run was inferior. Only the PTM method was able to process the largest data set of protobacteria, containing more that 25 thousand taxa.

A trace of a typical result is shown in Figure 5.8, the figure shows a search through a set of 6722 taxa. This trace only shows the TBR search after stepwise to the point in time when PTM returned an initial tree. The scores for the PTM search do not include any TBR refinement. For much of the search time the current tree score of PTM is poor. However, while PTM is exploring low scoring trees it is sampling from a broad area of tree space. It does this so that later phases of the search will not be caught in local minima. The value of this exploration is seen in how quickly the score improves, attaining a far superior answer in less time than traditional methods. The solution from PTM, before any TBR refinement, implies 900 fewer mutation events then the solution found by PAUP*. The solution found was then passed on to a TBR search where further improvements were made, though this is not shown in the figure.

5.6 Conclusions

Partial Tree Mixing is a method for producing an initial phylogenetic tree for use in common hill climbing methods. Current methods produce a tree built using only local information such as pairwise distances or stepwise parsimony. As the trees produced by these greedy

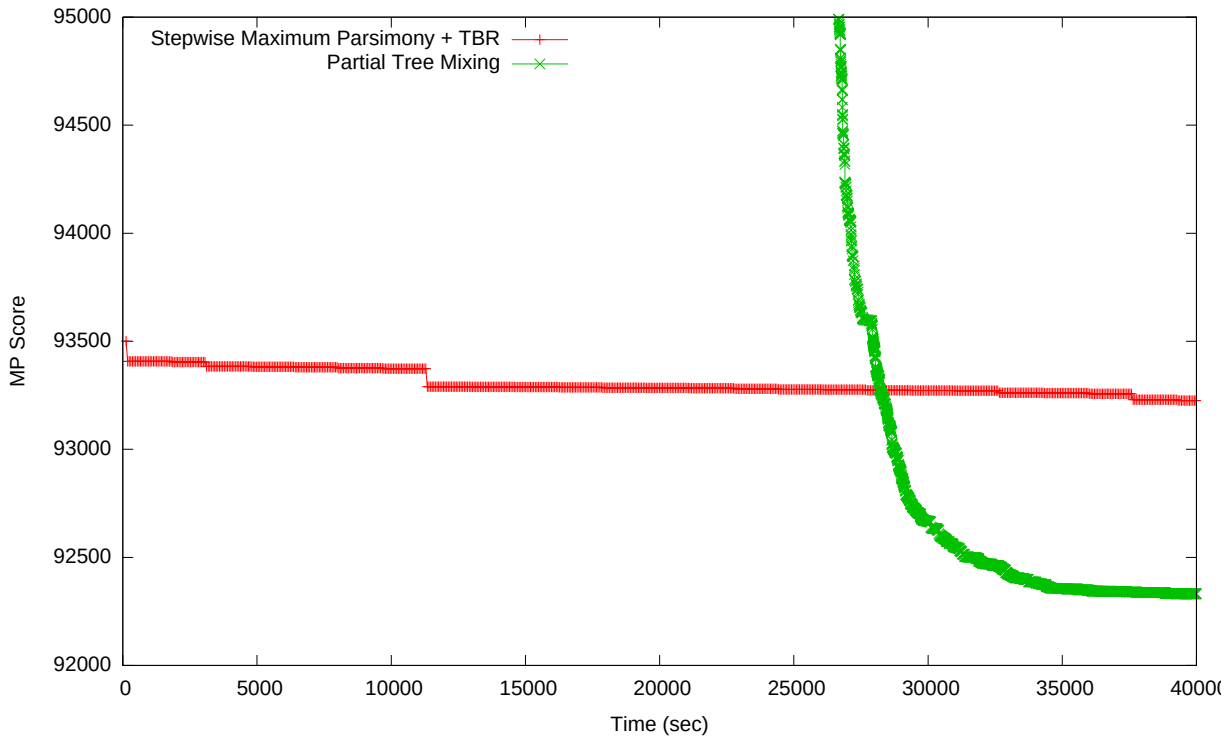


Figure 5.8: A comparison of scores found over time between Partial Tree Mixing (PTM) and PAUP*[54] (Stepwise Maximum Parsimony followed by TBR). Although PAUP* achieves better scores during the early phases of the search, PTM achieves significantly better results after 30000 seconds.

methods can limit the final score after a TBR search it is common practice to start many searches from different starting trees. A TBR search is much more expensive than any of the current starting methods and this duplication of effort outweighs the benefits of a quickly produced starting tree.

PTM produces a tree based on a global search of tree space guided by a partitioned based representation of all possible solutions. Although much more time is expended in producing this tree, results show that the tree produced is of better quality than a tree found using stepwise maximum parsimony followed by an equal amount of time spent in a TBR search. The exploratory nature of the PTM search greatly reduces the need for multiple searches, as PTM produces excellent starting trees. This in turn reduces the overall search time, as duplicate searches are not needed. Overall, a search started with a PTM produced tree finds better solutions in less time.

5.7 Proofs and Definitions

This section contains formal definitions of terms used in this work.

Definition 5.7.1. Tree: A tree is a connected acylclic graph with no vertices of degree two. A tree is **resolved** if its vertices are only of degree one or three, otherwise it is **unresolved**. The edges of this graph are also called **branches**. The vertices of degree one are called **leaves**. The leaves of a tree are labeled with taxa.

Definition 5.7.2. Partial Tree: A partial tree is a resolved tree whose leaves are labeled with a subset of the taxa.

Definition 5.7.3. Resolution of unresolved trees: A resolved tree(R) is a resolution of an unresolved tree (U) if the resolved tree can be iteratively constructed from the unresolved tree using the following operation. Select vertex v of at least degree four. Call the set of vertices directly connected to v, G. Remove v and all edges between v and any member of G

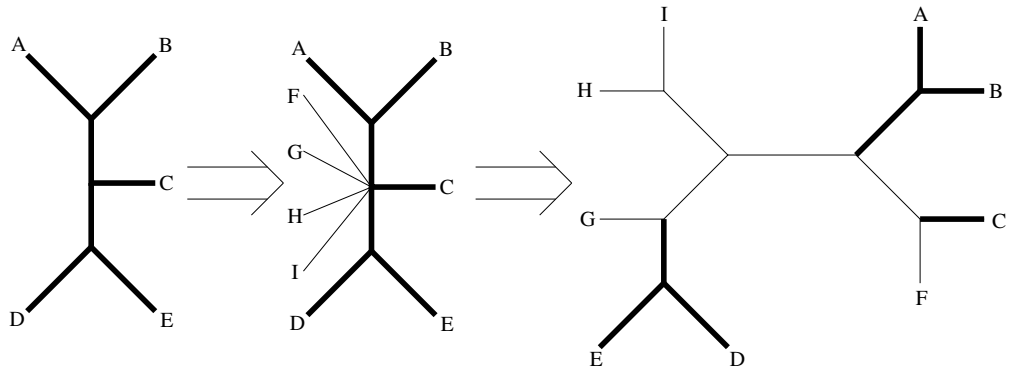


Figure 5.9: A partial tree containing only five taxa, is resolved by adding the missing taxa forming an unresolved tree nine taxa. The vertex of degree 7 is then divided as described in Definition 5.7.3 until a resolved tree of nine taxa has been constructed. This resolution of the first tree is contained in the image of that partial tree.

from the graph. Add two new vertices v_1 and v_2 and the edge (v_1, v_2) to the graph. Finally, for each element g of G add either (v_1, g) or (v_2, g) such that v_1 and v_2 are at least degree 3.

Definition 5.7.4. Resolution of partial trees: A resolved tree is a resolution of a partial tree (T) if it is the resolution of an unresolved tree (U) that can be constructed in the following manner: Let V be the set of vertices in T that are not leaves. For every taxa not in the partial tree add a vertex t labeled with the taxa and an edge (t, v) where $v \in V$. Figure 5.9 shows this process.

5.7.1 Images Under Cartographic Projections

Cartographic projections are used to build a representation of the global tree space. This section covers the properties of images of various tree constructs under this projection.

Definition 5.7.5. Properties of the Cartographic Projections:

- The projection maps branches to vectors in $\mathbb{R}^n$
- The components of these vectors are uniformly distributed from $[-1, 1]$
- Resolved trees are projected to the sum of the projections of their component branches, a point in $\mathbb{R}^n$

- All trees lie in $\mathbb{R}^n$, also referred to as global tree space

  See Chapter 3 for details.

Definition 5.7.6. Image of an unresolved or partial tree: The image of an unresolved or partial tree is defined as a volume which contains the image of all resolutions of this tree.

Theorem 5.7.7. *The image of an unresolved tree is a hypersphere.*

Proof. Consider an unresolved tree of n taxa which has $n-m-3$ branches. The location of the image of any resolution of this tree contains two components. The first is the sum of the images of the $n-m-3$ branches from the unresolved tree. This will be the same for all resolutions, and lies at the center of the hypersphere. The second is the sum of the images of the m branches constructed during resolution. The magnitude of the components of these vectors is at most 1. If the projection is into d dimensions then the maximal magnitude of such a vector is $\sqrt{d}$. With m such vectors, the magnitude of their sum can not exceed $m\sqrt{d}$. This is the radius of the hypersphere. The image of any resolution is the sum of the center of the hypersphere and some vector with magnitude less than or equal to the radius of the sphere. Clearly all such images are contained by this sphere. □

Theorem 5.7.8. *If two unresolved trees can be constructed from the same partial tree then the centers of their images are not separated by more than* $2(n-m-3)\sqrt{d}$.

Proof. Consider a partial tree with $n-m$ taxa. This tree has $n-m-3$ branches. The branches added when resolving a partial tree to an unresolved tree are identical. Thus, two such resolutions can at most differ by $n-m-3$ branches. If two such resolutions differed by every branch possible, and the vectors associated with the differences were all of the maximal magnitude and in opposite directions, the centers of the two images could not be separated by more than $2(n-m-3)\sqrt{d}$. □

Theorem 5.7.9. *The image of a partial tree is bounded by a hypersphere.*

Proof. Consider a partial tree with $n - m$ taxa. Unresolved trees which are resolutions of this tree will have $n - m - 3$ branches. The hyperspheres which contain the images of these unresolved trees will all be of radius $m\sqrt{d}$, where d is the dimensionality of the image space. By Theorem 5.7.8 the most distant unresolved trees are not separated by more than $2(n-m-3)\sqrt{d}$. Thus it is clear that all trees in the image of any of the hypersphere images of the unresolved trees can be circumscribed by a single larger hypersphere of at most radius $(n-3)\sqrt{d}$. □

Chapter 6

Overview

Evolution is widely accepted as a critical component for understanding biological systems. Phylogenetic search is the process of inferring these vital relationships. As such it is of interest to a wide variety of biological problems such as: biogeography [15], conservation efforts [44], drug discovery [6], epidemiology [12, 49], forensics [42] and viral transmission [13, 26]. Good phylogenies can help to find the evolutionary differences between resistant and non-resistant strains of bacteria and viruses. They can be used to help to determine which species are most genetically distinct, enabling them to be singled out for special study to improve our overall understanding of biological systems. They can be used to determine the paths along which viruses have been transmitted, helping to develop future prevention plans or to assign blame in related torts. In short, any field which benefits from an evolutionary perspective, benefits from better phylogenetic search.

In a typical problem, a set of descriptions for the organisms to be analyzed (often DNA sequences) is given, and the expected result is a tree structure. There is an enormous number of trees that can be used to explain the relationships between any set of taxa. In general for n taxa, there are $O(n!!)$ possible trees.

Phylogenetic search is known to be NP-Hard [11] with certain subsets known to be NP-complete [14]. Due to this explosively large search space, the use of heuristic methods is required for most problems of interest. Several competing heuristics exist, each with different trade-offs between accuracy, speed, and the largest trees that can be accommodated.

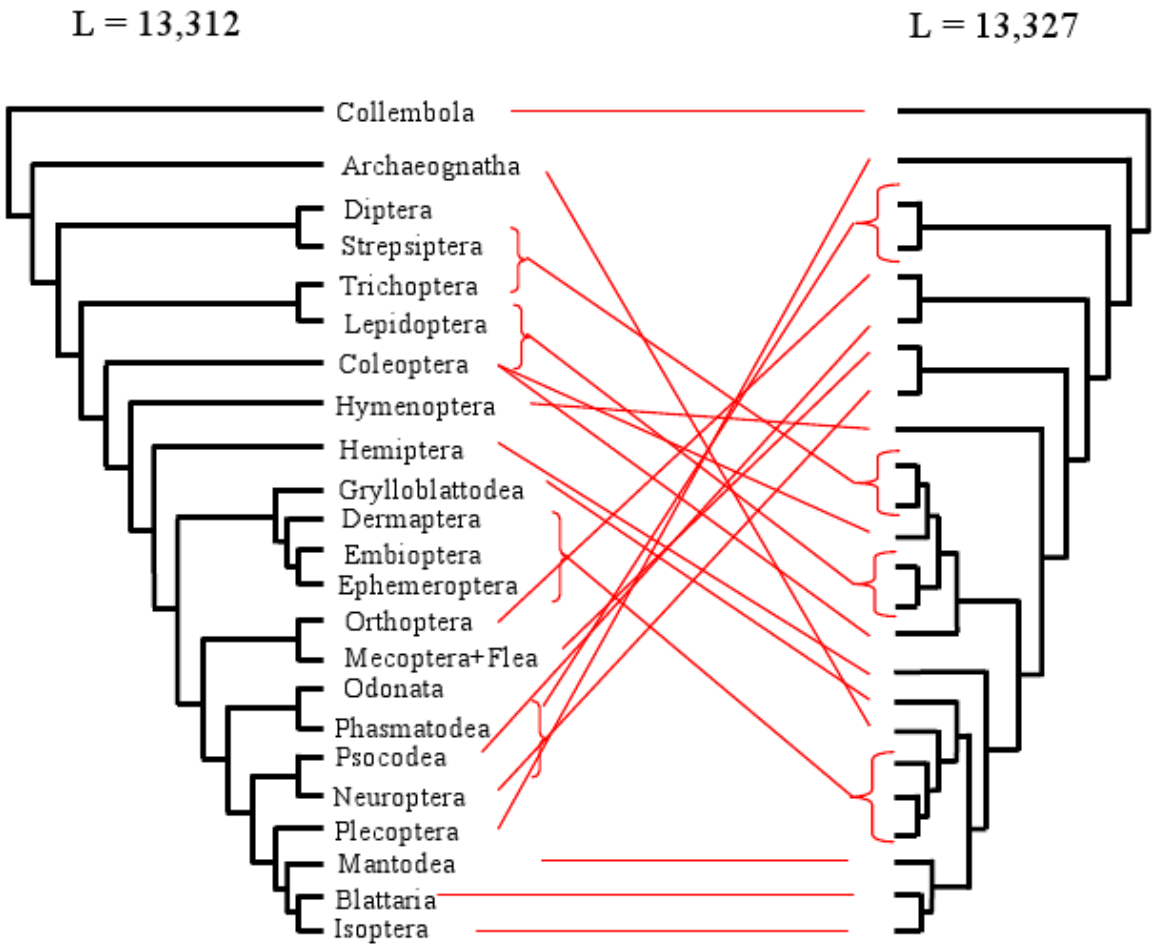


Figure 6.1: Topological differences between two similarly scored trees. The length L, is the number of mutation events implied by the topology.

However, a tree with only a slightly better score may have a very different topology. Figure 6.1 shows an example. In this example, Archaeognatha is closely related to the clade containing Dermaptera, Embioptera, and Ephemeroptera in one tree and distantly related in another. This highlights the importance of accuracy as conclusions are drawn based on tree topology. These trees differ by only 15 mutation events, but support very different evolutionary hypothesis. It is important that heuristic methods find the best trees possible. Otherwise, their results may be misleading to evolutionary researchers. The methods presented in this work have produced trees which have over 900 fewer mutation events than those produced by current methods.

6.1 Thesis Statement

This work aims to profitably unite two distinct subfields in phylogenetic search. The first area covers the design, development, and analysis of heuristic search methods. The second area studies the set of all trees and the underlying structure of phylogenetics.

Using the framework of tree space, searches that find equal or better trees in less time than currently known methods can be developed.

6.1.1 Heuristic Search Methods

Current heuristic searches can be broadly divided into two categories, distance methods and tree refinement methods. Distance methods are used to begin every search program. This is because, unlike the tree refinement methods, distance methods do not require an initial tree. These methods are greedy heuristics and depend on a distance measure such as pair-wise alignment scores. Current distance based methods do not sample from tree space. Partial Tree Mixing provides superior start trees, since it samples from the partition based tree space. This leads to starting trees that are less likely to become stuck in local minima.

Once an initial tree is found, tree refinement methods are employed to search the space of all trees. These methods take a tree and apply a transformation to form a new tree. This tree is then evaluated, if it is superior it becomes the initial tree and the search continues. Again, current methods do not use global information to guide this refinement and are prone to duplicate their efforts. Partition Space Sectorial Search uses a representation of partition space to prevent duplicate effort during this refinement process. This global representation also allows PSSS to behave like a gradient descent search.

6.1.2 Tree Space

This work defines a new description of tree space (Chapter 3), the set of all possible solutions. This space is based on the concept of considering trees as collections of bipartitions.

The goal of this space is to reveal exploitable structure. To accomplish this goal a few criteria must be met. The position of a tree must not depend on the position of any other tree. The position of a tree must be able to be efficiently computed. There must be meaning to distance and direction in tree space. The method presented in this work meets all of these criteria.

6.2 Solution

This work provides a new framework for phylogenetic search. Partition space and the hypersphere of trees provide new insight into the global structure of this important problem. Not only is a method presented for visualization and human inspection of this space, but the validity of the perspective is shown in the development of novel and effective search techniques. The developed heuristics produce more parsimonious trees in less time than previous methods.

Partition Space Sectorial Search (Chapter 4) is a systematic method for refining trees. This method uses a global representation of tree space to avoid duplication of effort. It can also be considered as an approximation to a gradient descent method.

Partial Tree Mixing (Chapter 5) addresses how initial trees for phylogenetic search are produced. This method again uses a global representation of tree space. Instead of using this representation to avoid duplicate effort, it is used to improve diversity and exploration. As a result the tree produced is of very high quality, making the use of multiple starting trees less important.

6.3 Validation

The most important performance metric is the score of the final trees found. A second metric is the amount of time required to find these trees. The methods presented in this work do well in both metrics, finding superior trees in less time. Figure 6.2 shows both PSSS and PTM compared to PAUP* on the Zilla data set.

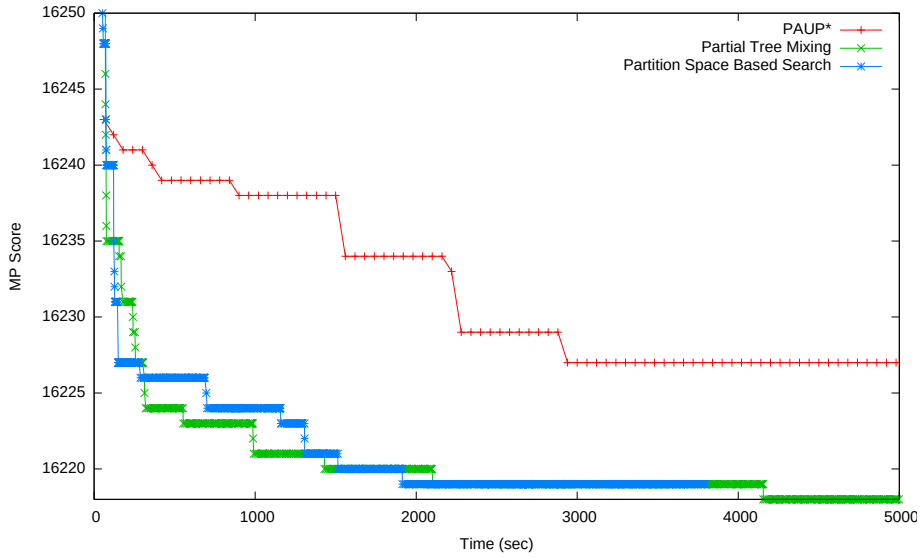


Figure 6.2: A comparison of PSSS, PTM, and PAUP* on the Zilla data set.

As important as it is to increase performance, this work also makes an effort to increase our fundamental understanding of phylogenetic search. This work uses the concepts of partition space and the hypersphere of trees to describe and analyze behavior of current methods. Furthermore, these concepts are used to develop and predict the behavior of new methods which are superior to previous methods. This new way of looking at phylogenetic search will revolutionize the research community by allowing researchers to examine data sets more accurately with greater understanding.

6.4 Future Work

Partition Space and the Hypersphere of Trees provide a framework for the study of phylogenetics. A number of questions could now be addressed. Some to be considered are:

- Bayesian analysis in phylogenetics is centered around partitions, can partition space be applied to this area also?
- How can reasoning about MP or ML be considered during a partition space based search?

- Do MP and ML searches follow the same or similar paths through tree space?

Partial Tree Mixing (Chapter 5) uses a search strategy which pulls ideas from genetic algorithms, hill climbing, and divide and conquer strategies. In this domain of clustering in a poor signal to noise environment, this method performs very well. A question to be addressed in future work is if this technique is applicable in a general way to optimization problems.

References

[1] B.L. Allen and M. Steel. Subtree Transfer Operations and Their Induced Metrics on Evolutionary Trees. *Annals of Combinatorics*, 5(1):1–15, 2001.

[2] N. Amenta, M. Godwin, N. Postarnakevich, and K. St. John. Approximating Geodesic Tree Distance. *Information Processing Letters*, 103(2):61–65, 2007.

[3] N. Amenta and J. Klingner. Case Study: Visualizing Sets of Evolutionary Trees. *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on Information Visualization*, pages 71–74, 2002.

[4] W. Basalaj. Incremental Multidimensional Scaling Method for Database Visualization. In *Proceedings of Visual Data Exploration and Analysis VI, SPIE*, volume 3643, pages 149–158, 1999.

[5] L.J. Billera, S.P. Homes, and K. Vogtmann. Geometry of the Space of Phylogenetic Trees. *Advances in Applied Mathematics*, 27(4):733–767, November 2001.

[6] J.R. Brown and P.V. Warren. Antibiotic Discovery: Is it in the Genes. *Drug Discovery Today*, 3:564–566, 1998.

[7] L.M. Bugayevskiy and J.P. Snyder. *Map Projections: a Reference Manual*. CRC, 1995.

[8] H. Carroll, M. Ebbert, M. Clement, and Q. Snell. PSODA: Better tasting and less filling than PAUP. In *Proceedings of the 4th Biotechnology and Bioinformatics Symposium (BIOT-07)*, pages 74–78, October 2007.

[9] R.A. Cartwright. DNA Assembly with Gaps (Dawg): Simulating Sequence Evolution. *Bioinformatics*, 21(Suppl 3), 2005.

[10] M.W. Chase, D.E. Soltis, R.G. Olmstead, D. Morgan, D.H. Les, B.D. Mishler, M.R. Duvall, R.A. Price, H.G. Hills, Y.L. Qiu, et al. Phylogenetics of Seed Plants: An Analysis of Nucleotide Sequences from the Plastid Gene rbcL. *Annals of the Missouri Botanical Garden*, 80(3):528–580, 1993.

[11] B. Chor and T. Tuller. Maximum Likelihood of Evolutionary Trees is Hard. In *Proceedings of the 9th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005)*, volume 3500, pages 296–310. Springer, 2005.

[12] A.G. Clark, K.M. Weiss, D.A. Nickerson, S.L. Taylor, A. Buchanan, J. Stengard, V. Salomaa, E. Vartiainen, M. Perola, E. Boerwinkle, and C.F. Sing. Haplotype Structure and Population Genetic Inferences from Nucleotide-Sequence Variation in Human Lipoprotein Lipase. *American Journal of Human Genetics*, 63:595–612, 1998.

[13] K.A. Crandall. Multiple Interspecies Transmissions of Human and Simian T-cell Leukemia/Lymphoma Virus Type I Sequences. *Molecular Biology and Evolution*, 13:115–131, 1996.

[14] W.H.E. Day, D.S. Johnson, and D. Sankoff. The Computational Complexity of Inferring Rooted Phylogenies by Parsimony. *Mathematical Biosciences*, 81(33-42):299, 1986.

[15] R. DeSalle. Molecular Approaches to Biogeographic Analysis of Hawaiian Drosophilidae. Hawaiian Biogeography (ed. by WL Wagner and VA Funk), pp. 72–89, 1995.

[16] J. Felsenstein. Maximum Likelihood and Minimum-Steps Methods for Estimating Evolutionary Trees from Data on Discrete Characters. *Systematic Zoology*, 22(3):240–249, 1973.

[17] J. Felsenstein. Evolutionary Trees from DNA Sequences: A Maximum Likelihood Approach. *Journal of Molecular Evolution*, 17(6):368–376, 1981.

[18] J. Felsenstein. Distance Methods for Inferring Phylogenies: A Justification. *Evolution*, 38(1):16–24, 1984.

[19] G. Ganapathy, V. Ramachandran, and T. Warnow. Better Hill-Climbing Searches for Parsimony. *Workshop on Algorithms in Bioinformatics*, 2003.

[20] G. Ganapathy, V. Ramachandran, and T. Warnow. Better Hill-Climbing Searches for Parsimony. *Algorithms in Bioinformatics*, 2812:245–258, 2003.

[21] P.A. Goloboff. Analyzing Large Data Sets in Reasonable Times: Solutions for Composite Optima. *Cladistics*, 15(4):415–428, 1999.

[22] P.A. Goloboff, J.S. Farris, and K.C. Nixon. TNT, a Free Program for Phylogenetic Analysis. *Cladistics*, 24(5):774–786, 2008.

[23] S. Guindon and O. Gascuel. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. *Systematic Biology*, 52(5):696–704, 2003.

[24] D. Gusfield. Efficient Algorithms for Inferring Evolutionary Trees. *Networks*, 21(1), 1991.

[25] M. Hasegawa, H. Kishino, and T. Yano. Dating of the Human-Ape Splitting by a Molecular Clock of Mitochondrial DNA. *Journal of Molecular Evolution*, 22(2):160–174, 1985.

[26] B.L. Herring, F. Bernardin, S. Caglioti, S. Stramer, L. Tobler, W. Andrews, L. Cheng, S. Rampersad, C. Cameron, J. Saldanha, , M.P. Busch, and E. Delwart. Phylogenetic Analysis of WNV in North American Blood Donors during the 2003-2004 Epidemic Seasons. *Virology*, 2007.

[27] D.M. Hillis, T.A. Heath, and K. St. John. Analysis and Visualization of Tree Space. *Systematic Biology*, 54(3):471–482, 2005.

[28] J.P. Huelsenbeck and B. Rannala. Phylogenetic Methods Come of Age: Testing Hypotheses in an Evolutionary Context. *Science*, 276(5310):227, 1997.

[29] A. Hultman. The Topology of Spaces of Phylogenetic Trees with Symmetry. *Discrete Mathematics*, 307(14):1825–1832, 2007.

[30] D.H. Huson, S.M. Nettles, and T.J. Warnow. Disk-covering, a Fast-Converging Method for Phylogenetic Tree Reconstruction. *Journal of Computational Biology*, 6(3-4):369–386, 1999.

[31] B. Jenkins. A New Hash Function for Hash Table Lookup. *Dr. Dobb's Journal*, 1997.

[32] T.H. Jukes and C.R. Cantor. Evolution of Protein Molecules. *Mammalian Protein Metabolism*, 3:21–132, 1969.

[33] J.M. Keith, P. Adams, M.A. Ragan, and D. Bryant. Sampling Phylogenetic Tree Space with the Generalized Gibbs Sampler. *Molecular Phylogenetics and Evolution*, 34(3):459–68, 2005.

[34] A. Kupczok, A. von Haeseler, and S. Klaere. An Exact Algorithm for the Geodesic Distance between Phylogenetic Trees. *Journal of Computational Biology*, 15:577–591, 2008.

[35] C. Lanave, G. Preparata, C. Sacone, and G. Serio. A New Method for Calculating Evolutionary Substitution Rates. *Journal of Molecular Evolution*, 20(1):86–93, 1984.

[36] M. Li, X. Chen, X. Li, B. Ma, and P.M.B. Vitányi. The Similarity Metric. *IEEE Transactions on Information Theory*, 50:12, 2004.

[37] D.R. Maddison. The Discovery and Importance of Multiple Islands of Most-Parsimonious Trees. *Systematic Zoology*, 40(3):315–328, 1991.

[38] R. Meier and F.B. Ali. Software Review. The Newest Kid on the Parsimony Block: TNT (Tree Analysis Using New Technology) . *Systematic Entomology*, 30(1):179, 2005.

[39] C.D. Michener and R.R. Sokal. A Quantitative Approach to a Problem in Classification. *Evolution*, 11(2):130–162, 1957.

[40] K. Nixon. The Parsimony Ratchet, a New Method for Rapid Parsimony Analysis. *Cladistics*, 15(4):407–414, 1999.

[41] M. Owen and J.S. Provan. A Fast Algorithm for Computing Geodesic Distances in Tree Space, 2010. in press.

[42] D. Patient. Molecular Epidemiology of HIV Transmission in a Dental Practice. *SCIENCE*, 256:1165, 1992.

[43] N.D. Pattengale, E.J. Gottlieb, and B.M.E. Moret. Efficiently Computing the Robinson-Foulds Metric. *Journal of Computational Biology*, 14(6):724–735, 2007.

[44] P. Posadas, D.R.M. Esquivel, and J.V. Crisci. Using Phylogenetic Diversity Measures to Set Priorities in Conservation: an Example from Southern South America. *Conservation Biology*, 15(5):1325, 2001.

[45] D.F. Robinson and L.R. Foulds. Comparison of Phylogenetic Trees. *Mathematical Biosciences*, 53(1-2):131–147, 1981.

[46] F. Rodriguez, JL Oliver, A. Marin, and JR Medina. The General Stochastic Model of Nucleotide Substitution. *Journal of Theoretical Biology*, 142(4):485–501, 1990.

[47] F. Ronquist and J.P. Huelsenbeck. MrBayes 3: Bayesian Phylogenetic Inference Under Mixed Models. *Bioinformatics*, 19(12):1572–1574, 2003.

[48] N. Saitou and M. Nei. The Neighbor-Joining Method: a New Method for Reconstructing Phylogenetic Trees. *Molecular Biology and Evolution*, 4(4):406, 1987.

[49] C.F. Sing, M.B. Haviland, K.E. Zerba, and A.R. Templeton. Application of Cladistics to the Analysis of Genotype-Phenotype Relationships. *European Journal of Epidemiology*, 8:3–9, 1992.

[50] A. Stamatakis. RAxML-VI-HPC: Maximum Likelihood-Based Phylogenetic Analyses with Thousands of Taxa and Mixed Models. *Bioinformatics*, 22(21):2688, 2006.

[51] M. Steel. The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees. *Journal of Classification*, 9(1):91–116, 1992.

[52] K. Sundberg, M. Clement, and Q. Snell. Visualizing Phylogenetic Treespace Using Cartographic Projections. In *Proceedings of Algorithims in Bioinformatics: 9th International Workshop (WABI 2009)*, page 321. Springer, September 2009.

[53] K. Sundberg, T. O'Connor, H. Carroll, M. Clement, and Q. Snell. Using Parsimony to Guide Maximum Likelihood Searches. *IEEE Symposium on BioInformatics and BioEngineering (BIBE)*, pages 774–779, 2007.

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

[55] J. Thompson, F. Plewniak, and O. Poch. BAliBASE: A Benchmark Alignments Database for the Evaluation of Multiple Sequence Alignment Programs. *Bioinformatics*, 15(1):87–88, 1999.

[56] R.A. Vos. Accelerated Likelihood Surface Exploration: the Likelihood Ratchet. *Systematic biology*, 52(3):368–373, 2003.

[57] M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall/CRC, 1995.

[58] D.J. Zwickl. *Genetic Algorithm Approaches for the Phylogenetic Analysis of Large Biological Sequence Datasets under the Maximum Likelihood Criterion*. PhD thesis, The University of Texas at Austin, 2006.