LARGE-SCALE ANALYSIS OF PHYLOGENETIC SEARCH BEHAVIOR

A Thesis

by

HYUN JUNG PARK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2007

Major Subject: Computer Science

LARGE-SCALE ANALYSIS OF PHYLOGENETIC SEARCH BEHAVIOR

A Thesis

by

HYUN JUNG PARK

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2007

Major Subject: Computer Science

ABSTRACT

Large-Scale Analysis of Phylogenetic Search Behavior. (August 2007)

Hyun Jung Park, B.S., Yonsei University, Korea

Chair of Advisory Committee: Dr.Tiffani L. Williams

Phylogenetic analysis is used in all branches of biology by inferring evolutionary trees. Applications include designing more effective drugs, tracing the transmission of deadly viruses, and guiding conservation and biodiversity efforts. 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. This is important, because a better understanding of local search behavior can facilitate the design of better heuristics, which ultimately leads to more accurate depictions of the true evolutionary relationships.

In order to access and analyze the tree search space, we implement an effective local search heuristic. Having an effective heuristic that can open the space is important, since no search heuristic in this field can effectively provide data collection control. So we have implemented and estimated a search heuristic, *Simple Local Search* or SLS, that works reasonably well in the space.

Our investigations led to several interesting observations about the behavior of a search heuristic and the tree search space. We studied the correlation of tree features of search path trees, where tree features refer to the parsimony score, the Robinson-Foulds distance and the homoplasy measure. Most importantly from the results, parsimony score was highly correlated with Robinson-Foulds distance only in trees that lie on the search path to a local optimum. We also note that the scores of neighborhoods along search paths improve together, as a local search progresses.

Correlations of tree features of search path trees are particularly useful in characterizing and controlling a search path. This paper proposes one possible stopping criterion to maximize the tree search results while minimizing computational time tested on three biological datasets using the correlation between the parsimony score and the RF distance value of search path trees. Also, the observation that scores of a neighborhood on a search path improve together gives us a significant amount of flexibility in selecting the next pivot of a search without losing performance.

Eventually, our long-term goal is developing an effective search heuristic that can deal with large scale tree space in reasonable time. Improved knowledge about the tree search space and the search heuristic can provide a reasonable starting point toward the goal.

To my soul mate Soyeon and family for their love and encouragement.

ACKNOWLEDGMENTS

I would like to express my thanks and gratitude to Dr. Tiffani L. Williams for her research guidance, patience and understanding and for sharing her knowledge and experience, as well as for serving as my advisor. Additionally, I would like to thank Dr. Jim Woolley and Dr. Sing-Hoi Sze for their knowledge and advice as committee members. I also wish to thank Dr. William J. Murphy and Dr. Matt Yoder for providing me with the biological datasets used in this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

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. The objective of phylogeny reconstruction is to produce a phylogenetic tree describing the evolutionary relationships between the organisms. But the problem is that there are so many possible trees in the search space that we cannot exhaustively navigate in order to find out the true tree. In parsimony criteria, this difficulty is converted into finding out the most parsimonious tree, and it's called the Maximum Parsimony problem. In order to address the problem, phylogenetic inference relies on effective heuristics for obtaining good-scoring trees. For now, a number of good heuristic have been suggested, but it is not enough, since any heuristic doesn't come from deep understanding about the search space or search path. The knowledge about the search space is important, because, the search basically takes place within this search space. More specifically, the understanding about the search space allows us to design a better heuristic that navigates the search space effectively, taking a full advantage of it.

Now, what is the effective navigation in the space? An effective navigation is the one that exploits more promising trees during a search. This statement drives three practical questions, what is a promising tree, how can we identify this from others in the space; and how can we make use of the knowledge? Local search heuristics lowers the score of the current path tree until a local optimum is achieved. So, from

The journal model is *IEEE Transactions on Automatic Control.*

the standpoint of a local optimum, search path trees are promising, since a series of path trees guide the search to the optimum. Eventually the final questions left for this paper to address are:

1. What characteristics do search path trees have? What does that mean? and why are they important?
2. How can the knowledge about search path trees contribute to making effective navigation?

In order to study quantitatively the behavior of local search heuristic related to search path trees, we first have to gather as much information as possible from search path trees of a local search heuristic. However, we are not allowed to capture all the information we need from PAUP*, since it has been commercially treated. Also, present open-source phylogenetic software such as Phylip [9] show that it performs poorly in comparison to PAUP* and that a reasonable search path cannot be guaranteed. In this regard, this study starts with developing a new local search heuristic that reasonably works and provides us with the data collecting capability. To verify the performance of our local search heuristic (*Simple Local Search* or SLS), we want to compare with other major heuristics such as PAUP* [32], Phylip [9], and libcov [3]. However, it is not precise that heuristic is estimated only with their output and time, since internal components of heuristics are so interconnected and data dependent that it is hard to tell the consistent performance from a few instances. Basically, a local search heuristic is composed of two essential parts, branch swapping and tree scoring basic blocks. We will compare the performance of our algorithm to three approaches by each basic block. Results from these estimations clearly show that *Simple Local Search* outperforms other open-source search heuristics both in time and in score from basic block level up to overall perspective. Based on this observation, SLS is allowed to have further investigation in that it can build a good search path in the search space.

After verifying SLS, we are going to address the first question arising from the previous paragraph, 'What characteristics do search path trees have?' and 'why are they important?' Knowing the characteristics of search path trees is important, because this knowledge can initiate the design of better heuristics that will utilize search path trees more effectively. 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. By controls SLS provides, we can collect a variety of data regarding the choices that SLS makes during a search, and examine the behavior of a collection of various data. In this study, we collect the data related to the topology and homoplasy information of search path trees. Also, we gather the information about their neighbors, especially MP scores of neighbors. Related to this collected data, we extend the previous question into a number of ones listed below about the behavior of local search path trees.

1. How are MP scores distributed in a neighborhood of search path trees as a search proceeds?
2. What is the correlation between MP scores of search path trees and their topological accuracy?
3. How does the "fit" of the data increase or decrease as the search progresses toward a local optimum?

The last part of this paper attempts to address the question, 'How can the knowledge about search path trees affect the performance of a local search heuristic?' Above all, two characteristics of search path trees observed from the previous experiment should be mentioned. The first observation is that scores of neighbors are also improving together as the search proceeds. That means there are more than one candidate tree in the neighborhood for the progression. So, we are interested in the effect of choosing another candidate tree as the next pivot rather than improving with the first tree. This question is given below more in detail. We are also interested in the application of the correlation coefficient between topological accuracy and parsimony score that had been discovered also from previous section. This observation is very interesting in that it shows whether the goals of a phylogenetic heuristic (i.e., finding the optimal-scoring tree) correspond to the actual goal of phylogenetics, which is depicting accurate relationships between organisms (i.e., topological accuracy). However, we want to extend this notion of correlation for more practical use by raising the following questions:

1. How much effort should be consumed in picking the next neighbor in a heuristic?
2. Is the performance of a search impacted if some of the neighbors are selected randomly?
3. How does the correlation coefficient between MP scores and their topological accuracy transform during the progression of a search?

This paper says the importance of background knowledge about the tree space itself and search path trees in the space. We know there are various types of trees and each tree has various types of features, but we vaguely know how they are distributed and what it means. The distribution of r_{FH} states that fitness and homoplasy of all trees in the space are highly correlated. On the contrary, from the perspective of r_{FD} and r_{DH}, correlatedness is not always given, only search path trees present this correlation. The first benefit of this knowledge is to understand the behavior of search path trees and the tree space. Local search heuristic has a direction that all local search paths commonly imply. MP search takes the path enhancing topological accuracy as well as reducing parsimony score. We also note that tree space consists of trees that have parsimony scores highly correlated with homoplasy values. Another contribution is the more practical one related to the questions we have raised. Search heuristic can significantly reduce the effort for generating and estimating neighbors using the observation that many neighbors are good. This is significant, because actually it took a lot of time for search heuristic to select the next pivot. Also we note that search heuristic can also be controlled by r_{FD}, since this estimate is consistently preserved throughout the search. In most cases of large-scale heuristic search that are extremely time-consuming, sometimes experiments are halted without any reasonable inference for the progression that the search made, because there has been no stopping criteria. But r_{FD} provides one way to estimate the status of the search.

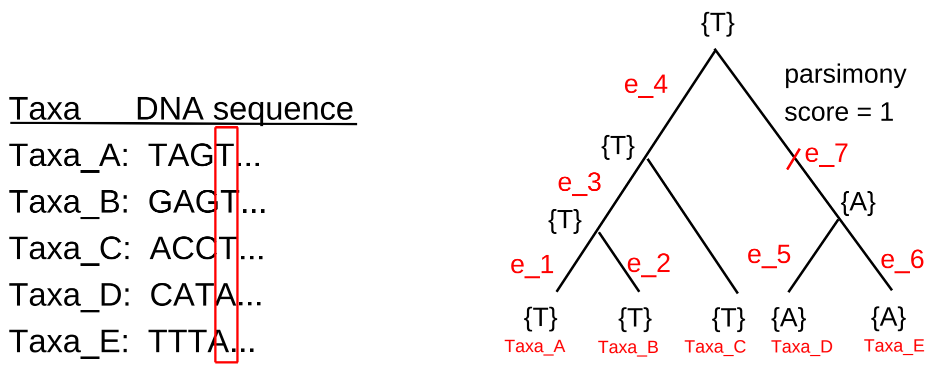
CHAPTER II

BACKGROUND

A. 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 $\sum_{e \in E(T)} H(e)$. From the given example in Fig. 1, $\sum_{e \in E(T)} H(e)$ is 1, because the edge (e_3) is the only place that has the evolutionary change ($H(e\_3) = 1$). 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 like ML(Maximum Likelihood), another major optimization problem used to reconstruct phylogeny reconstruction [11], [4], but the problem of assigning sequences to internal nodes of a fixed leaf-labelled tree is polynomial [10].



(a). DNA sequences *(b). An evolutionary tree*

Fig. 1. An evolutionary tree for five taxa (Taxa_A, Taxa_B, Taxa_C, Taxa_D, Taxa_E) is given, in which alphabet in {} represents DNA sequence assigned to taxa or internal nodes at the site marked red and *e_i* represents an edge ID that shows an evolutionary relationships between two nodes. Parsimony score at this site of this tree is 1, since one evolutionary change happens at edge (*e_*7). The actual parsimony score should be obtained by summing up all sites.

B. Branch Swapping Operations

To find the tree T with the minimum length, an MP search navigates the exponentially-sized tree space by moving from one point in tree space to another solution point. Here, each new solution point is created by rearranging the branches of a tree in some way. Below, we describe the three most popular branch-swapping operations with example moves in Fig. 2, even though some recent methods have used ECR coupled with TBR and see significant improvements both in speed and accuracy [13].

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. For a tree

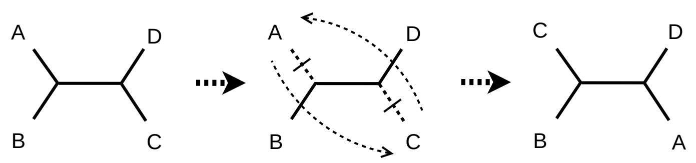
T with n taxa, $2(n-3)$ neighbors can be examined for each tree [1]. Local searches based strictly on NNI operations perform poorly in comparison to their SPR and TBR counterparts.

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. 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 $2(n-3)(2n-7)$.

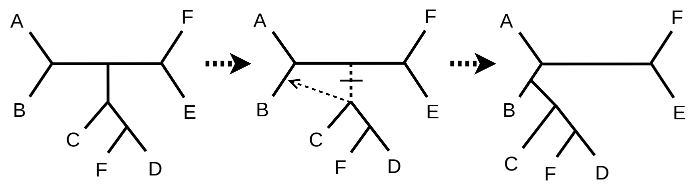
In a *tree-bisection and reconnection (TBR)* move, an interior branch is broken, and the two resulting fragments of the tree are considered as separate trees. All possible connections are made between a branch of one and a branch of the other. If there are n_1 and n_2 species in the subtrees, there will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

C. Robinson-Foulds Distance

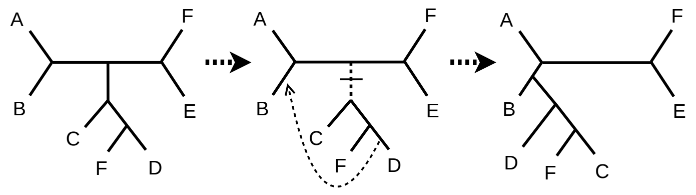
In our experiments, we compare trees found by our SLS algorithm to the best-known trees for the data under consideration or the best tree up to the point of a search progress. We use the Robinson-Foulds (RF) distance to measure the topological distance between trees. The RF distance between two trees 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



(a) Nearest Neighbor Interchange (NNI)



(b) Subtree pruning and regrafting (SPR)



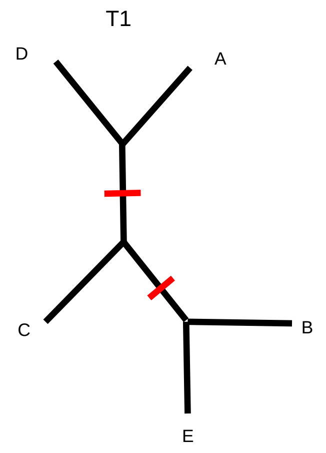
(c) Tree bisection and Reconnection (TBR)

Fig. 2. Examples of rearrangements. Alphabet nodes could represent terminal nodes or subtrees.

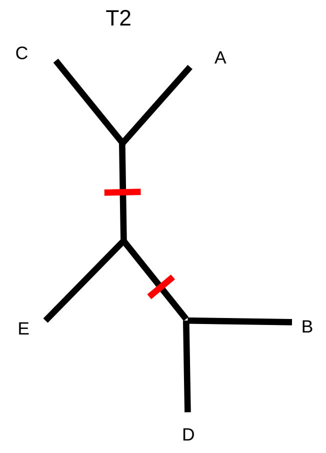
tree T. The RF distance between trees T_1 and T_2 is defined as

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

From the example in Fig. 3, the set $\Sigma(T_1)$ has $\{A, D \mid C, E, B\}$ and $\{A, D, C \mid E, B\}$, and $\Sigma(T_2)$ has $\{A, C \mid E, D, B\}$ and $\{A, C, E \mid D, B\}$ as their components. So, both $|\Sigma(T_1) - \Sigma(T_2)|$ and $|\Sigma(T_2) - \Sigma(T_1)|$ are 2, since they don't share any bipartition. So in this case, $d_{RF}(T_1, T_2)$ would be 2. In our experiment, we will 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%.



(a). $\Sigma(T_1) = \{\{A, D \mid C, E, B\}, \{A, D, C \mid E, B\}\}$



(b). $\Sigma(T_2) = \{\{A, C \mid E, D, B\}, \{A, C, E \mid D, B\}\}$

Fig. 3. An example of RF distance calculation between T_1 and T_2. Each tree has the same set of taxa (A,B,C,D,E) with a different evolutionary relationship. They don't share any bipartition, and this makes both $|\Sigma(T_1) - \Sigma(T_2)|$ and $|\Sigma(T_2) - \Sigma(T_1)|$ as 2. So, $d_{RF}(T_1, T_2) = \frac{|\Sigma(T_1)-\Sigma(T_2)|+|\Sigma(T_2)-\Sigma(T_1)|}{2} = \frac{|2|+|2|}{2} = 2$.

CHAPTER III

RELATED WORK

There have been similar studies identifying or classifying phylogenetic trees. Hendy et al. [16] discussed two methods of defining classes of trees. They formally defined a family of trees as "all trees within a fixed distance of a fixed tree T", where the distance between trees is measured by some tree-comparison metric. Hendy et al. also used complete-linkage cluster analysis, based upon the partition metric, to define clusters of trees. Maddison [22] 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.

Also, Stockham, Wang, and Warnow [30] 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 [18] 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.

Several researchers have explored the question of analyzing a collection of trees found by a phylogenetic search. But, we note that the research presented in this paper differs in four fundamental ways: (i) we are handling extremely large collections of trees, (ii) we limit our concern only to search path trees, not every trees in the space, (iii) but at the same time, we do not limit our search to the best-scoring trees, we look at all trees on the path; and (iv) the motivation for our work is understanding search behavior as a first step to design a better heuristics.

CHAPTER IV

EXPERIMENTAL METHODOLOGY

A. Biological Datasets

We used the following biological datasets as input to all our experiments.

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 [24]. 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)) [5]. PAUP* established a best score of 8,701 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 [14]. For this dataset, SLS established a best MP score of 7,440.

B. Starting Trees

A heuristic creates a random sequence addition (RSA) to create the initial starting tree for the search. 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) [28] or by generating a starting tree randomly. In our experiment, all methods were provided with the same set of starting trees to compare their search fairly. 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*, libcov, and SLS heuristics.

C. 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 [31]. All experiments were run on an Intel Pentium D platform with 3.0GHz dual-core processors and a total of 2GB of memory.

CHAPTER V

HEURISTIC MEASUREMENT

A. Introduction

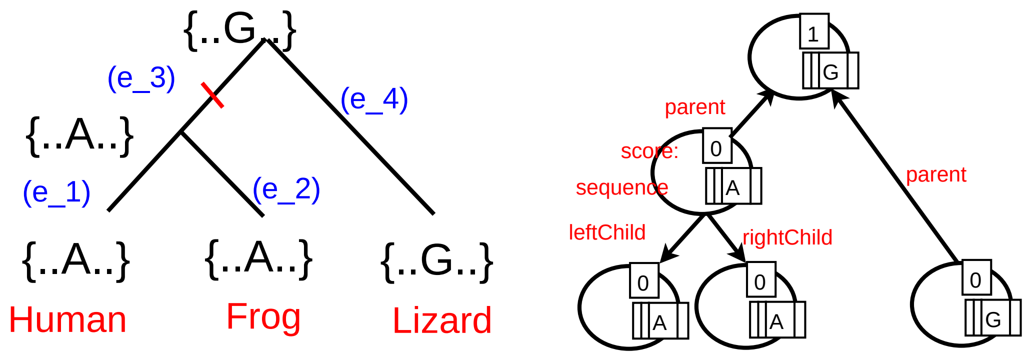
Here we implement and verify our local search heuristic, SLS for further study. First, we describe which data structure SLS uses in the implementation and how it works. And then, in order to verify the heuristic precisely, we discuss performance in local search heuristic and how it can be estimated more quantitatively. Also, we can take another viewpoint about the performance of search heuristic topologically. A topological investigation identifies how the result trees of heuristics are related between each other, since topological distance is calculated in pairs. By this, we want to make sure that SLS reasonably performs both in score and in time, and results of SLS are close to PAUP*. In addition, we also hope that these quantitative evaluation brings us the conceptual guideline for selecting the appropriate methodology when we have to have a search.

1. Simple Local Search

Simple Local Search is simple, but has a couple of effective techniques for improving scoring and traversing. Those are introduced here and investigated in the Results section piece by piece.

a. Data Structure

In SLS, data structure of a tree is composed of one root and a list of nodes connected by their topology relationship with pointers such that the parent node points to two descendents, which we call *leftChild* and *rightChild*. This is shown briefly at Fig. 4 (b)



(a) a phylogenetic tree *(b) a data representation in SLS*

Fig. 4. A conceptual phylogenetic tree and the data structure representation the tree in SLS.

as a data structure for a phylogenetic tree of Fig. 4 (a). Also, two descendent nodes point to their parent node by the pointer called parent. When being initialized, a tree has only root in the list of nodes, and nodes are added up according to their topological relationship in the tree. Leaf nodes in the list have DNA sequence assigned to an organism, and interior nodes have inferred sequences suggested from its descendents. The tree has a parsimony score derived from all organisms the tree has.

b. Accelerating Score Calculation

To improve the performance of SLS, we have employed two speedup techniques suggested by Ronquist [26].

The first technique is referred to as *making shortcut*. Normally, parsimony score is calculated while going up from each terminal node to the root node of the tree. However, while navigating rearrangement neighbors such as NNI, SPR or TBR neighbors, all the nodes don't always have to be updated. Usually, in the case of rooted implementation, the actual difference of the current tree, T_{new}, in score from the previous tree, T_{src}, starts from two nodes. The one is N_{one}, a node that has been taken out and be attached to the new place as a result of a rearrangement, and the other is $N_{sibling}$, a node that was a sibling node of N_{one} before clipping. In Fig. 5 (a), two nodes that start the score update process are marked as red triangles. So, when we try to attain the parsimony score of a tree after a rearrangement, the score should be updated only from N_{one} and $N_{sibling}$ up to the root of the tree. These paths from N_{one} and $N_{sibling}$ up to the root is called the shortcut. [26]

The second technique is *bit-wise calculation*. Parsimony score is defined as the sum of the total number of changes of states between parents and their children across all sites. Straightforwardly, this can be implemented using a sequence of set operations such as $\cap$ and $\cup$. However, from the implementation perspective, those operations are easily converted into similar operations in bit-wise environment such as the bit-wise operation & or | with minor manipulation. This conversion will save significant amount of calculation time, in that a bit-wise operation can handle large units of data in each clock cycle [26]. A scoring algorithm applying two techniques described above is given in the Algorithm 1 with comments on the line where speedups are used.

c. Removing Duplicated Rearrangements

In the middle of a search, rearrangement can be defined by two nodes in a tree. Basically, a rearrangement derived from two nodes should be unique under the condition that the tree is unrooted, and that there are only terminal nodes. However, from the implementation standpoint, an unrooted tree is difficult to maintain. So, SLS assumes that trees are rooted, and tries to match results of SLS these with unrooted cases when they are analyzed. Under this situation, it is possible that some rearrangements in SLS are duplicated, which substantially impairs the performance of search heuristic. SLS avoids duplicated rearrangements by filtering out moves that will cause duplication. A rooted representation of the TBR rearrangement of the tree of Fig. 2 (c) is given in Fig. 5 (a), and one of the moves that make the same topology is given in Fig. 5 (b).

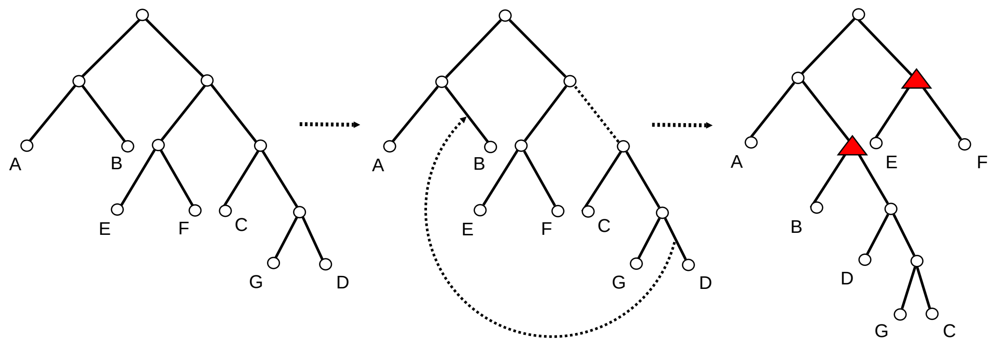
2. Measuring Performance

Performance studies evaluating the performance of MP heuristics have generally centered on two issues: speed and topological accuracy. Studies that explore speed have examined how quickly each heuristic can solve MP (or reach the current best known score) for specific real biological datasets (see [15, 27] for examples of such studies). Generally, an MP heuristic consists of two main components, a scoring mechanism and an algorithm for traversing neighbors using branch swaps, but with PAUP*, it is difficult to separate the two components. So, the two blocks will be taken together and evaluated. The scoring basic block of SLS employs two speed-up techniques; bit-wise calculation and making a shortcut for score updating. This experiment will present the impact of each technique on the overall performance of the heuristic. Also, neighbor traversing block of SLS detects and removes duplicated cases of re-

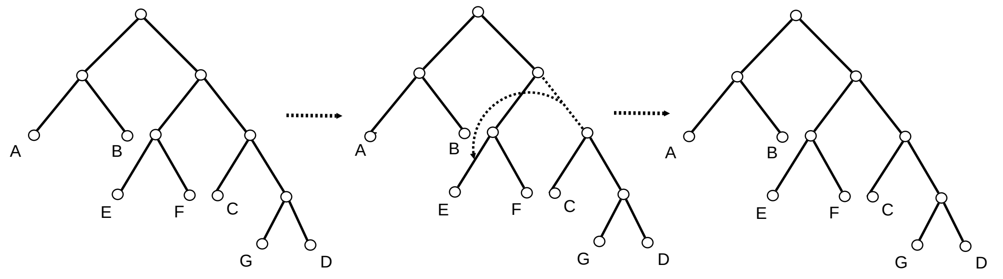
Algorithm 1 parsimony_score_with_speedups(T_{new}, T_{src}, N_{child}, $N_{sibling}$)

```
Require: $T_{new}$, the tree acquired from $T_{src}$ by one rearrangement started from two node $N_{child}$, $N_{sibling}$.
  {score update only on the shortcut from $N_{child}$ up to the root}
  $N_{score} = N_{child}$.
  $N_{left} = N_{score} \rightarrow leftChild$
  $N_{right} = N_{score} \rightarrow rightChild$
  $N_{score} \rightarrow parsimony\_score = N_{left} \rightarrow parsimony\_score$
  $N_{score} \rightarrow parsimony\_score += N_{right} \rightarrow parsimony\_score$
  while $N_{score}$ != $T_{new} \rightarrow root$ do
    $seq\_left = N_{left} \rightarrow sequence$
    $seq\_right = N_{right} \rightarrow sequence$
    $seq\_intersection = seq\_right \& seq\_left$ {bitwise calculation &}
    if $seq\_intersection$.count() == 0 then
      $N_{score} \rightarrow sequence = seq\_left \mid seq\_right$ {bitwise calculation |}
      $N_{score} \rightarrow parsimony\_score += 1$
    else
      $N_{score} \rightarrow sequence = seq\_left \& seq\_right$. {bitwise calculation &}
    end if$N_{score} = N_{score} \rightarrow parent$
  end while

  {score update only on the shortcut from $N_{sibling}$ up to the root}
  $N_{score} = N_{sibling}$
  $N_{left} = N_{score} \rightarrow leftChild$
  $N_{right} = N_{score} \rightarrow rightChild$
  $N_{score} \rightarrow parsimony\_score = N_{left} \rightarrow parsimony\_score$
  $N_{score} \rightarrow parsimony\_score += N_{right} \rightarrow parsimony\_score$
  while $N_{score}$ != $T_{new} \rightarrow root$ do
    $seq\_left = N_{left} \rightarrow sequence$
    $seq\_right = N_{right} \rightarrow sequence$
    $seq\_intersection = seq\_right \& seq\_left$ {bitwise calculation &}
    if $seq\_intersection$.count() == 0 then
      $N_{score} \rightarrow sequence = seq\_left \mid seq\_right$ {bitwise calculation |}
      $N_{score} \rightarrow parsimony\_score += 1$
    else
      $N_{score} \rightarrow sequence = seq\_left \& seq\_right$. {bitwise calculation &}
    end if$N_{score} = N_{score} \rightarrow parent$
  end while
```



(a) A rooted TBR



(b) A duplicated TBR

Fig. 5. A rooted representation of a TBR rearrangement, where score update goes up from two triangle nodes (a) and an example of TBR move that generates a duplicated case (b)

arrangements so that neighbors are effectively traversed. Putting those basic blocks together, local search heuristics try to reach the local optimum as soon as possible. So, our study will also show the output of various local search heuristic with the time consumed for the search and scores of the local optimum they have reached.

Assessing the topological accuracy of an inferred tree on a real dataset is difficult to estimate because the true tree cannot be known precisely. Alternatively, simulations are usually used for the estimations of topological accuracy. Simulation studies have been highly influential, and have suggested that good MP heuristics can produce reasonable estimates of trees, with acceptably low RF error rates (bounded by 10% or so) with respect to the true tree (see [17]). However our calculation requires neither the model nor simulated tree, since our calculation takes both resultant trees from different heuristics, and checks how far they are topologically apart from each other.

3. SLS Competitors

We compare the performance of SLS to three different local search heuristics—PAUP* [32], Phylip [9], and libcov [3]. Both Phylip and libcov are publicly 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, PAUP*(β) reflects PAUP* local search run with a β neighborhood, where $\beta \in \{\mathrm{NNI}, \mathrm{SPR}, \mathrm{TBR}\}$. We use the PAUP*4.0b10 commands for the PAUP(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(SPR) 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, 100, 10000] 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.

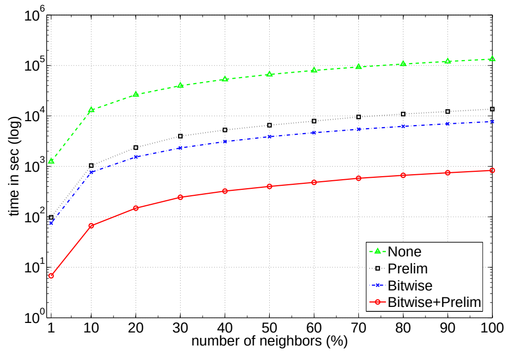
- **libcov:** Besides Phylip, the other open-source software package we used was libcov, which is a C++ library designed to manipulate protein structures, sequence alignments, and phylogenetic trees. For our purposes, we use the branch swapping modules and parsimony scoring routine to compose a local search heuristic, which follows the logic of our SLS implementation. As in SLS, there are no search parameters to set explicitly.

B. Results

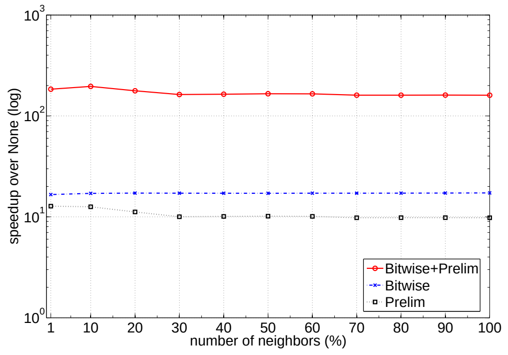
Basically, what a local search heuristic does is to retrieve a local optimum (or a good-scoring tree) in a certain amount of time. This can be easily considered to be the performance. But to be more precise, heuristic should be estimated by each basic block. Here we cannot divide scoring block and neighborhood traversing block in PAUP* execution, so performance will be monitored with two blocks together. The first part of experiment evaluates time spent for traversing neighbors and scoring them by different scoring techniques, and the second part by different heuristics. I also present the overall performance of heuristics in hill-climbing performance section. Topological approach for the performance evaluation presents heuristics' identifications in an interesting way.

1. The Impact of Scoring Techniques

MP local search heuristics operate by successively exploring the neighborhood of a current solution and moving to one of its neighbors based on their scores. So, scoring takes part in all decision-making processes, and this basic block is one of the most



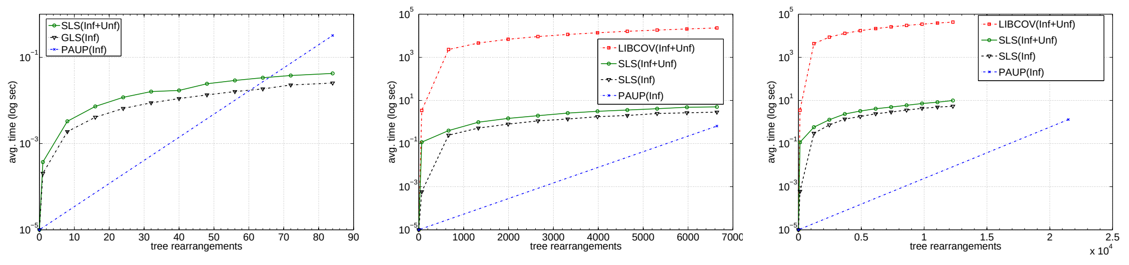
(a) performance



(b) speedup

Fig. 6. Comparison of performance between speedup techniques for scoring the same number of neighbors on Dataset #3 (174 taxa). Differences are plotted in log(time) scale, and speedup is measured over a straightforward algorithm that has no speedup technique.

essential basic blocks that decide the performance of local search heuristic. Here, two different kinds of scoring techniques are employed by SLS, shortcut and bit-wise [26]. Fig. 6 (a) shows the number of neighbors plotted on the x-axis and time taken for the process on the y-axis. Time is estimated for processing the same number of neighbors from the same starting tree of Dataset #3 by two different techniques separately and together. This figure shows the time in log scale. According to the Fig. 6, both bit-wise and shortcut techniques are much faster than a straightforward implementation. In particular, the bit-wise scoring technique is a little more effective than shortcut. However, it produces by far the best performance when they are combined. Fig. 6 (b) makes this point more clearly. The combined strategy increases the speed around thousand times, while each individual technique is around ten times better than the basic case.



(a) NNI neighbors *(b) SPR neighbors* *(c) TBR neighbors*

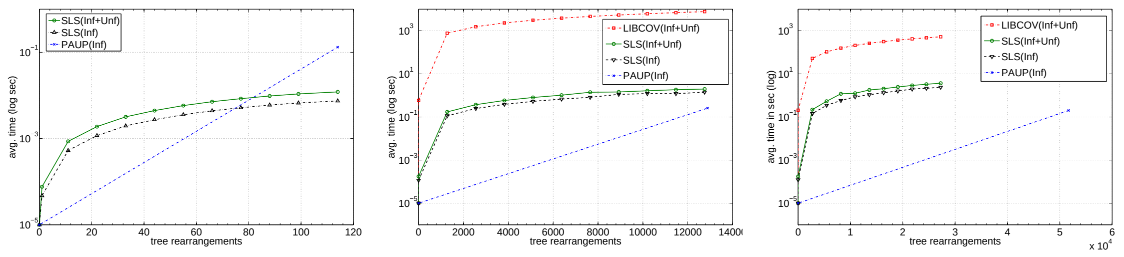
Fig. 7. Neighborhood performance on Dataset #1 (44 taxa). The number of informative (Inf) and uninformative sites (Unf) is 8,004 and 9,024, respectively. $Inf + Unf$ denotes the performance of a method based on scoring the entire sequence. Performance based on scoring only the informative sites is denoted Inf. We note that libcov is unable to generate an NNI neighborhood.

2. Scoring and Neighborhood Performance

This section attempts to state how much time each heuristic spend for scoring and traversing the same number of neighbors. Figs. 7, 8, 9 shows the time for retrieving all neighbors of a particular tree for Dataset #1, #2, #3, respectively. As in the previous section, x-axis represents the number of neighbors retrieved, and y-axis represents the time for retrieving that number of trees in log scale. Since the inside of PAUP* is hidden, PAUP*'s time is taken only at the starting and the ending point, while other heuristics are measured at intermediate points. In SPR, SLS retrieves the exact number of neighbors to PAUP* by taking more time. However, libcov takes much more time. This trend gets more clear in the TBR case in that libcov take about ten thousand times longer than SLS for the same number of neighbors.

3. Hill-climbing Performance

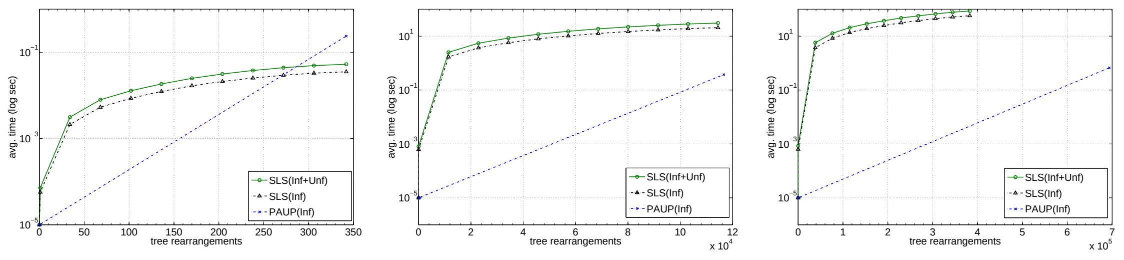
Simply put, an MP heuristic is a search heuristic looking for the tree that has a lower score. This "hill-climbing" process is controlled by a search strategy with two basic



(a) NNI neighbors *(b) SPR neighbors* *(c) TBR neighbors*

Fig. 8. Neighborhood performance on Dataset #2 (60 taxa). The number of informative (Inf) and uninformative sites (Unf) is 946 and 1,054, respectively. $Inf + Unf$ denotes the performance of a method based on scoring the entire sequence. Performance based on scoring only the informative sites is denoted Inf. We note that libcov is unable to generate an NNI neighborhood.

blocks mentioned above. SLS has the first-improvement and the best-improvement search strategy. The best-improvement strategy visits all neighbors and moves on to the best neighbors among them, while the first-improvement strategy moves to the first occurrence of a better tree than the current one. We present only the result of the first-improvement search strategy, because the first-improvement always outperforms the best-improvement. Fig. 10 (a) shows time taken by search on Dataset #1 on the x-axis and the final score from each MP heuristic on the y-axis, and the right figure presents the same result in terms of the number of rearrangements taken during search process. More specifically, NNI results from the heuristic SLS and PAUP* are separated from others, but they are closer to each other than others from other rearrangements. This is obvious in that the NNI search space is smaller than other search spaces. SPR and TBR results are mingled together. When considering the time criteria, PAUP* always outperforms SLS. However in terms of the number of rearrangements in Fig. 10 (b), PAUP* is not always better than SLS. This implies that if we have better implementation about our idea in SLS, we can improve the



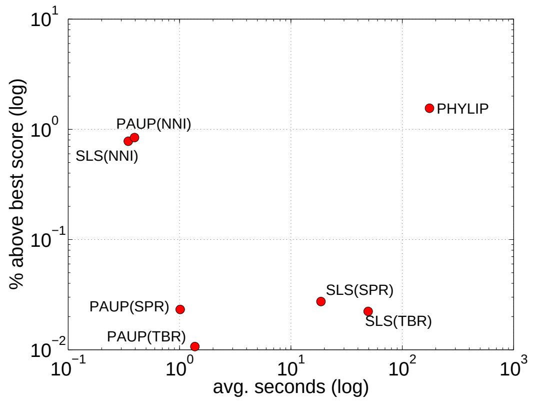
(a) NNI neighbors *(b) SPR neighbors* *(c) TBR neighbors*

Fig. 9. Neighborhood performance on Dataset #3 (174 taxa). The number of informative (Inf) and uninformative sites (Unf) is 912 and 955, respectively. $Inf + Unf$ denotes the performance of a method based on scoring the entire sequence. Performance based on scoring only the informative sites is denoted Inf. We note that libcov is unable to generate an NNI neighborhood.

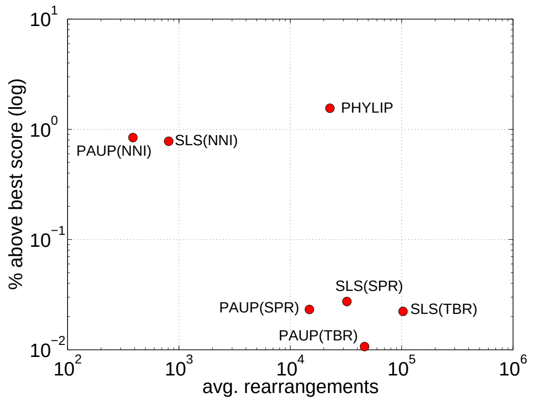
performance of SLS more closely to or even better than that of PAUP*. This trend is true for both Dataset #2 and #3, and we can check it in Figs. 11 and 12. The performance of Phylip is consistently lower not only in time but also in the number of rearrangements. Given that libcov and Phylip are among the few open-source implementations of MP heuristic, SLS is the only open-source implementation comparable to PAUP*, a widely used commercial program for MP.

4. Topological Performance

We use the Robinson-Foulds (RF) distance to measure the topological distance between trees. Fig. 13 shows resultant trees from a different viewpoint, RF distance between two trees. Each heuristic has five runs, and each run begins from the same starting tree across heuristics. The values of RF distances are averaged over five runs and displayed in a heatmap. This figure illustrates the kind of relationship resultant trees from different heuristic have in terms of topological manner. SPR and TBR results whether they're from SLS or PAUP* are clearly grouped together in topo-



(a) performance in time



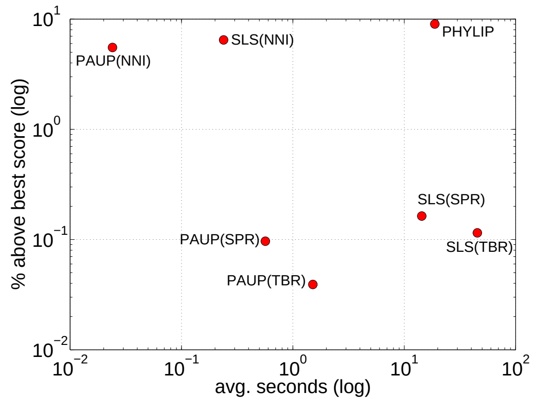
(b) performance in rearrangements

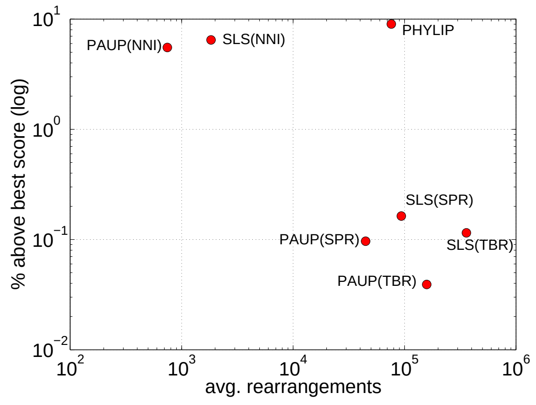
Fig. 10. Hill-climbing performance of SLS, PAUP*, and Phylip on Dataset #1 (44 taxa). PAUP* established the best score of 43,085 for this dataset.

logical distance. The distance between SPR and TBR is similar to distance within SPR neighbors themselves or within TBR neighbors themselves. On the contrary, NNI rearrangements between themselves are not as close as SPR or TBR rearrangements. This means that the NNI search does not guarantee sufficient diversity to bring different starting points to a close area. The Phylip results are highly closer between themselves, but they are far away from others. In conclusion, the Phylip search operates uniquely, NNI does not navigate the space sufficiently, and SPR and TBR results from PAUP* and SLS are very similar to each other.

C. Conclusions

Regarding scoring, it is clear that speedup techniques such as bit-wise and shortcut play an important role in improving heuristic performance. Performance is maximized when two techniques are applied together as is shown in Fig. 6. The shortcut reduces the number of nodes to be updated, and the bitwise score calculation does not interfere or affect with the shortcut, since they work in different hierarchies. For this reason,

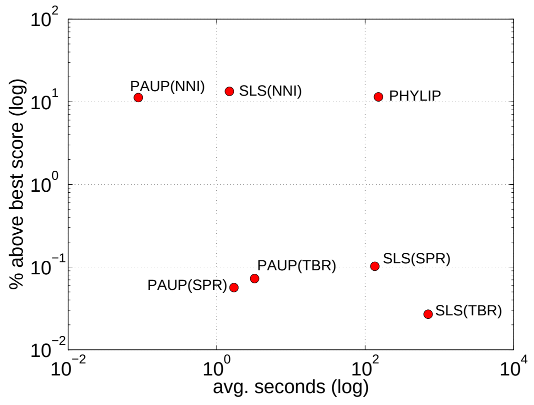




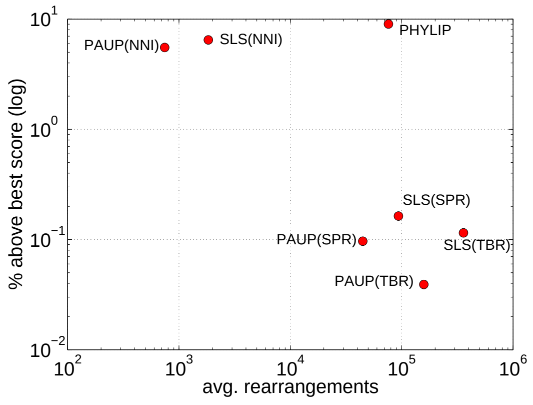
(a) performance in time *(b) performance in rearrangements*

Fig. 11. Hill-climbing performance of SLS, PAUP*, and Phylip on Dataset #2 (60 taxa). PAUP* established the best score of 8,701 for this dataset.

the effect is multiplied when they are together. When it comes to a search inside tree space, two decisions should be made, which rearrangement scheme and which heuristic are to be used. When it comes to the selection of a rearrangement scheme, each rearrangement has a unique score and topological relationship. So we have to consider these factors in order to meet our purpose most effectively, when we try to employ a rearrangement scheme for navigating our dataset. Normally, NNI is fast but not precise in finding good trees, while SPR and TBR results are grouped together, and both are good. Another important question for an efficient search is which heuristic we have to use. In order to get the correct answer, heuristic should be quantified or measured by the unit of building block. In this study, we measure the time for retrieving their neighbors and scoring them together, because they are the most essential steps in MP heuristic. In our experiment, difference of this value between PAUP* and SLS can partly account for the gap of the overall performance between them. Other than this difference, the direction of searching can be said to be similar in a topological sense in that RF distance between PAUP* and SLS is so



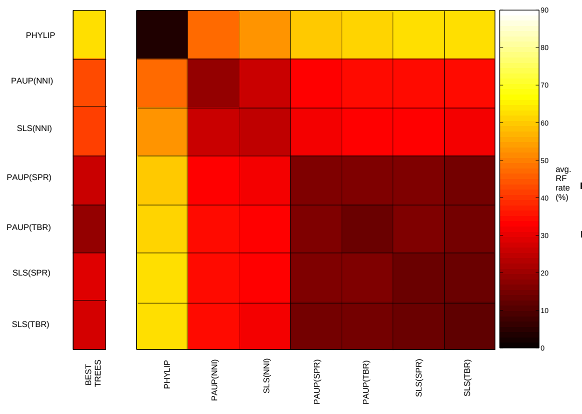
(a) performance in time



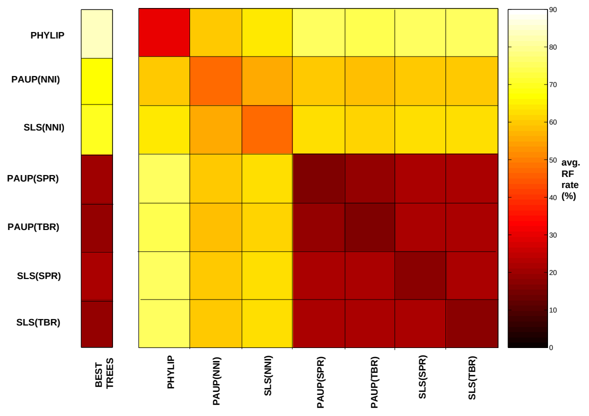
(b) performance in rearrangements

Fig. 12. Hill-climbing performance of SLS, PAUP*, and Phylip on Dataset #3 (174 taxa). SLS established the best score of 7,442 for this dataset.

close. So, based on the limitation that we cannot look inside of PAUP*, SLS is an efficient and reasonable heuristic for MP problems, especially for trying to capture events inside search processes, even though there still is a room for improvements. All later experiments will therefore use SLS.



(a) Dataset #1 (44 taxa)



(b) Dataset #3 (174 taxa)

Fig. 13. Heatmaps depicting the topological performance of the heuristics on our smallest (Dataset #1) and largest (Dataset #3) datasets. For each heuristic, we measure its RF rate to the best-known tree. Moreover, each heatmap shows the average RF rate between the trees found by every pair of heuristics. Here, the color scale ranges from dark (closely related trees) to light (distantly related trees).

CHAPTER VI

CORRELATION INSIDE THE LOCAL SEARCH SPACE

A. Introduction

Based on the information that SLS has captured during the local search procedures, we analyze the characteristics of the search path trees to distinguish them from others in the space. In order to describe the status of search path trees formally, we define search path tree and correlation measures.

1. Search Path Trees

We think of search path trees in the search history as promising trees that the search heuristic should focus on. The search history of a phylogenetic heuristic is the set of neighbors selected along the search path to the best tree. Fig. 14 conceptually describes how SLS search progresses. Let β represent the type of move used in a neighborhood. Hence, $\beta \in \{NNI, SPR, TBR\}$. P_β denotes the search path trees consisting of selecting the first-improving neighbor from a β neighborhood. The sequence of trees encountered along the search path using a β operation is defined as follows.

$$P_\beta = (t_1, \ldots, t_m).$$

For a path P_β, 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, P_β, where t_1 represents the initial (or starting) tree, and t_m is the final tree (e.g., local optimum). Thus, P_β represents the historical record of the phylogenetic search, and they will be usually estimated in experiments. Since TBR searches cover a superset of trees from both NNI and SPR [12], we take TBR neighbors as β.

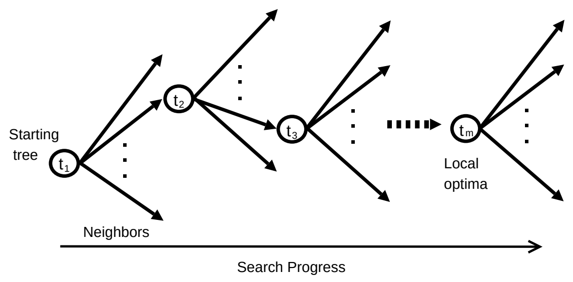


Fig. 14. A depiction of the trees $(t_1, t_2, \ldots, t_m)$ visited by the SLS algorithm on its way to reaching a final tree (e.g., local optimum). t_1 represents the starting tree and t_m is final tree (local optimum) found. Here, each tree t_i along the search path is the neighbor selected from tree t_{i-1}'s neighborhood.

2. Goodness of Fit

A *character* is any observable part, or attribute, of an organism. In a molecular sequence, the total number of characters is equal to the sequence length. The *basic premise* of parsimony is that taxa which share a common feature (or character) do so because they inherited that feature from a common ancestor. When conflicts with that assumption occur, then *homoplasy* occurs. Homoplasies are regarded as extra steps or hypotheses that are required to explain the data.

Three parameters are used to help define indices for quantifying the amount of homoplasy contained in a tree.

- s : length (number of steps) required by the character on the tree being evaluated;
- m : minimum amount of change that the character may show on any conceivable tree; and
- g : maximum possible amount of change that a character could possibly require on any conceivable tree (i.e., the length of the character on a star topology).

The *consistency index* [21] for a single character, c, equals m/s. Thus, if a particular tree explains the data as well as any tree possibly could, $c = 1$. The retention index, r, is defined as $(g - s)/(g - m)$. Thus, when a characters fits the data as poorly as possible, its retention index will be 0. Hence, the *retention index* [6] measures the amount of homoplasy which is locally informative in a dataset. Later, Farris proposed new indices, *rescaled consistency index* [7]. Farris recommends using r as a factor for scaling c between 0 and 1, defining the *rescaled consistency index* as the product of r and c ($=rc$). So we measure the amount of homoplasy with *rescaled consistency index* in our experiments.

Generally, there are two sources of homoplasy. Mis-coding of characters or mistakes in making a homoplasy statement can raise this problem, or parallelisms and reversals actually are real phenomena in nature. Then, what does this concept have to do with the parsimony score which is our main criteria for search? There have been many studies of the relationship between homoplasy and parsimony score. This controversy can be summaried into one question 'Must homoplasy be rare for parsimony to be justified?' As the answer for this question, Steve Farris said "No" [20], Joe Felsenstein said "Yes" [8], and Elliott Sober said "Maybe" [29]. Then, let's see how our dataset behaves as MP score decreases in terms of the goodness of fit, and determine how parsimony score and the fit of data relate to each other.

3. Measuring the Correlation between Trees

We try to determine the characteristics of space and trees using correlation measures. We compute the correlation between tree characteristics based on a measure proposed by Jones and Forrest for genetic algorithms [19]. Their measure computed the correlation between the fitness and Hamming distance between n individual solutions in a population. We extend their measure for use in a phylogenetic search. In particular, consider a set $X = \{x_1, x_2, \ldots, x_n\}$ and a corresponding set $Y = \{y_1, y_2, \ldots, y_n\}$ of

n solutions (or trees). We compute the correlation coefficient, r_{XY}, between the two sets X and Y as

$$r_{XY} = \frac{c_{XY}}{\sigma_X \sigma_Y}, \text{where}$$

$$c_{XY} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

is the covariance of X and Y, and σ_X, σ_Y, $\bar{x}$, and $\bar{y}$ are the standard deviations and means of X and Y, respectively.

The set of tree characteristics of interest that make up the sets X and Y are the parsimony score, the Robinson-Foulds (RF) distance, and the amount of homoplasy contained in a tree. Since there are three possible combinations of these characteristics, we study the performance of a phylogenetic search using the following three correlation coefficients:

- r_{FD}: fitness distance (FD) correlation coefficient
- r_{HD}: homoplasy distance (HD) correlation coefficient
- r_{FH}: fitness homoplasy (FH) correlation coefficient

$F = \{f_1, f_2, \ldots, f_n\}$ represents the fitness (parsimony scores) of the trees. $D = \{d_1, d_2, \ldots, d_n\}$ is the set of n RF distances to the nearest best-known tree. Finally, $H = \{h_1, h_2, \ldots, h_n\}$ is the amount of homoplasy present in the trees. (Of the three correlation equations, we note that the fitness distance correlation is the most closely related to the measure of Jones and Forrest.) A strongly positive (or negative) r coefficient, $-1 \leq r \leq 1$, indicates that the solution quality gives good guidance when searching for global optima. r values close to zero indicate no clear correlation between the two sets. Hence, the interpretation is that the smaller the deviation, that is, the better the solution, the closer we get to the global optima, on average.

In calculting distance value using RF distance, we compare trees found by our SLS algorithm to the best-known trees for the data under consideration. However, the best-known trees are not always available. So, in the next experiment, we will extend the concept of r_{FD} in order to address more practical problems.

B. Results

Our main objective here is to study the behavior of local search heuristics such as SLS for maximum parsimony. Of particular interest to us are: (i) the correlation between path tree scores and their RF distance from the best-known tree; (ii) the fit of the data to better-scoring trees; (iii) the correlation between path tree RF distances and their homoplasy estimates; and (iv) the distribution of search tree scores within the neighborhood of the current solution on the path. These questions are all about search path trees or the landscape around them.

1. Fitness-Distance

Figures 15, 16 and 17 show the correlation between MP scores and their topological distance from the best-known tree for Datasets #1,#2 and #3, respectively. The exact r_{FD} values for all cases are given in Table I. Each data point in Figs. 15, 16 and 17 represents the MP score (in terms of the percentage above the best-known score) in x-axis and its RF distance from the best-known tree for that dataset in y-axis. Lower values denote lower parsimony scores and better topological accuracy. Figures 15 (a), 16 (a) and 17 (a) clearly show that the MP score and RF distances from the best tree are highly correlated. In other words, the results imply that as the tree solutions are getting closer to the optimal tree, the topological accuracy of the trees improve. Figures 15 (b), 16 (b) and 17 (b) provide a closer look at the level of

Table I. The fitness distance correlation coefficients (r_{FD}) for all three datasets.

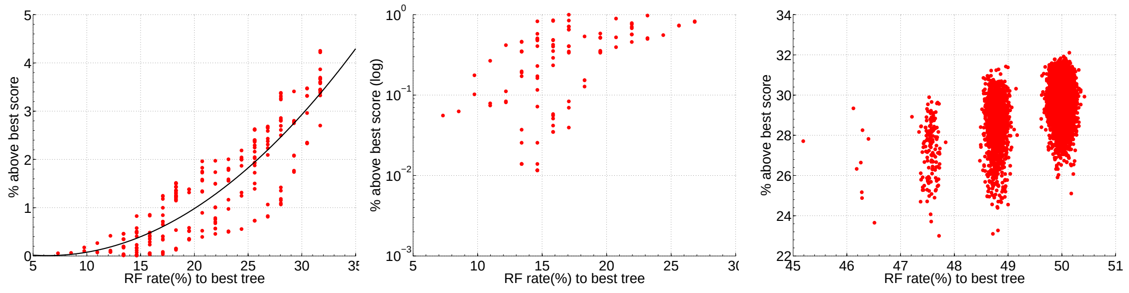
Fitness distance (r_{FD})

	P_{NNI}	P_{SPR}	P_{TBR}	P_{RAND}
Dataset #1 (44 taxa)	0.84	0.81	0.89	0.41
Dataset #2 (60 taxa)	0.82	0.92	0.95	0.28
Dataset #3 (174 taxa)	0.94	0.96	0.98	0.21

accuracy needed in the MP scores to reach the desired level of topological accuracy. However, Figure 15 (c), 16 (c) and Figure 17 (c) provide evidence that for a random search there is very little correlation between MP scores and distance.

2. Fitness-Homoplasy

Fig. 18 shows the correlation between MP scores and their RC values for Dataset #3. The exact r_{FH} values are given in Table II. Search path trees are plotted by their homoplasy values quantified by *rescaled consistency index* value in x-axis and MP scores normalized with the best score in y-axis. Since SLS performs a hill-climbing search, the actual search progresses from right to left in the direction that MP score is decreased. But from the Table II and figure 18, we note that a random walk also has a high correlation coefficient value (-0.97). This correlation gives us no significant information, because our purpose in here was to distinguish search path trees from others using this correlation. In other words, r_{FH} cannot identify search path trees, since the measure shows a consistent trend across all trees even in random trees in the space.

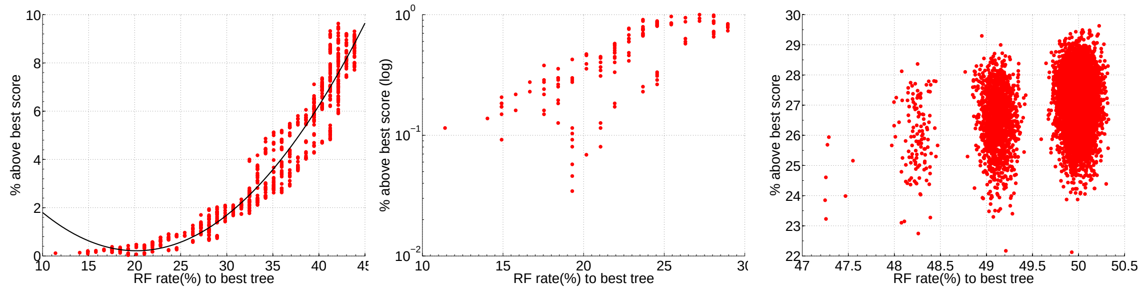


(a) search path trees *(b) trees within 1% of best score* *(c) random trees*

Fig. 15. Fitness distance correlation for Dataset#1 (44 taxa). (a) The fitness (MP score) and RF rate relative to the best-known tree of trees selected along the path to the local optimum under a TBR neighborhood. Here, $r = 0.89$, where r is the correlation coefficient value. (b) Trees along the path to the local optimal solution within 1% of the best score. (c) 10,000 random trees with a r of 0.48. For better display, we added a minimal amount of randomness to x-axis so that all points wouldn't be aligned into one line.

3. Homoplasy-Distance

Fig. 19 displays the correlation between homoplasy values and distance values of search path trees for Dataset #3, and Table III shows the actual r_{HD} values for the dataset. In the figure, the x-axis represents the RF rate values of search path trees from five search runs of search, and the y-axis represents the RC values from the same set of trees. From the previous observation that MP scores of search path trees are highly correlated with RF rate values, we can say search progresses from right to left in this figure. The result that RC and RF are highly correlated is predictable, since the behavior of RC values always match to that of MP scores. And of course, random trees shown in Fig. 19 have little correlation.

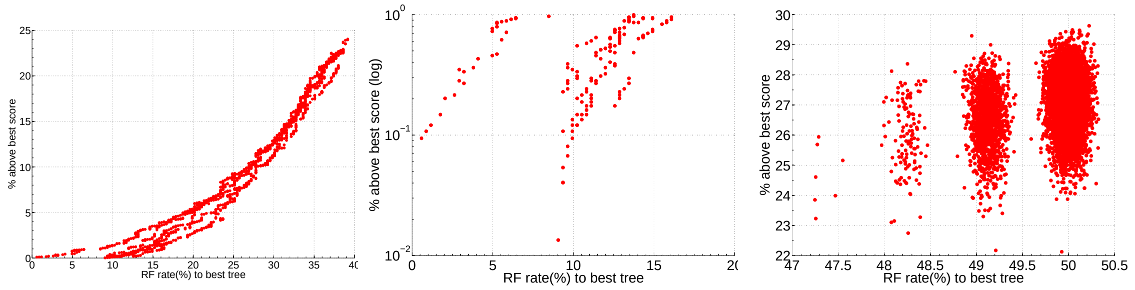


(a) search path trees *(b) trees within 1% of best score* *(c) random trees*

Fig. 16. Fitness distance correlation for Dataset #2 (60 taxa). (a) The fitness (MP score) and RF rate relative to the best-known tree of trees selected along the path to the local optimum under a SPR neighborhood. Here, $r = 0.93$, where r is the correlation coefficient value. (b) Trees along the path to the local optimal solution within 1% of the best score. (c) 10,000 random trees with a r of 0.28. For better display, we added a minimal amount of randomness to x-axis so that all points wouldn't be aligned into one line.

4. Neighborhood Fitness

Figure 20 shows the distribution of MP scores within the TBR neighborhood of the current solution. For example, the bottom histogram represents the TBR neighborhood of each of the five starting trees (0% search progress). The next interval (20% search progress), shows that the TBR neighborhood at this point has improved dramatically over the starting tree neighborhoods. As the search progresses, not only is the current MP score improved, but the tree scores in the surrounding neighborhood are improved as well. Furthermore, the range of the neighboring MP scores are becoming smaller. Our SLS approach uses the first-improvement strategy to select a neighbor. Thus the algorithm does not waste time searching for the best scoring tree in the entire neighborhood (best-improvement strategy). In fact, our runs with the best-improvement strategy produced worse scores and took much more time to terminate at a local optima (not shown). Overall, this result shows that the greedy



(a) search path trees *(b) trees within 1% of best score* *(c) random trees*

Fig. 17. Fitness distance correlation for Dataset #3 (174 taxa). (a) The fitness (MP score) and RF rate relative to the best-known tree of trees selected along the path to the local optimum under a SPR neighborhood. Here, $r = 0.96$, where r is the correlation coefficient value. (b) Trees along the path to the local optimal solution within 1% of the best score. (c) 10,000 random trees with a r of 0.21. For better display, we added a minimal amount of randomness to x-axis so that all points wouldn't be aligned into one line.

method using first-improvement is reasonable compared with the best improvement method since the score of the trees in the neighborhood is improved along with the score of the current tree.

C. Conclusions

In order to understand the behavior of local search heuristics, we investigate into the search path tree of SLS using moderately-sized datasets. For our experiments, Table C shows the number of trees on the search path or the length of search path for the various neighborhoods using our SLS algorithm.

Our experiments with SLS path trees show that there is a strong positive correlation between MP scores and topological distance. And we show that better-scoring trees fit the data better with increased RC values. At first, those two observations look similar in meaning. However, interestingly, what they imply is totally different.

Table II. The fitness homoplasy correlation coefficients (r_{FH}) for all three datasets.

Fitness homoplasy (r_{FH})

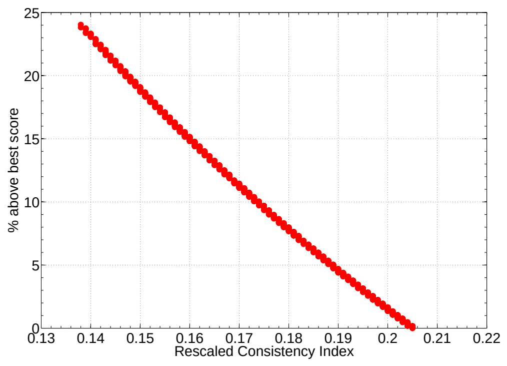
	P_{NNI}	P_{SPR}	P_{TBR}	P_{RAND}
Dataset #1 (44 taxa)	-1.00	-0.98	-1.00	-1.00
Dataset #2 (60 taxa)	-0.99	-1.00	-1.00	-0.98
Dataset #3 (174 taxa)	-1.00	-1.00	-1.00	-0.97

Table III. The homoplasy distance correlation coefficients (r_{HD}) for all three datasets.

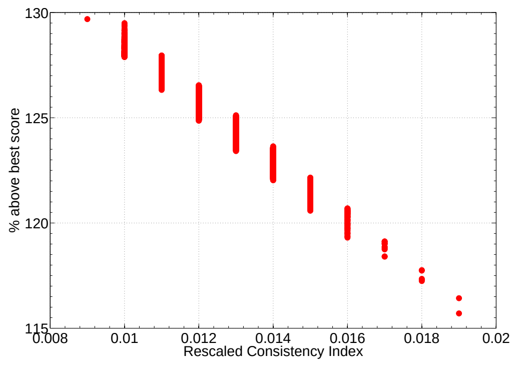
Homoplasy distance (r_{HD})

	P_{NNI}	P_{SPR}	P_{TBR}	P_{RAND}
Dataset #1 (44 taxa)	-0.84	-0.81	-0.90	-0.42
Dataset #2 (60 taxa)	-0.82	-0.92	-0.95	-0.28
Dataset #3 (174 taxa)	-0.94	-0.97	-0.98	-0.20

The measure r_{FD},r_{HD} shows high correlation only with search path trees, while r_{FH} is always correlated for all trees in the space. Given that our purpose was to separate search path trees from other trees, only r_{FD} and r_{HD} are meaningful to us. The strong correlation shown at r_{FD} is between the MP scores and the RF distances of search path trees. Calculating RF rate requires the best tree for the calculation. Therefore, RF rate quantifies how close the current tree might be located from the best tree. In the sense that a lower parsimony score is preferred in the local search, this orientation of RF matches the purpose of the search, and this correspondence of the direction of calculation makes them work together with correlated. However, r_{FH} deals with a little different situation. Unlike the RF calculation, MP and RC are calculated from



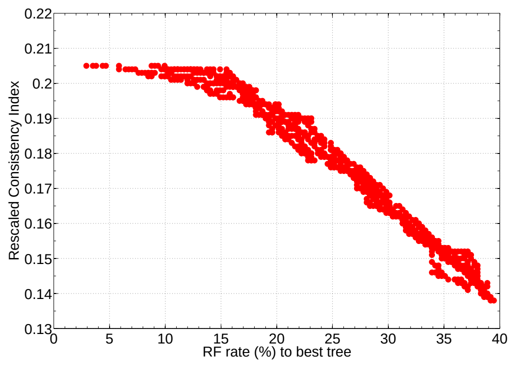
(a) search path trees



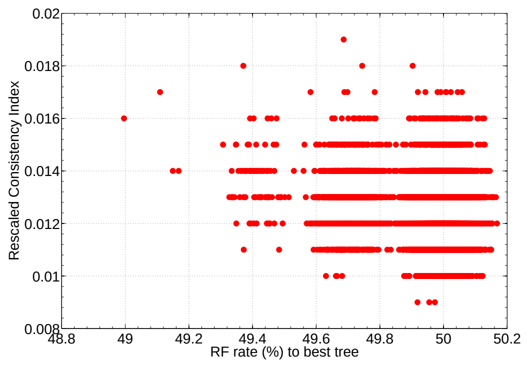
(b) random trees

Fig. 18. Fitness homoplasy correlation for Dataset#3 (174 taxa). (a) The fitness (MP score) and RF rate relative to the best-known tree of trees selected along the path to the local optimum under a TBR neighborhood. Here, $r_{FH} = -1.00$. (b) 10,000 random trees with a $r_{FH} = -0.97$.

what is available, not using the best tree in their calculation. In particular, both calculations commonly use the topology of tree as a tool for quantification. So, the values of MP and RC have to be related with each other, whether or not the tree under consideration is on the path. In addition, we investigate the neighborhood of search path trees. Since our local search heuristic is greedy, it is natural that parsimony scores improve as the search progresses toward a local optima. More enlightening, however, is that neighborhood scores surrounding the current best tree improve as well. As much as it helps us to understand about the first-improvement strategy as mentioned above, it will lead to more valuable application in the next chapter.



(a) search path trees



(b) random trees

Fig. 19. Homoplasy distance correlation for Dataset #3 (174 taxa). (a) The fitness (MP score) and RF rate relative to the best-known tree of trees selected along the path to the local optimum under a TBR neighborhood. Here, $r_{HD} = -0.98$. (b) 10,000 random trees with $r_{HD} = -0.20$. For better display, we added a little bit of randomness to x-axis so that all points wouldn't be aligned into one line.

Table IV. Total number of search path trees for the datasets under study.

Number of search path trees

	$\lvert P_{NNI}\rvert$	$\lvert P_{SPR}\rvert$	$\lvert P_{TBR}\rvert$	$\lvert P_{RAND}\rvert$
Dataset #1 (44 taxa)	166	224	265	10,000
Dataset #2 (60 taxa)	276	778	792	10,000
Dataset #3 (174 taxa)	748	1,698	1,870	10,000

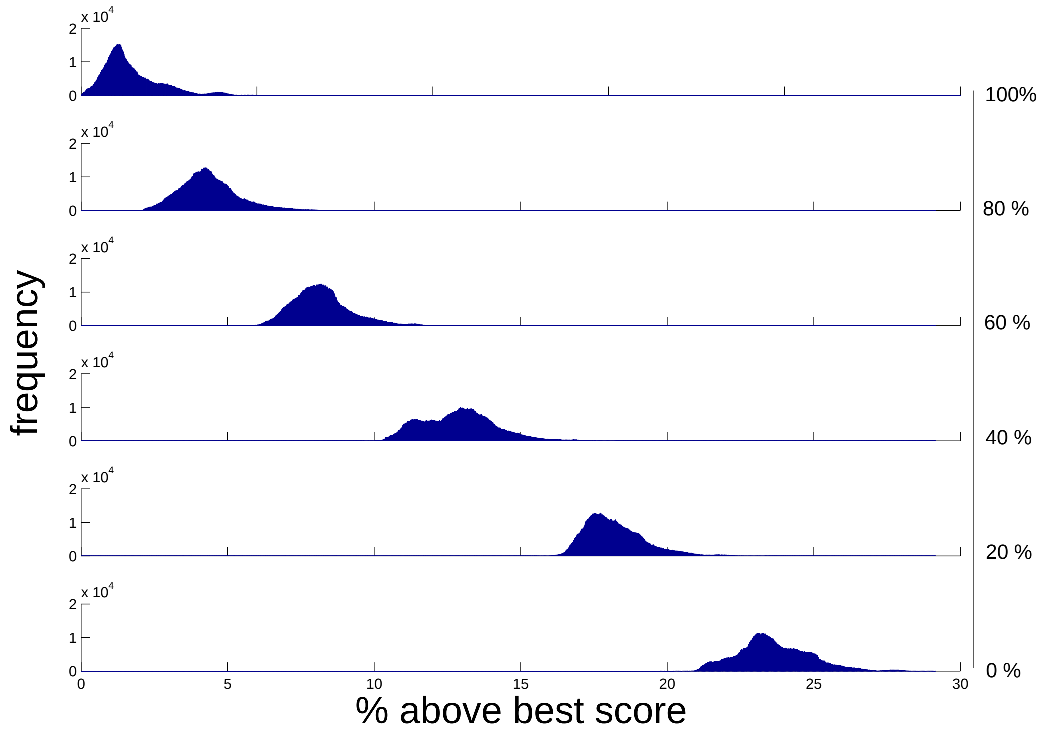


Fig. 20. Fitness (MP score) distribution of SPR neighborhoods on Dataset #3 (174 taxa). Each histogram depicts the MP scores of all neighbors at 20% intervals of the search for best trees. For each interval, all tree scores from the neighborhood of the current tree is shown for all five runs. We could not take TBR, because TBR neighbors for Dataset #3 are so numerous that current data processing technique cannot handle them directly.

CHAPTER VII

APPLYING THE CORRELATION TO THE SPACE

A. Introduction

From the beginning, we have been interested in developing a new technique that exploits search path trees effectively. For that, we have investigated the behavior of path trees from various directions so as to distinguish them from normal trees in the space. Now, it is the time to apply the knowledge in order to answer the second question, 'How can we make use of this knowledge to improve a heuristic?' We demonstrate how the knowledge about the space help developing a new heuristic.

1. Random Neighbor Selection

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 is always better than the previous one. However, we have no idea about how critical it is for a local search that a neighbor be greedily selected. So in our experiments, we introduce some amount of randomness to the search. At a given rate of time during a search, SLS search selects its next tree randomly, and reports what score the search finally has reached. Clearly, this application is related to the previous observation that the score of neighborhood is improving together during the progression of search. This observation tells us that there is not only the neighbor that guides the search to better score. So, we want to see how picking another neighbor affects the search performance.

2. Robinson-Foulds Distance with the Best Tree in the Search

As described earlier, RF(Robinson-Foulds) distance defines the distance between two trees. The initial purpose of this calculation was to check the distance of the current tree from the true tree. Given that it is impossible to know the true evolutionary history for a set of organisms, a reasonable substitute is to use the best-scoring tree found by any phylogenetic method as the *best-tree-overall*. The other possible target tree is the *best-tree-so-far* in that a phylogenetic heuristic may not always have access to the best overall tree—especially if the dataset of interest has been newly created. However, every heuristic will have access to its *best-tree-so-far*, which changes as the phylogenetic search makes improving moves based on fitness in its attempt to find the optimally-scoring tree. Thus, our study examines the behavior of our topological measure on biological datasets using both the *best-tree-so-far* as well as the *best-tree-overall* as target trees.

B. Results

1. Random Neighbor Selection

The previous studies demonstrated that as SLS improves upon the tree t_i, its neighbors improve as well. Local search heuristic invests significant amount of time for selecting a good neighbor. We were curious as to how sensitive the search is regarding selecting a neighbor. That is, what is the impact of randomly selecting a neighbor from $\mathcal{N}_{TBR}(t_i)$? Such a strategy would allow the search to potentially diversify its population of neighboring solutions, which in turn, could lead to better (or worse) tree scores.

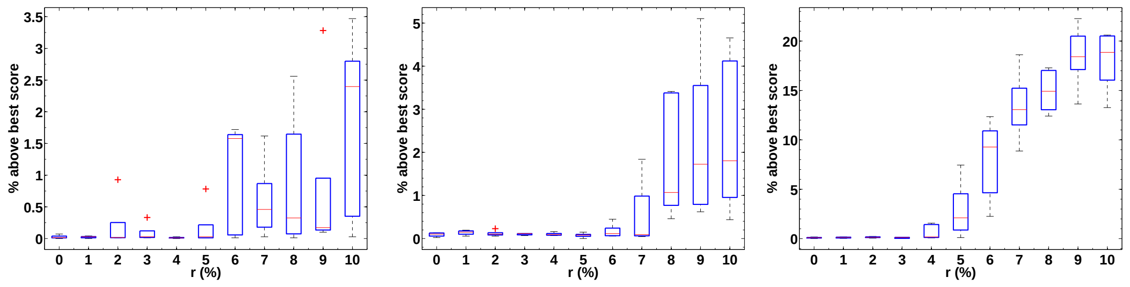
Fig. 21 shows the performance of the SLS algorithm when a random neighbor is selected $r\%$ of the time. Here, $r = 0\%$ represents our standard first improvement algorithm, each tree on the search path is based on the first neighbor that improves upon the current score. For $r \geq 1$, there is an r% 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 Fig. 21, 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. Fig. 22 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, Fig. 23 takes a look at the increased (or decreased) time in log that is required to terminate 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. That is, 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.

2. Robinson-Foulds Distance with the Best Tree in the Search

So far, the fitness-distance coefficient (r_{FD}) requires the access to the best (or optimal) solution. In phylogenetics, for datasets that have been heavily studied such as the `rbcL500` (a.k.a. Zilla data) [15, 25] this is not necessarily a problem. Moreover, on the



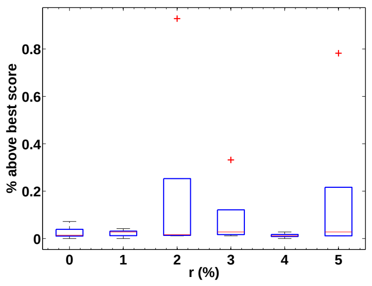
(a) Dataset #1 (44 taxa) *(b) Dataset #2 (60 taxa)* *(c) Dataset #3 (174 taxa)*

Fig. 21. 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.

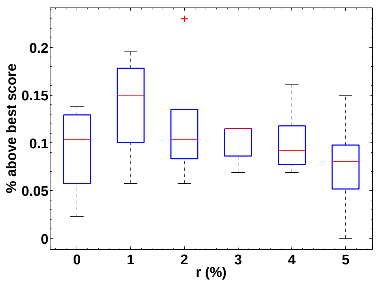
datasets used in this study, we have used numerous software packages (e.g., PAUP* and Phylip) to establish the *best-tree-overall*. It is highly likely that better trees do exist in tree space for these datasets. However, suppose we do not have access to a reliable *best-tree-overall*? How can the r_{FD} correlation coefficient be of use in this situation?

As a phylogenetic heuristic progresses through the search landscape, it will always have access to the *best-tree-so-far*. In other words, if a search has been running for time t, the search can return the fitness of the best-scoring tree that we have for a particular time point. Our next experiment looks at the r_{FD} coefficient of the search at different time intervals $(0\%, 20\%, \ldots, 100\%)$ of the search. The 0% time interval (or search progress) represents the starting trees. By Equation 1, this represents tree t_0 in the search path. The 20% time interval represents tree at $0.20 \cdot m$, where m is the length of the search path. The remaining tree interval points are found similarly.

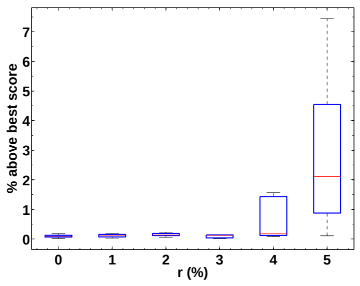
Fig. 24 (a) shows the r_{FD} values based on different search intervals of a TBR



(a) Dataset #1 (44 taxa)



(b) Dataset #2 (60 taxa)



(c) Dataset #3 (174 taxa)

Fig. 22. A closer look at the performance of the random neighbor selection experiments from Figure 21. Here, each plot show the performance of our SLS algorithm with r varying between 0% and 5%.

neighborhood. For example, to compute the r_{FD} values at 20% search progress, only trees labeled from t_0 to $t_{0.20 \cdot m}$ are used in the calculation. Furthermore, the RF distance between each of these trees is compared to the *best-tree-so-far*, that is the best-tree found within the 0% to 20% time interval.

r_{FD} values in Fig. 24 (a) decrease in the beginning and increase again at the 40% search mark. The initial r_{FD} values are high since there are not many points involved in the calculation. However, after 40% search progress, the r_{FD} value steadily increases showing a high positive correlation. Fig. 25 shows the scatter plots for the r_{FD} values for Dataset #3 at 0%, 20%, 40%, 60%, 80%, and 100% search progress.

According to Fig. 24 (a), the r_{FD} values are strongly correlated by 80% search completion. If the search were to stop early, what would be effect on the topological accuracy of the search as it relates to the *best-tree-overall*. Fig. 24 (b) shows the results. For each point, the RF distance between the *best-tree-so-far* at p% search progress is compared with the *best-tree-overall*. Clearly, at 80% search progress for the two smallest datasets, there is minimal (if any) loss in topological accuracy. Furthermore, there is a savings of 20% in overall computational time.

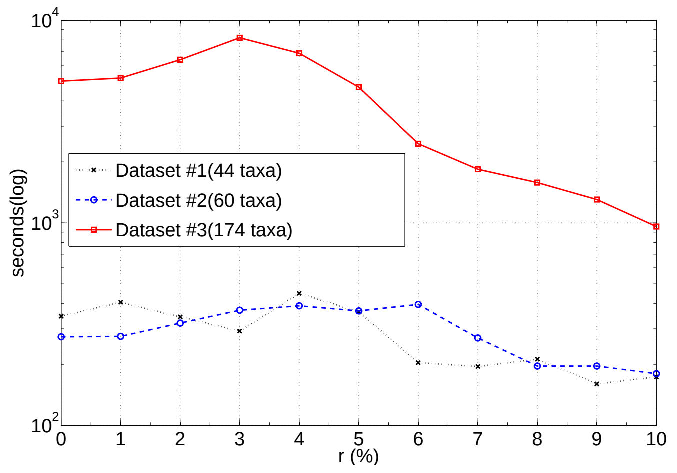
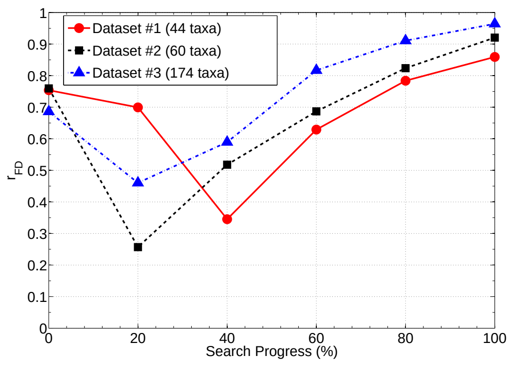


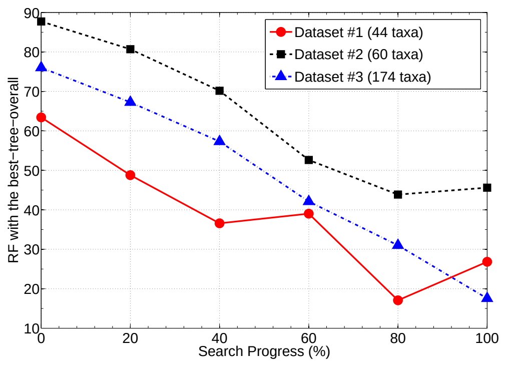
Fig. 23. The running time required for the SLS heuristic with different values of r. Each data point is the average of five runs.

C. Conclusions

The design of better phylogenetic heuristics can be initiated by analyzing the behavior of local searches. 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. This observation also provides evidence why search strategies such as parsimony ratchet [25], which takes backwards moves by reweighting the characters in the dataset, has been a highly successful search strategy. According to the result here, the search is quite robust to a small percentage of random neighbor selections. Using this robustness, search can diversify its population of neighboring solutions. Also, we extended the correlation coefficient called r_{FD} that was developed before. Based on a variety of different biological datasets, our previous results showed that improvements in fitness are strongly correlated ($r_{FD} > 0.8$) with topological distance to the



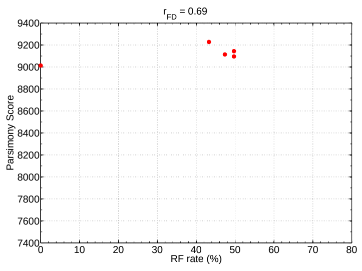
(a) r_{FD} at different search intervals



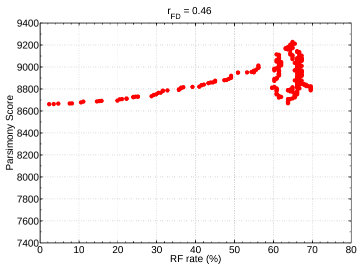
(b) RF rate to best-tree-overall

Fig. 24. r_{FD} are estimated with search path trees at each search progress on all dataset (a) and the best-tree-so-far at each search progress is compared with the best-tree-overall to see how the search goes.

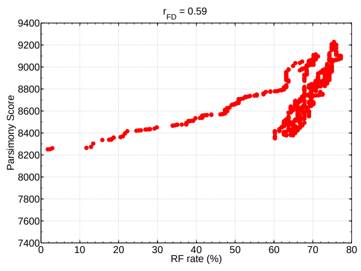
best-tree-overall. Here we investigated the use of the r_{FD} coefficient if the best overall tree is not available. Every run of a phylogenetic search can produce a *best-tree-so-far*. By monitoring the search at different time intervals, we also found that the r_{FD} coefficient shows strong positive correlation. Hence, the r_{FD} value is robust in that it does not need access to the *best-tree-overall.* As the search gets closer to terminating at a local optimum, the r_{FD} value increases accordingly. Hence, r_{FD} values could be used as a stopping criterion to determine when a search should stop. For Datasets #1 and #2, it would be safe to stop early (at the 80% search progress point) without any penalties in topological accuracy. Futhermore, a saving of 20% in computation time is saved without any corresponding loss in accuracy.



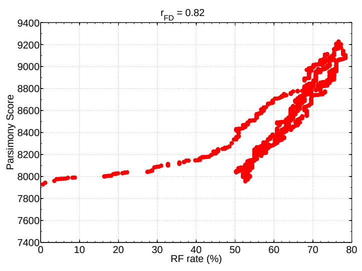
(a) 0% search progress



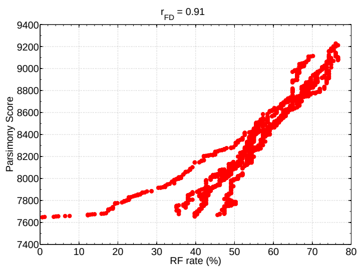
(b) 20% search progress



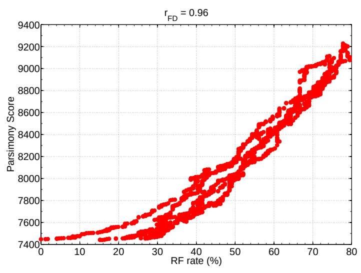
(c) 40% search progress



(a) 60% search progress



(b) 80% search progress



(c) 100% search progress

Fig. 25. Distribution of RF rate values of search path trees with their parsimony scores at various points in the search. r_{FD} at each progress is given on the top of each figure.

CHAPTER VIII

CONCLUSION AND FUTURE WORK

The task of knowing the true evolutionary history of all organisms can be converted into determining the structure of the Tree of Life in our community. But, it is impossible to exhaustively traverse the tree space to solve this problem. As a result, phylogenetic heuristic attempts to find the optimal-scoring tree in this exponentially-sized tree space. However, no heuristic in use can actually do this in reasonable time. So, we are in need of a more powerful heuristic, and we believe this should start from knowing the landscape of the space more thoroughly and in detail. We first focused to the search space. Deep understanding about the search space is essential in that all the possible trees reside there. In particular, we will look at promising trees in the space, search path trees in this paper.

Heuristic measurement clearly shows that SLS is the only appropriate and available local search heuristic for the purpose of our study in that it performs effectively and provides data collecting control. In terms of performance measurement, we go beyond just checking the score and search time of the final results. We break heuristics into basic blocks, and evaluate each of them separately. Lastly, we identify the relationship between each heuristic in topological sense. All these tests confirm that further research can be conducted with SLS. In addition, we present the impact of two speedup techniques used in SLS by profiling them. Based on this profiling and SLS software that is available upon request from the author, techniques for search heuristic can be investigated and tested further.

To analyze the search path trees, SLS retrieves trees on good local search paths and their associated information. For the question 'What characteristic do search path trees have?', we take them into account with interesting concepts such as r_{FD}, r_{FH} and

r_{HD}. As we have investigated their behavior in the previous chapter, their behaviors are not always obvious and straightforward. They are all correlated, but some of them are significant, and others are not. We will extract usefulness from those observations and apply them for more practical purposes. A particularly useful observation is that the neighborhood of search path trees improves, in terms of parsimony score, as the search progresses. We go toward the next question with these interesting observations.

The final question we addressed was 'How can the knowledge about search path trees contribute to making more effective navigation?'. For this, we considered two results from the previous chapter with a little manipulation. The first experiment selected the next pivot randomly at given r percent of time. This experiment clearly showed that some percentage of random neighbor selection would help diversifying its population of neighboring solutions. Second, we have updated r_{FD} updated from the previous chapter so that the calculation would be with the *best-tree-so-far*. This measure indicated how far the search progressed, because the trend in value remained consistent across all data.

In all our experiments, search path trees were investigated in order to know the behavior of local search heuristic, and random trees represent the tree space. Using these data, we can present two main contributions to the community. The first contribution is a deep understanding about the search space and local search heuristics in the space. Using our data, we determined that phylogenetic trees that are parsimonious always have less homoplasy throughout the space whether or not they are on a search path. On the other hand, the topological estimate was highly correlated with parsimony score only in search path trees. This result can be interpreted that the MP search heuristic refines its score while simultaneously approaching the best tree topologically simultaneously. As another contribution, this paper shows how the knowledge affect the performance of search heuristic. The knowledge about score distribution of neighborhood allows a heuristic not to have to spend as much time as it used to need, because there are many other options for the next progression in terms of the next pivot. Also, the knowledge about the correlation coefficient r_{FD} can tell where the search is at. Stated earlier, it doesn't provide any actual heuristic, but it provides one good starting point for more powerful and stable heuristic.

In the future, we plan to improve the performance (in terms of running time) of our SLS implementation and make it publicly available to the systematics community. From Figs. 10 and 12, SLS is slower than PAUP*, but has fewer number of rearrangements. This points out that SLS can be improved if implementation is optimized. So, improvement of SLS by optimizing code should be performed. Also not only for making further reliable and efficient heuristic but also for experimenting a delicate setup with the heuristic, we will open SLS source code and provide the chance to manipulate SLS according to their need.

Also, we will further investigate the value of r_{FD} and r_{DH} by examining larger datasets and additional phylogenetic heuristics. So far, we have analyzed the behavior of local search by looking at search path trees from SLS. But for moderately-sized datasets (> 250 taxa), more powerful approaches such as Parsimony Ratchet [25], Recursive-Iterative DCM3 [27], and TNT [15] are more commonly used because of their performance. By replacing the local search heuristic blocks with blocks from SLS, we can collect information going on in heuristics, and analyze them. This will provide a foundation for understanding (and appreciating) how these more powerful approaches operate.

REFERENCES

[1] B. Allen and M. Steel. “Subtree transfer operations and their induced metrics on evolutionary trees.” *Ann Comb.*, vol. 5, pp. 1-13, 2001.

[2] D. A. Bader, B. M. E. Moret, and L. Vawter. “Industrial applications of high-performance computing for phylogeny reconstruction.” In *Proceedings of SPIE Commercial Applications for High-Performance Computing*, vol. 4528, pp. 159-168, Aug. 2001.

[3] D. Butt, A. Roger, and C. Blouin. “libcov: A C++ bioinformatic library to manipulate protein structures, sequence alignments and phylogeny.” *BMC Bioinfomatics*, vol. 6, pp. 138-139, 2005.

[4] B. Chor and T. Tuller. “Maximum likelihood of evolutionary trees is hard.” In *Proc. 9th Ann. Int. Conf. on Computational Molecular Biology (RECOMB ́05)*, pp. 296-310. Cambridge, 2005.

[5] Andrew R. Deans and Joseph J. Gillespie and Matthe J. Yoder. “An evaluation of ensign wasp classification (*Hymenoptera: evanildae*) based on molecular data and insights from ribosomal RNA secondary structure.” *Syst. Ento.*, vol. 31, pp. 517-528, 2006.

[6] J. S. Farris. “The retention index and homoplasy excess.” *Systematic Zoology*, vol. 38, pp. 406-407, 1989.

[7] J. S. Farris. “The retention index and rescaled consistency index.” *Cladistics*, vol. 5, pp. 417-419, 1989.

[8] J. Felsenstein. “Parsimony in systematics: biological and statistical issues.” *Annu. Rev. Ecol. Syst.*, vol. 14, pp. 313-333, 1983.

[9] J. Felsenstein. “Phylogenetic inference package (PHYLIP), version 3.2.” *Cladistics*, vol. 5, pp. 164-166, 1989.

[10] W. M. Fitch. “Toward defining the course of evolution: minimal change for a specific tree topology.” *Syst. Zool.*, vol. 20, pp. 406-416, 1971.

[11] L. R. Foulds and R. L. Graham. “The Steiner problem in phylogeny is NP-complete.” *Advances in Applied Mathematics*, vol. 3, pp. 43-49, 1982.

[12] G. Ganapathy, V. Ramachandran, and T. Warnow. “Better hill-climbing strategies for parsimony.” In *Proceedings 3nd Int'l Workshop Algorithms in Bioinformatics (WABI'03)*, volume 2812 of Lecture Notes in Computer Science, pp. 245-258, Springer-Verlag, 2003.

[13] G. Ganapathy, V. Ramachandran, and T. Warnow. “On contract-and-refine-transformations between phylogenetic trees.” In *Proc. 15th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 900-909, Philadelphia, PA, 2004.

[14] 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.*, vol. 15, pp. 625-643, 2005.

[15] P. Goloboff. “Analyzing large data sets in reasonable times: solutions for composite optima.” *Cladistics*, vol. 15, pp. 415-428, 1999.

[16] M. Hendy, M. Steel, D. Penny and I. Henderson. ”Families of trees and consensus.” In H. Bock, editor, *Classification and Related Methods of Data Analysis*, pp. 355-362. Amsterdam, The Netherlands, Elsevier, 1988.

[17] D. Hillis. “Inferring complex phylogenies.” *Nature*, vol. 383, pp. 130-131, 1996.

[18] D. Hillis. "Analysis and visualization of tree space." *Syst. Biol.*, vol. 54, no. 3, pp. 471-482.

[19] T. Jones and S. Forrest. "Fitness distance correlation as a measure of problem difficulty for genetic algorithms." In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 184-192, San Francisco, CA, 1995.

[20] M. V. Kallersjo, A. Albert and J. S. Farris. "Homoplasy increases phylogenetic structure." *Cladistics*, vol. 15, pp. 91-93, 1999.

[21] A. G. Kluge and J. S. Farris. "Quantitative phyletics and the evolution of anurans." *Systematic Zoology*, vol. 18, pp. 1-32, 1969.

[22] D. Maddison. "The discovery and importance of multiple islands of most parimonious trees." *Syst. Bio.*, vol. 42, no. 2, pp. 200-210.

[23] M. L. Metzker, and 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*, vol. 99, no. 2, pp. 14292-14297, 2002.

[24] 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*, vol. 294, pp. 2348-2351, 2001.

[25] K. C. Nixon. "The parsimony ratchet, a new method for rapid parsimony analysis." *Cladistics*, vol. 15, pp. 407-414, 1999.

[26] F. Ronquist. "Fast fitch-parsimony algorithms for large data sets." *Cladistics*, vol. 14, no. 4, pp. 387-400, 1998.

[27] 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)*, pp. 98-109. IEEE Press, 2004.

[28] N. Saitou and M. Nei, “The neighbor-joining method: A new method for reconstructing phylogenetic trees.” *Mol. Biol. Evol.*, vol. 4, pp. 406-425, 1987.

[29] E. Sober, “Parsimony in systematics: Philosophical issue.” *Ann. Rev. Ecol. Syst.*, vol. 14, pp. 335-357, 1983.

[30] 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)*, pp. 285-293, 2002.

[31] S.-J. Sul and T. L. Williams. “A randomized algorithm for comparing sets of phylogenetic trees.” In *Proc. Fifth Asia Pacific Bioinformatics Conference (APBC'07)*, pp. 256-260, 2007.

[32] D. L. Swofford. “*PAUP*: Phylogenetic Analysis Using Parsimony (and Other Methods)*, Version 4.0.”, Underland, MA: Sinauer, 2002.

[33] T. L. Yates, J. Salazar-Bravo, and J. W. Dragoo. 2004, “The importance of the tree of life to society”, In J. Cracraft and M. J. Donoghue, (eds.) *Assembling the Tree of Life*, pp. 7-17, Cambridge, UK, Oxford University Press, 2004.

VITA

Hyun Jung Park

Education

- Master of Science, Computer Science, Texas A&M University, 2007
- Bachelor of Science, Computer Science, Yonsei University,
  South Korea, 2002

Contact Address

- *Permanent mailing address* : 301 Harvey R. Bright Building College Station, TX 77843-3112
- *E-mail* : justpark@tamu.edu, justpark78@gmail.com