

Texas A&M CS Technical Report 2009-12-1

December 14, 2009

Large-Scale Analysis of Phylogenetic Search Behavior

by

Hyun Jung Park¹

Department of Computer Science, Rice University

`hp6@cs.rice.edu`

Seung-Jin Sul and Tiffani L. Williams

Department of Computer Science and Engineering

Texas A&M University

`{sulsj,tlw}@cse.tamu.edu`

¹Most of this work was done while at Texas A&M University.

ABSTRACT

Phylogenetic analysis is used in all branches of biology with applications ranging from studies on the origin of human populations to investigations of the transmission patterns of HIV. However, inferring evolutionary trees is not a trivial task. Most analyses rely on effective heuristics for obtaining accurate trees. However, relatively little work has been done to analyze quantitatively the behavior of phylogenetic heuristics in tree space. A better understanding of local search behavior can facilitate the design of better heuristics, which ultimately lead to more accurate depictions of the true evolutionary relationships.

In this paper, we present new and novel insights into local search behavior for maximum parsimony on three biological datasets consisting of 44, 60, and 174 taxa. In particular, unlike traditional studies, we do not solely focus on analyzing the most parsimonious trees found during a search. By analyzing thousands of trees—which have a variety of scores—our study helps provides insights regarding a search’s behavior in tree space. In particular, our results show that as the search algorithm climbs the hill to a local optima, the trees in the neighborhood surrounding the current solution improve as well. Similar observations are found concerning the topological distances between trees. Furthermore, the search is quite robust to a small number of randomly selected neighbors. Thus, our work shows how to gain insights into the behavior of local search algorithm by exploring a large diverse collection of trees. arge diverse collection of trees.

1 Introduction

Phylogenetics is concerned with inferring the genealogical relationships between a group of organisms (or taxa). These evolutionary relationships are typically depicted in a binary tree, where leaves represent the organisms of interest and edges represent the evolutionary relationships. Phylogenetic trees have been used successfully in designing more effective drugs, tracing the transmission of deadly viruses, and guiding conservation and biodiversity efforts [2], [10], [18]. However, inferring evolutionary trees is not a trivial task. Since it is impossible to know the true evolutionary history for a set of organisms, the problem is often reformulated as an NP-hard optimization problem. Here, trees are given a score, where trees with better scores are believed to be better approximations of the truth. Given the exponential number of potential hypotheses (or trees) for a set of taxa, an exhaustive exploration of the tree space is not possible. Instead, phylogenetic inference relies on effective heuristics for obtaining good-scoring trees. However, relatively little work has been done to analyze quantitatively the *behavior* of phylogenetic heuristics in tree space. A better understanding of search behavior can drive the design of better heuristics that ultimately lead to more accurate reconstructions of phylogenetic trees.

The long-term objective of our work is to design high-performance phylogenetic heuristics to reconstruct large-scale phylogenies such as *The Tree of Life*, the evolutionary history of all known organisms. In this paper, we exploit the data mining and information visualization opportunities that exist among the large collection of trees found by a phylogenetic heuristic. For example, such a collection of trees could consist of:

- best-scoring trees found during a search; or
- all of the trees visited by a phylogenetic heuristic.

Traditional analysis techniques for phylogenetic heuristics solely focus on the set of best-scoring trees found in either a maximum parsimony or maximum likelihood analysis. How-

ever, such trees represent a very small fraction of the set of trees explored during a phylogenetic search. The *novelty* of our work is the emphasis on analyzing the entire set of trees examined by a phylogenetic heuristic (i.e., its search history), which can easily contain hundreds of thousand of trees. For us, all trees explored by a phylogenetic search are important—not only the equally most parsimonious ones. Without fully investigating where a heuristic has been (i.e., its behavior) in tree space, the search process itself is like a “black box” to the user.

In this paper, we study the behavior of a simple phylogenetic heuristic, which we call *Simple Local Search (SLS)*, for solving the maximum parsimony (MP) problem. Under MP, the tree that explains the data with the fewest evolutionary events (i.e., mutations) is the one that is preferred. For most dataset sizes of interest (> 20 taxa), heuristics must be used since MP is a NP-hard problem. Our SLS heuristic is a hill-climbing (or local search) heuristic that greedily selects trees based on their MP score until a local optimum is reached. Since we implemented SLS, we can collect a variety of data regarding the choices that the SLS heuristic makes during a search. Commercial phylogenetic software such as PAUP* [17] does not give us this type of control.

Our experiments with open-source phylogenetic software such as Phylip [5] show that it performs poorly in comparison to PAUP*. Our SLS algorithm, however, performs comparably to PAUP* and provides us with the data collecting capability we need to study quantitatively the behavior of local search heuristics. Moreover, for moderately-sized datasets (> 250 taxa), local search algorithms play an important role in the success of more powerful approaches such as Parsimony Ratchet [12], Recursive-Iterative DCM3 [13], and TNT [7]. So, understanding the behavior of local search heuristics such as SLS has a tremendous impact in providing a foundation for understanding (and appreciating) how these more powerful approaches operate.

To the best of our knowledge, this is the first experimental study to address collectively

the following questions related to the local search behavior of a phylogenetic heuristic.

1. How are MP scores distributed in a neighborhood during the progression of a search?
2. How different are the tree topologies during the progression of a search?
3. Is the performance of a local search impacted if some of the neighbors are selected randomly?

Local search heuristics move through tree space by selecting a single solution from a set of neighboring trees. Since most local search heuristics operate in a greedy fashion, each new tree selected on the path to the local optimum is better than the previous tree. However, what happens to the neighborhoods of these selected trees? Are they improving as well? Moreover, what is happening to the neighboring trees in terms of their topological distances between each other? Finally, how critical is it for a local search that a neighbor be greedily selected. If a worse scoring neighbor is selected, would that be detrimental to the performance of the phylogenetic search? Generally, selecting a “good” neighbor is a time consuming process in a phylogenetic search—especially as larger neighborhoods must be explored for larger datasets. If a search isn’t sensitive to how a neighbor is selected, then ideally search time can be saved, which translates into being able to analyze much larger datasets. It would also provide evidence why search strategies such as parsimony ratchet [12], which takes backwards moves by reweighting characters, has been a highly successful search strategy.

Our contributions

Our study is based on local search heuristics for maximum parsimony on three biological datasets of 44, 60, and 174 taxa. Several interesting and striking facts are uncovered from our study. Our first observation is that as a search climbs the hill to the local optima, the trees in the neighborhood surrounding the current solution improve as well. In fact, there is minimal overlap between the trees that are explored during a search. Hence, one can think

of a neighborhood as a population of solutions surrounding the current solution and the population improves as the search moves forward. Similar observations are found concerning the topological distances between trees. In other words, trees that are further apart (closer together) in score are further (closer) topologically as measured by their Robinson-Foulds distance. Since the neighborhoods of trees is improving along with the current solution, a small number of random neighbor selections does not impact significantly the performance of a search. In fact, with this strategy, our SLS heuristic established a new best-score on our 60 taxa biological dataset. Lastly, our analysis of search behavior is based on analyzing thousands of trees. Thus, our work shows how traditional visualization techniques can be used to gain essential insights from such a large collection of trees, in which very few of them are the most parsimonious trees found.

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 [9] 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 defines 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.

Stockham, Wang, and Warnow [15] present an alternative approach by using clustering

algorithms on the set of candidate trees. They propose 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 [8] explore 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 Simple Local Search (SLS)

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) [14] or by generating a starting tree randomly.

3.1 Tree rearrangement operators

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.

3.1.1 NNI

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). Afterwards, 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 [1]. Local searches based strictly on NNI operations perform poorly in comparison to their SPR and TBR counterparts.

3.1.2 SPR

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 $2(n - 3)(2n - 7)$ [1].

3.1.3 TBR

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). For TBR, there is no general formula for the exact number of neighbors since such a value depends on the shape of the underlying tree. However, it is estimated that there is at most $(2n - 3)(n - 3)^2$ TBR neighbors for a tree t [1].

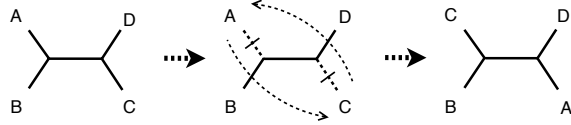


Figure 1: Nearest neighbor interchange (NNI)

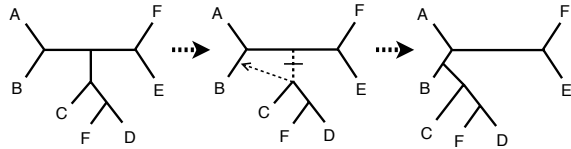


Figure 2: Subtree pruning and regrafting (SPR)

3.2 Neighbor selection

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. SLS could also operate extremely greedily by selecting the best tree from a neighborhood. Our experiments use SLS with a first-improvement strategy since it performs better than a best-improvement.

3.3 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 (see Figure 4). Formally, the sequence of trees encountered along the search path is defined as

$$P = (t_1, \dots, t_m). \quad (1)$$

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

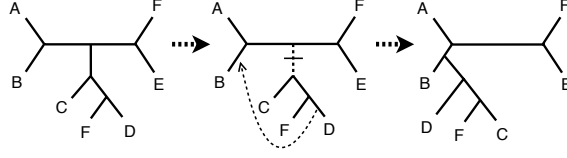


Figure 3: Tree Bisection and Reconnection (TBR)

the neighboring trees of t using rearrangement scheme β , where $\beta \in \{\text{NNI}, \text{SPR}, \text{TBR}\}$. For example, $\mathcal{N}_{\text{TBR}}(t)$ produces all of the TBR neighbors of tree t . To capture all of the β neighbors along a search path P , then

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

$\hat{\mathcal{N}}_\beta(P)$ 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.

For the search path P_i of run i , we are interested in the following trees,

$$\phi(P_i) = (t_{0\%}, t_{20\%}, t_{40\%}, \dots, t_{100\%}). \quad (3)$$

The ϕ operator selects the trees of interest along the search path P_i . That is, we are interested in the initial tree ($t_{0\%}$), the tree that represents the 20% completion point of the search ($t_{20\%}$), etc. If $|P_i| = 100$ trees, then $t_{20\%}$ would represent the 20th tree (t_{20}) of P_i . The final tree on the path represents the end of the search (100% search completion).

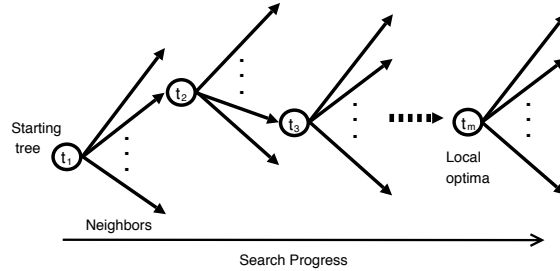


Figure 4: A depiction of the trees (t_1, t_2, \dots, t_m) visited by the SLS algorithm on its way to reaching a final tree (e.g., local optimum). t_1 represents the starting tree and t_m is final tree (local optimum) found. Here, each tree t_i along the search path is the neighbor selected from tree t_{i-1} 's neighborhood.

4 Experimental Methodology

4.1 Datasets

We used the following biological datasets as input to study the behavior of our SLS heuristic.

1. A 44 taxa dataset (17,028 sites) of placental mammals that includes 19 nuclear and 3 mitochondrial gene sequences for 42 placental and 2 marsupial outgroups [11]. In our experiments, both SLS and PAUP* established a best score of 43,085.
2. A 60 taxa dataset (2,000 sites) of ensign wasps composed of three genes (28S ribosomal RNA (rRNA), 16S rRNA, and cytochrome oxidase I (COI)) [4]. SLS established a best score of 8,698 on this dataset.
3. A 174 taxa dataset (1,867 sites) of insects and their close relatives for the nuclear small subunit ribosomal RNA (SSU rRNA) gene (18S). The sequences were manually aligned according to the secondary structure of the molecule [6]. For this dataset, SLS established a best MP score of 7,440.

4.2 Starting trees

All methods were provided with the same set of starting trees to begin their search for the most parsimonious trees. Both SLS and PAUP* can be provided with user trees. However, Phylip doesn't have this capability. So, we created the random sequence addition starting trees in Phylip. We modified Phylip so that it would output the starting tree that it generated. Afterwards, we fed those trees to PAUP and SLS.

4.3 Comparing tree topologies

In our experiments, we compare trees found by our SLS algorithm to the best-known trees for the data under consideration. 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

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

Our figures plot 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%.

4.4 Implementation and platform

Our SLS algorithm is implemented in C++. Our implementation took advantage of the libcov [3] phylogenetic software package to handle reading data matrices. However, we wrote our own branch-swapping routines as well as developed an algorithm for calculating the MP score more efficiently. We used the Hash-RF algorithm to compute the RF distances between trees [16]. Each heuristic was run five times on each of the biological datasets. All experiments were run on an Intel Pentium D platform with 3.0GHz dual-core processors and a total of 2GB of memory.

5 Results

5.1 Performance of our SLS implementation

We implemented the Simple Local Search (SLS) algorithm as a result of our interest in collecting a variety of data regarding the choices that SLS makes during the search. In order to test the effectiveness of our implementation, we compared its runtime performance to local search heuristics implemented in PAUP* [17] and Phylip [5] on the three biological datasets described in Section 4.1. Phylip is a freely available open-source packages that can be used to infer MP trees. PAUP* is a very popular package for phylogenetic analysis. It is commercially-available for a modest fee. Below, we show the main settings of the search parameters used in this study.

- **PAUP:** We ran a fast heuristic search in PAUP in which we save only one tree. The starting tree was provided to PAUP manually (it's a random sequence addition tree from Phylip) and PAUP was run with three different branch swapping algorithms. Hence, $\text{PAUP}(\beta)$ reflects a PAUP* local search using a β neighborhood, where $\beta \in \{\text{NNI}, \text{SPR}, \text{TBR}\}$. We use the PAUP*4.0b10 commands for the $\text{PAUP}(\text{TBR})$

heuristic.

```
set criterion=parsimony increase=no
    maxtrees=1; condense collapse=no;
hsearch start=current multrees=no
    swap=tbr;
```

The commands for PAUP(NNI) and PAUP(SCR) heuristics are defined similarly.

- **Phylip:** We use the following Phylip ver 3.65 commands.

```
Search for best trees? Yes
Search option? More thorough search
Number of trees to save? 1
Randomize input order of sequences? Yes
```

We varied the number of trees to save from 10^0 to 10^4 , but there was no impact on performance for the datasets used in this study.

- **SLS:** There are no search parameters that are given to our SLS implementation. The current tree t_i on the search path simply selects the first improving neighbor in its neighborhood. Similarly to PAUP, $SLS(\beta)$ reflects a SLS run using a β neighborhood, where $\beta \in \{NNI, SPR, TBR\}$.

Figure 5 shows that our SLS implementation performs comparably to PAUP* in terms of finding similar scoring MP trees. Our algorithm does require more time to find good-scoring trees. Given that our objective is understanding the behavior of search algorithms such as SLS, the actual runtime of a local search heuristic is not of primary concern here. We also note that both PAUP* and SLS established the best score for Dataset #1. SLS established

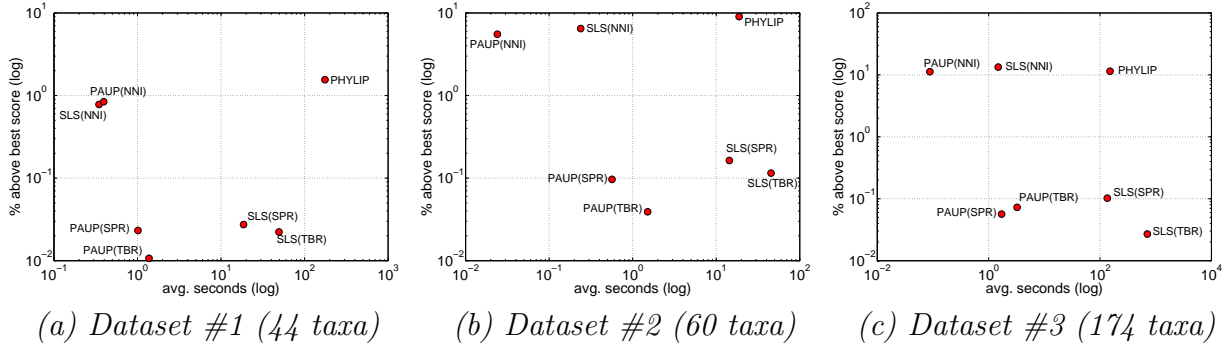


Figure 5: Hill-climbing performance of SLS, PAUP*, and Phylip on our three biological datasets. The same set of random addition sequence starting trees was used by each heuristic. The best scores found for each of the datasets (from smallest to largest) is 43085, 8698, and 7440. Each data point represents the average of five runs.

the best scores for the remaining two datasets. We note that Phylip is not very competitive when compared to either PAUP* or SLS.

Finally, we note that analyses based on our biological datasets have been published, we don't use such trees as the basis of our best-scoring trees as the analysis done are not quite comparable. For example on Dataset #1, the published tree is inferred using a Bayesian analysis. The MP score of this published tree is 43,128, which is 43 steps higher than the best-scoring trees found by SLS and PAUP*.

5.2 Properties of TBR neighborhoods

Given that TBR is the most popular neighborhood used in a phylogenetic search, we exclusively consider the *behavior* of the SLS (TBR) heuristic in the remainder of the paper. However, the trends found from analyzing SLS(TBR) apply to SLS(NNI) and SLS(SPR) as well.

Figure 6 shows all of the MP scores in $N_{TBR}(t_p\%)$, which is the set of trees in the TBR neighborhood of tree $t_p\%$. Here, $t_p\%$ represents the tree that represents the $p\%$ completion of the search. For example, $t_0\%$ is the initial tree and $t_{20}\%$ is the tree that represents the 20%

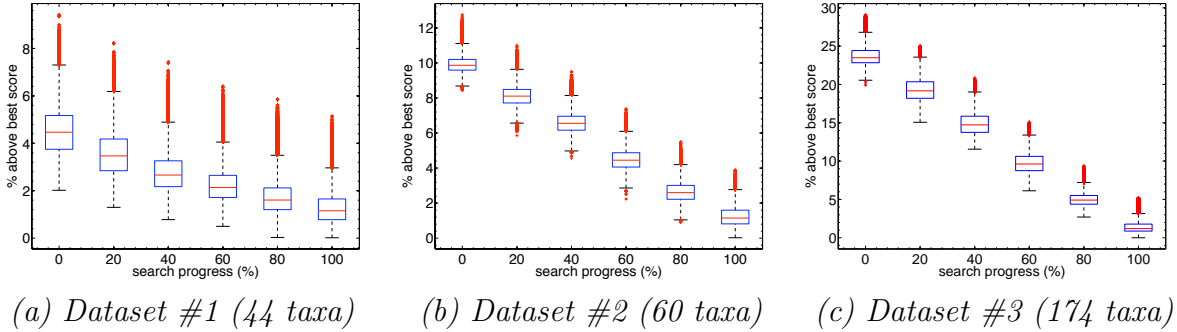
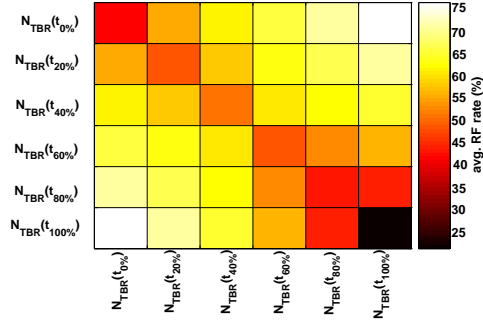
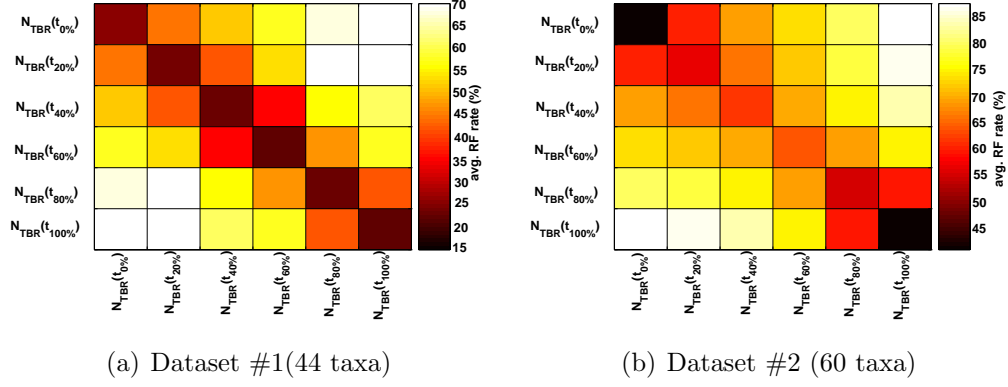


Figure 6: The distribution of the MP scores in a TBR neighborhood as the search progresses toward a local optima. For each interval (0%, 20%, ..., 100%), all tree scores from the neighborhood of the current tree is shown for all five runs.

completion point of the search (see Equation 3). For Dataset #1, the average number of MP scores depicted in each boxplot is 92,281.3. There are 233,120.8 and 250,000 MP scores represented by each box plot in Datasets #2 and #3, respectively. The actual number of neighbors for Dataset #3 is 3,716,369.7, but each box plot represent a sampling of 250,000 trees for this dataset.

The plots clearly show that the entire distribution of of MP scores in a TBR neighborhood improve as the search progresses toward the local optimum. Once the search reaches 100% search progress (i.e., a local optimum is reached), the plots show that the TBR neighborhood in terms of MP scores has improved dramatically over the TBR neighborhood of the starting trees. Thus, although the SLS heuristic is designed to improve each tree t_i selected on the search path P , Figure 6 clearly shows that the entire neighborhood is improving as well. So, not only is the best getting better, but the worse scoring trees in a neighborhood are getting better as well.

Figure 7 shows the topological differences between the trees in different TBR neighborhoods. Here, we use a heat map representation, where each value in the two-dimensional matrix is represented as a color. Darker (lighter) colors represented smaller (higher) RF rates, which is described in Section 4.3 . To make each heat map, we compare a sampling



(c) Dataset #3 (174 taxa)

Figure 7: RF distances between the neighborhoods visited during a SLS search on our datasets. Let $\mathcal{N}_\beta(t_i\%)$ represents the set of trees in the TBR neighborhood of tree $t_i\%$. Here, each entry (i,j) in the heat map represents the average RF rate between $\mathcal{N}_\beta(t_i\%)$ and $\mathcal{N}_\beta(t_j\%)$, where $i, j \in \{0, 20, \dots, 100\}$. Each headmanentry is the average of an all-to-all comparison of 10,000 trees sampled from the neighborhoods of interest.

of 10,000 trees from $\mathcal{N}_\beta(t_i\%)$ and $\mathcal{N}_\beta(t_j\%)$, where $i, j \in \{0, 20, \dots, 100\}$, compute the RF distance between every pair of trees in the sample, and take the average. According to the plots, trees from the same neighborhood are closer topologically than trees in different neighborhoods. For Dataset #1, trees from the same neighborhood are less than 25% different. Furthermore, as the search improves, the trees from the neighborhoods of the early portions of the search are quite different from trees in the later neighborhoods. The largest RF distances between trees occurs between the early neighborhoods ($\mathcal{N}_\beta(t_0\%)$ and $\mathcal{N}_\beta(t_{20\%})$) and the later ones ($\mathcal{N}_\beta(t_{80\%})$ and $\mathcal{N}_\beta(t_{100\%})$).

Figure 8 provides the topological distances between the neighboring trees along the search

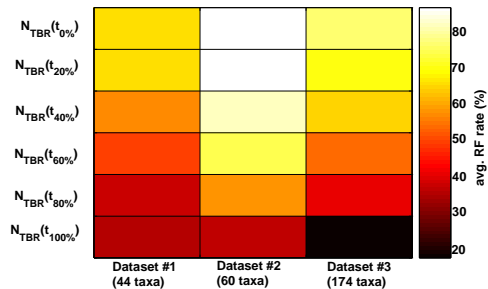


Figure 8: RF distances between the neighborhood trees and the best tree found for each dataset. Let $\mathcal{N}_\beta(t_i\%)$ represents the set of trees in the TBR neighborhood of tree $t_i\%$. Here, each entry (i,j) in the heat map represents the average RF rate between $\mathcal{N}_\beta(t_i\%)$, where $i \in \{0, 20, \dots, 100\}$ and the best known tree for each dataset. Each heat map entry is the average of an all-to-all comparison of 10,000 trees sampled from the neighborhoods of interest.

path P , and the best-known tree for a dataset. The plot clearly demonstrates that as the search progress by improving upon the current scores found, the neighboring trees also get topologically closer to the best-known trees. Trees that are in the neighborhood of the local optima have the smallest RF distances to the best-known trees. Hence, our set of heat maps clearly show that there is a strong correlation between MP scores and their RF distance from the best trees.

5.3 Random neighbor selection

The previous figures demonstrated that as SLS improves upon the tree t_i , its neighbors improve as well. Significant time during a phylogenetic search is spent generating and scoring neighbors in order to make a good decision regarding selecting the “best” neighbor. We were curious as to how sensitive the search is regarding selecting a neighbor. If the search isn’t that sensitive in terms of its overall performance, then time can be saved, which translates into being able to perform larger phylogenetic analyses.

Figure 9 shows the performance of the SLS algorithm when a random neighbor is selected

$r\%$ of the time. Here, $r = 0\%$ represents our standard first improvement algorithm, each tree on the search path is based on the first neighbor that improves upon the current score. For $r \geq 1$, there is an $r\%$ chance that the next tree (t_{i+1}) on the search path is selected randomly from $\mathcal{N}_\beta(t_i)$. (In case a local optimum is not reached, our r experiments used a search path limit of 1,000 trees so that the search would terminate. However, all of our experiments terminated on a local optimum.

In Figure 9(a), the SLS runs with $1 \leq r \leq 5\%$, result in median values that are similar to SLS runs with no randomly selected neighbors ($r = 0\%$). As r approaches 10%, the search cannot recover as the scores it finds are much further away from the best score. Similar trends occur in Datasets #2 and #3. Figure 10 provides a closer look at the random neighbor selection experiments for $0 \leq r \leq 5$. Our experiments show that r constrained to this range allows the search to make significant progress toward the best-scoring trees through tree space. We note that the best score for Dataset #2 was established by SLS with $r = 5\%$.

Finally, Figure 11 takes a look at the increased (or decreased) time that is required by our SLS heuristic when random neighbors are selected. When $r \leq 5$, random neighbor selection has a negative impact on performance in terms of running time. More specifically, the search needs more time to recover from the random selection. At around $r = 6\%$, the search time is significantly decreased for all datasets. Since random selection is a very inexpensive operation, the search completes very quickly. For example, our largest dataset, $r = 3$ requires approximately 2.5 hours. However, $r = 10$, results in a search that finishes in 17 minutes.

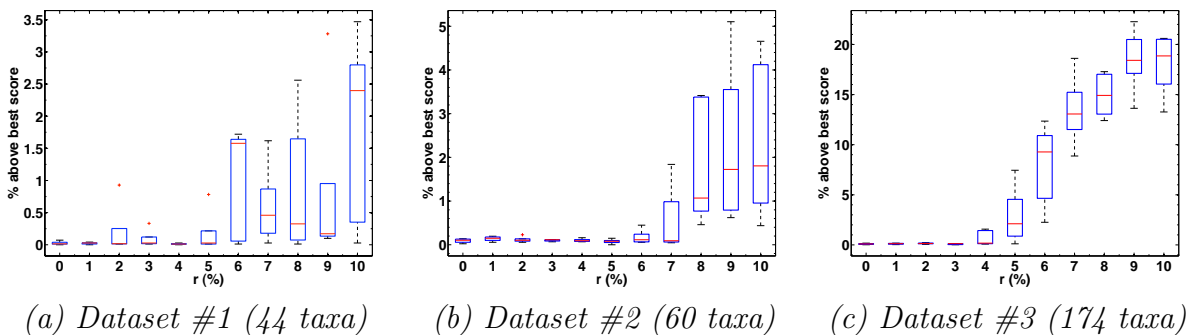


Figure 9: The performance of the SLS algorithm when random neighbors are selected. Our original SLS algorithm ($r = 0\%$), always chooses the first improving neighbor for its next move on the search path. However, for $r \geq 1$, there is an $r\%$ chance that the next tree (t_{i+1}) on the search path is selected randomly from the TBR neighborhood of t_i (i.e., $\mathcal{N}_\beta(t_i)$). Each box plot represents the distribution of five runs of the SLS heuristic for each r value.

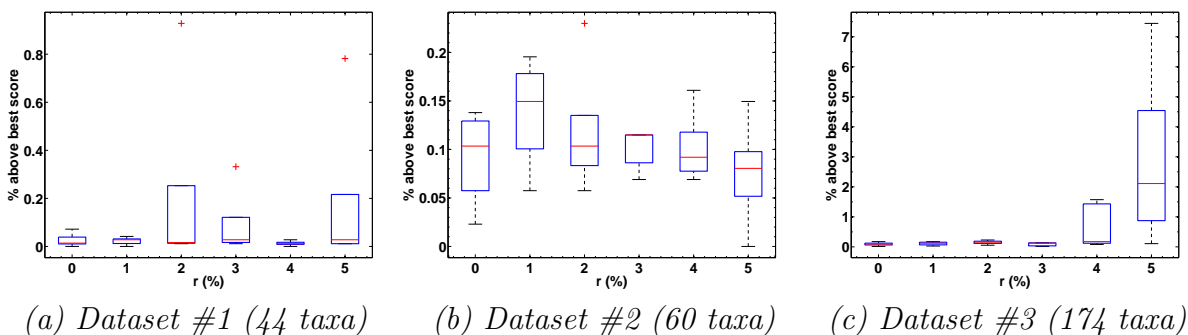


Figure 10: A closer look at the performance of the random neighbor selection experiments from Figure 9. Here, each plot show the performance of our SLS algorithm with r varying between 0% and 5%.

6 Conclusions

Since it is impossible to know the true evolutionary history for a set of organisms, the problem of inferring phylogenies is often reformulated into the NP-hard maximum parsimony problem. As a result, phylogenetic heuristics are used to find good-scoring trees since it is believed that better scoring trees are better approximations of the true evolutionary relationships. Although phylogenetic search is an important component of phylogenetics, few studies attempt to quantify the *behavior* of a local search as it makes its way through the exponentially-sized tree space.

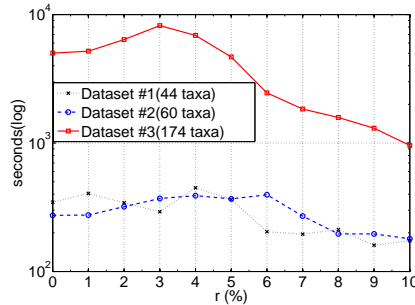


Figure 11: The running time required for the SLS heuristic with different values of r . Each data point is the average of five runs.

In order to understand the behavior of local search heuristics, we implemented SLS for solving the maximum parsimony problem on moderately-sized datasets. We show that our SLS algorithm performs comparably to PAUP*, a commercially available software package for phylogenetic inference. Thus, we lose minimal (if any) accuracy in using our algorithm for analyzing the behavior of local search heuristics. Furthermore, by analyzing thousands of trees with a diverse range of MP scores, our experiments with SLS show that there is a strong correlation between MP scores and topological distance. For example, as better scoring trees have a lower RF distance to the best-known tree. Of course, since our local search heuristic is greedy, parsimony scores improve as the search progresses toward a local optima. More enlightening, however, is that neighborhood trees surrounding the current best tree improve as well. In fact, the search is quite robust to a small percentage of random neighbor selections, which provides evidence why search strategies such as parsimony ratchet [12], which takes backwards moves by reweighting the characters in the dataset.

By analyzing the behavior of local searches, better phylogenetic heuristics can be designed. For example, by knowing that there are several good, but competing solutions within a neighborhood, a variety of different neighbor selection strategies (such as simulated annealing) are worthy of further investigation—especially in the context of investigating their behavior based on the analysis techniques presented here.

In the future, we plan to improve the performance (in terms of running time) of our SLS algorithm and make it publicly available to the systematic community. Furthermore, we plan to apply our approach to more powerful metaheuristics such as parsimony ratchet [12] and Rec-I-DCM3 [13], which will allow us to analyze much larger datasets.

7 Acknowledgements

The authors wish to thank Bill Murphy and Matt Yoder for providing us with the biological datasets used in this study.

References

- [1] B. Allen and M. Steel. Subtree transfer operations and their induced metrics on evolutionary trees. *Ann Comb.*, 5:1–13, 2001.
- [2] D. Bader, B. M. Moret, and L. Vawter. Industrial applications of high-performance computing for phylogeny reconstruction. In H. Siegel, editor, *Proceedings of SPIE Commercial Applications for High-Performance Computing*, volume 4528, pages 159–168, Denver, CO, Aug. 2001. SPIE.
- [3] D. Butt, A. Roger, and C. Blouin. libcov: A C++ bioinformatic library to manipulate protein structures, sequence alignments and phylogeny. *BMC Bioinformatics*, 6(138), 2005.
- [4] A. R. Deans, J. J. Gillespie, and M. J. Yoder. An evaluation of ensign wasp classification (Hymenoptera: Evanilda) based on molecular data and insights from ribosomal rna secondary structure. *Syst. Ento.*, 31:517–528, 2006.

- [5] J. Felsenstein. Phylogenetic inference package (PHYLIP), version 3.2. *Cladistics*, 5:164–166, 1989.
- [6] J. Gillespie, C. McKenna, M. Yoder, R. Gutell, J. Johnston, J. Kathirithamby, and A. Cognato. Assessing the odd secondary structural properties of nuclear small subunit ribosomal rna sequences (18s) of the twisted-wing parasites (Insecta: Strepsiptera). *Insect Mol. Biol.*, 15:625–643, 2005.
- [7] P. Goloboff. Analyzing large data sets in reasonable times: solutions for composite optima. *Cladistics*, 15:415–428, 1999.
- [8] D. M. Hillis, T. A. Heath, and K. S. John. Analysis and visualization of tree space. *Syst. Biol.*, 54(3):471–482, 2005.
- [9] D. Maddison. The discovery and importance of multiple islands of most parsimonious trees. *Syst. Bio.*, 42(2):200–210, 1991.
- [10] M. L. Metzker, D. P. Mindell, X.-M. Liu, R. G. Ptak, R. A. Gibbs, and D. M. Hillis. Molecular evidence of HIV-1 transmission in a criminal case. *PNAS*, 99(2):14292–14297, 2002.
- [11] 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.
- [12] K. C. Nixon. The parsimony ratchet, a new method for rapid parsimony analysis. *Cladistics*, 15:407–414, 1999.

- [13] U. Roshan, B. M. E. Moret, T. L. Williams, and T. Warnow. Rec-I-DCM3: a fast algorithmic techniques for reconstructing large phylogenetic trees. In *Proc. IEEE Computer Society Bioinformatics Conference (CSB 2004)*, pages 98–109. IEEE Press, 2004.
- [14] N. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4(406–425), 1987.
- [15] C. Stockham, L. S. Wang, and T. Warnow. Statistically based postprocessing of phylogenetic analysis by clustering. In *Proceedings of 10th Int’l Conf. on Intelligent Systems for Molecular Biology (ISMB’02)*, pages 285–293, 2002.
- [16] S.-J. Sul and T. L. Williams. An experimental analysis of robinson-foulds distance matrix algorithms. In *European Symposium of Algorithms (ESA’08)*, volume 5193 of *Lecture Notes in Computer Science*, pages 793–804. Springer-Verlag, 2008.
- [17] D. L. Swofford. PAUP*: Phylogenetic analysis using parsimony (and other methods), 2002. Sinauer Associates, Underland, Massachusetts, Version 4.0.
- [18] T. L. Yates, J. Salazar-Bravo, and J. W. Dragoo. *Assembling the Tree of Life*, chapter The Importance of the Tree of Life to Society. Oxford University Press, 2004.