On The Use Of Cartographic Projections In Visualizing Phylogenetic Treespace

Kenneth Sundberg Mark Clement Quinn Snell

June 8, 2009

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 datasets.

1 Introduction

Phylogenetic analysis has become an integral part of many biological research programs. These include such diverse areas as human epidemiology (Clark et al., 1998; Sing et al., 1992), viral transmission (Crandall, 1996; Herring et al., 2007), and biogeography (DeSalle, 1995). With the advent of new automated sequencing technologies, 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 (Day et al., 1986). The search problem itself, independent of scoring, is known to be NP-Hard (Chor and Tuller, 2005). 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 treespace. The most common method is to search treespace using tree rearrangements (Stamatakis, 2006; Meier and Ali, 2005; Swofford, 2003; Guindon and Gascuel, 2003). Other methods such as those based on Bayesian inference (Ronquist and Huelsenbeck, 2003), or genetic algorithms (Zwickl, 2006) 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 treespace.

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 treespace, and second, developing methods to display treespace 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 can not 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 structure suggests a space where good trees might be found, to be useful in searching it must be possible to quickly find trees in the suggested space.

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.

Section 3 presents a definition of treespace and section 3.3 presents an elegant projection of that space that has all four of these desirable properties. This projection is then used to visualize the treespace and expose structure that can be exploited to guide the searches of common, but computationally expensive, methods.

2 Related Work

Treespace 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 treespace that implicitly arises from these strategies is very cumbersome to work with. Treespaces 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 treespaces. Figure 1 contains a visual comparison of three treespaces that have been used previously and are discussed in the following sections.

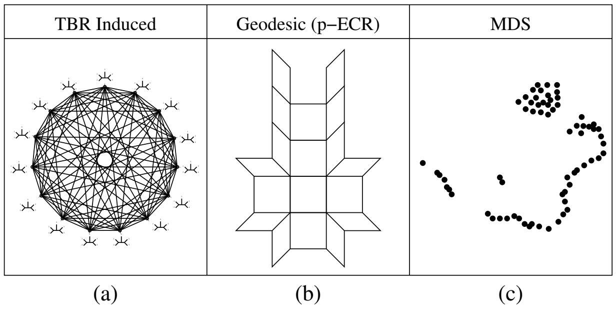


Figure 1: A visual comparison of three treespaces previously used. The graph structure induced by TBR moves is highly connected, in this five taxa example every possible tree is connected to all but two of the other trees. The geodesic structure consists of tiles of Euclidean space (orthants) each consisting 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) 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.

2.1 Subtree Transfer Induced Spaces

The most common treespaces used in phylogenetic search are the spaces implicitly defined by the subtree transfer operations, such as TBR or SPR, used during the search. These operations in turn induce distances between trees (Allen and Steel, 2001). These treespaces 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 treespace 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 Regraph (SPR), Tree Bisection and Reconnection (TBR), or p-Edge Contraction and Refinement (p-ECR) (Ganapathy et al., 2003). 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.* (Keith et al., 2005) 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 treespace 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 treespace 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 (Allen and Steel, 2001). These difficulties violate the third criterion. Finally this graph structure shown in Figure 1a does not exhibit exploitable structure, the fourth criterion, as trees of similar score are not grouped together. As shown in Figure 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 way to distinguish one such tree from another, without performing the rearrangement and examining the resulting tree.

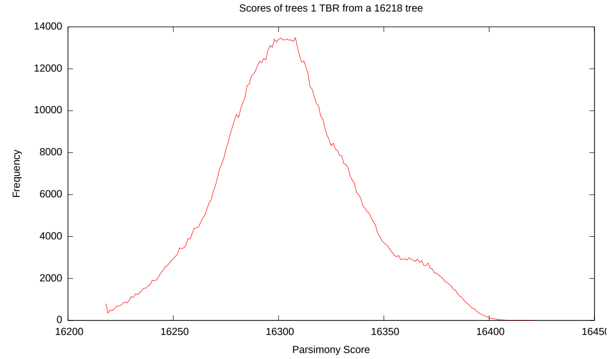


Figure 2: The frequency of various parsimony scores for trees found within 1 TBR rearrangement of a tree with a score of 16218, the best known score on the Zilla dataset. Note the wide spread of scores and that most neighbor trees are significantly worse than the initial tree.

2.2 Geodesic Tree Space

Billera *et al.* (Billera et al., 2001) introduced a new description of treespace, which has been further refined by Hultman (Hultman, 2007). 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 (multifricating) 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 1b.

This space is unlike the treespace induced by subtree transfer operations. The branch lengths of the trees are included and this treespace is continuous. However, because it is a geodesic, it can be difficult to calculate distances, though Amenta *et al.*(Amenta et al., 2007) do provide a linear time upper and lower bound that can be used to estimate the distances. Unfortunately, like the subtree transfer induced spaces used during phylogenetic search, Billera's geodesic 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 treespace defined by TBR there is no significant meaning to position between orthants. Thus, like the TBR induced treespace, this treespace defined by Billera 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.

2.3 Multidimensional Scaling

Multidimensional Scaling (MDS) has also been used to visualize treespace (Amenta and Klingner, 2002; Hillis et al., 2005). This method does not directly define a treespace, rather it uses the space induced by the distance metric used for the MDS. In the work of Hillis *et al.* (Hillis et al., 2005) Robinson-Foulds distance was used. 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.

Using this method Hillis *et al.* (Hillis et al., 2005) 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 the use of MDS. 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 The Hypersphere of Trees in Split Space

Another treespace 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 treespace and visualization. In the results section we show that this space also displays exploitable structure.

3.1 Split Space

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 treespace 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 3.1. 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 3.2. A trivial branch is a branch that connects a leaf node with an internal node.

Given n taxa there are $\sum_{i=2}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{i}$ 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 can not 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. 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 subclades which have already been built are joined at new 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 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.

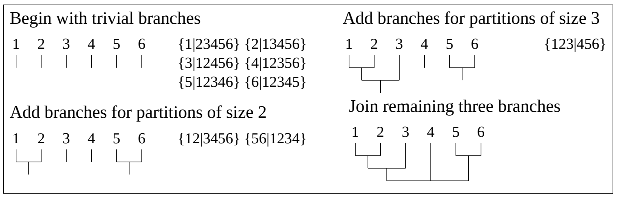

Begin with trivial branches
1 2 3 4 5 6
{1|23456} {2|13456}
{3|12456} {4|12356}
{5|12346} {6|12345}
Add branches for partitions of size 2
1 2 3 4 5 6
{12|3456} {56|1234}
Add branches for partitions of size 3
1 2 3 4 5 6
{123|456}
Join remaining three branches
1 2 3 4 5 6

Figure 3: Converting a six taxa partition set to an unrooted tree structure

3.2 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 3.4.

Lemma 3.3. *All n-trees have $2n-3$ branches, $n-3$ of these are nontrivial.*

Proof. By inspection all trees of n taxa have n trivial branches, one for each taxa.

$$branches_t(n) = n$$

A $n+1$ taxa tree can be constructed from a n taxa tree by inserting a new trivial branch. After this insertion the tree has one more nontrival branch.

$$branches_{nt}(n+1) = branches_{nt}(n)$$

By inspection it is clear that a 3 taxa tree has no nontrivial branches.

$$branches_{nt}(3) = 0$$

The formula which satisfies both of these conditions is

$$branches_{nt}(n) = n - 3$$

And finally:

$$\begin{aligned} branches(n) &= branches_{nt}(n) + branches_t(n) \\ &= 2n - 3 \end{aligned}$$

□

Theorem 3.4. *All n-trees lie on a hypersphere in $\mathbb{T}_n$.*

Proof. By Definition 3.1, n-trees are fully resolved. All fully resolved trees on n taxa have $n-3$ nontrivial branches by Lemma 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. □

3.3 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.

3.4 Sphere to Plane Projections

Cartographic projections 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 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 $n-$

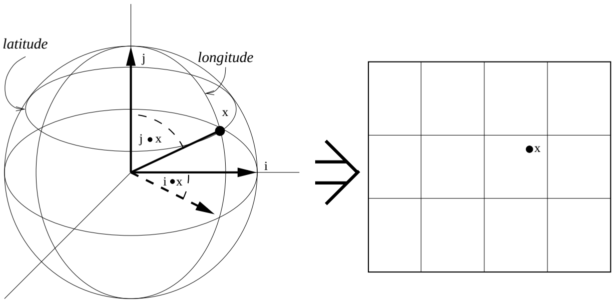


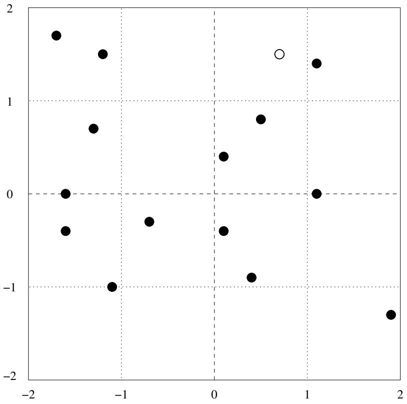
Figure 4: Cartographic projection of a sphere onto a plane, the most familiar of which is used in map making. Two vectors are selected, indicated as i and j. For any given point x on the sphere the inner products $i \bullet x$ and $j \bullet x$ are computed. These two quantities become the new coordinates of the point on the map.

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.

The spherical structure of trees in $\mathbb{T}_n$ shown in theorem 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. Section 4 shows the exploitable structure revealed by the 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.

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. 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 projected positions of all 5 taxa trees are shown in Figure 5 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).



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 5: A 2-D cartographic projection of all 5 taxa 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.

3.5 Implementation Details

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.

3.6 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.

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 natural numbers 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.

3.7 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 in appendix A.

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 A.5. The quality of the orthogonality property of the reference vectors is dependent on the quality of the hash function as shown in theorem A.6. Given the size of the representative vectors used (65535 elements) and only 20 taxa the reference vectors must be within 7.32×10^{-5} degrees of 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 A.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 (Jenkins, 1997) was used for the results in section 4.

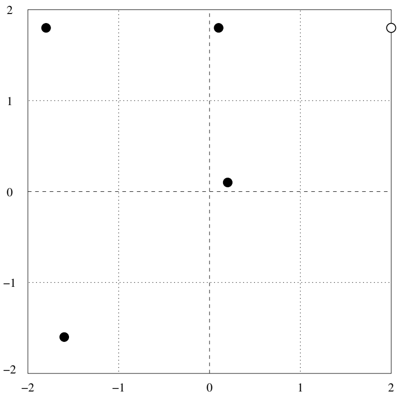
3.8 Calculating the Inner Product

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. 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.

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

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



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

Figure 6: A 2-D cartographic projection of all 5 taxa trees, with reference vectors represented through a simple modulo 3 hash. Under this poor hash there are only 5 locations which correspond to valid trees. The vector for the tree ((1,2),3,(4,5)) is also shown. The point corresponding to this tree is highlighted in the graph.

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 taxa numbers of the two taxa on one side, then divide this sum by three and take the remander as the value of the hash function. There are three possible values for this hash function 0,1, and 2. Figure 6 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 respond to any valid trees. These locations are all shown in Figure 6. 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 number of taxa in the tree. This upper bound on necessary storage makes the visualization of larger data sets feasible.

4 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.

4.1 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 Tree Set Vis (Hillis et al., 2005), and finally Multidimensional Scaling in three dimensions to account for any affects from the extra degree of freedom. The test case will be the exaustive set of all trees of 7 taxa, with each method run 100 times as they all have random elements. Once each projection is calculated, the nearest n neighbors for every tree are found, with n ranging from 0 to 25. A majority rule consensus tree is then constructed for each of these neighborhoods. 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 7 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 neigborhoods 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 treespace. Further this superiority is not due to the added flexibility of projecting onto three dimensions rather than two.

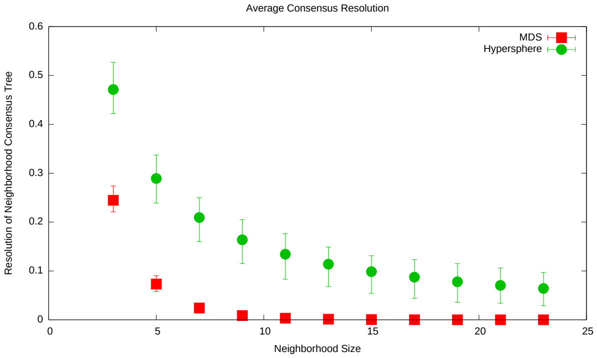


Figure 7: The average degree of consensus across near neighbors among all trees with 7 taxa, note that higher values are better. As both cartographic projections and MDS have a random component each point consists of 100 projections with the average, minimum and maximum values for the consensus across 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.

4.2 Results from Nine Taxa Set Exhaustive Searches

To explore the inherent structure of the maximum parsimony problem, several nine taxa data sets were fully analyzed. The size of nine taxa 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* (Swofford, 2003). 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 8, 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 8, clustering of scores is clear. Even though the gradient was much more complex, it would still be possible to use gradient descent.

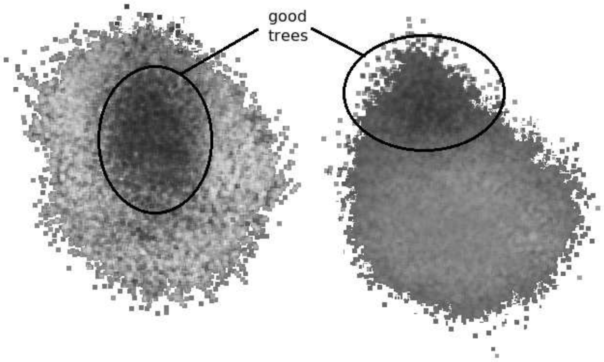


Figure 8: Two distinct 9 taxa datasets under cartographic projection. Dark points represent trees with better scores. The set on the left shows clear clustering with good trees near the center of the cloud. The set on the right shows a gradient, with good trees at the upper point of the cloud.

4.3 Visualizing an Exhaustive Search with MDS

The tool Tree Set Viz 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 taxa data set. An eight taxa subset was used instead. The program was run overnight to allow the program adequate time to converge to the mapping shown in figure 9.

A few features are noteworthy. First, the circular shape, which is a result of the hyperspherical nature of treespace. 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, however, the MDS formed two separate clusters and 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 can not be used to guide a search. Therefore any structure is inherently not exploitable structure.

4.4 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 (Carroll et al., 2007) 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.

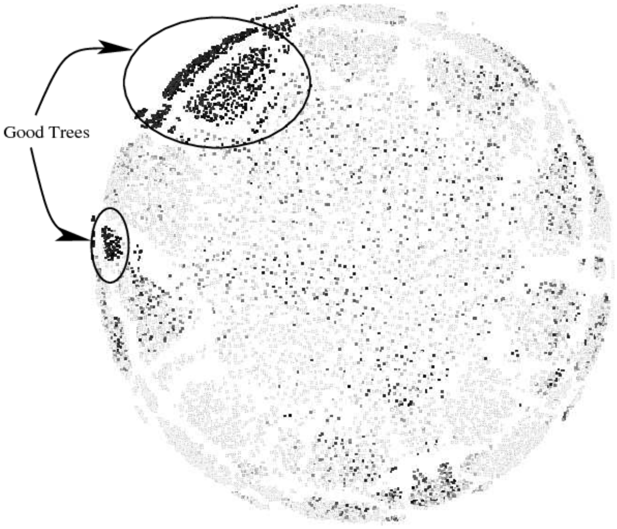


Figure 9: A multi-dimensional scaling (MDS) visualization of an exhaustive search of 8 Taxa. Dark points represent trees with better scores. Note that there is some clustering of good trees but that they can be found throughout the visualized set.

Figure 10 shows a projection of a TBR search with the Zilla data set (Chase et al., 1993) using cartographic projections. Again, a clustering of scores is apparent among the trees considered by the search, revealing exploitable structure in this difficult dataset.

5 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

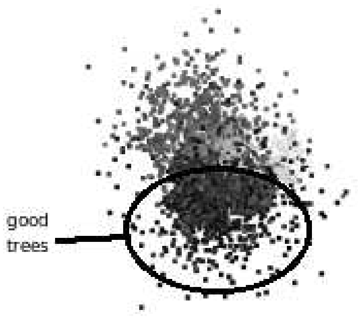


Figure 10: Projection of a search through the Zilla 500 taxa dataset.

structure. More importantly new search techniques can be developed that can exploit this structure.

5.1 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.

5.2 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.

6 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.

References

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

Amenta, N., M. Godwin, N. Postarnakevich, and K. St. John. 2007. Approximating geodesic tree distance. Information Processing Letters 103:61–65.

Amenta, N. and J. Klingner. 2002. Case study: visualizing sets of evolutionary trees. Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on Information Visualization Pages 71–74.

Billera, L. J., S. P. Homes, and K. Vogtmann. 2001. Geometry of the space of phylogenetic trees. Advances in Applied Mathematics 27:733–767.

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

Chase, M., D. Soltis, R. Olmstead, D. Morgan, D. Les, B. Mishler, M. Duvall, R. Price, H. Hills, Y. Qiu, et al. 1993. Phylogenetics of Seed Plants: An Analysis of Nucleotide Sequences from the Plastid Gene rbcL. Annals of the Missouri Botanical Garden 80:528–580.

Chor, B. and T. Tuller. 2005. Maximum likelihood of evolutionary trees is hard. Proceedings of the 9th Annual International Conference on Research in Computational Molecular Biology (RECOMB 2005 3500:296–310.

Clark, A., K. Weiss, D. Nickerson, S. Taylor, A. Buchanan, J. Stengard, V. Salomaa, E. Vartiainen, M. Perola, E. Boerwinkle, and C. Sing. 1998. Haplotype structure and population genetic inferences from nucleotide-sequence variation in human lipoprotein lipase. American Journal of Human Genetics 63:595–612.

Crandall, K. 1996. Multiple interspecies transmissions of human and simian t-cell leukemia/lymphoma virus type i sequences. Molecular Biology and Evolution 13:115–131.

Day, W., D. Johnson, and D. Sankoff. 1986. The computational complexity of inferring rooted phylogenies by parsimony. Mathematical Biosciences 81:299.

DeSalle, R. 1995. Molecular approaches to biogeographic analysis of Hawaiian Drosophilidae. Hawaiian Biogeography (ed. by WL Wagner and VA Funk), pp. 72–89.

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

Guindon, S. and O. Gascuel. 2003. A Simple, Fast, and Accurate Algorithm to Estimate Large Phylogenies by Maximum Likelihood. Systematic Biology 52:696–704.

Herring, B., F. Bernardin, S. Caglioti, S. Stramer, L. Tobler, W. Andrews, L. Cheng, S. Rampersad, C. Cameron, J. Saldanha, et al. 2007. Phylogenetic analysis of WNV in North American blood donors during the 2003-2004 epidemic seasons. Virology .

Hillis, D., T. Heath, and K. St. John. 2005. Analysis and Visualization of Tree Space. Systematic Biology 54:471–482.

Hultman, A. 2007. The topology of spaces of phylogenetic trees with symmetry. Discrete Mathematics 307:1825–1832.

Jenkins, B. 1997. A new hash function for hash table lookup. Dr. Dobb's Journal .

Keith, J., P. Adams, M. Ragan, and D. Bryant. 2005. Sampling phylogenetic tree space with the generalized Gibbs sampler. Mol Phylogenet Evol 34:459–68.

Meier, R. and F. Ali. 2005. Software Review. The newest kid on the parsimony block: TNT (Tree analysis using new technology) . Systematic Entomology 30:179.

Ronquist, F. and J. Huelsenbeck. 2003. MrBayes 3: Bayesian phylogenetic inference under mixed models. Bioinformatics 19:1572–1574.

Sing, C., M. Haviland, K. Zerba, and A. Templeton. 1992. Application of cladistics to the analysis of genotype-phenotype relationships. European Journal of Epidemiology 8:3–9.

Stamatakis, A. 2006. RAxML-VI-HPC: maximum likelihood-based phylogenetic analyses with thousands of taxa and mixed models. Bioinformatics 22:2688.

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

Zwickl, D. J. 2006. Genetic algorithm approaches for the phylogenetic analysis of large biological sequence datasets under the maximum likelihood criterion. Ph.D. thesis The University of Texas at Austin.

A 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.

A.1 Definitions

Definition A.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 A.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 A.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 A.4. The quality of the function h, ξ is a measure of how evenly the elements of X' and Y' are mapped by h onto X and Y

$$\exists \xi \forall i, \frac{d}{d'} + \xi \geq f_i$$

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

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

A.2 Theorems

Theorem A.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:

$$\forall s, sX \neq Y$$
$$\forall s \forall i \in \{\mathbb{N} < d\}, sX_i \neq Y_i$$
$$\forall k \exists j, X'_k = X_{h(j)} \quad \text{Definition A.2}$$
$$\forall k \exists j, Y'_k = Y_{h(j)}$$

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

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

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

Theorem A.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 A.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 $\langle X|Y\rangle = 0$, therefor either

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

and clearly

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

or

$$\begin{aligned}\exists i, X_i Y_i > 0 \\ \exists j, X_j Y_j < 0\end{aligned}$$

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 remaning 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

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

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 A.7

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

□

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 A.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 □