Cluster and Visualization Applications for Phylogenetic Tree Analysis

Cadran Cowansage and Tiffani L. Williams
Department of Computer Science
Texas A&M University

August 4, 2007

Abstract

Phylogenetic studies often produce thousands of trees that each represents a possible evolutionary history for a set of taxa. Large numbers of trees are difficult to analyze, particularly because there are few existing methods to do so. One common technique is to compute a representative strict consensus tree, but this approach can cause information about individual trees to be lost when their topologies are notably different. This paper investigates clustering as a tool to evaluate trees collected during the heuristic search process and employs a *cluster grid* to graphically interpret them. We identify some topological features of the trees within search spaces and present the cluster grid as a useful means for visualizing relationships among them.

1 Introduction

The process of inferring evolutionary histories among organisms with the ultimate goal of determining the relationships between all living things on earth is known as phylogenetic reconstruction [7]. Evolutionary history is often displayed in trees, with descendent species, or taxa, branching from ancestral ones. Determining the phylogenetic relationships that structure a tree can be difficult because many common ancestors are extinct and the fossil record is imprecise and does not include every species. Consequently, scientists infer relationships by examining common, inherited characteristics among species. It is thought that organisms that share more inherited characteristics are more likely to have descended from a common ancestor [7].

Often numerous hypotheses about evolutionary events are discovered and it can be difficult to compare the information contained in the resulting trees, particularly when the collection of trees is large. This paper explores clustering to compare these trees and graphical plots to represent the relationships among them.

Phylogenetic reconstruction is a computationally challenging problem that can be approached using heuristics. Maximum Parsimony (MP) is an NP-hard optimization problem that is used to search within tree-space for a phylogenetically true tree. Trees that minimize the number of branching (speciation) events within the tree, or minimize the tree length, have lower parsimony scores, and it is thought that trees with lower parsimony scores more accurately characterize evolution history among organisms [7].

The MP search method often produces a number of equally parsimonious solution trees for a given set of taxa. A common approach for interpreting solution, or *candidate* trees is to compute

one representative consensus tree. However, compacting the information from multiple trees into one summary tree can cause potentially valuable data to be lost [12].

In addition, during the heuristic search process, *search history* trees examined along the path to candidate trees are generally discarded without being analyzed. These trees can present a data-mining opportunity because they provide insight into search behavior in tree space [12]. A better understanding of search behavior can drive the design of better heuristics, which will ultimately lead to more accurate evolutionary trees.

Though interpreting large sets of trees is elemental in improving and understanding search heuristics, limited work has been done in this area. We use clustering to explore the topological differences within search history and candidate tree collections discovered during an MP search. We also present the cluster grid to visualize relationships among trees and enhance individual tree interpretation. We observed unique relationships within search history and candidate trees collected using different search parameters and we found that the cluster grid is an effective means to illustrate those patterns.

2 Related Work

Several researchers have explored the question of analyzing a collection of trees. However, we note that the research presented in this paper differs in three fundamental ways: (i) we are looking at extremely large collections of trees, (ii) we do not limit our tree collections to the most parsimonious trees; and (iii) the motivation for our work is on understanding search behavior in order to design better heuristics.

Maddison [8] explored another means of partitioning a collection of trees, based upon the lengths of trees and the number of branch rearrangements by which trees differ. He defined an *island* as a collection of trees less than or equal to a specified length that are topologically similar to one another. An island is a collection of interconnected short (parsimonious) trees that is separated from other islands by longer trees. Two trees are considered connected if they differ by a single rearrangement of branches. He suggested that each of these islands is an independent source of information that might be lost in a consensus tree.

Stockham, Wang, and Warnow [11] presented an alternative approach by using clustering algorithms on the set of candidate trees. They proposed bicriterion problems, in particular using the concept of information loss, and new consensus trees called characteristic trees that minimize the information loss.

Hillis, Heath, and St. John [5] explored the use of multidimensional scaling (MDS) of tree-to-tree pairwise distances to visualize the relationships among sets of phylogenetic trees. They found their technique to be useful for exploring "tree islands" (sets of topologically related trees among larger sets of near-optimal trees), for comparing sets of trees obtained from bootstrapping and Bayesian sampling, for comparing trees obtained from the analysis of several different genes and for comparing multiple Bayesian analysis.

3 Preliminaries

3.1 Maximum parsimony

Maximum parsimony (MP) is an optimization problem for inferring the evolutionary history of different taxa, in which it is assumed that each of the taxa in the input is represented by a string over some alphabet. The symbols in the alphabet can represent nucleotides (in which case, the input are DNA or RNA sequences), or amino-acids (in which case the input are protein sequences), or may even include discrete characters for morphological properties. It is also assumed that the strings are put into a multiple alignment, so that they all have the same length. Maximum parsimony then seeks a tree, along with inferred ancestral sequences, so as to minimize the total number of evolutionary events (counting only point mutations).

Formally, given two sequences a and b of the same length, the *Hamming distance* between them is defined as $|\{i : a_i \neq b_i\}|$ and denoted as $H(a, b)$. Let T be a tree whose nodes are labeled by sequences of length k, and let $H(e)$ denote the Hamming distance of the sequences at each endpoint of edge e. The *parsimony length* of the tree T is $\sum_{e \in E(T)} H(e)$. The MP problem seeks the tree T with the minimum length; this is the same as seeking the tree with the smallest number of point mutations for the data. MP is an NP-hard problem [4], but the problem of assigning sequences to internal nodes of a fixed leaf-labelled tree is polynomial [3].

3.2 Robinson-Foulds distance

In our experiments, we compare trees found by our Simple Local Search (SLS) algorithm and trees found using Paup*. We use the Robinson-Foulds (RF) distance to measure the topological distance between trees. The RF distance between two trees is the number of bipartitions that differ between them. It is useful to represent evolutionary trees in terms of *bipartitions*. Removing an edge e from a tree separates the leaves on one side from the leaves on the other. The division of the leaves into two subsets is the bipartition B_i associated with edge e_i. Let $\Sigma(T)$ be the set of bipartitions defined by all edges in tree T. The RF distance between trees T_1 and T_2 is defined as

$$d_{RF}(T_1, T_2) = \frac{|\Sigma(T_1) - \Sigma(T_2)| + |\Sigma(T_2) - \Sigma(T_1)|}{2}$$

We use the *RF rate*, which is obtained by normalizing the RF distance by the number of internal edges and multiplying by 100. (Assuming n is the number of taxa, there are $n - 3$ internal edges in a binary tree). Thus, the RF rate varies between 0% and 100%.

3.3 Simple Local Search Heuristic

Our Simple Local Search (SLS) heuristic operates by successively exploring the neighborhood of a current solution and moving to one of its neighbors. First, SLS creates a random sequence addition (RSA) to create the initial starting tree. To construct a RSA tree, we randomize the ordering of the sequences in the dataset. Afterwards, the first three taxa are used to create an unrooted binary tree, T. The fourth taxon is added to the internal edge of T that results in the best MP score. This process continues until all taxa have been added to the tree. Starting trees can also be based on neighbor-joining (NJ) [10] or by generating a starting tree randomly.

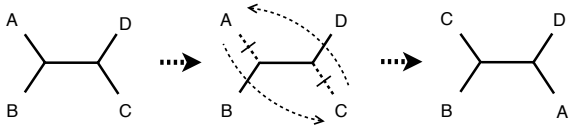

Figure 1: Nearest neighbor interchange (NNI)

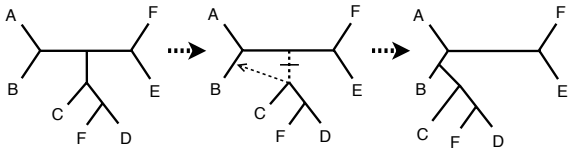

Figure 2: Subtree pruning and regrafting (SPR)

Once we have a tree T, we improve it by rearranging its edges in a way that improves its maximum parsimony score. There are three main types of rearrangement operations, which defines the neighborhood of T, used in phylogenetic search heuristics.

- The *nearest-neighbor interchange (NNI)* operation swaps two adjacent branches on the tree. In other words, it erases an interior edge on the tree, and the two branches connected to it at each end (so that a total of five branches are erased). Afterward, four subtrees are disconnected from each other. Four subtrees can be hooked together into a tree in three possible ways, where one of the trees is the original one (see Figure 1). For a tree T with n taxa, $2(n-3)$ neighbors can be examined for each tree. Local searches based strictly on NNI operations perform poorly in comparison to their SPR and TBR counterparts.

- A *subtree pruning and regrafting (SPR)* move consists of removing an edge from the tree with a subtree attached to it. The subtree is then reinserted into the remaining tree in all possible places, each of which inserts a node into a branch of the remaining tree (see Figure 2). Since there are n exterior edges and $n-3$ interior edges on an unrooted binary tree, the total number of solutions in the neighborhood is $4(n-3)(n-2)$.

- In a *tree-bisection and reconnection (TBR)* move, an interior branch is broken, and the two resulting fragments of the tree are considered as separate trees. All possible connections are made between a branch of one and a branch of the other (see Figure 3). If there are n_1 and n_2 species in the subtrees, there will be $(2n_1-3)(2n_2-3)$ trees in a TBR neighborhood.

With a mechanism for generating a neighborhood, we must decide which neighboring tree T' should be selected. SLS uses a *first improvement algorithm* to select a neighbor. If $score(T') < score(T)$, then T' is accepted to replace the current tree T. The search continues until there is no neighbor T' with a better score than the current tree, T. If no better neighbor can be found, a local optimum has been reached and SLS terminates.

3.4 Definition: Phylogenetic Search History

The search history of a phylogenetic heuristic is the set of neighbors selected along the search path to a local optimum. Formally, the sequence of trees encountered along the search path is defined

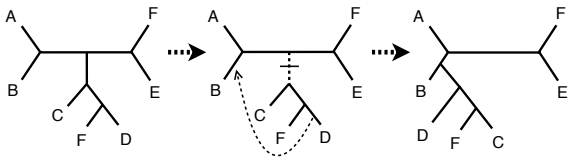

Figure 3: Tree Bisection and Reconnection (TBR)

as

$$P = (t_1, \ldots, t_m). \tag{1}$$

For a path P, the search examines tree t_i before tree t_j, where $0 \leq i < j \leq m$. There are m trees on the search path, where t_1 represents the initial (or starting) tree, and t_m is the final tree (e.g., local optimum). Consider a neighborhood relation $\mathcal{N}_\beta(t)$ which generates the neighboring trees of t using rearrangement scheme β, where $\beta \in \{\text{NNI, SPR, TBR}\}$. For example, $\mathcal{N}_{TBR}(t)$ produces all of the TBR neighbors of tree t. To capture all of the β neighbors along a search path P, then

$$\mathcal{N}_\beta(t) = \{t' | t' \text{ is a } \beta \text{ neighbor of } t\} \tag{2}$$

$\hat{\mathcal{N}}_\beta(P)$ is a mapping between a search path, P, and a list of neighboring tree sets, such that $\mathcal{N}_\beta(t)$ is the i^{th} tree set in $\hat{\mathcal{N}}_\beta(P)$ and t_i is the i^{th} tree in P.

Each run i of the heuristic results in a search path, P_i. The complete set of trees are $\mathcal{T} = \bigcup_{i=1}^{k} \hat{\mathcal{N}}_\beta(P_i)$, where k is the total number of runs. In our experiments, the number of runs, k, is 5.

4 Experimental Methodology

4.1 Collection of trees

A 44 taxa set of mammalian nuclear and mitochondrial gene sequences was used to gather trees [9]. SLS, Paup* and the `ape` package in the R statistical environment were used to generate datasets. SLS data sets are composed of search history trees, Paup* data sets contain candidate trees and R data sets contain trees with random topologies [1].

Three sets of SLS search history trees and three sets of Paup* candidate trees were obtained by MP searches employing NNI, SPR and TBR methods and will be designated `NNISLS`, `SPRSLS` and `TBRSLS`, `NNIPaup*`, `SPRPaup*` and `TBRPaup*`, respectively. To ensure uniformity, all search methods in Paup* and SLS were provided with the same five starting trees created using the RSA heuristic in Phylip. A final set of 250 random trees produced in R will be designated `Rand`.

A nexus format script the includes the following lines will produce a file containing 50 candidate trees for each MP search that is performed using Paup*:

```
begin paup;

...
set criterion=parsimony;

hsearch start=current multrees=yes swap=NNI nbest=50 savereps=yes;
...
```

```
end;
```

4.2 PAM algorithm

We implemented the Partitioning Around Medoids (PAM) algorithm, as described by Kaufman & Rousseeuw (1990). The algorithm seeks to partition the data into clusters in which the members of each cluster have a high level of similarity to each other and dissimilarity to the members of other clusters. The PAM algorithm consists of a *build phase* and a *swap phase*, which together search for k objects, each representing features that the objects in the cluster share and those that distinguish them from objects in other clusters. The clusters are built by assigning members of the dataset to the most representative of these k objects, or medoids [6].

Cluster assignments are based on a distance matrix for the observations that is used to minimize the sum of the dissimilarities of objects to their nearest medoids. We denote the distance between an object i and an object j as $d(i, j)$, where $d(i, j)$ can be computed using any number of metrics, including the Euclidean and Manhattan distance. In our case, however, we use the RF-distance.

The dissimilarity value for a specified i and j is designated by D_{ij}, is always non-negative, and small values near zero suggest i and j are more similar, whereas large values suggest dissimilarity. Additionally, we denote D_j as the dissimilarity of j to its closest medoid representative [2].

In the *build phase*, PAM selects k medoids. It chooses a candidate medoid i based on the difference between D_j, for some observation j, and the dissimilarity of i from j, or C_{ji}= max$(D_j - D_{ji}, 0)$. If the value of C_{ji} is positive, then i is more similar to j than any of the previously selected medoids. The total benefit of including i as a medoid is obtained by computing the sum of C_{ji} for all possible j, or $\sum_j C_{ji}$. The algorithm chooses k objects i that maximize ΣC_{ji} over all possible j. Every observation is assigned a cluster, based on the most similar medoid.

PAM then enters the *swap phase*, where it swaps a non-medoid object h with medoid i for all possible i and h to determine if there is an exchange that improves the quality of the clustering results. PAM computes this by calculating $C_{jih} = C_{jh} - C_{ji}$, for all j clustered with i. If C_{jih} is negative, than h is an improvement. If $\sum_j C_{jih}$ is negative, then element i is swapped with h. The algorithm continues until a local minimum is reached where no medoid can be replaced by an observation that further minimizes the sum of dissimilarities for a given cluster. Through this process each observation is assigned to a medoid in the solution set and a medoid and the objects assigned to it define a cluster.

We used the PAM algorithm from the `cluster` package in the R statistical environment. In our application, each cluster should represent a collection of trees with shared topological features that distinguish them from trees in other clusters, as measured by the RF-distance.

4.3 Clustering validation

A *silhouette* represents how well the medoid in a cluster characterizes the observations assigned to it. The average *silhouette width* is a measurement of how similar all the objects in a cluster are to the cluster center, as compared to how similar they are to the closest medoid of another cluster. This value can be used to determine the value of k that best suites the dataset. The clustering that yields the largest width has the most well defined clusters and the most suitable value for k (see Figure 4).

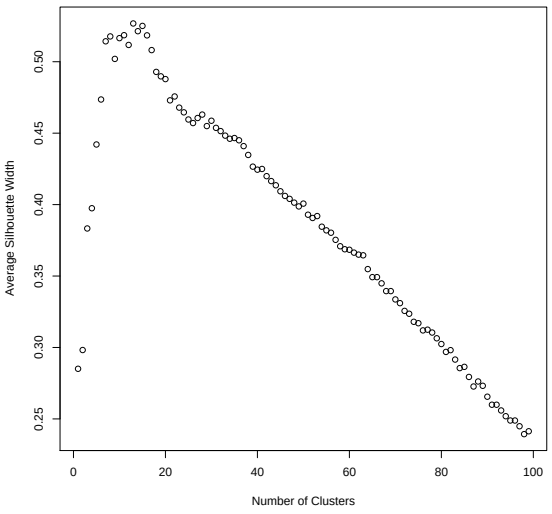

Figure 4: Silhouette cluster optimization for NNISLS trees.

If i is an observation and A is the cluster it is assigned to, then the average dissimilarity of i to all other objects of A is denoted $a(i)$. Moreover, if C is some cluster that i does not belong to, then let $d(i, C)$ equal the average dissimilarity of i to all members of C. The cluster B that minimizes $d(i, C)$ for i has a dissimilarity b =min $d(i, C)$, where $C \neq A$ and $b(i) = d(i, B)$. The silhouette value $s(i)$ is thus defined as

$$S(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)}$$

Therefore, the range of silhouette widths is $-1 \leq s(i) \leq 1$, where a value close to -1 suggests i has been poorly assigned to its cluster, one near 0 suggests i fits equally well in two clusters, and an $s(i)$ near 1 suggests i has been well clustered. The overall average silhouette width S for the entire clustering is the average of $s(i)$ over all observations. The range of S is from 0 to 1, where $0 \leq S \leq .25$ indicates that no structure has been found within the dataset, $.26 \leq C \leq .50$ implies that a weak and potentially artificial structure has been found, $.51 \leq C \leq .7$ implies that there is a reasonable one and $.71 \leq C \leq 1$ indicates that a strong structure has been found and the clustering is well defined.

4.4 Graphical Interpretation

We introduce the cluster grid as a means of graphically displaying the clustering information obtained from sets of trees. The cluster grid was produced in the R environment and the clusterGrid function is available from the authors upon request. Each cell in the cluster grid represents an observation, or in this case a tree. The tree in the ith row and the jth column of the m x n grid can be designated $T_{i,j}$, and its numerical location within the dataset equals $(i-1) * n + j$.

The function takes a dissimilarity matrix or a table of observations as input, computes the clustering assignments internally, and produces a plot based on user-specified parameters.

Given a dissimilarity matrix, the user may specify the method of ordering and coloring cells within the grid. If a table of observations is given as input, the user must specify the characteristic that defines the clustering, the method of ordering the values and the metric to compute the distance matrix. The function can compute distance matrices based on the Euclidean or Manhattan distances between data points. Trees can be clustered by parsimony score or RF distance, they can be ordered by parsimony score or time stamp within sequential search histories, and they can be colored to distinguish between different searches (trials) or between the quartiles that define the range of data values.

In Figure 5, the `NNISLS` data set is plotted in a cluster grid. Each consecutive search that begins from a different starting tree is noted with a different color. For example, the cells of trees from the first search history are colored red. We also note that $i = 3$ and $j = 12$ for the final, optimized tree in the first run. Thus it is known that $T_{3,12} = 38$, which means there are 38 trees in the search path for the first run. Cluster assignments can also be noted by the alphanumeric symbol on each cell. The symbol for $T_{3,12}$, for example is $C3$, which identifies it as being a member of the third cluster.

The data in this cluster grid uses the optimal value of k, but the user may specify some other k, along with values of n and m that determine the number of cells in the grid, and a vector of integers that defines the size of each search history path (or the trial size).

Figure 6 depicts a cluster grid of the same set of trees, but the color and alphanumeric labels for each cell define different features. The colors in this grid represent individual tree cluster assignments and the symbol in each cell conveys the run (or trial) that each tree was produced during. For example cells $T_{1,1}$ through $T_{1,13}$ are colored red, which means that trees 1 through 13 are members of the first cluster. These trees are also all labeled $R1$, which means that they are part of the first SLS search history, or first SLS run.

5 Experimental Results

5.1 Cluster analysis for SLS trees

The `NNI-SLS` trees have an optimal clustering around 14 medoids, with a silhouette width of 0.53. The trees from each of the five runs are assigned to separate clusters (see Figures 5 and 6). In the first run, for example, the first 34% of the trees are assigned to cluster 1, the following 34% to cluster 2 and the final 32% are assigned to cluster 3. Clusters 1, 2 and 3 contain trees taken solely from the first run. Similarly, the second run is isolated in clusters 4 and 5, the third run in clusters 6 and 7, the fourth run in clusters 8, 9, 10 and 11, and the fifth run in clusters 12 and 13. As in the first run, trees from the four other runs cluster according to their locations in the search paths.

When a value of $k = 5$ is imposed on the clustering, the silhouette width decreases to 0.4. The trees from each run do not cluster independently and they do not always cluster according to their locations in the search paths (see Figure 7).

The `SPR-SLS` dataset has a significantly larger number of medoids than the `NNI-SLS` dataset, with $k = 44$ and a silhouette width of 0.52. Again each run clusters independently, with the exception that the trees from the last 10% of the first run, the last 6% of the second run and the last 11% of the fifth run cluster together, and trees from the last 12% of the second run clusters with trees from the last 5% of the fourth run (see Figures 8 and 9).

When `SPR-SLS` is clustered around 5 medoids, it has a silhouette width of 0.23. The trees

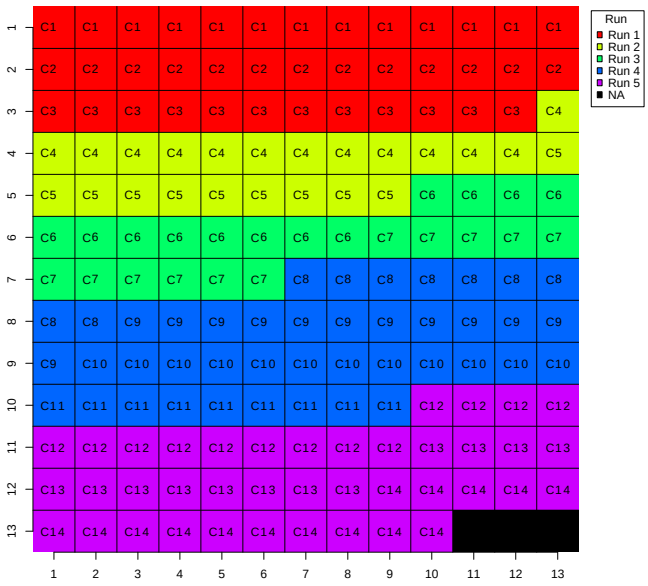

Figure 5: NNI-SLS trees colored by run (k=14, silhouette width= 0.53).

from each run do not cluster independently and instead intermittent segments of more than one run share the same cluster (see Figure 10). While only the optimal and near-optimal trees from `SPR-SLS` have runs that share the same clusters when $k = 45$, no such pattern exists when $k = 5$.

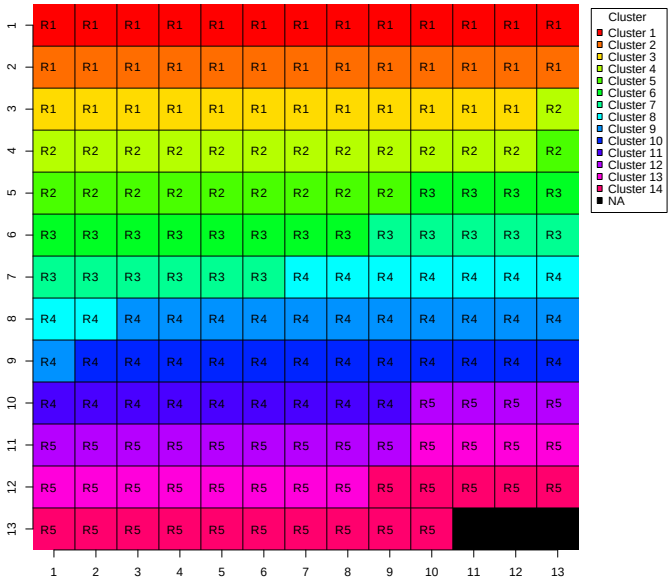

Figure 6: NNI-SLS trees colored by cluster assignment (k=14, silhouette width= 0.53).

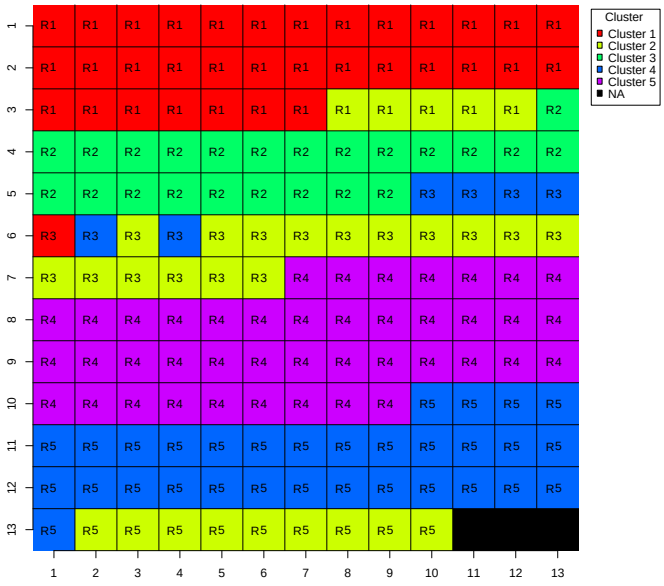

Figure 7: NNI-SLS trees colored by assignment to a manually selected number of clusters (k=5, silhouette width= 0.40).

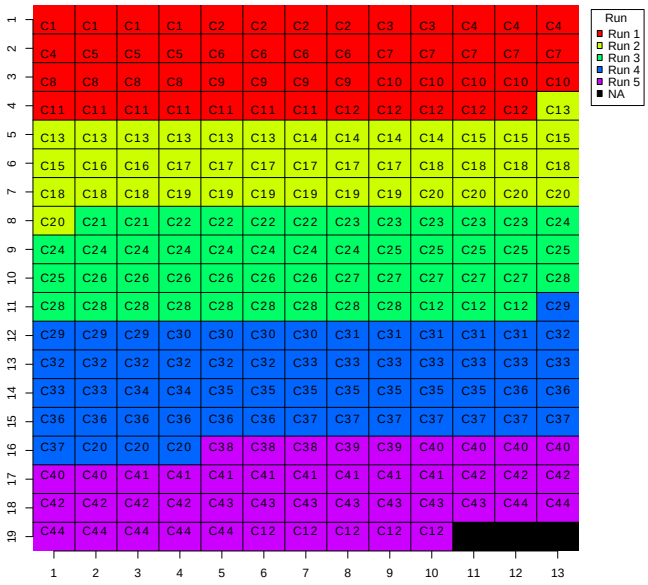

Figure 8: SPR-SLS trees colored by run (k=44, silhouette width= 0.52).

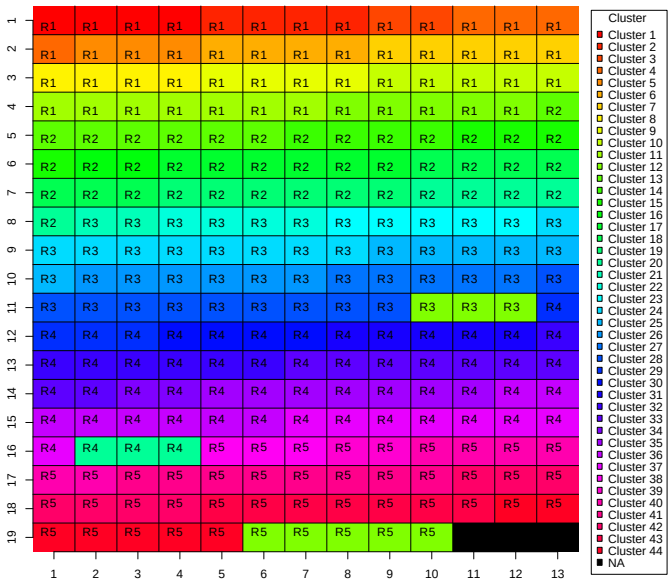

Figure 9: SPR-SLS trees colored by cluster assignment (k=44, silhouette width= 0.52).

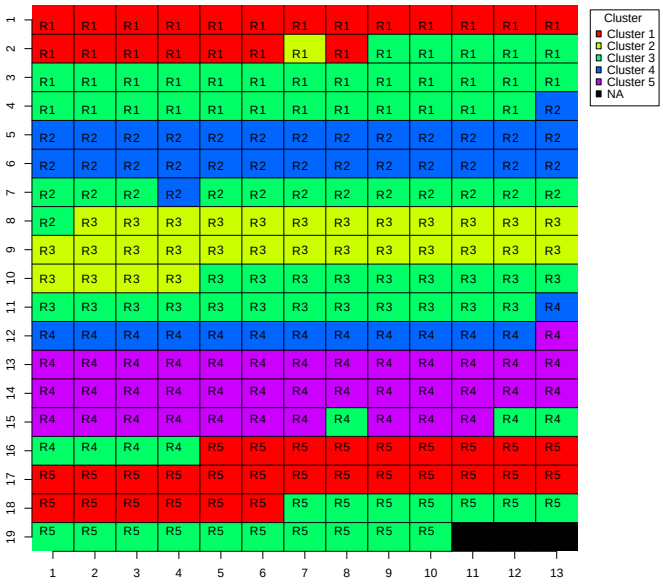

Figure 10: SPR-SLS trees colored by assignment to a manually selected number of clusters (k=5, silhouette width= 0.23).

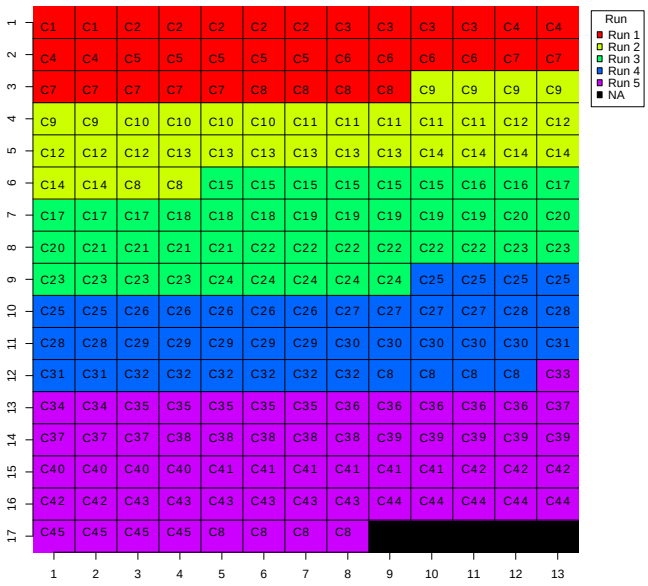

Figure 11: TBR-SLS trees colored by run (k=45, silhouette width=0.55).

Just as with SPR-SLS, the value of $k = 45$ for TBR-SLS also gives it a significantly larger number of medoids than NNI-SLS, and the existence of structural qualities within the clusters is verified with a silhouette width of 0.55. Just as with the optimally clustered trees for NNI-SLS and SPR-SLS, each run forms isolated clusters based on the location of trees within the specified search. Exception is made for the trees from the last 11% of the first run, 9% of the second run, 10% of the fourth run and 7% of the fifth run, which cluster together (see Figures 11 and 12).

When TBR-SLS is coerced into clustering around five medoids, the silhouette width decreases to 0.23 and the trees no longer exhibit a significant clustering pattern (see Figure 13)

5.2 Cluster analysis for Paup* trees

NNI-Paup* has five optimal medoids that are validated with a silhouette width of 0.75. The trees from each run form one cluster that is isolated from the other runs (see Figures 14 and 15). Since the NNI-Paup trees have a preferred k of 5, no coercion is performed.

The SPR-Paup* trees cluster the most tightly around 2 medoids and have a silhouette width of 0.61. In this case, every tree from the first, second, fourth and fifth run clusters around one medoid and all the trees from the second run are assigned to another medoid (see Figures 16 and 17).

The silhouette width decreases to 0.43 when SPR-Paup* is clustered around 5 medoids. Trees from the first, second and third runs are mixed in three clusters, but trees from the third run compose an isolated cluster, as do trees from the fifth run (see Figure 18).

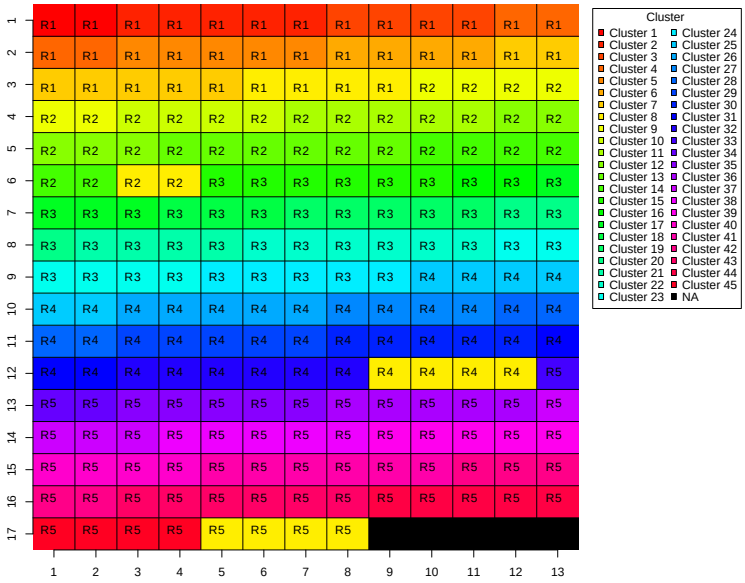

Figure 12: TBR-SLS trees colored by cluster assignment (k=45, silhouette width=0.55).

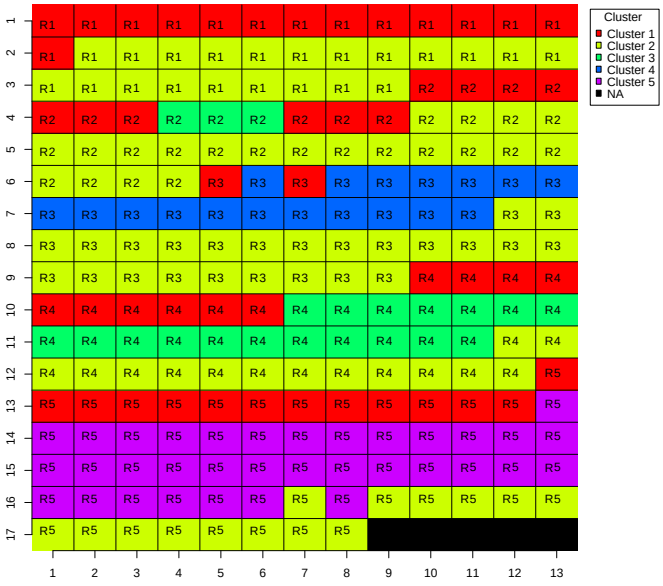

Figure 13: TBR-SLS trees colored by assignment to a manually selected number of clusters (k=5, silhouette width=0.23).

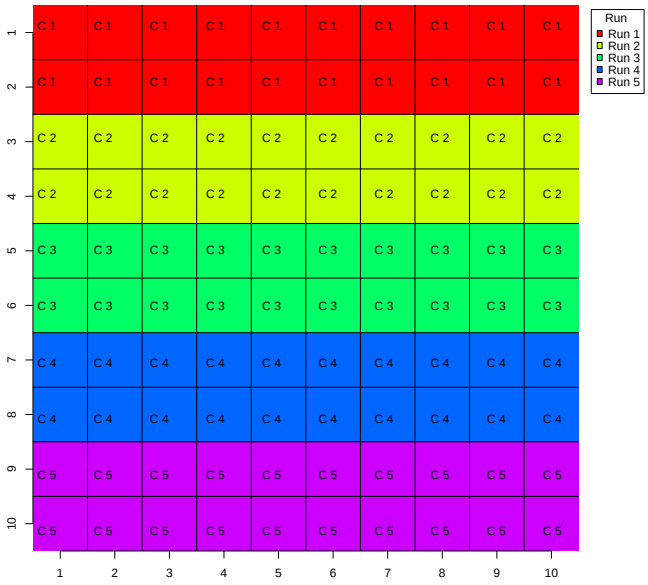

Figure 14: NNI-Paup* trees colored by run (k=5, silhouette width= 0.75).

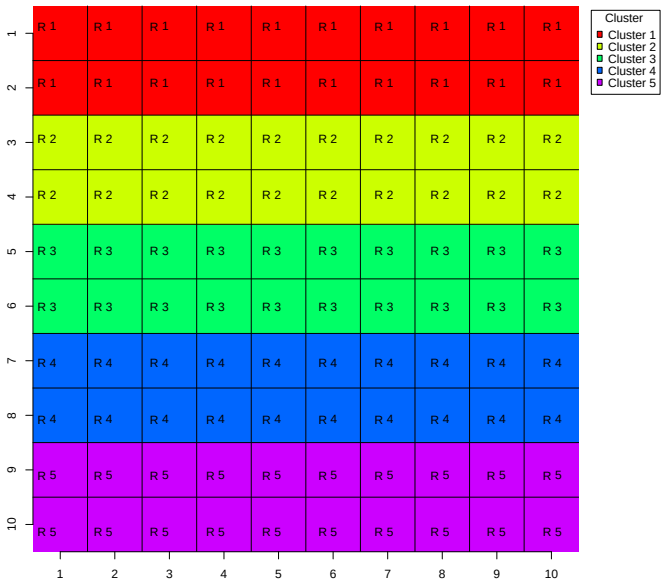

Figure 15: NNI-Paup* trees colored by cluster assignment (k=5, silhouetted width= 0.75).

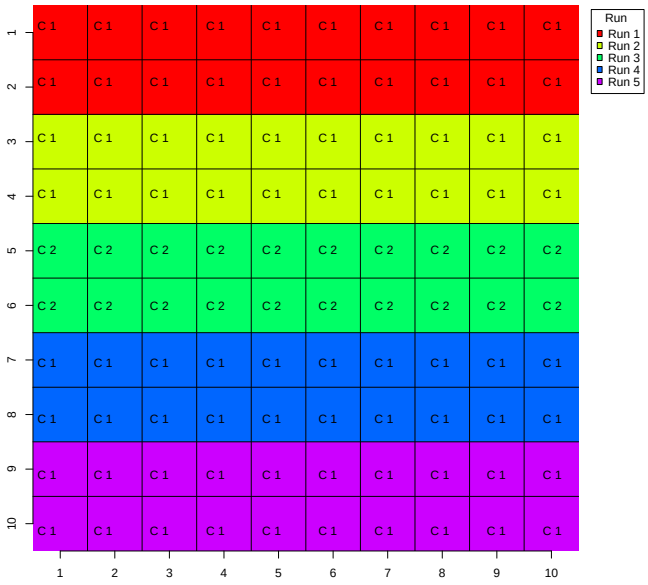

Figure 16: SPR-Paup* trees colored by Paup run (k=2, silhouette width =0.61).

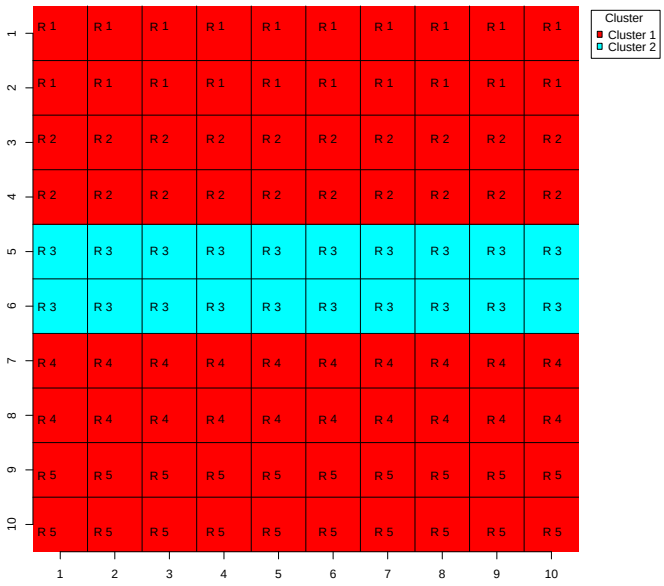

Figure 17: SPR-Paup* trees colored by cluster assignment (k=2, silhouette width =0.61).

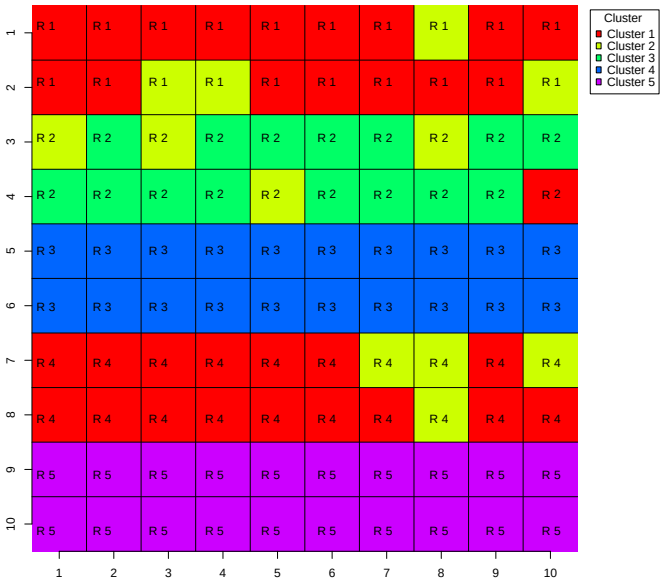

Figure 18: SPR-Paup* trees colored by assignment to a manually selected number of clusters (k=5, silhouette width= 0.43).

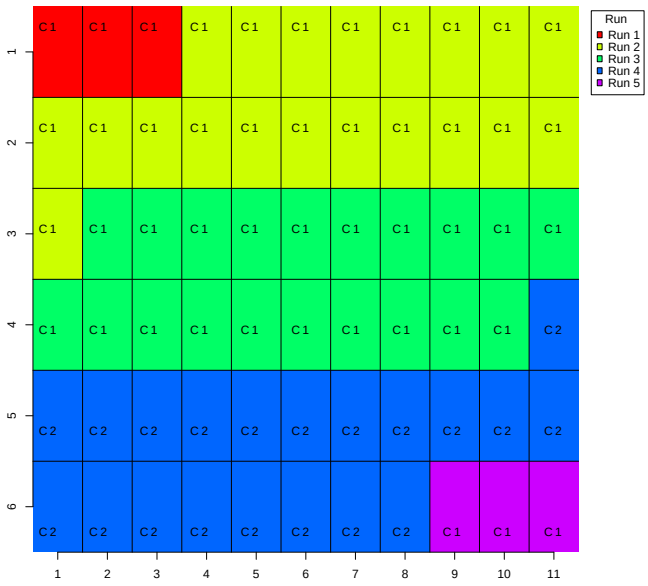

Figure 19: TBR-Paup* trees colored by run (k=2, silhouette width=0.81).

The `TBR-Paup*` trees have an optimal k of 2 and a silhouette width of 0.81. The first and fourth run form isolated clusters, while the second, third and fifth run all cluster together (see Figures 19 and 20) .

The silhouette width of `TBR-Paup*` is reduced to 0.30 when k is 5. Under these conditions no clearly defined clustering behavior is observed (see Figure 21).

5.3 Cluster analysis for random trees

The `Rand` trees have an optimal silhouette width of 0.006 when there are 66 medoids and a silhouette width of 0.0005 when a value of 5 is enforced for k. Consequently there is no evidence of any structurally relationships within the collection of trees (see Figures 22 and 24).

6 Discussion

The silhouette width of the optimal clusterings for `NNI-SLS`, `SPR-SLS`, `TBR-SLS`, `NNI-Paup*`, `SPR-Paup*` and `TBR-Paup*` suggest that the clusterings are a reasonable characterization of the topological relationships within the sets of trees. We use the cluster grids to observe that the trees produced from each branching method have distinguishing cluster patterns for the optimal values of k, but cluster poorly for the manually selected k value of 5, with the exception of those cases when k has an optimal value of 5. The imposed value is based on the number runs that are performed to gather each set of trees in order to examine whether the trees cluster according to one of the 5 runs they are collected from.

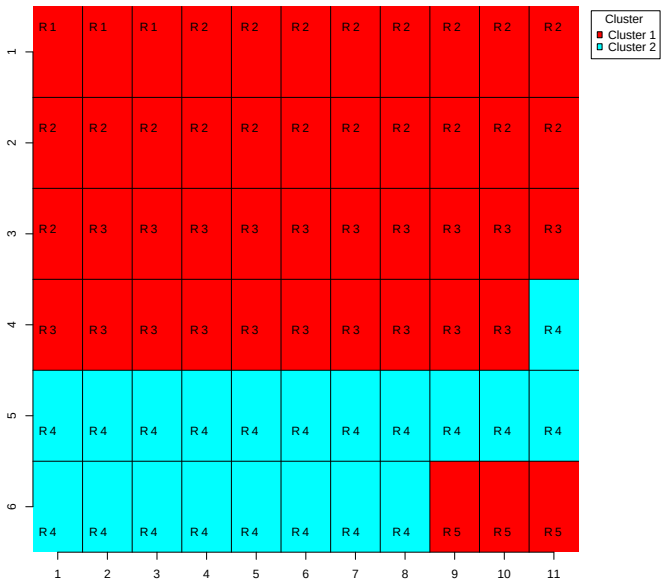

Figure 20: TBR-Paup* trees colored by cluster assignment (k=2, silhouette width=0.81).

6.1 Comparison of SLS trees

In the case of `NNI-SLS`, each cluster is composed of an individual run, and each run breaks down into between two and five clusters in which tree-medoid assignments are related to the time in the path that the individual trees are found. This correlates with the parsimony scores of each tree because those trees selected earlier have lower parsimony scores than those trees selected later. Logically those trees that are neighbors have more topological similarities and will thus be regarded as more similar from a clustering perspective. The trees found at the same time in different runs do not cluster together, which indicates that each search follows a unique path and collects trees that are topologically unique to that search history. Further, the optimal trees from each run cluster separately, and this fact can act as confirmation that distinct tree islands do exist in tree collections. On the other hand, this might also suggest that the limited nature of the NNI search neighborhoods cause the search to get stuck in local minima and terminate before honing in on those structurally similar and most optimal trees.

`SPR-SLS` runs have a tendency to form isolated clusters, however the near optimal and optimal trees form one isolated cluster, which is a notable exception to this trend. It may indicate that although the four searches take topologically distinct paths, they end up finding similar optimal trees and are less prone to getting stuck on local minima. Additionally, perhaps the optimal clustering for `SPR-SLS` is significantly larger than for `NNI-SLS` because of the increased structural diversity among the `SPR-SLS` trees.

Again in the case of `TBR-SLS`, each run has isolated clusters based on the search time within a history, with the exception of the near optimal and optimal trees from four of the runs that cluster together and independently from the rest of the trees. These findings can be attributed to search characteristics that are similar to those of a search implementing the SPR method. `TBR-SLS` also has an optimal value of k that is similar to that of `SPR-SLS` and which may suggest that the `TBR-SLS` runs and the `SPR-SLS` runs are composed of similar ranges of tree topologies that are more

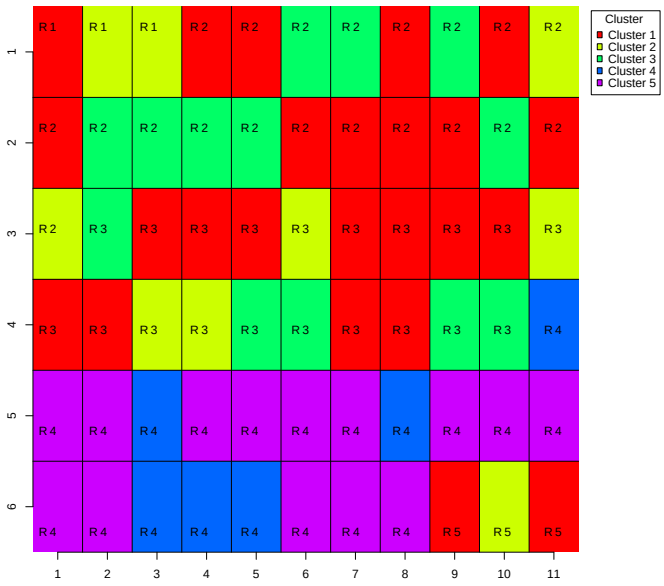

Figure 21: TBR-Paup* trees colored by assignment to a manually selected number of clusters (k=5, silhouette width= 0.30).

diverse than that found among the `NNI-SLS` trees.

`NNI-SLS`, `SPR-SLS`, `TBR-SLS` do not cluster well when the number of medoids is preselected to be 5 because the silhouette widths are not large enough to validate the existence of structure within the clusters. This notion is supported by inconsistencies within the clustering assignments. Though remnants of the clustering patterns for optimal values of k exist, they become less definable.

Thus, each of the five runs for each of the branch swapping methods does not have enough unique qualities to distinguish it as an entire single cluster unit from the other runs. However, when more medoids are permitted and the runs can divide into subgroups, more subtle features within a search history are identifiable. Once the runs are broken down into these characterized segments they are distinguishable, with the exception of near optimal and optimal trees in the previously mentioned cases.

6.2 Comparison of Paup* trees

The exclusive quality of all the `NNI-Paup*` clusters suggests that each Paup* search finds a topologically distinct set of solution trees. Just as in the case of `NNI-SLS`, this finding verifies that unique tree islands do exist within the solution trees and can act as further evidence that consensus methods may not be a information-rich means of summarization. Further, the `NNI-Paup*` trees exhibit the same clustering patterns as the low-scoring search history trees from `NNI-SLS`. Since the low-scoring trees from the SLS search are in theory analogous to the candidate trees from the Paup* search, the cluster uniformity between the two sets of trees confirm that the two heuristics discover trees with similar topological relationships.

Just as with the candidate trees from most of the `SPR-SLS` and `TBR-SLS` searches, the candidate trees from the majority of the Paup* runs from `SPR-Paup*` and `TBR-Paup*` do not appear to have distinct topological features. This is proven in both datasets by the assignment of trees

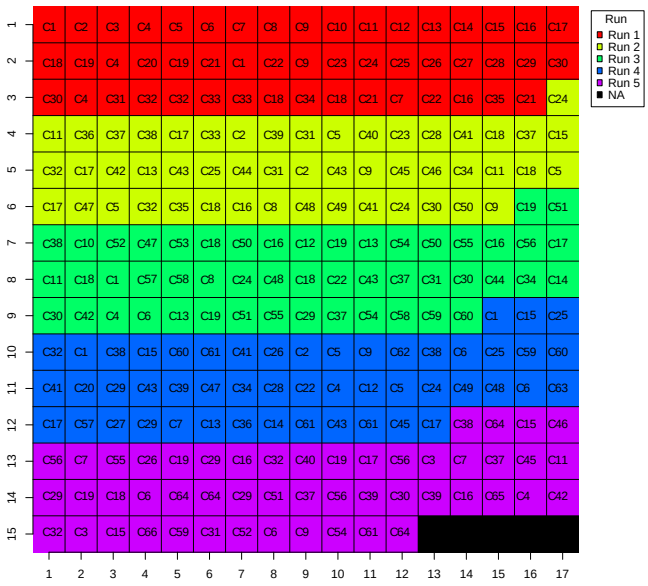

Figure 22: Rand trees colored by run (k=66, silhouette width=0.006).

from four runs to inter-run cluster. This may mean that one general tree structure is in fact the most parsimonious, and the SPR and TBR searches are able to discover it in the majority of instances. In some cases, however, the search will produce a set of topologically distinct solutions, as demonstrated by the single run from each data set that clusters separately.

Furthermore, forcing the `SPR-Paup*` and `TBR-Paup*` trees to be identified with more than the optimal number of clusters appears to have a similar effect among both sets of trees. It degrades the structural validity that was observed in the previous clusterings, and thus the observations within these clusters are not considered significant.

Therefore, in the case of the NNI branch swapping method, the runs can be distinguished based on the candidate trees each one produces. This does not hold true for the other two branching methods, and instead the candidate trees from most of these runs share similar features.

6.3 Comparison of random trees

Finally, the random trees we examine behave as an experimental control. There should not be a relationship among the topologies of random trees and the significantly low silhouette width and the lack of visual relationships in the associated cluster grids supports this supposition.

7 Conclusions and Future Work

We found clustering to be an effective means of identifying relationships among trees within heuristic searches. Unique search behavior is observed for each of the three branching methods, with NNI trees showing the most topological similarity within each search run and SPR and TBR trees containing a larger variety of tree structures. The clustering technique can also be used to identify which regions of the search space are represented in a run, and which runs become stuck in a local minima.

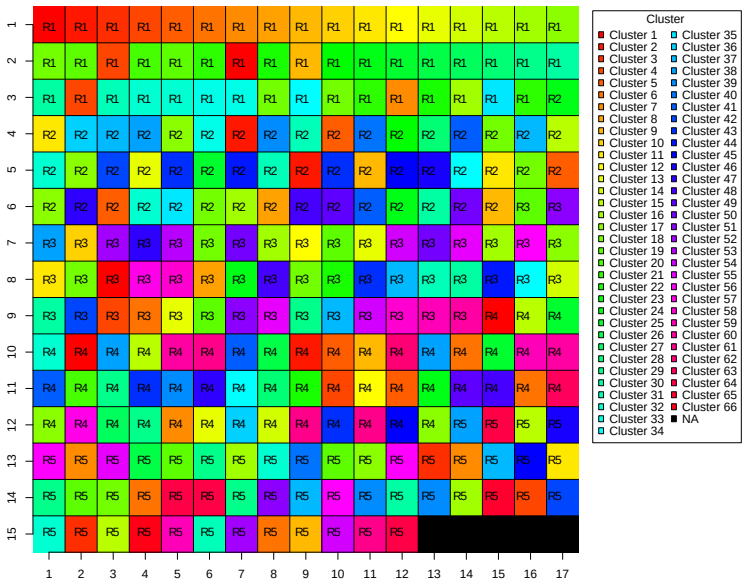

Figure 23: Rand trees colored by cluster assignment (k=66, silhouette width=0.006).

Our grid plot is a valuable tool for summarizing the information contained in individual trees. The use of color to identify clusters and runs allows for structural relationships to be quickly identified. The ability to recluster and reorder the trees according to user-defined specifications makes the plot a flexible addition to the phylogenetic visualization tools currently available.

Color becomes a restrictive feature of the grid plot when there are a large number of clusters because it becomes difficult to distinguish between the shades of a color that represent different clusters . Furthermore, the plot may become too complex when thousands of trees are analyzed.

The *tessellation plot* is a tentative means of providing more individual tree information in one plot than the grid plot, and the associated *summary plot* would be functional for extremely large collections of trees (see Figures 25 and 26). Two triangles in each cell of the tesselation plot represents a tree. The upper triangle is colored and labeled according to the tree's cluster assignment and the bottom triangle is colored and labeled according to each tree's run. The labeled box in the top right corner of the cell includes the tree identification number and the labeled box in the bottom left corner includes information about the parsimony score and the RF distance from the best scoring tree in the collection. The summary plot includes one cell to represent each cluster-run combination that exists in the larger tessellation plot. In 26, the cell that pairs the first run with cluster 1 represents all the trees from the first run that are assigned to cluster 1. The plot displays the cluster and run numbers in the same fashion as the tessellation plot, but it also provides summary information about the cluster composition and density for each pairing. The upper triangle displays the cluster's average silhouette width ($s(i)$), and the bottom triangle includes two ratios to encapsulate the cluster makeup. The ratio in the bottom left corner relates the number of trees represented by the cell to the overall number of trees in the cluster. In Figure 26, the value (2/2) means that two trees from the first run are assigned to cluster 1, which contains a total of 2 trees. The ratio in the bottom right corner compares that same number of represented trees to the overall size of the run. Again, in Figure 26, 2 of the 34 trees in the first run are assigned to cluster 1 and so the cell is labeled (2/34).

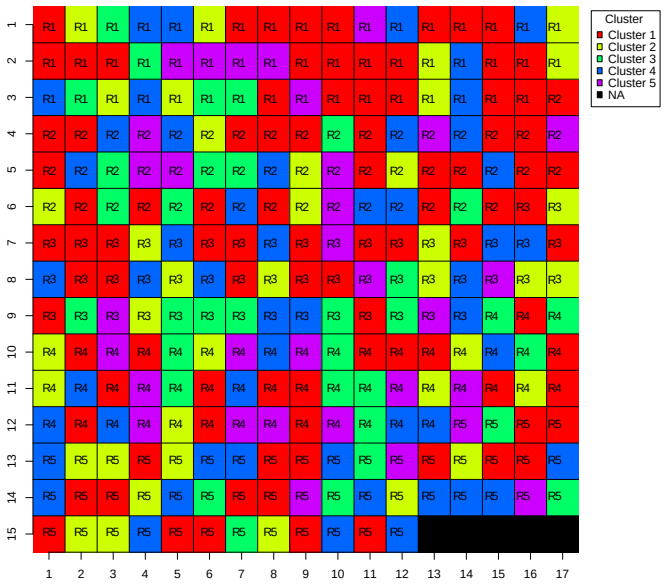

Figure 24: Rand trees colored by forced cluster assignment (k=5, silhouette width= 0.0005).

The tessellation plot augments the MDS cluster visualization that Hillis, Heath, and St. John [5] investigated. Where their technique provides a birds-eye view of a collection of trees, our approach offers a detailed representation of individual trees. The MDS method relates the trees as unlabeled points in 2-dimensional space. Points may be quite far apart if the trees are extremely topologically different or they may overlap if the trees have many branches in common. The tessellation plot, on the other hand, magnifies the amount of tree information available by providing an identification number, parsimony score, RF value, run membership and cluster assignment for every tree. It also gives trees equal representation within the plot by ordering them according to their locations within a search run. Thus, topological distinctions between trees are made with cluster colors and labels instead of spatial positions. Our work provides a foundation for future research about the tessellation plot, and more research is needed to evaluate its functionality.

8 Acknowledgements

The authors wish to thank Bill Murphy and Matt Yoder for providing us with the biological dataset used in this study. We would also like to thank Hyun Jung Park for providing SLS code and Seungjin Sul for providing the Hash RF algorithm to compute the RF distance matrix.

References

[1] J. C. E. Parasdis, J. Claude, and J. Strimmer. APE: analyses of phylogenetics and evolution in r language. *Bioinformatics*, 20:289–290, 2004.

[2] S. Craighead and B. Klemesrud. Stock selection based on cluster and outlier analysis. In *15th International Symposium on the Mathematical Theory of Networks and Systems*, 2002.

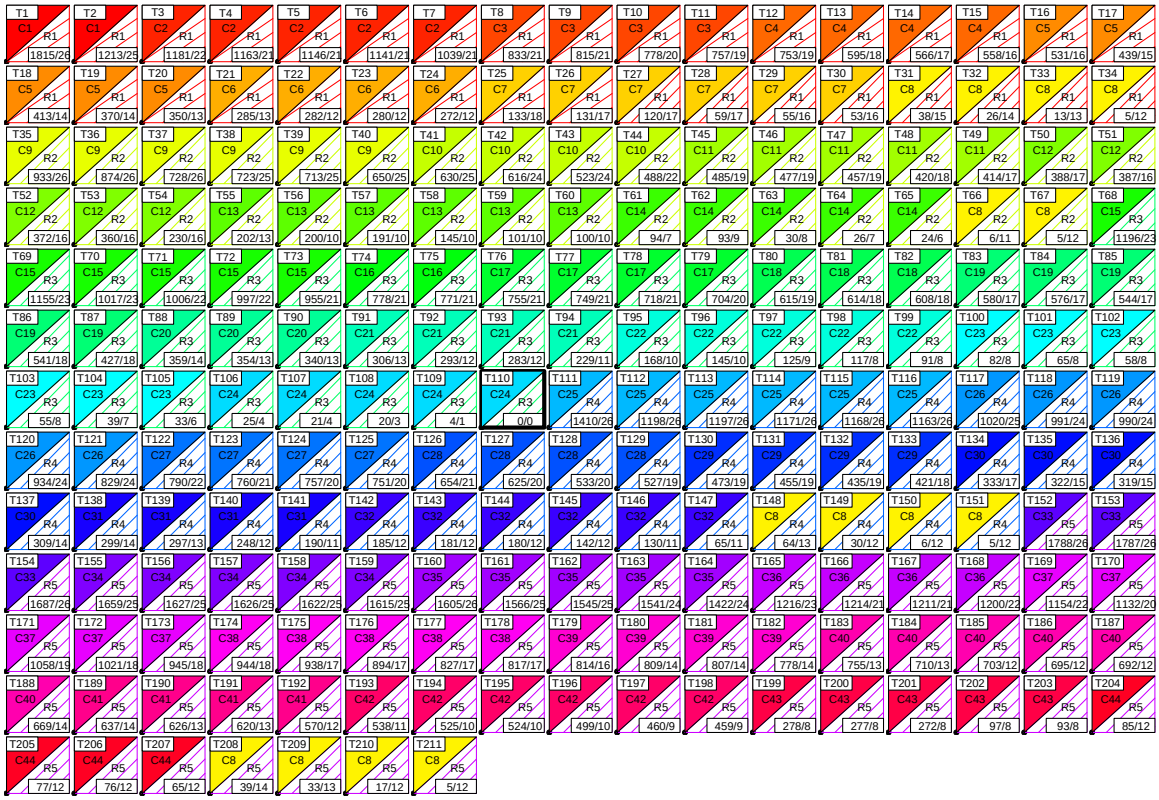

Figure 25: TBR-SLS tessellation plot.

[3] W. M. Fitch. Toward defining the course of evolution: minimal change for a specific tree topology. *Syst. Zool.*, 20:406–416, 1971.

[4] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3:43–49, 1982.

[5] D. M. Hillis, T. A. Heath, and K. S. John. Analysis and visualization of tree space. *Syst. Biol*, 54(3):471–482, 2005.

[6] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: an introduction to cluster analysis*, chapter Partitioning Around Medoids (Program PAM). John Wiley & Sons, Inc., New York, 1990.

[7] C. Linder and T. Warnow. Overview of phylogeny reconstruction. In S. Aluru, editor, *Handbook of Computational Biology*, Handbook of Computational Biology. Chapman & Hall, 2005.

[8] D. Maddison. The discovery and importance of multiple islands of most parsimonous trees. *Syst. Bio.*, 42(2):200–210, 1991.

[9] W. J. Murphy, E. Eizirik, S. J. O'Brien, O. Madsen, M. Scally, C. J. Douady, E. Teeling, O. A. Ryder, M. J. Stanhope, W. W. de Jong, and M. S. Springer. Resolution of the early placental mammal radiation using bayesian phylogenetics. *Science*, 294:2348–2351, 2001.

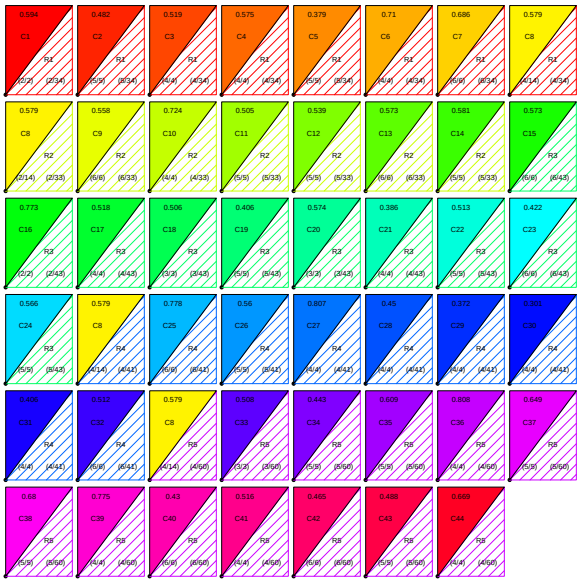

Figure 26: TBR-SLS summary plot.

[10] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructiong phylogenetic trees. *Mol. Biol. Evol.*, 4(406–425), 1987.

[11] C. Stockham, L. S. Wang, and T. Warnow. Statistically based postprocessing of phylogenetic analysis by cluste ring. In *Proceedings of 10th Int'l Conf. on Intelligent Systems for Molecular Biology (ISMB'02)*, pages 285–293, 2002.

[12] S. Sul and T. L. Williams. A randomized algorithm for comparing sets of phylogenetic trees. In *Proc. Fifth Asia Pacific Bioinformatics Conference (APBC'07)*, 2007.