The Dissertation Committee for Derrick Joel Zwickl
Certifies that this is the approved version of the following dissertation:

GENETIC ALGORITHM APPROACHES FOR THE PHYLOGENETIC ANALYSIS OF LARGE BIOLOGICAL SEQUENCE DATASETS UNDER THE MAXIMUM LIKELIHOOD CRITERION

Committee:

David M. Hillis, Supervisor

David C. Cannetella

Robin R. Gutell

Robert K. Jansen

Tandy Warnow

GENETIC ALGORITHM APPROACHES FOR THE PHYLOGENETIC ANALYSIS OF LARGE BIOLOGICAL SEQUENCE DATASETS UNDER THE MAXIMUM LIKELIHOOD CRITERION

by

Derrick Joel Zwickl, B.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May, 2006

Dedication

To John Conaway and Maria Truong.

GENETIC ALGORITHM APPROACHES FOR THE PHYLOGENETIC ANALYSIS OF LARGE BIOLOGICAL SEQUENCE DATASETS UNDER THE MAXIMUM LIKELIHOOD CRITERION

Publication No._____

Derrick Joel Zwickl, Ph.D.

The University of Texas at Austin, 2006

Supervisor: David M. Hillis

Phylogenetic trees have a multitude of applications in biology, epidemiology, conservation and even forensics. However, the inference of phylogenetic trees can be extremely computationally intensive. The computational burden of such analyses becomes even greater when model-based methods are used. Model-based methods have been repeatedly shown to be the most accurate choice for the reconstruction of phylogenetic trees, and thus are an attractive choice despite their high computational demands. Using the Maximum Likelihood (ML) criterion to choose among phylogenetic trees is one commonly used model-based technique. Until recently, software for performing ML analyses of biological sequence data was largely intractable for more

than about one hundred sequences. Because advances in sequencing technology now make the assembly of datasets consisting of thousands of sequences common, ML search algorithms that are able to quickly and accurately analyze such data must be developed if ML techniques are to remain a viable option in the future.

I have developed a fast and accurate algorithm that allows ML phylogenetic searches to be performed on datasets consisting of thousands of sequences. My software uses a genetic algorithm approach, and is named GARLI (Genetic Algorithm for Rapid Likelihood Inference). The speed of this new algorithm results primarily from its novel technique for partial optimization of branch-length parameters following topological rearrangements. Experiments performed with GARLI show that it is able to analyze large datasets in a small fraction of the time required by the previous generation of search algorithms. The program also performs well relative to two other recently introduced fast ML search programs.

Large parallel computer clusters have become common at academic institutions in recent years, presenting a new resource to be used for phylogenetic analyses. The P-GARLI algorithm extends the approach of GARLI to allow simultaneous use of many computer processors. The processors may be instructed to work together on a phylogenetic search in either a highly coordinated or largely independent fashion. Preliminary experiments suggest that analyses using the P-GARLI software can result in better solutions than can be obtained with the serial GARLI algorithm.

Table of Contents

List of Tables ix

List of Figures x

Chapter 1: Background 1

1.1 Motivation 1

1.2 Phylogenetic inference 2

1.3 The Maximum Likelihood criterion 7

1.4 Maximum Likelihood heuristics 13

Chapter 2: The Serial GARLI Algorithm 22

2.1 Introduction 22

2.2 Algorithm Overview 22

2.3 The Fitness Function 28

2.4 Mutation operators 31

2.5 Branch-length optimization 40

2.6 Memory requirements 51

Chapter 3: Performance of the Serial GARLI Algorithm 52

3.1 Introduction 52

3.2 Methods 54

3.3 Results and Discussion 66

3.4 Conclusions 86

Supplemental Information 88

Chapter 4: The Parallel GARLI Algorithm 89

4.1 Introduction 89

4.2 The P-GARLI Algorithm 90

Chapter 5: Performance of the Parallel GARLI algorithm 99

5.1 Introduction 99

5.2 Methods 99

5.3 Results and Discussion 103
5.4 Conclusions 108

References 109

Vita 115

List of Tables

Table 2-1. Relative reproduction probabilities of two individuals differing by 1.0 lnL unit for various values of the selection intensity, *s*. 30

Table 3-1. Times required to obtain parsimony starting topologies 66

Table 3-2. Optimized log-likelihood and parsimony scores of topologies selected as GARLI starting points.....70

Table 3-3. Results of GARLI analyses performed with parsimony starting topologies. 72

Table 3-4. Results of GARLI analyses performed with random starting topologies 73

Table 3-5. P-values resulting from Kruskal-Wallis nonparametric tests 73

Table 3-6. Results of RAxML and PHYML analyses performed on the five real datasets. 81

Table 3-7. Topological similarity of results obtained by GARLI, RAxML and PHYML.....85

Table 3-8. Topological accuracy of analyses performed on simulated datasets.....85

Table 5-1. Optimized scores and runtimes of P-GARLI analyses of the Gutell3845 dataset. 104

List of Figures

Figure 2-1. A schematic view of the GARLI algorithm. 23

Figure 2-2. The reconnection distance 33

Figure 2-3. Reconnection distances on a real topology 34

Figure 2-4. The relationship between the length of three arbitrarily chosen branches and the total tree log-likelihood of the ML topology for the 64-taxon dataset of Hillis and Wilcox (1995).....42

Figure 2-5. Progression of the GARLI branch-length optimization algorithm over a topology following an SPR rearrangement 48

Figure 3-1. The relationship between parsimony and likelihood scores of topologies obtained by parsimony searches in PAUP*.....67-69

Figure 3-2. Results of GARLI runs performed on the Angio dataset 74

Figure 3-3. Results of GARLI runs performed on the Rbcl dataset 75

Figure 3-4. Results of GARLI runs performed on the 1000ARB dataset 76

Figure 3-5. Results of GARLI runs performed on the Gutell3845 dataset 77

Figure 5-1. Optimized lnL scores of the solutions resulting from P-GARLI analyses of the Gutell3845 dataset 105

Figure 5-2. Runtimes vs. lnL scores for parallel analyses of the Gutell3845 dataset 106

Chapter 1: Background

1.1 MOTIVATION

It is now widely accepted among biologists that an understanding of the evolution of a group of organisms requires an understanding their phylogenetic relationships. In addition to their importance in systematic studies, phylogenetic methods present many other applications such as prediction of the evolution of infectious diseases (Bush et al., 1999), the discovery of new drugs (Brown and Warren, 1998), planning conservation priorities (Faith, 1992), and even forensics (Ou et al., 1992). The recent shift toward an appreciation of the importance of phylogenetics in many biological fields is further underscored by the large amount of NSF funding put toward projects such as "Assembling the Tree of Life" (ATOL: atol.sdsc.edu) and "Cyberinfrastructure for Phylogenetic Research" (CIPRES: www.phylo.org).

It has also become well accepted that probabilistic model-based methods are the superior choice for phylogenetic inference, in part because of studies demonstrating their higher accuracy (Huelsenbeck and Hillis, 1993; Huelsenbeck, 1995; Rosenberg and Kumar, 2001; Zwickl and Hillis, 2002). These methods include those based on the maximum likelihood criterion (Felsenstein, 1981) and more recently developed Bayesian techniques (Rannala and Yang, 1996; Mau et al., 1999; Huelsenbeck and Ronquist, 2001). The primary drawback of these methods is higher computational costs relative to alternative methods.

When first suggested as an optimality criterion for phylogenetic tree inference (Felsenstein, 1981), maximum likelihood (ML) analyses were impractical due to limited computational resources. As computing power and availability have grown tremendously over the last decade, it has allowed ML analyses to become commonplace. However,

increases in available sequence data currently outpace increases in computing speed, leaving phylogeneticists once again limited in their ability to perform ML analyses. The motivation for the development of the GARLI algorithm (Genetic Algorithm for Rapid Likelihood Inference) is to ensure that ML phylogenetic inference will continue to be a reasonable option despite large increases in dataset sizes. This necessity has been recognized by a number of researchers, and has resulted in several new ML inference programs that increase the speed of model-based inference through a variety of algorithmic enhancements (e.g. MetaPIGA: Lemmon and Milinkovitch, 2002; PHYML: Guindon and Gascuel, 2003; RAxML: Stamatakis et al., 2005; IQPNNI: Vinh and von Haeseler, 2004).

This chapter presents background material important for the GARLI algorithm, and for understanding its place in the field of phylogenetic inference. I begin with a general orientation to the problem of phylogenetic inference, and of the techniques available to perform such inference. Methodological details relating to the use of the ML optimality criterion will then be discussed. Finally, a discussion of heuristic methodology and a survey of relevant phylogenetic search algorithms will be presented, with special emphasis on ML heuristics.

1.2 PHYLOGENETIC INFERENCE

Terminology

The goal of phylogenetic inference is to obtain a branching structure representing the evolutionary history of a set of organisms using a set of data that describes some aspect of the organisms. Discussion here will be limited to unrooted, strictly-bifurcating tree topologies, although other potential types of evolutionary relationships between organisms exist (e.g., see Moret et al., 2004). I will begin by defining the component

parts of a tree. Differing sets of terminology are used by researchers in the field. Tree will be used interchangeably with the term topology. The tips of the tree, representing the observed data on which the inference is based, will be termed terminal nodes or taxa (also leaves or tips). The connections between taxa will be referred to as branches (also edges), each of which may be assigned a numerical value representing its length. Branch lengths represent some measure of evolutionary distance between nodes, with the exact definition depending on the inference method. Branches are also referred to as bipartitions, because each branch divides the taxa on either side of it into two non-overlapping sets. Branches intersect at internal nodes or vertices, which represent hypothetical ancestral taxa. By definition, all internal nodes of an unrooted bifurcating tree will be of degree three (i.e., connected to exactly three branches).

Orienting a topology with respect to time, known as rooting, is not typically considered part of the topology search *per se*. Most tree inference methods do not seek to distinguish between potential rootings of a particular unrooted topology, and thus the task of orienting the tree with respect to time usually occurs after tree inference. The most common method is to include outgroup taxa which are assumed *a priori* to be less closely related to any of the ingroup taxa of interest than the ingroup taxa are to one another. Topologies represented in figures and computer implementations are often oriented by choosing an arbitrary internal node to be designated as the effective root, although the tree technically remains unrooted. No additional meaning is assigned to this node, and the score of the tree is independent of the node chosen. The root lies at the "bottom" of the tree, and all internal and terminal nodes lie above it.

Restricting consideration to only bifurcating topologies is a choice of convenience. The consideration of trees containing nodes of degree greater than three, termed polytomies or multifurcations, increases the number of possible topologies and

complicates computer implementations. Polytomies are typically used to represent uncertainty about the relationships in a region of a topology. This uncertainty may arise because of either insufficient or conflicting signal present in the data. Note that computer implementations allowing zero or near-zero branch lengths are able to represent topologies essentially equivalent to those containing polytomies while still maintaining their technically bifurcating nature. Any unrooted bifurcating tree for n taxa will contain exactly $(2n-3)$ branches and $(n-2)$ internal nodes.

Input data

Phylogenetic inference may be performed using a variety of data types. The data are typically represented in the form of a matrix in which rows represent taxa and columns represent individual characters. Each character may be thought of as an independent data point containing information about the relationships between the taxa. Early phylogeneticists primarily used morphological characters, the physical attributes of the organisms. In theory, inference may be performed using any characters that are inherited and are able to change over evolutionary time. Modern phylogenetic inference is performed primarily on biological sequence data, such as DNA, RNA or protein sequences representing the same gene in each of the organisms. Sequence data are composed of a series of characters drawn from a limited alphabet. These characters are referred to as bases or nucleotides in the case of DNA (A, C, G and T) and RNA (A, C, G and U) sequence, and amino acids in the case of protein sequence (A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W and Y). The preference for sequence data is primarily due to the ease and speed with which many characters can be gathered and the tractability of statistically modeling the process of sequence evolution.

Before phylogenetic inference may be performed on sequence data, the sequences obtained for each taxon must first be aligned such that all characters within a column of

the data matrix are homologous. This means simply that the characters are related by a common evolutionary history. Note that this step of alignment would be unnecessary (or at least easier) if the processes of character insertion and deletion did not cause variation in the length of the sequences among the taxa. Alignment methods range from fully automated heuristic algorithms that attempt to find the best alignment based on a parsimony criterion (e.g., Higgins et al., 1994), to more intensive hands-on methods that incorporate information about the structure and function of the sequences being aligned (e.g., Woese et al., 1980**).** Regardless of how the alignment is generated, it is typically fixed once it has been entered into phylogenetic software. Note, however, that several methods for simultaneously inferring alignment and phylogeny have recently been developed (Redelings and Suchard, 2005; Lunter et al, 2005; Fleissner et al., 2005). The remainder of this work will be concerned solely with phylogenetic inference performed on aligned nucleotide sequence data.

Phylogenetic inference methods

A number of distinct methodologies exist for choosing the "best" tree topology given a set of aligned sequences. The methodologies may be roughly divided into three main categories. First, distance-based methods use sequence similarity to obtain an estimate of the pairwise evolutionary distance between taxa, and use those distances to construct or choose a topology. Second, parsimony-based methods seek the topology that minimizes the number of inferred evolutionary changes. Third, model-based methods (i.e., maximum likelihood and Bayesian techniques) use an explicit model of character evolution to identify topologies that are highly probable given the data. Over the past 30 years researchers have argued over which of the methods should be preferred based on both philosophical and performance based arguments. Often these disagreements have not been cordial, and some have gone so far as to describe them as "wars" (see Hull,

1998). In more recent years there has been a general movement toward the acceptance of model-based methods, based largely on studies demonstrating their statistical consistency (Rogers, 2001), superior accuracy (Huelsenbeck and Hillis, 1993; Huelsenbeck, 1995; Rosenberg and Kumar, 2001; Zwickl and Hillis, 2002), and robustness to model violations (Fukami-Kobayashi, and Tateno, 1991**).**

Large tree inference

Both the increased availability of DNA sequence data and advances in computational power are making it possible to assemble and analyze datasets of sizes previously thought impossible. It has frequently been shown that increasing the number of sequences analyzed can increase the accuracy of the inferred trees under realistic conditions (e.g., Hillis, 1996; Poe, 1998; Zwickl and Hillis, 2002), especially when the amount of evolutionary divergence between taxa is large. However, some continue to refute this viewpoint (e.g., Rosenberg and Kumar, 2001; 2003; Rokas and Carroll, 2005). Although the accuracy of inference may sometimes increase with increased sampling of taxa, the magnitude of the inference problem doubtlessly increases to a much greater degree. Define *T(n)* as the number of possible unrooted bifurcating topologies for *n* taxa:

$$T(n) = \prod_{i=3}^{n} (2i - 5)$$

Thus, *T* grows factorially with increasing *n*, and becomes almost unimaginably large for $n > 50$ (e.g., $T(50) \approx 2.84 \times 10^{76}$). Enumeration and evaluation of all possible topologies clearly cannot be considered an option, regardless of the optimality criterion used. The only feasible approach is to apply heuristic methods that seek the optimal solution but have no guarantee of finding it.

1.3 THE MAXIMUM LIKELIHOOD CRITERION

The basic premise of the ML optimality criterion is to choose the topology that makes the observed data the most probable. The simplest parsimony and pairwise distance-based methods are able to calculate the unique score of a topology T given only the data. In contrast, calculating the likelihood score of a topology requires branch-length values for all (n-3) branches and an explicit model describing how sequences evolve over time. Define B as a parameter vector containing the lengths of all branches, and M as a vector of evolutionary model parameters. Branch lengths are typically measured in units of the expected numbers of substitutions per nucleotide site, as will be discussed further below. The number of elements in M and their definitions vary depending on which of the many described nucleotide models is chosen.

More formally, the ML criterion seeks to maximize the likelihood (L) of the variables (T, B and M) given the data, D. The likelihood is proportional to the probability of D given the variables:

$$L(T,M,B \mid D) \propto P(D \mid T,M,B)$$

Often it is an estimate of T that is the primary goal of phylogenetic analyses, with B and M considered "nuisance parameters". However, because the likelihood of any topology depends also on the values of B and M, these parameters are important even if their values are not of particular interest. For this reason, when the likelihood scores of topologies are compared, it is generally on the basis of their maximized likelihood. This is simply the largest possible likelihood score for a particular T across all possible values of B and M. The values of B and M that result in the maximized likelihood will be denoted $\hat{B}$ and $\hat{M}$, and represent the maximum likelihood estimates of those parameters

given that tree. There is not an analytical way of solving for $\hat{B}$ and $\hat{M}$ given a topology, and so the maximized likelihood is obtained by one of a number of numerical optimization techniques such as the Newton-Raphson method or Brent's method (see Press et al., 1992). It is this numerical optimization step that constitutes the majority of the computation in typical ML analyses.

Models of sequence evolution

The sequence evolution models typically used in ML phylogenetic analyses are classified as time-reversible Markov models; they are not directed with respect to time, and that the probability of any substitution depends solely on the current state. Models have been developed for various sequence types, such as sequences of nucleotides, amino acids, and codons. The present discussion will concern only nucleotide-based models. The most general time-reversible nucleotide model is termed, aptly enough, the General Time-Reversible model (GTR: Lanave et al., 1984). The parameters of the GTR model include equilibrium frequencies of the four nucleotides (π_A, π_C, π_G and π_T) and six parameters representing the relative rates of substitution between each nucleotide pair (a, b, c, d, e, f). Note that although this amounts to ten parameters, only eight of the parameters may be considered free because the equilibrium frequencies must sum to one and the f rate is typically fixed to ensure identifiability of the remaining relative rates. These parameters are used to define a matrix $\boldsymbol{Q}$, where $\boldsymbol{Q}_{ij}$ represents the instantaneous rate of substitution from base *i* to base *j*. Using the convention of ordering the bases in alphabetical order (A, C, G, T):

$$Q = \begin{matrix} - & \mu a\pi_C & \mu b\pi_G & \mu c\pi_T \\ \mu a\pi_A & - & \mu d\pi_G & \mu e\pi_T \\ \mu b\pi_A & \mu d\pi_C & - & \mu f\pi_T \\ \mu c\pi_A & \mu e\pi_C & \mu f\pi_G & - \end{matrix}$$

The diagonal elements of $\boldsymbol{Q}$ are the negative sum of the other elements in each row, and are omitted for clarity. μ represents the mean instantaneous substitution rate. $\boldsymbol{Q}$ must be rescaled such that the mean substitution rate is equal to one in order to maintain the proper definition of branch lengths as the expected number of substitutions per site.

Given model parameters describing the instantaneous substitution process, the probabilities of observing state *i* at one end of a branch of length *t* and state *j* at the other may be obtained from the matrix $\boldsymbol{P}$, which is defined thusly:

$$\boldsymbol{P}=e^{\boldsymbol{Q}t}$$

There is no closed form expression for $\boldsymbol{P}$ under the GTR model, and it must be calculated via eigen-decomposition or other numerical methods. Note that all reversible nucleotide substitution models are nested within the GTR model, and may be obtained by fixing some of the parameters of this most general model. The present work is concerned only with the GTR model.

It has been repeatedly demonstrated that a very important component of models of sequence evolution is the accommodation of rate heterogeneity, variation in evolutionary rate between nucleotide sites (Sullivan et al., 1995). Some form of rate heterogeneity is now implemented in nearly all phylogenetic software, although there are multiple approaches. Two will be discussed here, referred to as discrete gamma rate heterogeneity (GRH: Yang, 1993, 1994) and site-specific rate heterogeneity (SSRH: Olsen et al., 1994). In both forms the branch lengths associated with a topology retain their meaning as the mean number of substitutions per site. Variation in rates is achieved by multiplying these mean values by rate modifiers representing rates either faster or slower than the mean.

GRH assumes that the distribution of relative rates of substitution across sites can be described by a gamma distribution with a fixed mean of one. Given a value of the gamma shape parameter, α, the gamma probability distribution is divided equally into *k* categories (often four). The rate multipliers are taken as the means of these categories. The likelihood of each site is then defined as the average likelihood of that site over the rate multipliers. Because GRH requires averaging likelihoods across *k* rates, it requires *k* times more computation than assuming constant rates across sites. The SSRH model differs primarily in the fact that a particular rate modifier is assigned to each site, either by *a priori* specification or by estimating the rate modifier outright. Because each site is only scored for a single rate, use of SSRH does not result in a significant increase in computation. Note that including GRH in a model adds only a single extra parameter, the α shape parameter of the gamma distribution. In contrast, the relative rate of each site becomes a separate parameter under SSRH. Thus, each method has its own advantages, and there is not a clear way of choosing between them. GRH should be preferred on statistical grounds and SSRH on computational grounds.

Another aspect of rate heterogeneity observed in real datasets is presented by sites that are invariable, or unable to change over the tree. These sites may be accommodated in sequence evolution models by adding a class of sites with a rate modifier of zero, and estimating the proportion of sites in that category (Gu et al., 1995). This model parameter may be used alone or in conjunction with GRH, and in either case the total likelihood is calculated as a weighted average of the likelihood values across all rate categories.

Model choice

Often the numerical values and the meaning of evolutionary model parameters are not of particular interest. This does not suggest that they are unimportant, as it has been

shown that ML tree inference is most accurate when an appropriate model is used (e.g., Sullivan and Swofford, 1997, Posada and Crandall, 2001). Choosing a model that is too simple (underfitting) causes the model to be unable to fully capture the process of substitution that generated the data, and can reduce the accuracy of tree inference. Choosing a model that is overly complex (overfitting) may reduce the power of tree inference because some parameters may fit random noise present in the data rather than true phylogenetic signal. With real biological data, any model we choose will inherently be a simplification, and so our task is to choose the best available model, rather than "true" model. The two methods of model choice commonly used by phylogeneticists are the hierarchical likelihood ratio test (Goldman, 1993) and Akaike's information criterion (Akaike, 1973). These methods differ significantly in their philosophy, but it practice generally give similar results.

Calculating the likelihood

I will now briefly discuss the methodology used to calculate the likelihood of a topology T, given a set of branch lengths B and a set of model parameters M. A more detailed treatment is given by Swofford et al. (1996). First, consider the likelihood of a "tree" consisting of a single branch of length t connecting two sequences. M and t are first used to calculate the matrix $\boldsymbol{P}$, which gives the probability of observing each possible pair of bases at opposite ends of the branch. The likelihood of a single site, for example an A at one end and a C at the other end of the branch, may then be obtained directly from $\boldsymbol{P}$. Because sites are assumed to evolve independently, the likelihood of the two sequences is simply the product of the likelihoods of each site.

Now consider the calculation of the likelihood for a single nucleotide site over a tree consisting of greater than two taxa. The site likelihood is then equal to the sum of the likelihoods of the observed data across all possible combinations of states at internal

nodes. Fortunately, the pruning algorithm of Felsenstein (1981) allows the calculation of the site likelihood without requiring calculation of the likelihoods of each of the $4^{(n-2)}$ possible assignments of unobserved states at the internal nodes. To perform the calculation, the topology is first oriented by choosing an arbitrary internal node as the effective root. The algorithm then requires the calculation of conditional likelihoods at each internal node. The conditional likelihoods at each node represent the likelihood of observing the terminal states within the subtree defined by that node, conditional on a particular unobserved state at that node. The calculation of the conditional likelihoods for a particular internal node depends only on the branches and nodes (internal or terminal) immediately above it. Thus, a post-order traversal of the tree may be used to calculate all of the conditional likelihoods. The conditional likelihood values at the root node represent the likelihood of the entire topology, conditional on each potential state at the root. Finally, the total site likelihood is equal to the sum over root states of the likelihood conditional of each root state times the probability of each root state (the equilibrium base frequencies). When the likelihood of multiple characters is calculated over a topology, conditional likelihoods must be calculated for each site independently. Because sites are assumed to be independent, the total likelihood of a set of sequences may be obtained as the product of the likelihoods of each column of the data matrix, known as the site likelihoods. Because this product of the site likelihoods quickly becomes a very small number, it is typically represented as a log-likelihood, or lnL.

The conditional likelihoods calculated for each internal node are of particular interest because they may be reused in future likelihood calculations. If no change is made to the topology or branch lengths within the subtree on which the conditional likelihoods are based, the previous values remain valid and do not require recalculation. Efficient ML implementations take advantage of this fact by storing the previous

computations and only recalculating those conditional likelihoods that have changed. Note that because the evolutionary model parameters by definition apply to the entire tree, any change to the components of *M* requires the recalculation of all conditional likelihoods.

1.4 MAXIMUM LIKELIHOOD HEURISTICS

Heuristic methodology

Because it is computationally intractable to evaluate more than a small proportion of the possible topologies for a dataset of any size, the only feasible approach is to apply heuristics**.** Typical ML branch-swapping heuristics, as typified by PAUP* (Swofford, 2002) and fastDNAml (Olsen et. al, 1994), search by obtaining a starting topology and then making rearrangements to it in an effort to find more optimal topologies. One common way of generating the starting topology is by the stepwise-addition method (see Swofford et al., 1996), which sequentially adds one taxon at a time to a growing tree. All possible attachment points of each successive taxon are attempted, and each is added at the location that results in the optimal score. When searching begins, each topology generated by a rearrangement is evaluated, and if a better scoring topology is encountered it becomes the topology to which further rearrangements will be made. This type of heuristic approach is described as "greedy" (because once a choice is made it cannot be reversed) and "hill-climbing" (because only new topologies that improve the score are considered).

Heuristic methodology depends on the existence of a correlation between the score of a particular topology and the score of other topologies that are "near" it in some sense. Most simply, we may judge the closeness of two solutions in the search space by the number of rearrangements necessary to convert between them. Thus, the geography

of the search space (generally termed "tree-space" in the phylogenetic context) may be defined by the method used to move through the space. Phylogenetic heuristics most commonly terminate when a topology is reached from which no rearrangement will create a better scoring solution. Such a position is an optimum in tree-space, although it may not be the global optimum that is the ultimate goal of the search. The presence of local topological optima is an unfortunate but not uncommon feature that complicates phylogenetic searches. Multiple optima have been studied primarily in the context of parsimony methods (e.g., Maddison, 1991), but exist for the ML criterion as well. Note that for heuristic methods that apply rearrangements in a deterministic and ordered fashion, the identity of the starting topology uniquely determines the optimum at which the search will terminate if multiple optima exist. Typically such searches are run multiple times from different starting topologies to increase the probability that the global optimum is found.

Several standard topological rearrangement types are used in phylogenetic heuristics (see Swofford et al., 1996 for more details). These are termed Nearest Neighbor Interchange (NNI), Subtree Pruning and Regrafting (SPR) and Tree Bisection and Reconnection (TBR). The three rearrangement types are nested, such that the NNI rearrangements are a subset of the SPR rearrangements, which are in turn a subset of the TBR rearrangements. The number of unique NNI, SPR and TBR rearrangements possible from any bifurcating unrooted topology containing n taxa are $O(n)$, $O(n^2)$ and $O(n^3)$, respectively (Allen and Steel, 2001).

Enhanced branch-swapping heuristics

Typical ML branch-swapping heuristics calculate the maximized likelihood of each topology examined by performing full branch-length optimization. Optimization of model parameters may also optionally be performed. Some programs (e.g., PAUP*) only

perform this full optimization step if an initial rough estimate of the likelihood is within a specified threshold of the likelihood of the current topology. Regardless, this branch-length optimization step is the limiting factor in the number of topologies that may be evaluated. This fact has been recognized by the developers of two recently released ML heuristic search algorithms, PHYML (Guindon and Gasquel, 2003) and RAxML (Stamatakis et. al, 2005). Both programs implement enhancements that reduce the burden of branch-length optimization in the evaluation of new topologies. Although these programs are not widely used, they are currently the best available ML heuristics. A more detailed examination of the details of these algorithms will prove instructive.

The PHYML algorithm begins by constructing a starting tree by the fast distance-based BIONJ method (Gasquel, 1997) and estimating model parameters on that tree. It then calculates the lnL scores obtainable by applying each possible NNI rearrangement to the topology. However, instead of fully optimizing all branches of each of the topologies generated by an NNI, PHYML only finds the optimal length for the one new bipartition created by each swap. The algorithm then ranks the increases in lnL attainable by all of the rearrangements, and applies a proportion of those that increase the likelihood over that of the starting tree. The resulting topology is then used as the starting point for the next round of the algorithm, and model parameters are estimated again.

The PHYML algorithm has several features of note. First, only very localized rearrangements and branch-length optimizations are performed. Second, because score improvements due to rearrangements in different parts of the tree are assumed to be largely independent and additive, multiple rearrangements may be applied simultaneously. Third, because both the BIONJ method of obtaining the starting topology and the swapping algorithm are completely deterministic, PHYML will always return the same topology for a particular dataset unless the user provides a different starting tree.

The RAxML program is derived from fastDNAml (Olsen et al., 1994), but makes important algorithmic enhancements. It first creates a starting tree using stepwise addition under the parsimony criterion, and roughly optimizes model parameter values. SPR branch-swapping is then performed, with the constraint that the branches to which a pruned subtree may be reattached are limited by their distance from the branch where the subtree was initially pruned. All allowable SPR rearrangements from the current topology are attempted, and the lnL scores of the resulting topologies are calculated after only applying optimization to the three branches directly affected by the rearrangement. The 20 best scoring topologies resulting from the rearrangements are then subjected to full branch-length optimization, and the most optimal is accepted as the next topology. Model parameters are then reestimated and the entire procedure is repeated. Note that unlike the PHYML algorithm, the ability to use random taxon-addition sequences to obtain the stepwise addition starting tree allows RAxML to return different topologies if run multiple times, even though the swapping algorithm is deterministic. Like PHYML, RAxML localizes both rearrangements and branch-length optimization.

Stochastic heuristics

The ML heuristics discussed thus far are deterministic in their application of branch swapping, and only attempt each rearrangement once. Another class of heuristic methods incorporates stochasticity in the search strategy. This class includes simulated annealing approaches and evolutionary algorithms. These methods apply rearrangements in a random fashion, meaning that both the path of the search through tree space and the final solution obtained can vary even when the search is started from the same topology. This property could be viewed as either a benefit or drawback. Regardless, the intent of introducing stocasticity into the search is not to cause this lack of repeatability, but to exploit other theoretical benefits.

Simulated annealing algorithms are similar in nature to the hill-climbing algorithms discussed thus far, with the important distinction that they are able to accept rearrangements that decrease the likelihood score. Although it is somewhat counter-intuitive that accepting topological changes that reduce the likelihood should be desirable, it is only by allowing such movements that a local optimum can be escaped. The probability of accepting a topology that worsens the score depends both on the difference in lnL between the current and proposed states and on a control parameter. This control parameter is slowly changed over the course of a search, reducing the probability of accepting rearrangements that worsen the likelihood. It has been shown (although not specifically for phylogenetic problems) that if the control parameter is changed at the optimal rate the simulated annealing algorithm is guaranteed to return the optimal solution (Lundy and Mees, 1986). Simulated annealing algorithms for ML phylogenetic searches have been implemented by Salter and Pearl (2001) and Stamatakis (2005).

Genetic algorithms

Genetic algorithms (GAs) belong to a class of heuristic methods known collectively as evolutionary computation. The other general categories of evolutionary computation are known as evolutionary strategies and evolutionary programming. The distinctions between these techniques are rather subtle, and unimportant here. All depend on the concept of evolving a more optimal solution to a problem through the repeated application of mutation, selection and reproduction. The primary differences between them lie in the selection scheme and in whether a crossover or recombination technique is used to create new solutions by combining existing ones.

Unlike the phylogenetic search algorithms discussed thus far, the state of a genetic algorithm at any point in time is represented not by a single solution in the search

space, but rather by a group of individual solutions. The individual solutions will be referred to simply as individuals (also commonly called "chromosomes"), with the collection of individuals existing at the same point in time making up a population. Typical population sizes range from 50 to 1000 (Mitchell, 1996). The unit of time over which one round of mutation, selection and reproduction occur is termed a generation. Selection is applied to the population through a selection function that relates the optimality of each individual to its probability of reproduction. Thus, the individuals in the population compete to be the parents of future generations of individuals. Selection functions may be divided into two main types, termed rank-based and fitness-proportional. The definitions of these are intuitive, with the first basing the probability of reproduction solely on the rank of the individual's score in the population and the second taking into account the magnitude of the differences in scores.

Functions acting on the individual solutions are termed "operators". Mutation operators alter some portion of the individual solutions, changing their score. The first GAs (Holland, 1975) encoded their solutions as simple binary strings, and mutation operators simply changed individual bits in the strings. Unfortunately, such a tractable representation has not been developed for phylogenetic trees, making the implementation of mutation operators more complicated. The standard rearrangement types discussed previously function as reasonable topological mutation operators. However, the best way of obtaining branch-length values after a rearrangement is not obvious, and presents one of the largest challenges in the implementation of a GA for ML tree inference. If model parameters are also encoded as part of the individual, their continuous nature allows for easy implementation of mutation operators. An important part of most traditional GA implementations is the use of a recombination or "crossover" operator that can be used to bring together highly fit portions of two separate individuals into a single solution. The

optimal implementation of recombination operators in the context of a phylogenetic GA is also not obvious.

GAs share with simulated annealing algorithms the ability to escape from local optima by moving through valleys of lower fitness in the search space. The use of recombination operators to bring together portions of multiple solutions also allows for "jumping" through the search space and escaping local optima. Typical GAs seek to maintain a large amount of variation in the solutions present in the population over the course of a search, and expect to lose this variation as the population eventually approaches the global optimum. This process is termed convergence, and loss of variation in the population before reaching the global optimum is known as premature convergence. The rate of convergence is dependent primarily on the strength of selection and the population size.

Genetic algorithms for phylogenetic inference

The most important details of any GA implementation are the selection function and the implementations of the mutation and recombination operators. A number of researchers have developed GAs to search the space of phylogenetic trees, under both the parsimony (Moilanen, 1999; Congdon, 2001) and ML criteria (Matsuda, 1996; Lewis, 1998; Lemmon and Milinkovitch, 2002; Brauer et al., 2003). I will focus on the most recently developed ML algorithms.

The GAML program (Lewis, 1998) and its parallel implementation (Brauer et al, 2003) served as the progenitors to GARLI, both in terms of their general algorithm structure and in their code base. The code for GAML has been almost entirely rewritten, and significant differences in the mutation types, implemented models and selection process warrant its new name. GAML used a rank-based fitness function, and implemented only the HKY model of sequence evolution (Hasegawa, Kishino and Yano,

1985), with empirical base-frequency estimates and gamma-distributed rate heterogeneity. SPR rearrangements were the sole topological mutation operator, and after a rearrangement branch-length values maintained their previous values. Model parameters and branch lengths were mutated by multiplying them by a gamma-distributed random variable with mean one. GAML implemented a recombination operator in which a subtree is first randomly chosen in one individual. A second individual is then randomly chosen, and the taxa present in the chosen subtree are removed from the second individual. The chosen subtree is then grafted onto the second individual at a random branch. The parallel implementation of GAML applied an identical algorithm, but distributed the task of calculating the likelihood of topologies generated by the mutation and recombination operators across multiple processors. It has been noted that the fact that GAML performed no branch length optimization following rearrangements almost certainly hampered its effectiveness (Brauer et al., 2003).

The most recently developed ML genetic algorithm is MetaPIGA (Lemmon and Milinkovitch, 2002). This sophisticated Java program implements a number of interesting techniques. Foremost among these is the simultaneous evolution of a number of partially independent "meta-populations" of individuals. Such approaches have been well studied in the GA literature (see Cantu-Paz, 2001), and can be very effective in increasing search efficiency by exploring a larger area of the search space and proving a diversity of solutions for recombination operators to act upon. MetaPIGA takes special advantage of meta-populations in its novel "consensus pruning" technique. The idea of consensus pruning is that topological features (bipartitions) found by multiple independent populations are well supported and will appear in the final optimal topology. MetaPIGA constrains such bipartitions so that no topological mutations are considered that would remove them from the topology. This effectively breaks the tree into much smaller

portions that can be searched within, drastically reducing the search space. Another novel feature of MetaPIGA is its treatment of branch lengths. Terminal branches are fixed throughout the search at values derived from a simple distance-based method, and branch-length mutations may only affect internal branches. Some unfortunate characteristics of the MetaPIGA software implementation are its fairly extreme memory requirements and its inability to run in a noninteractive or batch mode. These factors make investigation of its effectiveness in the analysis of large datasets difficult.

Chapter 2: The Serial GARLI Algorithm

2.1 INTRODUCTION

GARLI (Genetic Algorithm for Rapid Likelihood Inference) is a heuristic search algorithm for performing phylogenetic inference under the maximum likelihood (ML) criterion. The algorithm was developed with the goals of increasing both the speed of ML inference and the size of the datasets that could reasonably be analyzed. This is achieved primarily through algorithmic enhancements that allow effective ML topology searching while performing only a small fraction of the numerical optimization required by older software.

I begin with a general overview of the GARLI algorithm. GARLI is derived in part from GAML (Lewis, 1998), and as such special attention will be paid to distinctions between the two algorithms. A general outline of the algorithm appears in section 2.1, and more detailed discussions of particular aspects follow in sections 2.2 through 2.6.

2.2 ALGORITHM OVERVIEW

Terminology

Given a set of aligned nucleotide sequences, the GARLI algorithm seeks to evolve high quality solutions representing the evolutionary relationships between those sequences. As is typical of genetic algorithm (GA) approaches, this is done through the evolution of a *population* of solutions, each termed an *individual*. Each individual encodes a topology representing a set of potential evolutionary relationships, a corresponding set of branch-length parameters for that topology and a set of parameters representing the model of sequence evolution. The sequence evolution model implemented is the general time-reversible model (GTR: Lanave et al., 1984), with

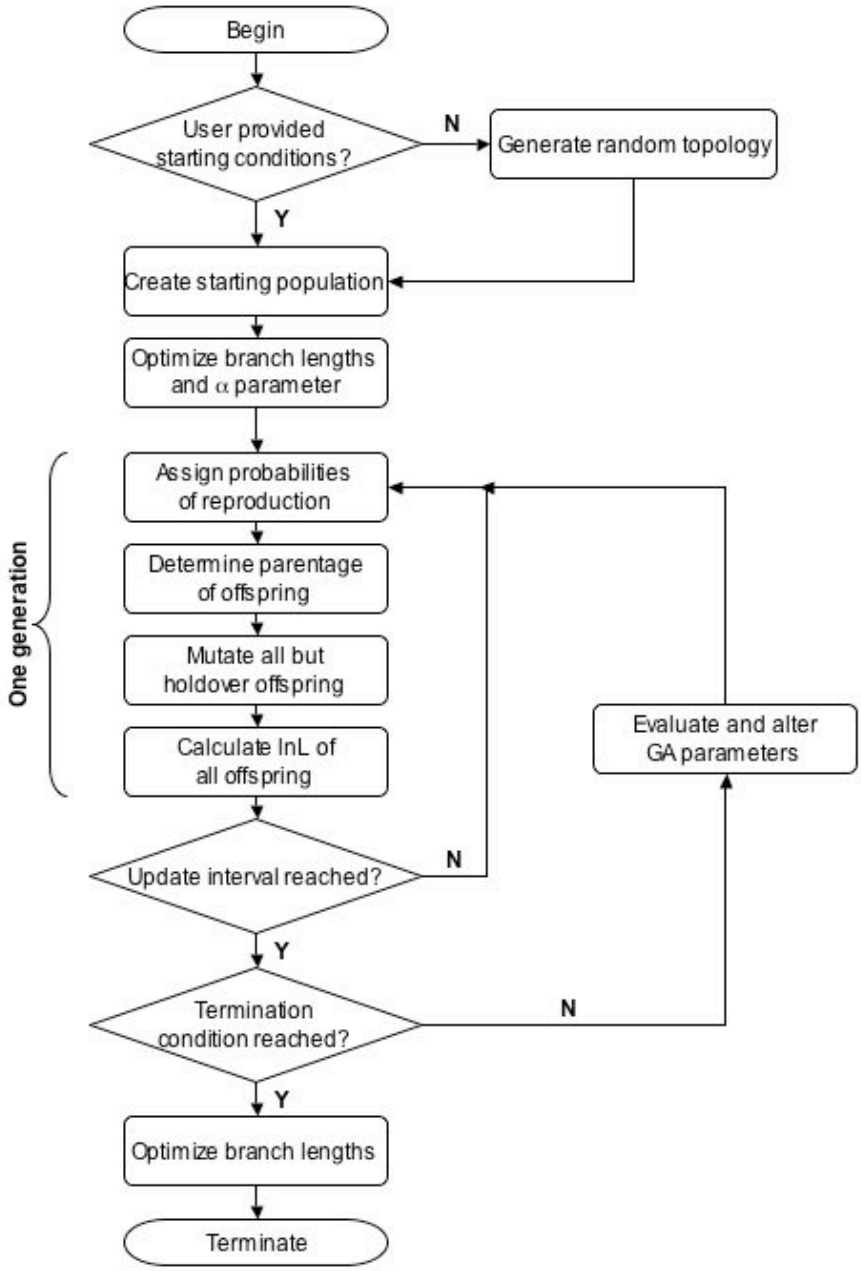

Figure 2-1. A schematic view of the GARLI algorithm.

discrete gamma-distributed rate heterogeneity (Yang, 1994) and an estimated proportion of invariable sites (Gu et al., 1995). The *fitness* of each individual is a function of its log-likelihood score (lnL) on the given dataset.

The *starting conditions* of the algorithm consist of the initial population of solutions, and may be user-specified or generated randomly. I will use the term *GA parameters* to refer to all variables that control the details of the genetic algorithm, including aspects of the selection process, mutation operator details and mutation operator probabilities. The most important GA parameter is P_b, which controls the extent of deterministic branch-length optimization that follows topological mutations. Two important timescales exist: the *generation*, which consists of a single round of replication, mutation and scoring of individuals, and the *update interval,* which consists of one hundred generations. At the end of each update interval certain GA parameters are evaluated and may be adaptively altered. Individuals of the previous generation are referred to as the *parent* individuals (although not all will necessarily leave offspring), and those of the current generation are referred to as the *offspring* individuals. The offspring of the current generation become the parents of the following generation. The *stopping criterion* refers to the conditions under which the search will terminate. An entire execution of the algorithm from start to termination is termed a *run*. A high-level schematic of the algorithm appears in figure 2-1.

Starting conditions

The algorithm begins by creating a population of identical individuals using either a user-specified or random topology. Starting branch lengths and model parameters may also be user-specified. When a starting topology is not specified, one is generated equiprobably, and all branch-lengths are set to an arbitrary length of 0.05. If a topology is specified without branch-lengths, the length of all branches is again set to 0.05. If model

parameter values are not specified, the algorithm infers reasonable estimates from the data for the equilibrium base frequency parameters and the proportion of invariant sites, and uses arbitrary default values for the GTR relative rate parameters and the α parameter of the rate heterogeneity distribution. Specifically, the observed base frequencies are used as the starting values of the equilibrium base frequencies parameters, and the proportion of invariable sites is set to 1/4 of the observed proportion of invariant sites. The GTR relative rate parameters are set to the equivalent of a transition:transversion rate ratio of 4:1 (i.e., a=c=d=f=1.0 and b=e=4.0) and the α parameter of the gamma rate heterogeneity distribution is set to 0.5.

Before beginning stochastic optimization through the GA, a phase of rough deterministic optimization is performed on the branch lengths and the α parameter of the starting individuals. Strong interactions exist between the shape of the rate heterogeneity distribution and branch-length parameters. Initial experiments suggest that when these parameters start far from their optimal values, stochastic mutations performed by the GA have a difficult time moving them toward globally optimal values, and may sometimes result in entrapment in local parameter optima. This initial optimization step does not attempt to bring these parameters to their optimal values, but rather to bring them close enough that stochastic mutations may proceed to optimize them. This step is also important for GARLI's branch-length optimization algorithm, which will be discussed in section 2-5.

Note that in contrast to deterministic topological search algorithms such as PAUP* (Swofford, 2002), RAxML (Stamatakis et al., 2005) or PHYML (Guindon and Gascuel, 2003), the use of a particular starting condition in GARLI does not uniquely determine the final solution. This lack of determinism stems from the algorithm's stochastic search strategy, and should not be viewed as a negative aspect.

The generation cycle

Nearly the entire runtime of the algorithm is spent in the generation cycle. Generations are non-overlapping, such that there is full turnover of all individuals each generation. The number of individuals in the population, *N*, is constant over time. Within each generation the following steps are performed:

1. The fitness of each individual is calculated from the relative lnL scores of the population using a fitness function. The details of the fitness function are discussed in section 2.2.
2. The parent with the highest fitness is automatically chosen to leave *k* offspring, termed the "holdover" individuals. In GA terms this is known as an elitist strategy (Mitchell, 1996).
3. The parents of each of the remaining *N-k* offspring are chosen randomly, in proportion to the parent's fitnesses. At this point the offspring are exact copies of their parents.
4. Each offspring besides one holdover individual is subjected to one of the mutation operators, chosen randomly. Retaining an unmutated copy of the best individual ensures that it is never lost due to chance (i.e., genetic drift).
5. The lnL score of each mutated offspring is calculated.

Typical runs consist of many thousands or tens of thousands of generations.

The update interval

Each update interval is a period of one hundred generations. During each update interval information is tabulated on beneficial topological changes and the performance of each mutation operator. At the end of each update interval this stored information is evaluated and various GA parameters may be altered in an attempt to favor those

operators that have proved beneficial over recent generations. This procedure will be discussed in detail in section 2.4.

Stopping criterion

Both the serial and parallel GAML implementations (Lewis, 1998; Brauer et al., 2003) left open the issue of the most appropriate conditions under which to terminate a GA run. Runs were terminated when a pre-specified number of generations or runtime limit has been reached. Deterministic branch-swapping algorithms such as those implemented by PAUP* and RAxML typically terminate when all allowable swaps have been attempted without encountering a better scoring topology. In the GA context, there is no point at which the algorithm "runs out" of allowable branch swaps. Because successive mutations may interact synergistically (non-additively) and allow escape from local topological optima, the fact that an attempted rearrangement was not beneficial once does not necessarily suggest that it should be disallowed in the future.

Although the specification of an automated stopping criterion for a GA is not as clear-cut as it is for some other search strategies, reasonable criteria do exist. Because the primary aim of the search is an estimate of the topology, a lack of topological change in the best scoring individual over a large number of generations is one obvious component. However, because topology estimation is intimately tied to model and branch-length parameter estimation, some assurance that these parameters are near their optimal values for the current topology is also desirable. GARLI implements an automated stopping criterion which requires three conditions to be met. First, no topological mutations may have created a significantly higher fitness individual (lnL improvement of greater than 0.05) over a specified number of generations. Second, the total increase in score over a number of update intervals must be less than a specified amount. Third, the GA parameter controlling the degree of branch-length optimization (P_b, discussed in section

2.5) must have reached its minimum value. In addition to this automated stopping criterion, GARLI may also be instructed to terminate after a specified number of generations or runtime has passed. This deterministic stopping criterion supersedes the automated stopping criterion.

Note that if the algorithm is allowed to continue running after the particular stopping criterion has been reached, it is likely that better solutions will eventually be found (unless the search has located the global optimum). As such, reaching the stopping criterion should not be viewed as a guarantee that the global optimum has been reached, but rather as a suggestion that the run has reached a point in the search space from which improvement is difficult. Allowing the run to continue longer may eventually result in further improvement, but at some point computational resources may be better spent on performing additional independent runs from the same or different starting conditions.

2.3 THE FITNESS FUNCTION

The procedure used to determine an individual's probability of being selected as a parent from its lnL score is termed the fitness function. The choice of a particular fitness function is an important consideration for any genetic algorithm. Many methods exist in the GA literature, but the primary distinctions are between rank-based and fitness-proportionate methods. In effect, the choice of fitness function determines how "greedy" the search algorithm is. Highly biased fitness-proportionate functions strongly favor the best individual and result in aggressive hill-climbing searches. Rank-based functions may be thought of as more conservative, spreading the probability of reproducing across individuals regardless of their exact scores.

The GAML algorithm applied a fully rank-based fitness function. In contrast, GARLI uses a tunable fitness-proportionate method. This change was motivated by an examination of the distribution of lnL changes caused by topological rearrangements.

Once a reasonable topology is found, the vast majority of rearrangements tend to very drastically worsen the likelihood. At such a point, rearrangements that are able to increase the likelihood are quite scarce, and only result in very meager score increases. The application of a rank-based fitness function in this situation gives individuals that have just experienced a mutation causing a large decrease in lnL a good chance of reproducing, despite the fact that the probability of them experiencing a compensatory beneficial mutation is quite small. GARLI chooses to strongly favor the current best individual at the potential cost of intra-population variability.

The probability of each individual of the previous generation being selected as the parent of each offspring individual is a simple function of the parents' relative lnL scores and the user-defined selection intensity, *s*. The procedure is as follows:

1. Given the set of lnL scores of all *N* individuals, $F=\{lnL_1, lnL_2, \ldots lnL_N\}$, calculate $lnL_{max} = \max(F)$
2. Calculate the relative lnL score of each individual, $\Delta lnL_i = lnL_i - lnL_{max}$
3. Calculate the fitness of each individual, $f_i=\exp(s\Delta lnL_i)$
4. Finally, the probability of being selected as a parent is proportional to the individuals' fitnesses:

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

The selection intensity parameter *s* controls the strength of selection, the sensitivity of the fitnesses to the magnitude of the differences in lnL. Large and small values of *s* represent strong and weak selection, respectively. Table 2-1 displays the relative probabilities of

reproduction for two individuals differing by a score of 1.0 lnL over a range of reasonable values of *s*.

Selection intensity (*s*)	Ratio of reproduction probabilities
0.05	0.95:1.0
0.10	0.90:1.0
0.25	0.78:1.0
0.50	0.61:1.0
0.75	0.47:1.0
1.00	0.37:1.0
2.00	0.14:1.0

Table 2-1. Relative reproduction probabilities of two individuals differing by 1.0 lnL unit for various values of the selection intensity, *s*.

The GARLI fitness function is a generalization of the procedure used to calculate Akaike weights (see Burnham and Anderson, 2002, p. 75), and is equivalent to it when *s* is equal to 0.5. The Akaike weights procedure is typically used to calculate the evidence in favor of a number of competing models based on their relative lnL scores and the number of parameters estimated by each. The procedure may used in model averaging approaches to obtain an estimate of a parameter across models by weighting the parameter estimates under each model by its Akaike weight. Note that technically the Akaike weights procedure requires that the likelihoods for each model be compared with all free parameters at their maximum-likelihood estimates. As it is used here, the contributions of the current values of the model and branch-length parameters to an individual's likelihood are also incorporated in the calculation of its fitness. To my knowledge the use of an Akaike weights type procedure in determining the probabilities of reproduction in a GA is novel, but seems reasonable and flexible.

2.4 MUTATION OPERATORS

The mutation operators implemented in GARLI are of four general types: topological mutation, evolutionary model parameter mutation, branch-length parameter mutation and topological recombination. The mutation operators are mutually exclusive, such that each mutated individual experiences only one per generation. Each operator will be described in turn, with the exception of topological recombinations. Due to the very limited amount of variation maintained in a single GA population with small population sizes (Brauer et al., 2003), recombination is expected to be of little benefit, and is not utilized in the serial GARLI algorithm. However, in the parallelized multiple-population implementation, topological recombination is utilized and will be discussed in detail in chapter 4.

Topological mutations

Choosing an effective distribution of topological mutations for use in a phylogenetic GA is an important issue. It is intuitive that all possible topological rearrangements are not equally likely to cause increases in lnL score. Once a reasonable topology has been obtained, small adjustments are more likely to result in new topologies with high fitness than are very radical adjustments. This argument is akin to Fisher's Geometric Model of Adaptation (1930), which predicts that as a population nears a fitness peak, mutations of smaller effect are necessary to allow further progress toward the peak. Although mutations of small effect are important in this context, they are also inherently less likely to allow the discovery of other disjoint fitness peaks. For these reasons, a stochastic search strategy must seek a balance between rearrangements that fine-tune a topology and those that allow a more extensive exploration of treespace.

To further explore the issue of an optimal distribution of topological mutation effects, we must first choose some measure of the effect size of a topological mutation.

One such measure is the Robinson-Foulds distance (RF: Robinson and Foulds, 1981) between the original topology and the topology resulting from the rearrangement. In this context the RF distance is equal to twice the number of bipartitions present in the original tree that are no longer present in the topology after the rearrangement.

For simplicity, the following discussion will consider only the class of SPR rearrangements, implicitly including NNI rearrangements that are a special case. For SPR rearrangements there is a simple relationship between the details of the rearrangement and the RF distance induced by it. Define the *reconnection distance* of a rearrangement as the number of nodes that must be crossed on the path between the point at which a subtree was pruned for an SPR rearrangement and the branch to which it is reattached (see Figure 2-2). The induced RF distance is equal to twice the reconnection distance. NNI rearrangements are simply those with a reconnection distance of one. The minimum and maximum RF distances between two topologies for a dataset with *T* taxa are two and 2(T-3). However, except in the case of topologies that have very peculiar shapes (i.e., are completely pectinate), the maximum distance that may be induced by an SPR rearrangement is significantly less than this maximum.

If we accept the RF distance as a reasonable measure of the effect size of topological rearrangements, examining the distribution of RF distances across all possible SPR rearrangements for a particular topology may be instructive. The exact distribution of reconnection distances will vary depending on the number of taxa and the tree shape, but some generalizations may be made. Consider the topology of Figure 2-3a, the ML tree for 64 frog taxa published by Hillis and Wilcox (1995). The distribution of reconnection distances for all possible SPR swaps on this topology appear in Figure 2-3b. The reconnection distances of the 14,792 possible SPR rearrangements range from one to seventeen, with a mean of approximately 7.4. This suggests that although the spectrum of

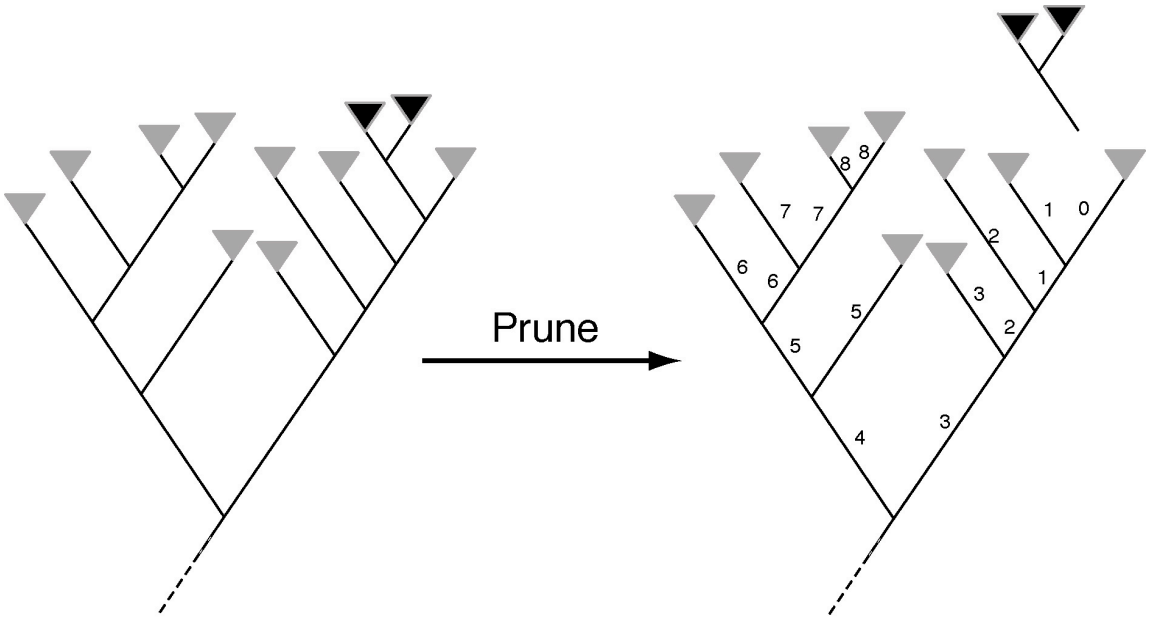

Figure 2-2. The reconnection distance. After pruning at the branch ancestral to the two black clades, each potential reattachment branch is assigned a reconnection distance. The distance to each branch is equal to the number of nodes that must be crossed on the path between that branch and the branch to which the subtree was originally connected.

SPR mutations includes some of small effect, a rearrangement chosen at random is likely to be relatively radical. Further, as the number of taxa in the topology increases, the proportion of swaps with small reconnection distances will decrease yet more. For this fairly small topology the proportion of swaps of smallest effect (NNI rearrangements) is less than one percent, and this proportion will become vanishingly small as total taxon numbers grow into the hundreds and thousands. This fact is not problematic when rearrangements are chosen deterministically and exhaustively, because all allowed rearrangements are attempted in turn. However, in the context of a stochastic search strategy steps must be taken to ensure that sufficient mutations of small effect are included in the mutational spectrum regardless of the number of taxa in the analysis.

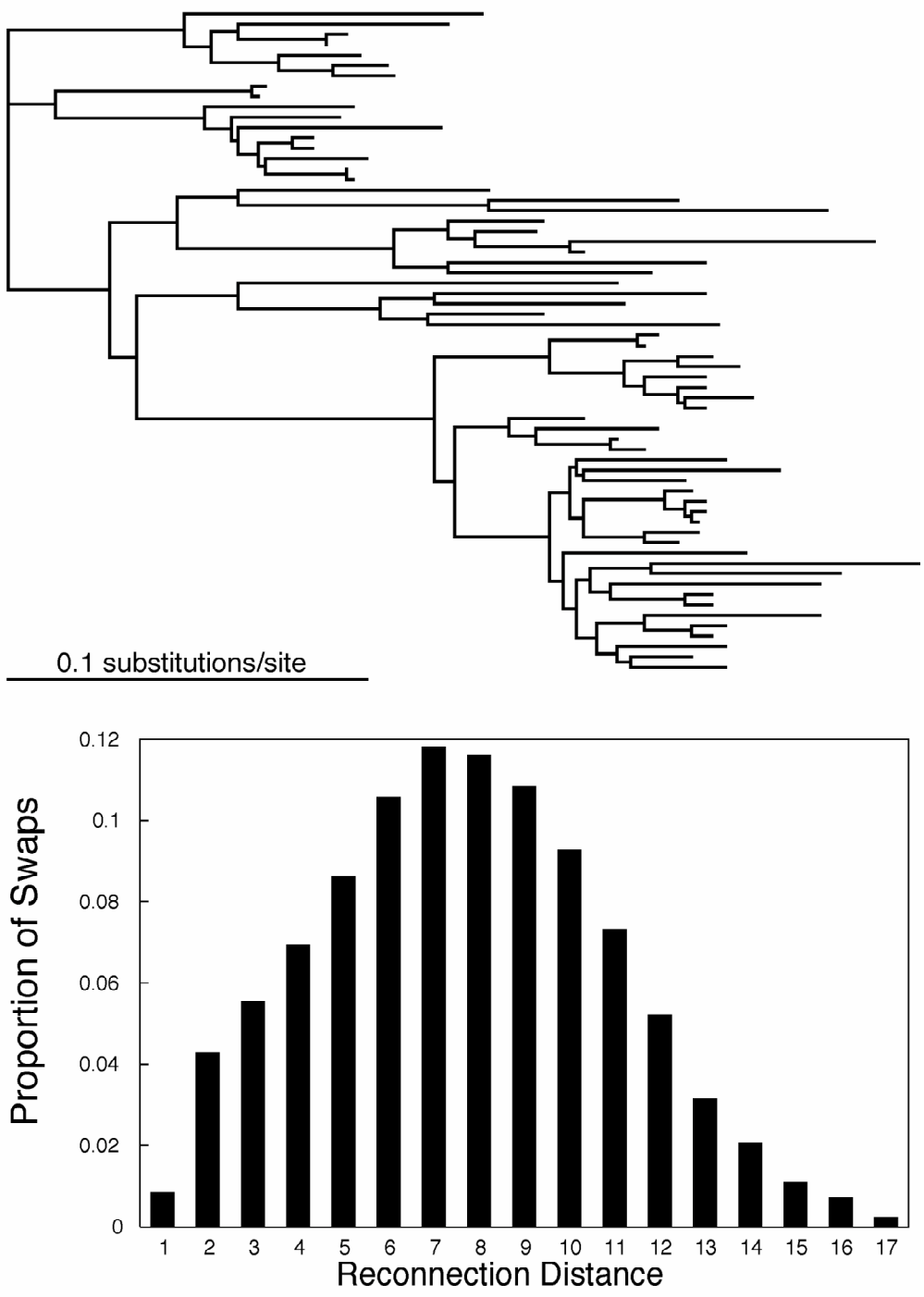

Figure 2-3. Reconnection distances on a real topology. (a.) The ML topology for the 64 taxon dataset of Hillis and Wilcox (2005). (b.) The distribution of SPR reconnection distances on the topology of (a).

GARLI allows flexible control of the distribution of topological rearrangements by implementing three distinct rearrangement operators. The first is the standard NNI rearrangement, which provides minor topological changes. The second is a localized form of SPR, referred to here as limSPR. This operator only allows reattachment of the pruned subtree to branches with reconnection distances of less than a specified value. All branches with distances less than the specified value have equal probability of being chosen as the reconnection point, after branches with a reconnection distance of one have been removed from consideration (rearrangements equivalent to NNIs). Similar localization of rearrangements is also available in other software such as PAUP* and RAxML. The third operator performs a standard SPR rearrangement in which the reattachment branch is chosen equiprobably from all branches of the tree.

The probability of applying each of the three operators during a run may be specified individually, allowing simple control of the proportion of rearrangements that cause minor topological changes (NNIs), moderate topological changes (limSPRs) and more radical topological changes (SPRs). In addition, the proportions of the three operators may be adaptively altered over the course of a run, allowing the most beneficial operator to be favored at any point in time. The adaptation procedure will be discussed later in this section.

Model parameter mutations

When a model parameter mutation occurs, the parameter to be altered is chosen with equal probability from the 13 parameters described below. Mutation operators affecting the evolutionary model parameters simply multiply the current parameter value by a number drawn from a gamma distribution with mean 1.0 and a specified shape parameter. The model parameters of the GTR+I+Γ model consist of six relative rates, four equilibrium base frequencies, the proportion of invariable sites and the α shape

parameter of the gamma distribution of rate heterogeneity. Some parameters have restrictions such as the need to sum to one (the equilibrium base frequencies) or to maintain indentifiability (the f rate of the GTR model is fixed at 1.0). When the value of one of these parameters is altered, rescaling occurs such that the appropriate restrictions are maintained.

In addition to operators affecting the aforementioned 12 parameters, one other important parameter operator is implemented. This operator effectively mutates the mean rate of substitution (μ), and in some senses might also be considered a branch-length operator. Because μ is subsumed into the definition of the branch lengths (i.e., branch length = μ x length of time represented by the branch), it is not directly identifiable in ML phylogenetic inference. However, it may be effectively mutated simply by multiplying all branch lengths of a topology by a single gamma-distributed variable. Allowing this operator in a stochastic search is important because of strong correlations between optimal branch lengths and optimal values of rate-heterogeneity parameters. Mutations to the α parameter or the proportion of invariable sites cause the optimal lengths of all branches to increase or decrease by nearly the same proportion. In the absence of a μ parameter operator, many successive random branch length mutations would be necessary to bring the branch lengths back into line because of the lack of deterministic parameter optimization. A similar operator is implemented as a proposal type in MrBayes (Huelsenbeck and Ronquist, 2001) for similar reasons.

Branch-length parameter mutations

As with model parameter mutations, branch-length mutations involve multiplying a branch length by a gamma-distributed variable with mean 1.0 and a specified shape. Branch-length mutations differ from model parameter mutations in that multiple branch lengths may be altered simultaneously. When a branch-length mutation occurs, the

number of branches affected is drawn from a binomial distribution with specified mean M. This number of branches is then randomly selected from the topology and each is altered with an independently drawn multiplier. Note that drawing the number of branches to be altered from a binomial distribution with mean M is equivalent to specifying a per-branch mutation probability of $M/(T\text{-}3)$. I find the binomial mean more intuitive because the expected number of branches to be mutated does not depend on the number of taxa in the analysis.

Computational costs of mutation operators

Nearly all of GARLI's computational effort is spent in calculating the lnL scores of altered individuals, as is typical of ML search algorithms. Assuming that sufficient memory is available to store previous computations, the amount of computation necessary to score a mutated individual is proportional to the number of internal nodes at which the previous computations have changed and must be recalculated. The mutation operators differ in the number of nodes that they require to be recalculated, and therefore also differ strongly in the amount of computation necessary to score the mutated individuals. Operators altering any of the model parameters are particularly costly computationally. A change in any model parameter alters the matrix of instantaneous rates of nucleotide substitution, requiring the recalculation of all computations for the entire topology. In contrast, topological and branch-length mutations tend to require far fewer recalculations because many of the previous computations remain valid and may be reused. It is important to note that the amount of recalculation necessary following model parameter mutations grows more quickly with increasing taxon numbers than does the recalculation necessary for the other mutation types. This suggests that as the number of taxa increases, the computational cost of optimizing the model parameters stochastically through the GA may at some point outweigh the benefit. The implications of fixing

model parameters at reasonable estimates and not allowing stochastic optimization will be investigated in chapter 3.

Adaptive tuning of mutation operator probabilities

GAML required that the probabilities of the mutation operators be fixed at the outset of a run. However, it is likely that the most advantageous mutation types will vary dataset to dataset, as well as during the course of a run on a single dataset. For this reason, GARLI implements a mechanism that allows the relative probabilities of some of the operators to be adaptively biased toward those that have been the most beneficial over recent generations.

Each operator has what I will term an *operator weight* calculated for it, with the probability of applying each operator proportional to these weights. The operator weights are dependent on two terms: the user-specified *prior weights* and the calculated *performance weights*. The prior weights determine the expected proportions of the operators and the performance weights summarize the benefit of each operator over recent generations. The operator weights are recalculated at the end of each update interval. The same procedure is used to determine the proportions of the three primary operator types (topological, model parameter and branch length), as well as the proportion of the specific operators within the class of topological mutations (NNI, SPR and limSPR).

The prior weights (w_i, $i \in \{t, m, b\}$ for topology, model parameter and branch-length mutations, respectively) are set before the start of a run. The performance weights for each operator type are calculated by summing any increases in the population best score that are due to that type over a number of update intervals (s_i), and dividing by the total number of times that operator was applied (n_i). Note that this is not equivalent to the average change in score due to each mutation type because mutations that worsen the

score are not included in the sum. This distinction is important, because the intent is to capture a measure of the inherent potential for lnL improvement of each mutation type, rather than its average effect on fitness.

More formally, the probability of performing each mutation type is updated at each update interval such that

$$P(i, i \in t, m, b) = \frac{\frac{s_i}{n_i} + w_i}{\sum_{j}^{t,m,b} \left(\frac{s_j}{n_j} + w_j\right)}$$

Thus, the ratios of the prior weights control the expected probabilities of the mutation operators, and their magnitudes control the sensitivity of the probabilities to the observed performance of each operator. Large values of the prior weights cause the observed performance of the operators to be unimportant, making the operator probabilities essentially constant. Very small values of the weights may cause the probabilities to vary significantly over the course of a run. A minimum probability of 0.02 is enforced for each of the mutation types, regardless of the values of the weights.

Allowing adaptive changes in the operator probabilities is beneficial in part because of the different computational costs of the mutation operators. Model-parameter mutations are much more computationally costly that the other mutation types and it is intuitive that they should only be performed when a significant benefit is expected. The fact that optimal model parameter values often do not vary significantly across reasonable topologies further suggests that the prior weight on model mutations may be set quite low, so that their high computational cost only becomes apparent when the parameter values are far from optimal.

An alternative approach to adaptively change mutation probabilities is common in the GA literature. This approach consists of encoding the mutation operator probabilities as part of the individual, and allowing mutations of the mutation probabilities themselves (e.g., Foster, 2001). This should theoretically allow the optimal mutation probabilities (i.e., those that allow the greatest increases in fitness) to evolve naturally in the population. I have investigated this approach, and have not found it to work well in the context of a phylogenetic GA. In practice, allowing the mutation probabilities to be selected as part of the individual resulted in evolution toward a very high probability of branch-length mutations. I believe that this is because the mean lnL change caused by topological and model mutations is very strongly negative, while the mean change caused by branch-length mutations is only slightly negative. Thus, by evolving toward higher frequencies of branch-length mutations, the average lnL score of the population increases, but at the expense of the generally more important mutation types.

2.5 BRANCH-LENGTH OPTIMIZATION

Motivation

ML phylogenetic searches must involve the joint optimization of tree topology, branch lengths, and evolutionary model parameters, despite the fact that researchers are often primarily interested in the topology. Whether or not the values of the continuous parameters are of interest in and of themselves, they are crucial in evaluating the merits of alternative topologies. The facts that optimal model parameter values may vary across topologies and that branches appear and disappear as topologies change are among the most difficult issues in the implementation ML phylogenetic searches.

Accurate model parameter estimates are known to be very important for accurate ML phylogeny estimation (Sullivan and Swofford, 1997). Fortunately, it is often

observed that optimal model parameter values do not vary significantly between reasonable topologies, and conversely that topologies are ranked nearly identically when scored using the same set of reasonable model parameters. These facts contribute to the effectiveness of the "successive approximation" approach to ML phylogeny estimation (see Sullivan et al., 2005), in which model parameter values are estimated on an initial topology, fixed during the course of a topology search, re-estimated on the resulting topology, and fixed for another round of searching. Although the optimization of model parameters on each topology encountered during a search may be reasonably avoided, the optimization of at least some branch-length parameters on newly created topologies cannot be.

To obtain a sense of the importance of branch-length values in the comparison of topologies, consider the relationship between a topology's total lnL score and the length of individual branches within that topology (see Figure 2-4). If even a single branch length is far from its optimal value, the lnL score of the entire topology may be reduced by tens of lnL units. This is clearly an important issue, because it is often necessary to distinguish between topologies whose optimal lnL scores differ by less than one lnL unit during a topology search. Because GARLI stochastically optimizes branch-length parameters through mutation and selection, a topology's branch lengths will become more optimal the longer that topology exists in the population. It is therefore unreasonable to expect a newly created topology to compare favorably to an existing one unless the new topology's branch lengths are also made reasonably optimal. The GAML algorithm performed no branch-length optimization following topological rearrangements, simply leaving existing branches at their previous lengths and assigning newly created branches an arbitrary length. This inability to assign reasonable branch

lengths following a rearrangement was perhaps GAML's greatest shortcoming, as noted by Brauer et al. (2003).

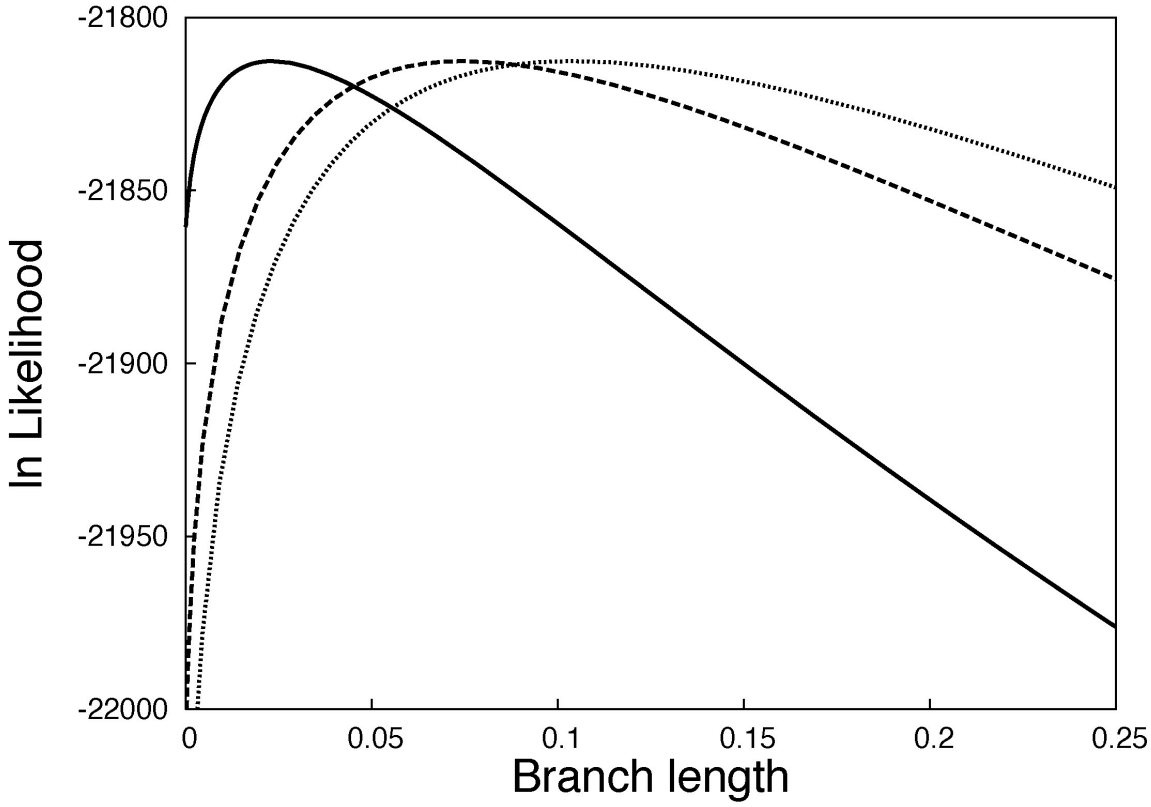

Figure 2-4. The relationship between the length of three arbitrarily chosen branches and the total tree log-likelihood of the ML topology for the 64-taxon dataset of Hillis and Wilcox (1995).

In non-stochastic ML searches, topologies are typically compared to one another on the basis of their maximized likelihood ($\ln \hat{L}$), the highest log-likelihood attainable for each topology across all possible values of the branch-length parameters. Model parameters may also be optimized in the calculation of $\ln \hat{L}$, or it may be calculated conditional on a fixed set of model parameters. The computation of $\ln \hat{L}$ values provides an unambiguous means of comparing the merits of topologies, but comes at a huge

computational cost. Because there is not an analytical solution for the branch-length values that maximize the likelihood, these values must be obtained by one of several computationally intensive numerical optimization techniques such as the Newton-Raphson method or Brent's method (see Press et al., 1992, for details of numerical optimization techniques). All of these techniques guarantee convergence on a parameter optimum, but often require many iterations and only approach the optimum asymptotically.

If we consider the minimum requirements for the implementation of an ML hill-climbing search, it is evident that obtaining $\ln \hat{L}$ values for every topology is not strictly necessary. Only the ability to rank topologies on the basis of $\ln \hat{L}$ values is required. In other words, the algorithm must simply determine whether a newly created topology is more or less optimal than the current topology. Thus, a scoring procedure that performs less optimization but provides reasonable estimates of the $\ln \hat{L}$ score of topologies should be sufficient. More formally, we seek an alternative scoring procedure that returns estimated likelihood scores ($\ln \tilde{L}$) having the following property for all topologies *i* and *j*:

$$\ln \tilde{L}_i > \ln \tilde{L}_j \text{ iff } \ln \hat{L}_i > \ln \hat{L}_j \; \forall \; i, j$$

By definition, the relationship between $\ln \tilde{L}_i$ and $\ln \hat{L}_i$ is

$$\ln \hat{L}_i = \ln \tilde{L}_i + \Delta opt_i$$

where Δopt_i represents the increase in score attainable by performing full branch-length optimization on topology *i*. This value may be thought of as a measure of the optimality

of the entire set of branch lengths of the topology. We desire an accurate estimate of $\Delta \ln \hat{L}_{ij}$, the difference in maximized log-likelihood between topologies *i* and *j*:

$$\Delta \ln \hat{L}_{ij} = \ln \hat{L}_i - \ln \hat{L}_j$$

substituting and rearranging we obtain:

$$\Delta \ln \hat{L}_{ij} = (\ln \tilde{L}_i + \Delta opt_i) - (\ln \tilde{L}_j + \Delta opt_j)$$
$$\Delta \ln \hat{L}_{ij} = (\ln \tilde{L}_i - \ln \tilde{L}_j) + (\Delta opt_i - \Delta opt_j)$$

Thus, the error inherent in estimating $\Delta \ln \hat{L}_{ij}$ using the alternative scoring procedure, *E*, is equal to the term $(\Delta opt_i - \Delta opt_j)$. *E* will clearly will be minimized when $\Delta opt_i = \Delta opt_j$. This suggests that accurate scores may be obtained by devising an optimization algorithm that assures that the set of branch lengths on each topology scored is equally optimal (i.e., Δopt is constant across topologies).

The degree of optimality of a set of branch lengths on a topology *i* must be related to the degree of optimality of each individual branch length *b* within that topology, suggesting the following:

$$\Delta opt_i \propto \sum_{b=1}^{b \in B} \Delta opt_i^b$$

However, assuming such a simple relationship is a gross simplification because it requires that the values of Δopt_i^b for each individual branch are independent of one another, which is not the case. When one branch is optimized, the potential for score improvement due to optimization of every other branch will change, and may either

increase or decrease. Nonetheless, the notion of a per-branch optimality measure is useful in devising an optimization algorithm that results in consistent values of Δopt across topologies. In particular, such a relationship suggests that an equal degree of optimality of the entire set of branch lengths on each topology scored may be attained by making all branch lengths within each topology equally optimal.

Several other observations are also important in understanding GARLI's optimization methodology. First, we may expect that the magnitude of E will be proportional to the amount of numerical optimization of branch lengths that is applied to the topologies. As the number of optimization passes applied in the calculation of $\ln \tilde{L}$ approaches the number required to obtain $\ln \hat{L}$, E must by definition approach zero. Second, note that the magnitude of E that may be tolerated in scoring two topologies while still accurately ranking them is proportional to the difference in maximized likelihood between the topologies. In other words, when one topology is much more optimal than another, highly accurate estimates of scores are not needed to recognize that fact. Finally, if a new topology is generated from an existing one by a topological rearrangement, the two topologies will share a large amount of their branching structure. This suggests that if branch lengths were relatively optimal in the parent individual, most will remain relatively optimal in the new topology and will not require optimization.

Optimization algorithm

I will first describe how a single branch length is optimized, and then detail the functioning of GARLI's optimization algorithm a whole. Individual branch lengths are optimized by the second order Newton-Raphson (NR) method. This method requires analytical calculation of the first and second derivatives of the likelihood function with respect to the branch length of interest (*d1* and *d2*, respectively). Given a branch's current length, *t*, the NR estimate of the optimal value, *t'*, is

$$t' = t - \frac{d1}{d2}$$

Multiple iterations of the NR procedure (with *t'* becoming *t* for the next round) may be needed to reach a point very near the fully optimal branch length, especially if the branch length was initially far from its optimal value. After the derivatives have been calculated for a given branch *b* in topology *i*, an estimate of Δopt_i^b, denoted $\Delta \widetilde{opt}_i^b$, may be calculated via a Taylor series:

$$\Delta \widetilde{opt}_i^b = d1(t'-t) + \frac{d2}{2!}(t'-t)^2 = d1(-\frac{d1}{d2}) + \frac{d2}{2}(-\frac{d1}{d2})^2 = -\frac{d1^2}{2d2}$$

This estimate is of critical importance to the optimization algorithm. Because the aim is to make all branch lengths of the topology equally optimal, the value $\Delta \widetilde{opt}_i^b$ allows the algorithm determine when the optimization of each branch may be terminated. Specifically, optimization passes are performed on each branch until $\Delta \widetilde{opt}_i^b$ is less than a threshold value, defined by the GA parameter P_b. Note that $\Delta \widetilde{opt}_i^b$ is calculated *before* any change to the branch length has taken place, and thus represents an estimate of the score increase that would be obtained by changing the branch length from its current length *t* to the NR estimate *t'*. This means that if the value of $\Delta \widetilde{opt}_i^b$ calculated on the first optimization pass on a particular branch is less than P_b, no change is made to the branch length because it is already sufficiently optimal.

During the normal course of a run, GARLI only applies branch-length optimization to parts of a topology directly affected by a topological rearrangement. It thus depends on the fact that the branch lengths of all parent individuals are already within the proper threshold of their optimal lengths. This is ensured at the beginning of a

run by performing repeated optimization passes over all branches of the starting topology until $\Delta\widetilde{opt}_i^b$ for every branch is less than P_b. After each rearrangement is performed during a run, the algorithm attempts to return the optimality of all branch lengths to within the threshold value.

The progression of GARLI's optimization algorithm following a topological rearrangement is divided into three phases. First, the *localized* phase performs optimization on the branches directly affected by the rearrangement. Second, the *propagation* phase "sweeps" optimization over successive branches outward from the branches optimized during the localized phase. Third, the localized branches and any branches whose lengths were altered during the propagation phase are revisited for final optimization passes during the *refinement* phase.

The following discussion traces the progression of the branch-length optimization algorithm over a topology following the sample SPR rearrangement depicted in Figure 2-5. This rearrangement prunes branch *b1* from its initial location and reattaches it along branch *b2*. This divides *b2* into two new branches, *b2a* and *b2b*, and effectively fuses two branches into one at the previous attachment point of *b1*. This topological change may have an impact on optimal branch-length values in the vicinity of *b1*'s new attachment point as well as it in the vicinity of its previous location. In each of these regions the optimization algorithm centers on a single node. These are termed the *attach node* at *b1*'s new location and the *prune node* at *b1*'s former location (see figure 2-5). The attach node is the node joining branches *b1*, *b2a* and *b2b*. The prune node is the node ancestral to *b1*'s former location. The opposite end of each branch connected to these two nodes is termed its *distal* end. Note that the optimization algorithm progresses in the same fashion following any of the three types of topological mutation operators, except that no prune node is involved in the case of NNI or limSPR rearrangements. Thus, any mention of the

prune node in the following discussion is only relevant in the case of standard SPR rearrangements.

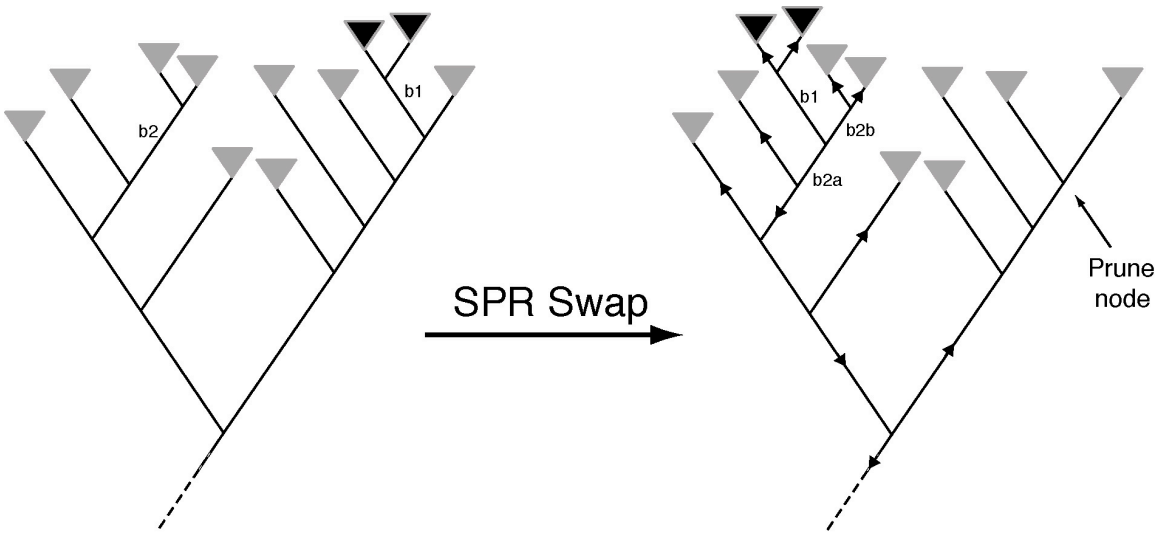

Figure 2-5. Progression of the GARLI branch-length optimization algorithm over a topology following an SPR rearrangement. This rearrangement prunes branch *b1* and reattaches it along branch *b2*, creating branches *b2a* and *b2b*. Optimization is first applied to branches *b1*, *b2b* and *b2a*. Optimization then propagates upward from branches *b1* and *b2b*, and downward from branch *b2a*, as indicated by the arrows. If the value of $\Delta\widetilde{opt}_i^b$ calculated for a particular branch is less than P_b, optimization does not continue to propagate to further branches.

The localized optimization phase begins at the attach node. The branch length of *b1* is initially set to an arbitrary length of 0.05 because its previous length does not have meaning in its new topological context. The previous length of *b2* is divided equally between *b2a* and *b2b*. NR branch-length optimization is performed first on *b1*, followed by *b2a* and *b2b*. The three branches surrounding the prune node are then optimized in a similar fashion. The optimization algorithm then enters the propagation phase at the attach node. The two branches connected to the distal end of *b1* are optimized first. If a branch requires optimization (i.e., $\Delta\widetilde{opt}_i^b$ is greater than P_b), its value is updated and the

optimization algorithm propagates to the two branches connected to the distal node of that branch. This procedure continues to propagate recursively to branches in a directed fashion away from the attach node. Propagation along a series of branches stops when $\Delta o\tilde{p}t_i^b$ is less than P_b or a terminal node is reached. When all optimization has propagated to its full extent on the branches distal to *b1*, optimization propagates in the same manner in a directed fashion from the distal end of *b2a* and then *b2b*. If the lengths of any of the branches surrounding the prune node required optimization during the localized phase, the same propagation procedure is applied around the prune node.

It is important to note that throughout the optimization routine the optimal length of each branch is conditional on the current values of all other branch lengths. This means that even if a branch length is brought to its optimum value after a rearrangement, subsequent optimization of other branch lengths may cause that optimum to shift. Fortunately, the degree to which the lengths of branches affect one another is closely related to the proximity of the branches in the topology. If a particular branch length requires significant adjustment following a rearrangement, it is likely that other branches adjacent to it may as well. This fact is exploited by GARLI to identify branches that may need optimization during the propagation phase.

Finally, the optimization algorithm enters the refinement phase. This phase is necessary because any optimization performed during the first two phases was conditional on the lengths of all branches at that time, and subsequent optimization may have shifted the optima of some branches. The branches directly connected to the attach and prune nodes are placed into the *optimization queue*. Any branches whose lengths were altered during the propagation phase are also added to this queue. The refinement phase consists of cycling through the branches in the queue in the order in which they were added. For each branch, $\Delta o\tilde{p}t_i^b$ is calculated and its length is optimized if necessary.

When the length of a branch requires optimization it is moved to the back of the queue; otherwise it is removed. This procedure continues until there are no branches remaining in the queue, and full optimization is complete.

When a search is far from a topological optimum, many potential rearrangements may allow large gains in fitness. During this period intensive branch-length optimization is not necessary because the beneficial rearrangements are recognizable even without accurate score estimates. As the run progresses these "easy" rearrangements become rarer, and the beneficial rearrangements remaining offer a much smaller increases in score. To recognize these rearrangements of smaller benefit, more accurate estimation of likelihood scores is necessary, implying that a greater effort should be put into branch-length optimization. Toward this end, the algorithm reduces P_b as the run progresses. The initial value of P_b is specified by the user, as is its minimum value. Additionally, the user specifies the number of times that P_b should be decreased (by default, 40). When no topological mutation has created a more optimal individual over a specified number of update intervals, the current P_b value is reduced geometrically. Thus, P_b remains constant while beneficial topological changes are occurring and is reduced when none are found at the current degree of optimization. Beneficial topological mutations often begin to be found again following a reduction in P_b. Immediately after each reduction of P_b, deterministic optimization passes are made over the topologies to ensure that all branch lengths are within the new optimality threshold. Reduction of P_b continues throughout the run whenever there is a lack of beneficial topological mutations. Eventually it reaches its specified minimum value, which is one of the criteria required by the automatic termination criterion.

2.6 MEMORY REQUIREMENTS

As is sometimes the case in computational problems, the GARLI algorithm is confronted with a tradeoff between memory requirements and computational efficiency. Typically the programmer decides how to balance these two factors, and this choice is not flexible at runtime. Unfortunately, if memory usage is minimized the software will be usable on more systems, but will not be optimally efficient where large amounts of memory are available. Conversely, if the choice is made to maximize computational efficiency, typical computer systems will be unable to analyze the largest datasets.

To my knowledge, GARLI is unique among phylogenetic search software in that its memory requirements are flexible. The user specifies the maximum amount of system memory that should be devoted to the storage of likelihood computations. As long as this meets a modest minimum requirement, GARLI works within this limit. If sufficient memory is available, the likelihood computations for all individuals in the population may be stored. This allows maximal reuse of previously computed values. When smaller amounts of memory are provided, GARLI stores as many computations as possible. This process is biased so that the most fit individual has its computations retained whenever possible. Because the most fit individual is both automatically copied into the next generation and the most likely to leave offspring, this biased retention assures that the computations most likely to be reused are retained, even when relatively little memory is available. Note that the minimum and optimal amounts of memory are dependent on both the number of taxa in the dataset and the number of unique character patterns.

Chapter 3: Performance of the Serial GARLI Algorithm

3.1 INTRODUCTION

Measuring heuristic performance

The goal of a maximum likelihood (ML) phylogenetic search is to find the topology that maximizes the log-likelihood score (lnL). This makes the comparison of those scores the most obvious criterion for measuring algorithm performance. However, the optimality of the final solution must be weighed against the time required to obtain it. It might be expected that there will be a tradeoff between search time and the quality of the solution obtained. When presented with the choice between algorithms that are slow and result in optimal solutions or fast and result in less optimal solutions, the choice is ultimately up to the researcher. Each of the algorithms could be considered preferable, depending on the goals of the study and the differences in search time and the optimality of solutions.

Repeatability of results is an important practical consideration in choosing a search algorithm. Because of the stochastic nature of the GARLI algorithm, there can be significant variation between runs, even when they are performed using the same settings and starting conditions. This variation appears both in the form of differences in the sequence of mutations occurring over the course of the run, and in the quality of the final solutions found. Ideally, the algorithm would always reach the same global optimum when given a set of appropriate settings, but in practice this may not be the case. Similar variability may be encountered in deterministic branch-swapping approaches when different starting topologies (often generated by different orders of random stepwise addition) are used. Thus, a lack of exact repeatability should not be viewed as a

drawback of a method *per se*. Stochastic methods that do not always return the same solution must be judged against other algorithms by observing the range of the variability of the solutions, and gauging the probability of each algorithm returning a better solution.

The concept of solution accuracy is closely related to solution optimality, but is not equivalent. By choosing the ML optimality criterion and a particular model of evolution, we are in effect trusting that there will be a strong correlation between solution optimality and solution accuracy. Solution optimality must necessarily be accepted as a proxy for solution accuracy, because accuracy itself cannot be measured when the "true" solution is not known. Numerous simulation-based studies have shown that the correlation between optimality and accuracy is greater for ML than for other optimality criteria (e.g., Huelsenbeck and Hillis, 1993; Huelsenbeck, 1995), at least when the chosen evolutionary model is reasonably close to the true process of evolution. Analyses of simulated data can be important in investigating the performance of search algorithms because both optimality and accuracy may be measured when there is a known "true" solution.

There are three primary aims of this chapter. The first is to explore how the use of different starting conditions affect runtimes, repeatability and the optimality of solutions obtained by GARLI in the practical analysis of real biological datasets. The second is to compare the performance of the program to other software that has been established as the currently best available for the analysis of large datasets under the maximum-likelihood criterion. The third is to examine how the accuracy and runtimes of GARLI and other software scale with increasing numbers of sequences in the analysis of simulated datasets.

3.2 METHODS

Computational setup

Except for where noted otherwise, all analyses were performed on Lonestar, a Rocks version 4.0 Linux cluster (www.rocksclusters.org) maintained by the Texas Advanced Computing Center (www.tacc.utexas.edu). Lonestar comprises 512 dual processor compute nodes, with a mixture of 3.06 GHz and 3.2 GHz Intel Xeon processors. To assure consistency of results, only 3.06 GHz processors were used for experiments, and only one processor was used per node (the other processor remained idle). Each node contains 2 GB of system memory. Other incidental analyses (searches to starting topologies and to optimize final results) were performed on Phylocluster2, a small Rocks version 3.2 cluster with five dual processor compute nodes. Apart from the fact that Phylocluster2 contains 2.8 GHz Xeon processors, the two clusters used are essentially identical.

The source code of GARLI and RAxML (Stamatakis, 2005) were compiled with the Intel icc compiler, version 8.0. The compiler flags specified were "–O2 –ip –fno-alias", which proved to result in the fastest code execution. The Phyml software (Guindon and Gascuel, 2003) is distributed as a binary executable, so it was not able to be compiled equivalently to the other programs.

Biological datasets

Five real datasets of varying sizes were analyzed to examine the performance of the GARLI algorithm and to compare it to similar algorithms such as PHYML and RAxML. These datasets were:

- Rana: A 64 taxon x 1976 column dataset of mitochondrial rRNA from North American frogs (Hillis and Wilcox, 2005).

- Angio: A 228 taxon x 4811 column dataset that combines nuclear ssu rRNA genes and the chloroplastic rbcL and atpB genes from Angiosperm plants. Data obtained from Soltis et. al (1999) and edited to represent the same set of taxa used by Hillis (1996). Previously analyzed by Brauer et al (2003).
- Rbcl: A 500 taxon x 1398 column dataset of rbcL gene sequences from Chase et al. (1993). This dataset, also known as "Treezilla" has been used extensively in testing the performance of heuristic search algorithms.
- 1000ARB: A 1000 taxon x 5547 column dataset of nuclear small subunit rRNA sequences, extracted from the ARB database (Ludwig et. al, 2004). This dataset was extracted and used by Stamatakis et al. (2005) as a benchmark dataset for RAxML, although how particular sequences were selected and the exact taxonomic distribution are unclear.
- Gutell3845: A 3845 taxon x 1866 column dataset of nuclear small subunit rRNA sequences sampled across the Eukaryota, structurally aligned by the Gutell lab at the University of Texas (www.rna.icmb.utexas.edu). These sequences were selected from a larger dataset of 5844 eukaryotic sequences by removing identical sequences and ones that did not have an associated NCBI taxonomic hierarchy string (see http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html).

The most appropriate evolutionary model was selected for each dataset by applying a hierarchical likelihood-ratio test (Goldman, 1993). The model chosen was GTR+I+G in all cases except the Gutell3845 dataset, which favored GTR+G.

GARLI settings

GARLI version 0.942 was used for all analyses presented here. As discussed in chapter 2, GARLI allows a large number of variables controlling the details of the GA to

be specified by the user. The experiments presented here will not attempt to examine the implications of varying the majority of these settings, but rather will focus on a set of defaults that have been determined through the development and testing of the program. A listing of some of the more important settings follows, and a complete list of all settings used appears in the supplemental information at the end of this chapter. These settings were used on all serial runs. Population sizes were four, with one holdover individual. The selection intensity (s) was 0.5. The starting and minimum values of the branch-length optimization parameter (P_b) were 0.5 and 0.01, respectively. P_b transitioned from its starting to minimum values in forty steps. The prior weights of topological, model and branch-length mutations were 1.0, 0.05 and 0.2, respectively, resulting in expected proportions of 80%, 4% and 16% for these three classes of mutations. Weights of 0.2, 0.5 and 0.3 were assigned to NNI, limSPR and SPR topological mutations, respectively. The maximum reconnection distance for limSPR mutations was five. The stopping criterion was set such that runs would terminate when no topological mutations increasing the lnL by more than 0.05 lnL units had occurred over the last 20,000 generations and the total increase in lnL over the last 500 generations was less than 0.05. Initial deterministic optimization of the branch lengths and the α parameter were performed at the outset of all runs. A maximum of 1.6 gigabytes of memory were allowed for storage of the conditional likelihood arrays for all datasets.

Experimental design

Effect of starting conditions

Determining how the starting conditions provided to GARLI affect the final solutions returned is of obvious practical interest. As with most phylogenetic search software, the algorithm may be given a starting point in tree-space in the form of a tree

topology estimated by a fast distance or parsimony-based inference method. In addition, starting model parameters may be estimated via ML using a program such as PAUP* (Swofford, 2002) and provided to the algorithm. When such initial estimates are not provided, GARLI starts its search from a random starting topology and a set of reasonable default model parameters. Providing the program with a reasonable starting topology gives the algorithm a large head start in terms of its initial lnL scores relative to starting with a random starting topology, but this does not necessarily mean that the run will have shorter runtimes or more optimal final results.

One of the reasons for choosing ML as an optimality criterion is its ability to avoid certain biases apparent in other phylogenetic inference methods. The most commonly cited of such biases is the well studied "long-branch attraction" bias of parsimony methods (Felsenstein, 1978). When a starting topology is obtained from another inference method, the possibility exists that methodological biases are introduced into the GA. When local topological optima under the ML criterion are not common, any methodological biases introduced by a starting tree are likely to be eliminated by simple hill-climbing optimization in the GA. However, if the starting tree lies at or near a local topological optimum under the ML criterion, it may be difficult to overcome these influences. Starting the algorithm from a random topology provides a sure way of avoiding any biases inherent in other inference methods, but requires the search to move a longer distance through tree-space.

Because of the relatively high computational cost of optimizing model parameters through the GA and the typical lack of variation in the optimal parameter values across reasonable topologies, providing GARLI with fixed model parameter estimates is an attractive option. However, the computational cost of initially obtaining the model estimate must also be considered, and is not trivial for large datasets. Additionally, there

are two potentially detrimental effects of fixing model parameters during a search. The first could arise if the model parameters initially estimated are far enough from the globally optimal values that they will favor a different topology. This could mean that the best scoring topology under the fixed parameter values is found, but that it is not identical to the globally optimal topology that is the goal of the search. The second effect of fixing model parameters might arise if changes in model parameters assist the GA in moving more easily through tree-space. If subtle changes in parameter values are able to change the relative lnL ranking of topologies, model mutations might change the likelihood surface enough to allow the search to escape from a local topological optimum. Fixing model parameters could then hinder the search by eliminating this beneficial effect. The experiments performed here will investigate the importance of these concerns.

To investigate the effects of the starting topology on the performance of the GARLI algorithm in the analysis of real data, six topologies were chosen per dataset using the methodology detailed below. The topologies were selected from a large population of topologies obtained by parsimony searches performed in PAUP* using a variety of branch-swapping settings. In addition to allowing a range of starting trees for GARLI, the procedure used also allows the investigation of several other interesting issues. The correlation between the parsimony and likelihood scores of topologies on large datasets will be observable, as well as the effect of intensive branch swapping under the parsimony criterion on parsimony and likelihood scores. For the Angio dataset, Brauer et al. (2002) found that intensive parsimony searches resulted in topologies with higher lnL scores than did GAML runs started from random trees. Here it will be possible to compare the degree to which parsimony searches are able to compete with GARLI in obtaining trees that score well under the ML criterion.

The following procedure was used to select the six parsimony starting topologies for each dataset:

1. Searches were performed in PAUP* under the unweighted parsimony criterion, starting from stepwise-addition trees with random addition sequences. One hundred searches were performed for each of four levels of branch swapping: no swapping (stepwise-addition only), NNI swapping, SPR swapping and TBR swapping. All searches were continued until all possible rearrangements of the given type had been exhausted without finding a better scoring individual. Multiple trees of equal score were not stored during each search (multrees=no), and the best tree from each of the one hundred replicates was retained (savereps=yes).
2. Model parameters for the GTR+I+G model were estimated by ML using PAUP* on the tree with the best parsimony score over all of the 400 searches (or one of the best in the case of ties).
3. All trees obtained from the searches had lnL scores obtained under the GTR+I+G model with the model parameters fixed at the estimates obtained in step 2.
4. The parsimony and lnL scores of all trees were plotted against one another, and a linear regression was calculated using Microsoft Excel.
5. For each dataset, six trees were selected by eye along the regression line, with an attempt made to ensure equal spacing over the full range of scores.
6. For each of the six starting topologies, optimal model parameter values were estimated by ML in PAUP*.

Replicated GARLI runs were performed on each dataset using three distinct setups, which will be referred to as the *starting conditions*:

- RAND: Random starting topology and default model parameters. All parameters optimized by GARLI (ten replicates per dataset, not performed on Gutell3845 due to computational limitations).
- TOPO: Parsimony starting topology and default starting model parameters. All parameters optimized by GARLI (four replicates per starting tree, per dataset).
- TOPO+MOD: Parsimony starting topology and ML model parameter estimates provided. Branch lengths and topology optimized by GARLI, model parameters fixed (four replicates per starting tree, per dataset).

These replicated allowed investigation of the effects of both starting topology and fixed model parameters on final optimality and runtimes. However, because of the large stochastic component of GARLI's search strategy, there can be significant variation in runtimes and final optimality even when using exactly the same starting conditions. This fact makes evaluating the effect of run conditions by eye problematic, and calls for more rigorous statistical testing. Unfortunately, most parametric statistical tests (such as ANOVA) that could be used to test for an effect of the starting topology or conditions on lnL scores or runtimes make the assumption that the data are normally distributed with equal variances. Because it is not clear that lnL scores or runtimes obtained in replicated GARLI runs meet these conditions, the less powerful nonparametric Kruskal-Wallis test was used. This test first requires ranking the raw data (runtimes or final lnL scores) across groups (topologies or conditions). The average rank of the samples in each group was then tested against a null distribution of ranks expected in the absence of any effect of the groups. By defining six groups (one per topology) consisting of four replicates each, the effect of starting topology may be tested. This was done separately within the TOPO and TOPO+MOD conditions on each dataset. The effect of fixing model

parameters was tested by pooling the data across all six starting topologies into two groups representing the TOPO and TOPO+MOD conditions, with 24 samples in each.

Note that the RAND condition does not directly fit into the scheme of these replicated tests because each replicate was started from a different random topology. Runs using this condition were performed as a test of the feasibility of performing GARLI searches without supplying a precomputed starting topology, and performance was compared to the other runs by eye.

Comparison to alternative software

A comparison of the performance of the GARLI algorithm relative to other ML phylogenetic search programs is of interest, but presents some difficulties. The comparisons presented here are to two recently developed programs that have been shown to be the fastest currently available for ML inference. These programs are PHYML (Guindon and Gasquel, 2003) and RAxML (Stamatakis et. al, 2005). The versions used were PHYML 2.4.4 and RAxML-VI-HPC ver 1.0. Performing comparisons to these programs is made more difficult by differences in the implemented evolutionary models and by a number of computational issues.

Although PHYML implements the same GTR+I+Γ model of sequence evolution as GARLI, the details of the implementation are subtly different. This results in different (although in theory equally valid) likelihood scores. However, comparable scores may be obtained by using a common program to fully optimize model and branch length parameters on the final topology obtained by each of the algorithms. I chose to use PAUP* for this purpose, because it is the most stable and probably most commonly used program for ML inference. Note that the implementations of the GTR+I+Γ model in GARLI and PAUP* are identical, resulting in equivalent lnL scores. Because the programs being compared differ both in their intent and ability to optimize model

parameters and branch lengths, common scoring also has the benefit of ensuring that topologies are compared on the basis of their fully maximized lnL score.

Comparison to the RAxML program presents further complications. RAxML implements site-specific rate heterogeneity (SSRH: Olsen et al, 1994), which offers a large benefit in speed and memory requirements over the more commonly used discrete gamma-distributed rate heterogeneity implementation (GRH: Yang, 1994). This advantage arises because SSRH only requires the computation of each site likelihood for a single rate, while GRH must average the site likelihoods over a number of rates (most commonly four, as in GARLI). The use of different models makes comparison of results between these programs potentially problematic, because there is no guarantee that the models will favor the same topologies. However, the developer of RAxML notes that the SSRH model is used primarily for its computational advantages, and agrees that the GRH model should be used to assess final topology quality (A. Stamatakis, pers. comm.). In practice, performing inference using the SSRH model in RAxML has proven to be an excellent method for obtaining topologies that score well under the GRH model. For the aforementioned reasons, the performance of all programs was compared on the basis of the lnL scores of their final topologies after optimization in PAUP*.

Other problems presented by performing experiments with PHYML and RAxML are purely computational. PHYML has by far the most extreme memory requirements of the three programs, to the degree that it limits the datasets on which it can be run. Because all experiments were performed on systems containing 2 Gb of system memory, PHYML runs could not be performed on the largest real datasets, nor on the majority of the simulated datasets. RAxML presents a different computational problem. The Lonestar cluster only allows analyses to be run for a maximum of 24 hours. Both GARLI and RAxML require more than that amount of time to perform analyses on the largest

datasets. After 24 hours of a run have elapsed, GARLI may be easily restarted by giving it the best topology and model parameters that were found before the run was terminated. GA parameters such as the branch length precision parameter may also be set it their final values. Although this is not exactly equivalent to allowing a single run of greater than 24 hours to continue until completion, it has no important implications for the experiments presented here. Unfortunately, RAxML is not able to be restarted in this way. Although the program offers the option to output checkpoints which could theoretically used to restart the program, when a RAxML run is started with a checkpoint it treats it as a completely new search and "forgets" all previous parameter values (A. Stamatakis, pers. comm.). For this reason, RAxML runs which required greater than 24 hour runtimes were run on Phylocluster2, which does not enforce a runtime limit. Because the only difference between the systems is in the processor speed, correcting the runtimes of analyses performed on Phylocluster2 by a factor of 0.915 (2.8 GHz divided by 3.06 GHz) should result in runtimes that are comparable to those obtained on Lonestar. All results presented here will be standardized in this way when necessary.

Multiple PHYML and RAxML runs were attempted for each of the five datasets. Analyses were performed beginning both from the default starting points of each program and beginning from the most optimal of the parsimony starting trees provided in GARLI runs. Only a single default run was performed using PHYML because it uses the deterministic BIONJ method to obtain a starting topology. Three default runs were performed with RAxML because its use of parsimony random stepwise addition can generate a variety of starting topologies. Starting each of the programs from a topology also used to start GARLI runs allows differences in performance between the programs due to their ability to create a quality start tree and to search effectively to be differentiated.

Evaluation of topological accuracy

A series of large simulated datasets were generated to allow investigation of the topological accuracy of solutions obtained by GARLI, RAxML and PHYML. These datasets range in size from 500 sequences to 4000 sequences, allowing the relationship between dataset size and accuracy to be measured for each of the programs. Simulated datasets were generated as follows:

1. A 4000-taxon simulation topology was generated using software written by Tracy Heath. This software is based upon the birth-death model implemented in PhyloGen (evolve.zoo.ox.ac.uk/software/PhyloGen/main.html), but additionally allows the birth and death rates to evolve over the topology. Starting rates were: birth = 0.3 death = 0.2.
2. This ultrametric tree was scaled to have a root-to-tip length of 0.5 and variable rates of evolution were simulated on the tree. This was done using the rate evolution model of Kishino, Thorne and Bruno (2001), implemented in software by Tracy Heath. Rates were evolved over the tree with a starting rate at the root of 1.0. At each node, new rates were drawn from a lognormal distribution centered on the rate of the parent node with variance = (ν * edge length). ν is a constant that determines the amount of autocorrelation among the rates of parent and descendent nodes, and in this case was set to 1.0. The substitution rate applied to each branch is the average of the rates of the nodes at each end.
3. Two distinct "genes" of 1398 nucleotides each were simulated on the topology with Seq-Gen (Rambaut and Grassly, 1997), using simulation parameters corresponding to estimates obtained from the Rana and Rbcl datasets. Parameters were estimated in PAUP* under the GTR+I+G model on the best-known tree for each dataset, and

appear in the supplemental information at the end of this chapter. The two genes were concatenated to form the 4000sim dataset of 2796 nucleotides.

4. Sequences were randomly subsampled from the 4000sim dataset, creating smaller datasets of various sizes. By using the edge lengths of the true simulation tree, it was ensured that all subsampled datasets had the same tree diameter (maximum pathlength) as the original 4000 taxon tree. The datasets created by subsampling ranged from 500 to 3500 taxa in increments of 500. All sequences were chosen randomly from the full set of 4000 sequences for each dataset size. This means, for instance, that the 500 taxa of the smallest dataset do not all appear in the 1000 taxon dataset.
5. For each dataset, 100 parsimony searches using NNI branch swapping were performed in PAUP*. The tree with the best score under the parsimony criterion was selected as the starting tree to be used for all runs on that dataset.

The intent of using this procedure was to create a series of relatively realistic simulated datasets that were directly comparable. Data simulated under two different sets of model parameters were concatenated to introduce additional heterogeneity and reduce the inference model's ability to exactly capture the true generating process. One run was performed per program per dataset, in each case using the same starting topology. Due to PHYML's memory requirements, runs were only able to be performed on the 500 and 1000 sequence datasets.

3.3 RESULTS AND DISCUSSION

Starting tree searches

The times required for the PAUP* parsimony searches used to obtain starting topologies for each dataset appear in Table 3-1. The search times were trivial for the three smaller datasets, but grow extremely quickly with increasing taxon numbers for SPR and TBR branch swapping. The one hundred replicate searches using TBR branch swapping for the Gutell3845 dataset required almost 17 days to complete. The relationships between the likelihood and parsimony scores of all resulting topologies for each dataset are depicted in figure 3-1(a-e). All datasets show a correlation between scores under the two criteria, although this relationship is fairly weak for the Rana dataset. The relationship between the thoroughness of branch swapping and the optimality of the resulting topologies is as would be expected, with more intensive searching resulting in more optimal topologies as measured under either criterion. The results suggest that very little extra optimality is gained by performing TBR branch swapping relative to SPR branch swapping, despite the fact that the searches using TBR took significantly longer. For the larger datasets a significant gap in scores exists between the topologies obtained by NNI swapping and by SPR swapping.

	Search Time (minutes)				Ave. Opt. Time
Dataset	SA	NNI	SPR	TBR	(minutes)
Rana	0.01	0.01	0.07	0.10	1.98
Angio	0.28	0.40	4.30	7.07	47.58
Rbcl	0.38	0.65	4.95	12.02	41.38
1000ARB	6.68	14.22	840.38	1069.97	572.33
Gutell3845	51.92	183.53	16236.20	24375.31	797.92

Table 3-1. Times required to obtain parsimony starting topologies. Total search time of one hundred parsimony searches performed for each level of branch-swapping and average time required to optimize branch lengths and model parameters on a single topology.

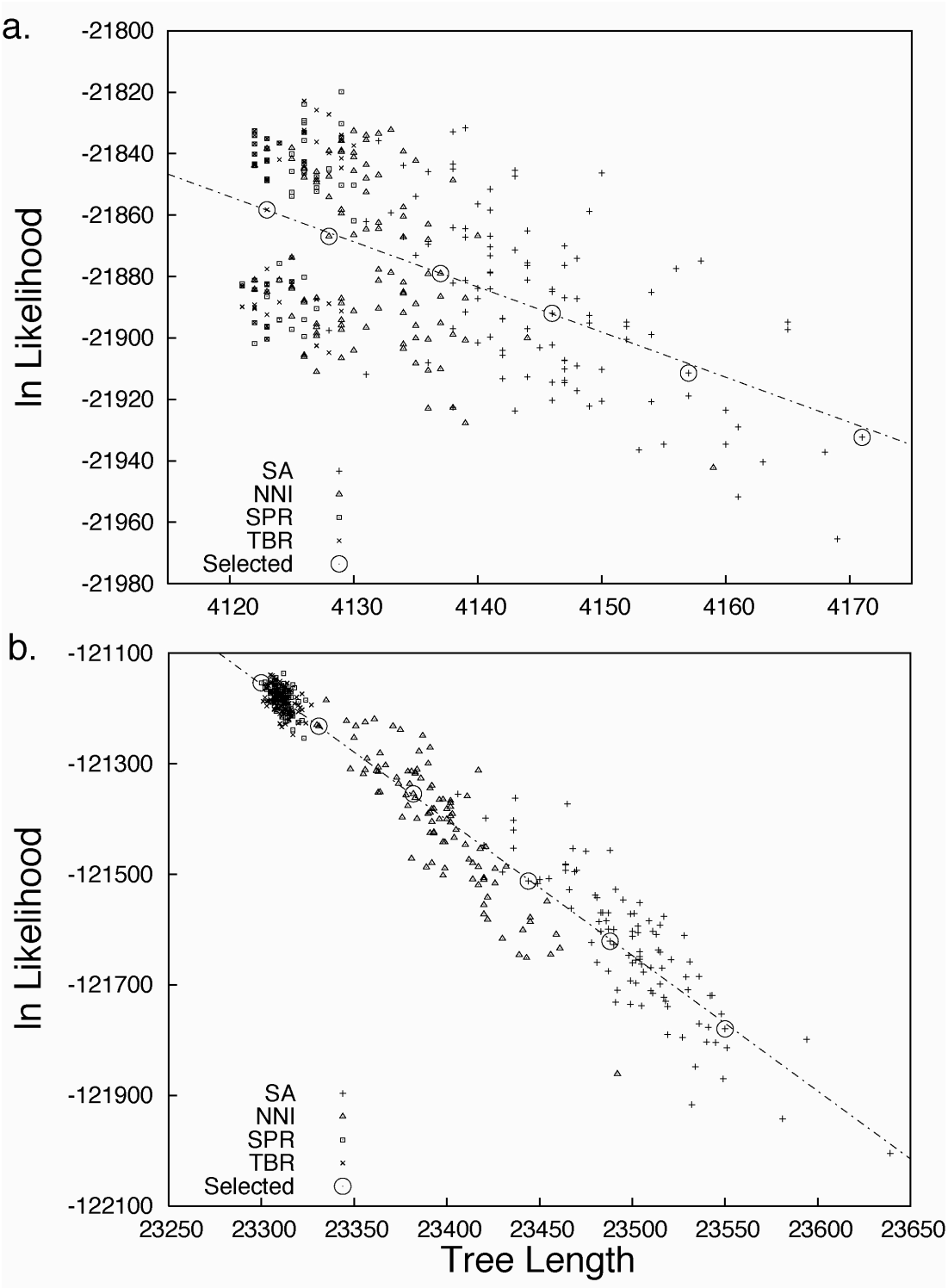

Figure 3-1. (a.)Rana. (b.)Angio.

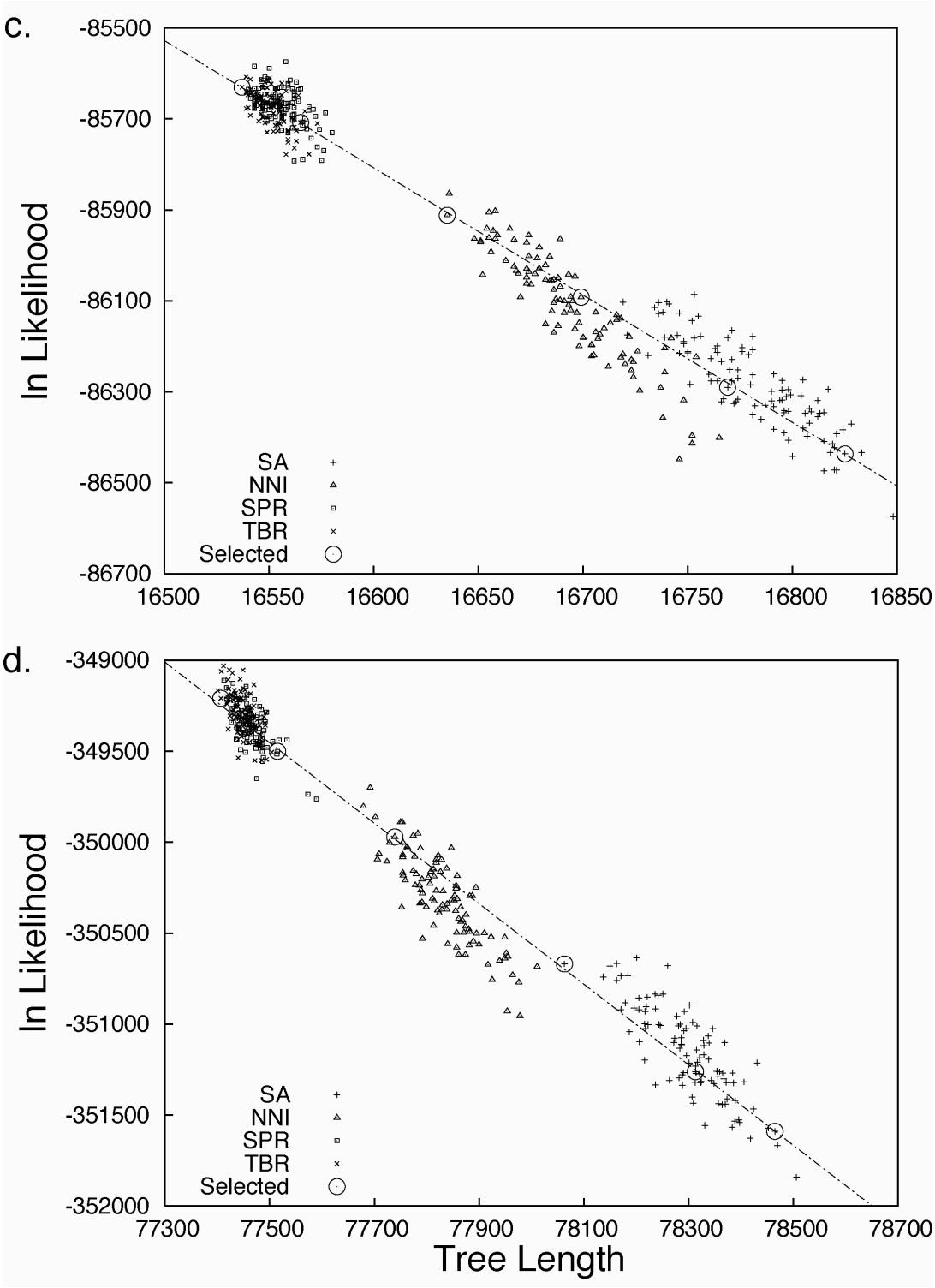

Figure 3-1. (continued) (c.) Rbcl. (d.) 1000ARB.

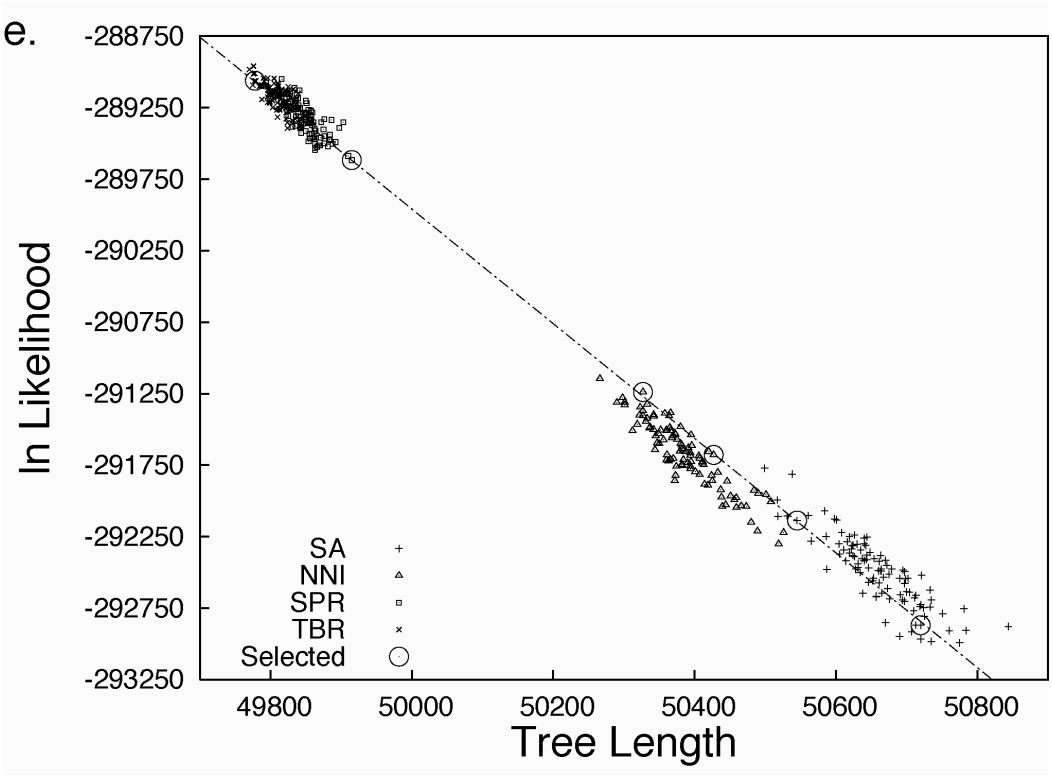

Figure 3-1. (continued) The relationship between parsimony and likelihood scores of topologies obtained by parsimony searches in PAUP* with the indicated types of branch swapping. (a.) Rana, (b.) Angio, (c.) Rbcl, (d.) 1000ARB, (e.) Gutell3845. The topologies selected for use as GARLI starting topologies are indicated by circles.

The scores of the six topologies per dataset selected as GARLI starting points appear in table 3-2, and appear circled in figure 3-1(a-e). The average time required to estimate optimal model parameters and branch lengths for each of the six chosen starting topologies ranged from 2 minutes for the Rana dataset to over 13 hours for the Gutell3845 dataset. A relatively small range of lnL scores was observed among the topologies chosen for the Rana dataset, but the range of scores between topologies was quite large for the other datasets.

Tree#	Dataset									
	Rana		Angio		Rbcl		1000_ARB		Gutell3845	
	lnL	Length	lnL	Length	lnL	Length	lnL	Length	lnL	Length
1	-21858.29	4123	-121153.71	23300	-85630.66	16537	-349208.91	77407	-262114.30	49778
2	-21866.94	4128	-121232.02	23331	-85708.19	16565	-349500.32	77515	-262574.61	49915
3	-21879.03	4137	-121354.46	23382	-85911.87	16635	-349970.73	77739	-263841.14	50327
4	-21891.99	4146	-121512.26	23444	-86092.16	16699	-350668.84	78063	-264183.15	50427
5	-21911.43	4157	-121621.07	23488	-86289.85	16769	-351261.51	78313	-264445.61	50545
6	-21932.35	4171	-121779.57	23550	-86436.37	16825	-351588.97	78465	-265084.87	50720

Table 3-2. Optimized log-likelihood and parsimony scores of topologies selected as GARLI starting points.

GARLI search results

Runtimes and the final lnL scores obtained after parameter optimization in PAUP* for all runs using parsimony starting topologies appear in table 3-3. Results of runs using random starting topologies appear in table 3-4. Across all runs there was a clear correlation between the number of sequences in the dataset and the amount of variation observed in both final lnL scores and runtimes. Figures 3-2 through 3-5 graphically display the final lnL values and runtimes for each dataset besides Rana. Due to the lack of variation in final lnL scores and runtimes for the Rana dataset, results from this dataset were omitted from the figures. For the same reason the results from the Rana dataset were omitted from the statistical tests of the effect of topology and conditions on final lnL. The results of all Kruskal-Wallis tests performed on the other datasets are summarized in table 3-5.

Effect of starting topology

The five datasets studied show a variety of relationships between the starting topology used for the search and the final lnL values attained. Starting topology appears to have little effect on the smaller datasets, but a significant effect on the larger ones.

4			Topology 1		Topology 2		Topology 3		Topology 4		Topology 5		Topology 6	
Dataset	Condition	Rep	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)
Rana	TOPO	1	-21812.64	15.8	-21812.67	15.7	-21812.67	15.7	-21812.67	15.9	-21812.67	15.8	-21813.44	16.2
		2	-21813.44	16.2	-21812.64	15.5	-21812.67	15.7	-21812.64	16.0	-21812.67	15.9	-21812.64	15.9
		3	-21812.67	15.8	-21812.67	15.8	-21812.64	15.6	-21812.67	16.1	-21812.67	15.7	-21812.64	15.6
		4	-21813.44	16.3	-21812.64	15.5	-21812.64	15.7	-21812.67	16.0	-21813.44	16.3	-21812.67	16.4
	TOPO+MOD	1	-21813.44	15.9	-21812.64	15.5	-21812.67	15.7	-21812.67	15.8	-21812.64	15.6	-21812.67	15.7
		2	-21812.64	15.4	-21812.67	15.5	-21812.67	15.4	-21812.64	15.8	-21812.64	15.8	-21812.64	15.9
		3	-21813.44	15.8	-21812.67	15.5	-21812.67	15.5	-21812.67	15.7	-21812.64	15.5	-21812.67	15.9
		4	-21812.64	15.4	-21812.64	15.4	-21813.44	15.9	-21812.64	16.0	-21812.64	15.4	-21812.67	15.6
Angio	TOPO	1	-121011.03	113.6	-121010.27	104.6	-121002.86	171.2	-121002.77	206.0	-121013.26	143.6	-121012.95	104.4
		2	-121010.38	129.0	-121011.43	129.6	-121005.07	105.5	-121012.02	194.5	-121011.22	102.9	-121010.72	163.1
		3	-121015.27	129.0	-121010.27	130.0	-121010.54	105.8	-121010.38	195.2	-121002.66	102.5	-121010.72	140.0
		4	-121012.49	159.6	-121010.38	105.2	-121003.16	113.4	-121021.51	133.7	-121010.72	138.5	-121004.73	138.0
	TOPO+MOD	1	-121013.26	90.6	-121010.38	101.7	-121013.24	89.7	-121015.47	97.7	-121010.64	101.5	-121004.95	90.4
		2	-121010.27	123.6	-121010.27	130.7	-121012.73	132.7	-121015.45	85.3	-121012.78	115.0	-121012.95	92.8
		3	-121010.54	86.0	-121010.27	91.7	-121010.98	110.0	-121015.65	112.4	-121013.91	106.0	-121010.96	113.7
		4	-121015.79	84.7	-121010.27	88.6	-121010.38	91.9	-121002.66	115.7	-121015.79	91.2	-121022.94	93.4
Rbcl	TOPO	1	-85416.68	100.7	-85440.51	107.4	-85415.28	105.2	-85421.03	127.3	-85413.63	105.3	-85414.89	154.2
		2	-85412.93	97.9	-85432.83	111.6	-85422.06	120.8	-85431.17	106.4	-85419.75	152.7	-85417.01	79.7
		3	-85414.66	91.8	-85429.12	113.2	-85417.04	125.0	-85427.64	152.7	-85412.37	129.1	-85422.78	137.3
		4	-85419.17	97.1	-85440.27	129.7	-85418.94	119.7	-85424.22	107.7	-85418.78	115.7	-85427.14	96.8
	TOPO+MOD	1	-85422.27	61.8	-85444.46	129.7	-85419.24	91.6	-85418.50	94.7	-85416.96	68.6	-85415.93	108.9
		2	-85414.92	84.2	-85447.38	157.1	-85417.41	84.5	-85424.12	117.7	-85417.68	70.5	-85415.21	79.1
		3	-85415.25	94.6	-85457.78	82.2	-85422.47	106.5	-85420.29	107.3	-85426.95	65.5	-85422.27	120.0
		4	-85418.99	63.4	-85440.84	97.5	-85414.16	92.9	-85417.27	117.4	-85417.18	60.8	-85417.24	94.0

Table 3-3. (Continued on next page)

			Topology 1		Topology 2		Topology 3		Topology 4		Topology 5		Topology 6	
Dataset	Condition	Rep	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)
1000ARB	TOPO	1	-348436.79	315.2	-348460.95	513.4	-348500.05	681.4	-348508.93	535.2	-348551.97	777.4	-348549.47	662.9
		2	-348463.77	265.5	-348547.25	510.2	-348545.11	564.9	-348455.69	510.5	-348488.78	793.7	-348513.57	536.5
		3	-348476.00	348.4	-348569.63	484.0	-348595.10	629.8	-348524.32	451.5	-348535.36	572.1	-348527.87	391.0
		4	-348419.36	725.0	-348496.29	628.3	-348528.69	520.0	-348455.56	848.6	-348676.86	569.2	-348476.97	716.9
	TOPO+MOD	1	-348478.11	451.4	-348420.32	533.0	-348534.31	394.3	-348537.35	322.9	-348443.82	731.3	-348459.23	584.3
		2	-348450.20	282.6	-348485.16	521.4	-348507.03	322.4	-348536.10	315.0	-348544.99	731.6	-348613.38	520.4
		3	-348415.47	417.0	-348505.52	425.1	-348528.91	424.6	-348501.10	242.1	-348479.62	675.2	-348608.93	380.6
		4	-348475.54	321.6	-348422.56	550.5	-348542.09	277.1	-348464.83	383.8	-348552.26	423.7	-348509.56	693.8
Gutell3845	TOPO	1	-261023.44	2248.9	-261065.14	2979.0	-260990.96	2312.4	-260997.11	2712.4	-261049.18	2985.9	-261167.28	3108.6
		2	-261023.91	2103.3	-261059.99	3159.7	-261196.86	2861.1	-261305.72	2264.0	-261061.57	1983.8	-261273.68	2727.1
		3	-260990.96	2462.7	-261189.48	1839.9	-261126.12	2563.0	-261279.81	2349.6	-261176.99	2472.5	-261181.89	3099.8
		4	-260997.11	2354.0	-261104.08	3076.3	-261291.31	1818.9	-261253.69	2810.1	-261022.00	2263.2	-261386.56	2341.9
	TOPO+MOD	1	-261003.87	1728.5	-261127.93	1379.4	-261253.60	1945.7	-261294.37	1834.1	-261114.02	1654.3	-261408.40	1413.8
		2	-261000.47	2098.7	-261158.25	1361.4	-261179.69	2177.5	-261282.63	2019.9	-261138.87	1946.6	-261363.55	1922.9
		3	-261011.10	1417.5	-261104.60	2091.0	-261209.70	2052.6	-261328.93	1679.2	-261170.53	1758.6	-261375.64	1776.7
		4	-260967.72	2179.5	-261102.49	1552.5	-261322.07	1816.1	-261201.43	2406.8	-261132.33	1666.4	-261439.80	1625.8

Table 3-3. Results of GARLI analyses performed with parsimony starting topologies. Log-likelihood (lnL) scores represent the score obtained after optimization of branch lengths and model parameters in PAUP*. The six starting topologies are ordered from the most optimal (Topology 1) to least optimal (Topology 6).

	Rana		Angio		Rbcl		1000ARB	
Rep	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)
1	-21812.67	16.1	-121010.27	140.6	-85439.97	189.4	-349644.71	1103.9
2	-21812.64	16.0	-121002.66	144.8	-85495.40	113.9	-350294.35	1867.1
3	-21812.67	16.5	-121005.83	122.8	-85430.51	156.3	-348659.68	1744.5
4	-21812.67	16.8	-121010.27	111.3	-85436.04	98.8	-349905.73	1439.9
5	-21812.67	17.0	-121020.91	150.6	-85442.27	154.0	-350236.01	1361.2
6	-21812.64	17.1	-121002.66	115.1	-85487.77	131.9	-349345.51	1002.0
7	-21812.64	17.1	-121010.27	113.2	-85520.19	111.3	-351515.70	818.0
8	-21812.67	16.7	-121010.27	114.8	-85423.12	113.3	-350282.06	1151.1
9	-21812.67	16.0	-121003.85	142.1	-85422.94	163.9	-348517.18	1424.5
10	-21812.64	16.2	-121165.14	116.4	-85423.08	146.6	-348582.00	1439.8

Table 3-4. Results of GARLI analyses performed with random starting topologies. Log-likelihood (lnL) scores represent the score obtained after optimization of branch lengths and model parameters in PAUP*.

	Effect of starting topology on lnL		Effect of starting topology on runtime		Effect of fixed model parameters on:	
Dataset	within TOPO	within TOPO+MOD	within TOPO	within TOPO+MOD	lnL	Runtime
Angio	0.362	0.312	0.155	0.842	0.527	0.018*
Rbcl	0.007*	0.048*	0.210	0.016*	0.995	0.096
1000ARB	0.044*	0.154	0.240	0.012*	0.918	0.297
Gutell3845	0.072	0.001*	0.344	0.214	0.569	0.003*

Table 3-5. P-values resulting from Kruskal-Wallis nonparametric tests. P-values of less that 0.05 indicated with asterisks.

This is consistent with the notion that larger numbers of sequences lead to larger tree-spaces that are more difficult to search, making higher quality starting points in that space more beneficial. The lack of variation in scores obtained for the Rana dataset suggests that starting topology is not important for such small datasets, and runs performed from random starting topologies perform as well as those in which a topology was provided.

Analyses performed on the Angio dataset result in more variation in scores, but no evidence that the starting topology provided has a consistent effect on the final score. Interestingly, the lnL scores obtained across runs on this dataset suggest the presence of a

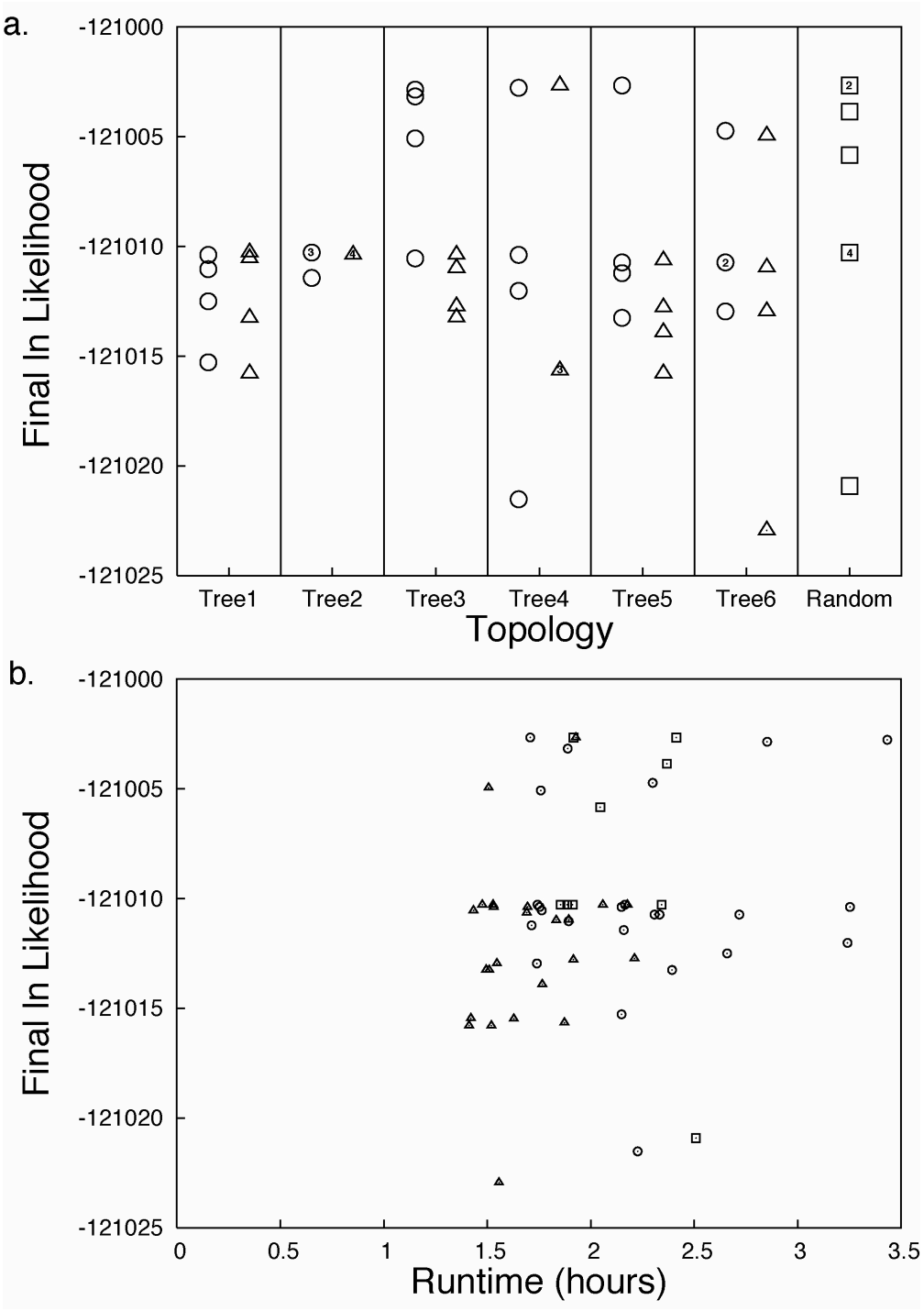

Figure 3-2. Results of GARLI runs performed on the Angio dataset. Key: circles=TOPO condition, triangles=TOPO+MOD condition, squares=RAND condition.

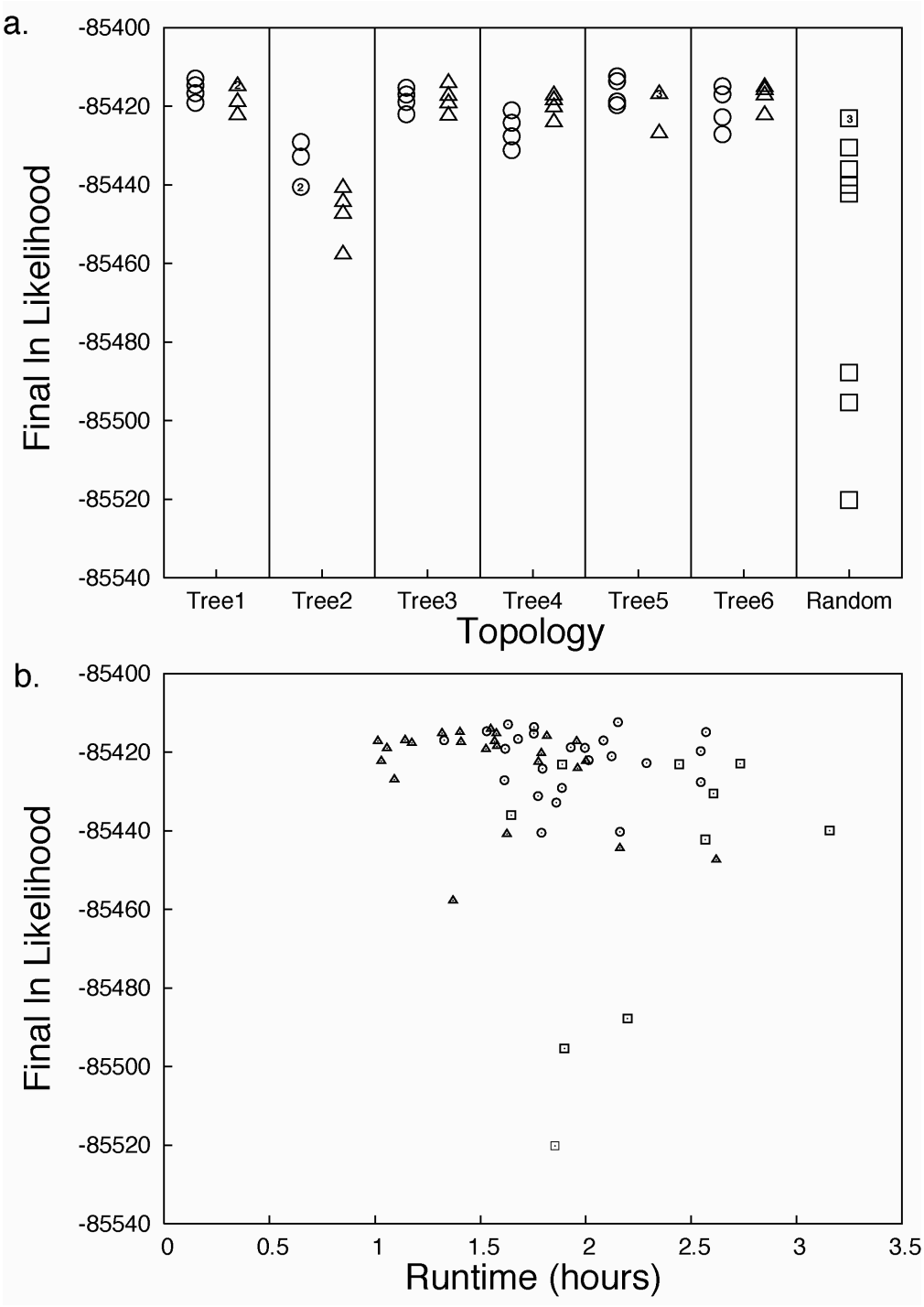

Figure 3-3. Results of GARLI runs performed on the Rbcl dataset. Key: circles=TOPO condition, triangles=TOPO+MOD condition, squares=RAND condition.

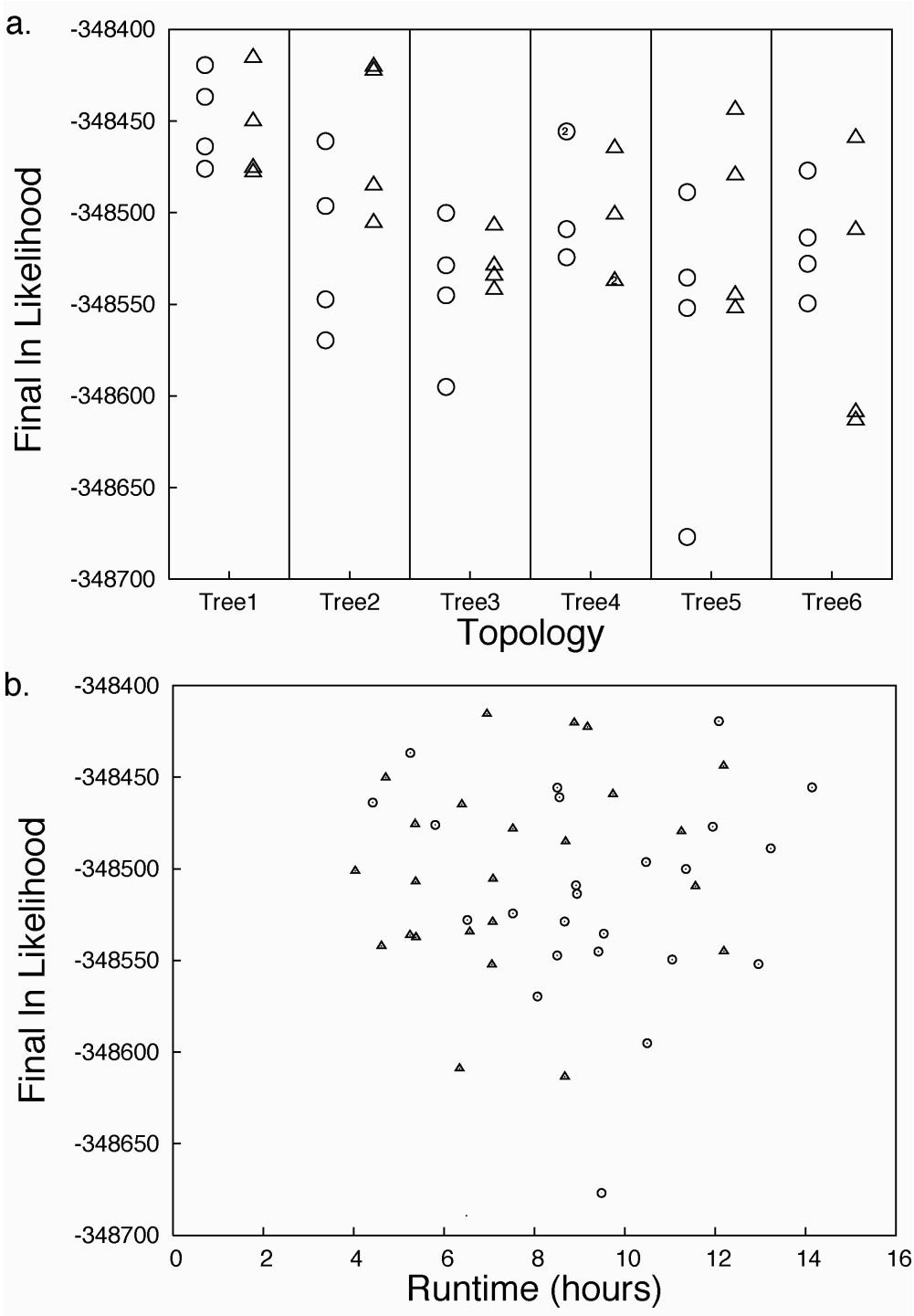

Figure 3-4. Results of GARLI runs performed on the 1000ARB dataset. RAND condition runs are omitted for clarity. Key: circles=TOPO condition, triangles=TOPO+MOD condition.

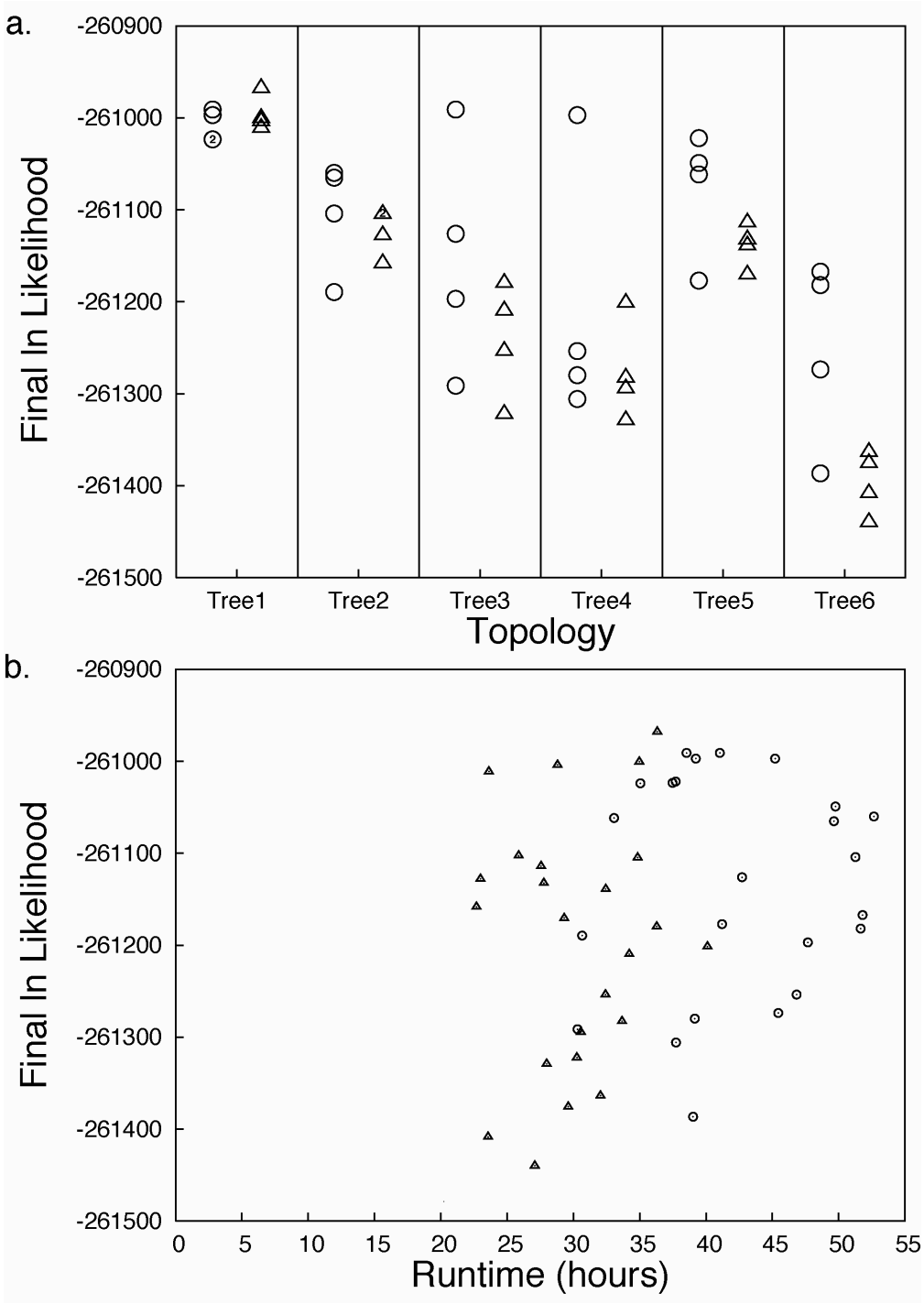

Figure 3-5. Results of GARLI runs performed on the Gutell3845 dataset. Key: circles=TOPO condition, triangles=TOPO+MOD condition.

local optimum with a score of approximately -121,010 lnL that relatively few runs escape. A higher proportion of runs started from random topologies on the Angio dataset avoid entrapment in this optimum than do runs starting from parsimony trees (4 of 10 = 40% vs. 8 of 48 = 16.7%). Although this effect is only weakly significant, it is consistent with the idea that random starting topologies can help avoid entrapment in local optima.

The results of analyses performed on the Rbcl dataset show a weakly significant effect of starting topology on final lnL for both the TOPO ($p=0.007$) and TOPO+MOD ($p=0.048$) starting conditions. Interestingly, it appears that it is the identity of the starting topology rather than its degree of optimality that is important. Tree2, the second most optimal starting topology clearly performs the most poorly, suggesting that searches beginning from this topology are prone to entrapment in a local optimum. These results indicate that performing runs with a variety of starting points is helpful in finding more optimal results, but that this effect is not necessarily directly related to the optimality of the starting tree. On this dataset, results obtained using random starting topologies do not fare as well relative to ones obtained using parsimony starting topologies as they did for the two smaller datasets.

The 1000ARB dataset shows a weakly significant effect of starting topology on final optimality for the TOPO starting condition ($p=0.044$), but not for the TOPO+MOD condition. This dataset also presents much greater variation in final likelihood scores than did the smaller datasets. The RAND starting condition performs quite poorly on this dataset, and is omitted from figure 3-4. The best results obtained searching from random starting topologies are on par with the worst scoring of the runs using parsimony starting trees, and the worst of the RAND runs are thousands of lnL units short of the best known trees. The feasibility of using random starting trees in GARLI clearly plummets as taxon numbers grow.

The Gutell3845 dataset shows a fairly strong effect of starting topology on final lnL scores within the TOPO+MOD condition (p=0.001). Unlike some of the smaller datasets, a strong linear relationship between the score of the starting topology and the final lnL obtained is apparent, although Tree5 does not fit into this general pattern. As with the 1000ARB dataset, there is a large amount of variation in lnL scores across runs.

Apart from the small Rana dataset, variation in runtimes is observed for all datasets both in run using different starting topologies and in replicated runs performed using a single topology. However, only the Rbcl and 1000ARB datasets show a significant effect of starting topology on runtimes (p=0.016 and p=0.12, respectively), and only under the TOPO+MOD condition. The effect appears to be weakly correlated with starting topology optimality, with more optimal starting topologies resulting in shorter runtimes on average. Runtimes for the 1000ARB and Gutell3845 datasets are extremely variable, representing more than a three-fold variation in times. Runs under the RAND condition require fairly similar runtimes to those under the TOPO condition for the smaller datasets, but for the 1000ARB dataset take approximately twice as long on average.

Effect of fixed model parameters

None of the datasets demonstrate that fixing model parameters has a consistent effect on final tree optimality. This suggests that the model parameter values estimated on the starting topologies are close enough to the globally optimal values that they do not favor significantly different topologies. Fixed model parameters do have a consistent effect in shortening average runtimes for all but the smallest dataset, although this effect is only significant for the Angio and Gutell3845 datasets. When model parameters are estimated during a search the average runtimes are approximately 25% longer for the Rbcl and 1000ARB datasets, about 33% longer for the Angio dataset and nearly 40%

longer for the Gutell3845 dataset. This effect is impressive, although if the amount of time required by the model estimation in PAUP* is included (divided by four because each estimate was used by four replicates), the effective speed increase due to fixing model parameters is significantly less.

Overall, the results suggest that fixing model parameters can be an effective way of moderately reducing runtimes without affecting solution quality. If model parameters are estimated a single time on a high quality topology and used in conjunction with many starting topologies, the overhead inherent in this pre-estimation step will be considerably less. Another potential starting condition that was not investigated here would involve starting with a random topology but fixing model parameters at pre-estimated values. This condition might perform quite well, at least on the smaller datasets.

Comparison to alternative ML search software

The runtimes and optimized lnL scores of the topologies returned by PHYML and RAxML searches appear in Table 3-6. Because these programs optimize model parameters during the search and begin searching from a topology estimated by a fast tree inference method, the fairest comparison is to GARLI runs performed with the TOPO starting condition. Any mention in this section of results obtained by GARLI refers only to the results obtained from the TOPO runs. The "Performance Relative to GARLI" entries in 3-6 represent the proportion of GARLI runs for that dataset whose scores were exceeded by each solution. Although many more GARLI runs were performed than were performed with the other programs, this measure should give some indication of relative performance. As noted previously, PHYML's memory requirements made it impractical to perform analyses on the two largest datasets.

The two runs performed with PHYML (one with PHYML's default BioNJ start tree, and one with the best parsimony tree used to seed GARLI) on the three smaller

		Rana			Angio		
Program	Topology	lnL	time (min)	Performance Relative to GARLI	lnL	time (min)	Performance Relative to GARLI
RAxML	Provided	-21812.84	1.8	0.17	-121014.92	53.8	0.08
	Default1	-21813.05	2.0	0.17	-121020.30	67.6	0.04
	Default2	-21831.44	1.6	0.00	-121021.42	48.7	0.04
	Default3	-21831.44	1.6	0.00	-121021.28	44.3	0.04
PHYML	Provided	-21821.78	6.1	0.00	-121151.76	51.0	0.00
	Default	-21860.47	5.6	0.00	-121090.95	70.8	0.00

		Rbcl			1000ARB		
Program	Topology	lnL	time (min)	Performance Relative to GARLI	lnL	time (min)	Performance Relative to GARLI
RAxML	Provided	-85433.22	52.8	0.08	-348424.35	955.8	0.67
	Default1	-85455.25	61.0	0.00	-348484.23	788.5	0.96
	Default2	-85520.52	63.3	0.00	-348724.51	763.0	0.04
	Default3	-85465.45	59.4	0.00	-348467.47	887.5	0.83
PHYML	Provided	-85557.74	32.0	0.00	-	-	-
	Default	-85661.84	34.8	0.00	-	-	-

		Gutell3845		
Program	Topology	lnL	time (min)	Performance Relative to GARLI
RAxML	Provided	-260945.92	2561.2	1.00
	Default1	-261193.52	2001.4	0.29
	Default2	-261386.68	1816.0	0.00
	Default3	-261204.35	2137.4	0.25

Table 3-6. Results of RAxML and PHYML analyses performed on the five real datasets. "Performance Relative to GARLI" represents the proportion of 24 GARLI runs performed under the TOPO condition whose scores are exceeded by the results of each search. Due to its memory requirements, PHYML analyses were not performed on the largest datasets.

datasets result in final solutions with very different optimality. This suggests that the starting topology has a strong effect on the results of PHYML runs. It is thus somewhat unfortunate that the program only implements the deterministic BIONJ algorithm to obtain a starting topology, as this algorithm cannot produce a variety of topologies. On the Rana and Rbcl datasets the run beginning with a provided parsimony tree performed better, while the default BIONJ tree was superior on the Angio dataset. All searches

performed with PHYML resulted in topologies that are lower scoring that any of the 24 GARLI runs performed on the same datasets under the TOPO condition. Because the two algorithms implement the same evolutionary model, these differences in scores cannot be due to different rankings of topologies. The average runtimes for PHYML searches are much shorter than those of GARLI, generally taking between 30% and 50% of the time required by GARLI.

The optimality of the results obtained in RAxML searches are much more comparable to those obtained by GARLI. However, the relative performance of the two programs varied across datasets. Unlike GARLI, RAxML does not provide consistent results on the Rana dataset. Although two of the runs resulted in solutions with scores almost identical to GARLI's, the other two runs resulted in a tree that scores nearly 20 lnL units worse. RAxML performed more consistently when analyzing the Angio dataset, but its scores were less optimal than almost all of the GARLI runs performed under the TOPO condition. These differences in score were all less than 20 lnL. GARLI clearly outperformed RAxML in analyses of the Rbcl dataset. Only one of the RAxML runs resulted in a topology with an optimized score that overlapped with the range of scores obtained by GARLI under the TOPO condition. RAxML runtimes on the three smaller datasets were all much shorter than GARLI's, and shorter than PHYML's runtimes on the Rana and Angio datasets.

The relative performance of RAxML and GARLI is less clear on the two largest datasets. In analyses of the 1000ARB dataset, three of the RAxML results were more optimal than the majority of the GARLI runs. However, the four topologies returned by RAxML on this dataset scored more poorly than all but one GARLI run. Results for the Gutell3845 dataset show a similar pattern, but are even more extreme. On this dataset, the scores of the topologies obtained by RAxML fully bracket the scores of topologies

obtained by GARLI. RAxML runs resulted both in a topology that was more optimal than all GARLI results as well as a topology that was less optimal than all GARLI results. These results demonstrate that RAxML is clearly less consistent than GARLI in obtaining high quality solutions on very large datasets. However, because RAxML's most optimal topologies outscore GARLI's, RAxML may be a better choice for the analysis of these datasets when multiple runs from different starting topologies are performed.

Although it is possible to rank the performance of ML heuristics by simply comparing the final lnL scores of solutions, such a comparison ignores the fact that more optimal solutions may not necessarily be *significantly* more optimal. The issue of what constitutes significant differences between topologies is not a simple one. There is not a standard number of lnL units that may be considered significant across topologies because the distribution of lnL scores depends very strongly on details of the particular dataset. The tests that may be used to assess the significance of likelihood differences between topologies are discussed in detail by Goldman et al. (2000). The most appropriate test in the present case is the Shimodiara-Hasegawa test (SH: Shimodaira and Hasegawa, 1999). This test takes a set of topologies as input and indicates which trees may be considered significantly less optimal. This test was performed in PAUP* using the RELL resampling method. For each dataset the set of trees tested contained all PHYML and RAxML results and all GARLI results under the TOPO condition, for a total of 30 trees. Unfortunately, due to a bug in PAUP*, it is not possible to perform an accurate SH test on very large datasets (D. Swofford, pers. com.). For this reason only the topologies obtained for the Rana, Angio and Rbcl datasets were tested for significance. The results of the SH tests show that almost none of the trees tested on any of these three datasets can be rejected as being significantly less optimal. The only topologies that were rejected resulted from PHYML searches. These were the trees resulting from both

PHYML searches on the Rbcl dataset, and the search using a parsimony starting tree on the Angio dataset.

Another comparison that may be made between the three programs is the degree of topological similarity between the results returned by different runs. Using the same sets of 30 trees tested above, the Robinson-Foulds tree distance metric (RF: Robinson and Foulds, 1981) was calculated between all pairs of topologies. By calculating the average pairwise RF distance between all trees obtained by a single program, a measure of that program's topological consistency was obtained. By calculating the average pairwise distances of topologies between programs, the degree of similarity of the results was measured. The maximum RF distance between topologies for a dataset of *n* taxa is 2(*n*-3). The average pairwise RF distances were transformed into the average proportion of shared bipartitions by first subtracting them from the maximum distance and then dividing by the maximum. The resulting measures of topological similarity are presented in Table 3-7. These results show that all of the programs consistently return fairly similar topologies. PHYML seems to have the least consistency, but because only two trees were included from the program the values may not be very accurate. The average proportion of the topology that is recovered consistently clearly drops for GARLI and RAxML as the datasets become larger.

Simulated dataset analyses

The results of the analyses performed on simulated datasets are summarized in Table 3-8. Although efforts were made to make the simulated datasets as realistic as possible by using a realistic cladogenesis model and simulating under multiple evolutionary models, it is clear that they fall short of reality in terms of the difficulty of tree inference. Even fast searches using NNI branch swapping under the parsimony criterion were able to obtain topologies with less than 20% of the bipartitions inferred

	Average proportion of bipartitions shared in pairwise topological comparisons between					
Dataset	GARLI x GARLI	RAxML x RAxML	PHYML x PHYML	GARLI X RAxML	GARLI x PHYML	RAxML x PHYML
Rana	0.94	0.95	0.81	0.93	0.85	0.87
Angio	0.95	0.90	0.79	0.88	0.78	0.80
Rbcl	0.93	0.88	0.83	0.85	0.78	0.78
1000ARB	0.84	0.86	-	0.82	-	-
Gutell3845	0.63	0.69	-	0.59	-	-

Table 3-7. Topological similarity of results obtained by GARLI, RAxML and PHYML. The values represent the average proportion of bipartitions shared between topologies in pairwise comparisons, with all topologies obtained from each program grouped together.

	Proportion of bipartitions incorrect			
Number of sequences	Parsimony NNI search	PHYML	RAxML	GARLI
500	0.22	0.11	0.12	0.11
1000	0.19	0.14	0.13	0.13
1500	0.16	-	0.12	0.12
2000	0.16	-	0.12	0.11
2500	0.14	-	0.11	0.11
3000	0.15	-	0.11	0.11
3500	0.15	-	0.11	0.12
4000	0.16	-	0.12	0.12

Table 3-8. Topological accuracy of analyses performed on simulated datasets.

incorrectly on all but the 500 sequence dataset. The accuracy of topologies inferred by RAxML, GARLI and PHYML (on the two datasets that were possible with its memory requirements) were very similar. All of the programs attained between 85% and 90% accuracy on all of the datasets ranging from 500 sequences through 4000. All ML analyses resulted in topologies with lnL scores that exceeded that of the true tree, although the scores of the parsimony starting topologies did not. A slight effect of increased taxon sampling on accuracy is apparent in the parsimony starting trees, but for

ML the accuracy is essentially constant across dataset sizes. Runtimes follow the pattern observed in the analysis of the real datasets, with PHYML being the fastest, followed by RAxML and GARLI. GARLI's runtimes for the simulated dataset analyses were similar to the runtimes observed for the real data analyses of datasets of similar sizes.

3.4 CONCLUSIONS

The results presented here demonstrate that the GARLI algorithm is able to perform efficient and consistent ML heuristic searches on very large datasets. It compares quite favorably with two recently developed ML programs that have previously been shown to outperform older software such as PAUP* and fastDNAml (Guindon and Gascuel, 2003; Stamatakis et al., 2005). GARLI clearly outperforms PHYML in terms of solution optimality, although it cannot compete in terms of runtimes. GARLI performs fairly comparably to RAxML across a range of datasets, with GARLI appearing to have an edge in terms of optimality on datasets consisting of less than one thousand sequences.

These results also suggest a set of recommendations about how GARLI should optimally be run. Although the effect of the starting topology on the outcome of searches is somewhat dataset specific, using a wide variety of topologies is clearly to be recommended. For smaller datasets it appears that the optimality of the starting topologies is not particularly important, although it does become so as the number of sequences grow. For datasets of less than several hundred sequences random starting topologies are also a viable option. If accurate estimates of model parameters can be obtained, fixing them during a GARLI search can have a beneficial effect on runtimes with little or no detrimental effect on optimality.

The effects of the large number of parameters controlling the details of the genetic algorithm were not investigated here. GARLI's performance can most likely be improved further by finding the optimal settings of parameters controlling the selection scheme,

mutation probabilities and distribution of topological rearrangements. Future experiments will need to be performed toward this end.

SUPPLEMENTAL INFORMATION

Default Garli Settings

population size = 4 individuals
holdover = 1 individual
selection intensity = 0.5
initial $P_b = 0.5$
minimum $P_b = 0.01$
number of P_b reductions = 40
topology mutation weight = 1.0
model mutation weight = 0.05
branch-length mutation weight = 0.2
NNI mutation weight = 0.2
SPR mutation weight = 0.3
limSPR mutation weight = 0.5
limSPR reconnection range = 6
mean number of branch-length mutations = 5
gamma shape of model mutation multiplier = 1000
gamma shape of branch-length mutation multiplier = 1000
update interval length = 100
update intervals stored = 5
maximum memory for conditional likelihoods = 1600 mb

Model parameters used for simulation

Rana: Rate Matrix: (2.21, 8.21, 2.37, 0.91, 16.54, 1.0)

Base Frequencies: (0.35, 0.21, 0.15, 0.29)

Alpha shape parameter of gamma rate distribution: 0.56

Proportion of invariable sites: 0.34

Rbcl: Rate Matrix: (1.08, 3.72, 0.34, 1.61, 3.03 1.0)

Base Frequencies: (0.30, 0.19, 0.15, 0.36)

Alpha shape parameter of gamma rate distribution: 0.58

Proportion of invariable sites: 0.23

Chapter 4: The Parallel GARLI Algorithm

4.1 Introduction

As large computer clusters have become less costly and more common in recent years, they have presented a new resource to be applied to phylogenetic inference. The notion that multiple processors can be used to explore the extremely large search spaces presented by phylogenetic tree searches is an attractive one. It becomes even more attractive as the number of sequences analyzed increases or when computationally intensive probabilistic methods are used. Systematists should recognize that common search methodologies such as performing multiple independent searches from distinct starting topologies are inherently parallel in nature. Beyond such embarrassingly parallel independent searches, a nearly infinite number of schemes may be imagined in which multiple processors work in a coordinated fashion to perform a search. Several simple and easily controllable schemes of coordinated searching have been implemented in the parallel GARLI algorithm (henceforth P-GARLI).

A fine (yet important) distinction may be made between classes of parallel algorithms. What might be termed a "typical" parallel algorithm is created by taking a particular serial algorithm and dividing the required computation between multiple processors. By definition, this parallel algorithm must result in the same solution as the serial algorithm on which it is based, but ideally in a shorter period of time. The best case scenario in such implementations is to develop a parallel algorithm that allows a linear speedup, meaning that the parallel runtime is equal to the serial runtime divided by the number of processors used. In practice, various sources of computational overhead cause parallel algorithms to fall short of this ideal goal.

Other parallel algorithms do not generalize a particular serial algorithm to allow the use of multiple processors, but rather develop a different algorithm that may leverage the power of multiple processors in unique ways. This sort of parallel application may improve upon serial algorithms in terms of solution quality as well as in runtime. Improvements in solution quality might come in the form of an increased probability of returning high quality solutions, or in the ability to return solutions more optimal than can be obtained with serial analyses. It is with these goals in mind that P-GARLI was designed.

A number of programs able to perform ML phylogenetic inference on parallel computer systems are available. These include parallel versions of fastDNAml (Stewart et al., 2001), GAML (Brauer et al., 2003), and RAxML (Stamatakis et al., 2005). Each of these is a direct parallelization of the respective serial algorithm, and thus the application of parallel resources acts only to decrease runtimes. The approach taken by P-GARLI is to implement a more coarse-grained form of parallelism, in which each processor performs something akin to the serial GARLI algorithm. This approach is particularly attractive in the GA framework, where the coexistence of multiple partially independent populations has been shown to be an efficient search strategy (see Cantu-Paz, 2001). Such a meta-population GA approach is also taken by MetaPIGA (Lemmon and Milinkovich, 2002), although it is implemented for execution on a single processor.

4.2 THE P-GARLI ALGORITHM

Algorithm overview

An astounding number of approaches to the implementation of a parallel GA have been presented in the GA literature. The parallel GA taxonomy of Nowostawski and Poli (1999) divides these approaches first by whether they implement a single population or

multiple partially independent populations, also termed demes or metapopulations. Multiple population approaches are further categorized by the number of individuals in each population and the scheme of inter-population migration. The approach taken by P-GARLI uses a moderate number of populations, with a small number of individuals in each. A single population is designated the master, and all other populations are slaves. Migration is hierarchical, such that the master may communicate with all slaves, but each slave may communicate only with the master. Note that although I will term this a master-slave migration scheme, the term "master-slave parallelization" is used in the GA literature to refer to a single population implementation in which the slave processors are used to evaluate the fitness of individuals (as in the parallel implementation of GAML).

P-GARLI's communication scheme was chosen partly because of its ease of implementation, but also because it lends itself well to bestowing the global decision making power of the parallel algorithm on a single processor. A run of the parallel algorithm may be considered a single distributed search, with the master serving to collect the solutions obtained by the slaves and optionally to exert control over where in the space they search. The master maintains its own population of solutions and performs the typical GA activities of reproduction, mutation and selection, in addition to tasks related to the management of the slaves. One clear downside of a master-slave communication scheme is that it does not scale well to large numbers of processors because the single master becomes overwhelmed. One potential solution to this scalability problem is to add another level to the hierarchy. In such a scheme, groups of slave populations would communicate with sub-master populations that in turn act as slaves to a single "super-master" population. I have not yet attempted this approach in P-GARLI..

The parallel algorithm begins by creating starting populations of individuals for each processor. As with the serial algorithm, the starting topologies may either be random or user-specified. Different starting topologies may be specified for each slave population, allowing a large amount of topological variation to be present from the outset. Each population then begins evolving more optimal solutions using a process very similar to the serial GARLI algorithm. In addition to the (serial repertoire) the master also applies an operator that creates new topologies by recombining existing ones. Migration between populations occurs under certain conditions that will be discussed below. The termination condition is the same as that of the serial algorithm (i.e., no changes in topology that increase the lnL score significantly over a large number of generations), but the possible topology changes also include new topologies received in migration and those created by recombination.

Inter-population communication

Communication between populations in P-GARLI is implemented through calls to standard MPI libraries (Message Passing Interface: see www.unix.mcs.anl.gov/mpi/). MPI is a flexible (although somewhat archaic) standard for passing arbitrary messages between processors connected by any type of network. Two general types of MPI communication may be distinguished, non-blocking and blocking (also asynchronous and synchronous). By analogy, a non-blocking message is similar to mailing a letter, in that the sender may move on to other tasks immediately after the message has been sent. Blocking messages, on the other hand, may be likened to a telephone call in which the sender cannot perform other activities until the receiver answers and the message is communicated. Blocking messages are typically called for when detailed coordination between processes is necessary, usually taking the form of messages being passed back and forth that are conditional on one another. The potential drawback of blocking

messages is that processors may not perform any computation until messages have been sent or received. As will become apparent, the P-GARLI algorithm does not require this sort of detailed coordination between populations. For this reason, P-GARLI makes use of non-blocking messages only, ensuring that processors never waste time waiting for the receiver of a message to be available.

P-GARLI implements a very simple means of transmitting the information constituting an individual solution to another population. The topology of the individual to be sent is simply transformed into Newick tree representation, including branch lengths, and then sent as a string of characters. These messages represents a very modest amount of network traffic, as the string representing a 1000 taxon tree requires a message of less than 30 kilobytes. A short message encoding the model parameter values is also sent, but in binary rather than character format. Once a message is received, it is transformed back into an internal representation using the same function used to read starting topologies from file.

Master-slave roles

The tasks performed by each of the slaves are nearly identical to those performed by the serial GARLI algorithm. All settings used to specify aspects of the serial algorithm, such as operator probabilities, operator details and selection intensity may be set independently for the master and slaves. In addition, the slaves adaptively alter mutation probabilities and reduce the branch-length optimization parameter (P_b) independently. The only added task for the slave populations is the requirement to communicate with the master. At the end of each generation, the slave checks if a pre-specified send interval (measured in seconds) has been exceeded. If it has, the slave sends a copy of its currently most optimal individual to the master. At the end of each generation the slave also checks to see if a new individual has arrived from the master. If

so, the string representation is converted into an individual. The new individual then replaces the least fit individual in the population at that point. Once it is inserted into the population, the new individual behaves identically to the other individuals. The new individual will quickly dominate the population if it is of higher fitness than had previously been encountered by the slave, and will quickly be lost otherwise.

The master is tasked with coordinating the searches performed by the slaves, and also functions as a melting pot for the diversity of solutions found by them. However, when population sizes are small, variation may easily be lost by chance (i.e., genetic drift). In order to ensure the master population's ability to maintain the full level of variation generated by the slaves, I have implemented a technique that I refer to as migrant shielding. This simply amounts to maintaining a single copy of the most recent migrant from each slave within the master population, and shielding those individuals from both mutation and selection. The shielded individuals are copied unchanged into each successive generation, and are replaced when a new migrant arrives. The master population thus consists of a number of individuals identical in behavior to those of the serial algorithm, plus one shielded individual for each of the slave populations. In terms of leaving offspring, the shielded individuals act identically to all others, doing so in proportion to their fitness. Maintaining the shielded individuals in the population has little cost from either a computational or search efficiency perspective. These individuals need only have their score evaluated once, and if they are of low fitness they simply will not reproduce. When the fitness of one or more shielded individuals is near the best in the master population, they may have a large impact on future generations of solutions. Most importantly, the shielded individuals provide a large amount of variation to be operated upon by the recombination operator, the details of which will be discussed shortly.

At the end of each generation, the master checks if messages have arrived from any slaves. If so, the message is converted into an individual which replaces the individual previously received from that slave. The lnL score of the newly received individual is immediately evaluated, and represents the highest fitness attained by the slave that sent it. The master compares this score (lnL_s) to the score of the most fit individual that has yet been encountered (lnL_{max}), and may choose to send a copy of the globally best individual to that slave. Specifically, the master sends an individual if the difference in score is greater than a specified replacement threshold, R:

$$(lnL_{max} - lnL_s) > R$$

R effectively determines the global behavior of the multi-population search, and the implications of specific values will be discussed below. After sending an individual, the master immediately moves on to evaluate individuals received from other slaves or to proceed with the next GA generation.

Beyond coordinating the efforts of the slaves, the primary goal of the master is to generate new solutions by using the diversity of solutions provided by the slaves. This is done by creating one new individual per generation through the application of a simple recombination operator. One recombination partner is chosen randomly from all individuals in the population, using the same selection function used to choose the parents of offspring individuals. The second recombination partner is chosen from among the shielded individuals only, using a selection function with very low selection intensity (s = 0.01). Thus, the first recombination partner is likely to be among the best solutions yet found, and ideally the second partner will contribute novel topological variation found by one of the slaves. Once the recombination partners have been chosen, the

recombination operator finds a bipartition that is present in both recombination partners, but ensures that the subtrees defined by the bipartition do not have identical branching structures. If such a bipartition can be found, the subtree of the first individual is removed and that of the second copied in its place. Finally, the full branch-length optimization procedure described in chapter 2 is applied, centered on the node at the base of the swapped subtree. Note that the recombination procedure results in only one recombinant offspring individual because the subtree is only swapped in one direction.

P-GARLI's mode of recombination is essentially the same as that applied in MetaPIGA (Lemmon and Milinkovich, 2002), which the authors term RCM. This type of recombination has been noted to greatly outperform the recombination operator implemented in GAML (Lewis, 1998), which often resulted in very drastic topology changes. In the P-GARLI implementation, the two recombination partners are specifically chosen to represent one of the best known solutions and a potentially very different solution whose selection is only weakly determined by its fitness. The rationale behind this is that large topologies may contain novel, highly optimal portions even if they are not very optimal on the whole.

Parallel strategies

There are a number of ways that multiple GA populations might be managed to improve search performance on large datasets. As noted previously, the replacement threshold *R* controls the global search strategy of the parallel algorithm. The potential strategies may be envisioned as lying along a continuum representing the degree of independence of the searches performed by the slave populations. At one extreme lies a strategy of complete slave independence, and at the other a strategy of very tight coordination of slaves. At the independence extreme, each slave performs its own search from start to finish, never receiving input from the master. Such an approach would be

specified by a very large value of *R*. This approach ensures that the maximal amount of variation is maintained in the master population, to be acted upon by recombination. At the extreme of no independence, specified by a small value of *R*, the master population very frequently sends the best known individual to the slaves. This amounts to the master frequently redirecting the search effort across all populations to the most promising region of the search space that has yet been found. Recombination is not expected to be of significant benefit with this strategy because little variation is maintained. Between these extremes, moderate values of *R* allow the master to give independence to those slaves who are producing promising results, but to redirect those who are doing the most poorly.

The worst case scenario for the independent P-GARLI search strategy would result if no additional increases in fitness were achieved through recombination. In such a case the search would be expected to perform no worse than if a number of serial runs equal to the number of slave populations had been performed. The worst case scenario for the highly coordinated search strategy is more worrisome. Because this parallel search simply directs all search effort to a small region of the search space, it may be quite possible for it to become entrapped in a local optimum. Thus, a coordinated parallel search could potentially perform no better than a single serial search, despite its use of more processing power.

On a conceptual level, the optimal value of *R* depends largely on the topography of treespace for the dataset being analyzed. If local topological optima are not prevalent, the GA search will be limited in its ability to increase the score by the rate at which beneficial rearrangements are performed. Because rearrangements are chosen randomly, some may not be attempted for some time due to chance. In such a case, a low value of *R* will cause more concentrated application of rearrangements to the best solution, speeding

hill-climbing. On the other hand, if local optima are common, their avoidance is critical if the search is to attain a highly optimal solution. A large value of R then allows the searches to explore many areas of treespace, effectively hedging its bets.

P-GARLI also implements a mechanism that allows the value of R to decrease over the course of a run. This approach, termed a hybrid search, is appealing in that it allows for very dispersed exploration of the search space early in a run, and allocates more of the search effort to the most promising solutions as the run progresses. Thus, it exploits the best aspects of the independent and coordinated parallel search strategies. The rate of decrease of R could be tied to a variety of measurable attributes of the search. For the sake of simplicity, the current P-GARLI implementation causes it to be reduced according to the same conditions as the branch-length optimization parameter, P_b. Starting and minimum values of R are specified by the user. Each time the master population reduces P_b, R is reduced as well. R is reduced from its starting to minimum values in the same number of steps specified for the reduction of P_b.

Chapter 5: Performance of the Parallel GARLI algorithm

5.1 INTRODUCTION

In contrast to many parallel algorithms, the primary aim of P-GARLI is not to reduce runtimes relative to the serial algorithm. Instead, the algorithm seeks to find more optimal solutions by coordinating the searches of multiple GA populations to perform a more thorough search of tree space. Ideally, the algorithm should function at least as well as the sum of its parts. For the parallel algorithm to be a viable choice for analysis, the solution obtained when utilizing a certain number populations in P-GARLI should on average be at least as optimal as the best solution obtained in an equal number of runs of the serial GARLI algorithm. If this is not the case, then the use of the parallel algorithm presents no benefit over the use of serial GARLI. As was the case with experiments performed using the serial algorithm in chapter 3, the primary means of gauging algorithm performance will be to compare solution optimality and runtimes.

5.2 METHODS

Computational setup

All parallel analyses were performed on Lonestar, a Rocks version 4.0 Linux cluster (www.rocksclusters.org) maintained by the Texas Advanced Computing Center (www.tacc.utexas.edu). Lonestar comprises 512 dual processor compute nodes, with a mixture of 3.06 GHz and 3.2 GHz Intel Xeon processors. To assure consistency of results, only 3.06 GHz processors were used for experiments, and only one processor was used per node (the other processor remained idle). Each node contains 2 GB of system memory. Other incidental analyses (searches to obtain starting topologies and to optimize final results) were performed on Phylocluster2, a small Rocks version 3.2 cluster with

five dual processor compute nodes. Apart from the fact that Phylocluster2 contains 2.8 GHz Xeon processors, the two clusters used are essentially identical. The source code of P-GARLI was compiled with the Intel icc compiler, version 8.0. The compiler flags specified were "–O2 –ip –fno-alias", which proved to result in the fastest code execution.

Because Lonestar enforces a maximum runtime limit of 24 hours, parallel runs taking longer that this had to be restarted where they left off. This was done simply by starting each of the populations with the last tree that had been written to file by each. Parameters such as the branch-length optimization parameter were also set to their last values. The program unfortunately does not yet implement checkpointing, which would make this process simple and automated.

Dataset

The P-GARLI search algorithm is most likely to be of benefit on very large datasets, where the larger and more complicated search spaces may lead the single population algorithm to become entrapped in local optima. For this reason, as well as because of computational limitations, only a single large dataset was used in the experiments presented here. This was the Gutell3845 dataset, the largest examined in chapter 3.

Starting conditions

One of the potential strengths of allowing multiple partially independent populations is the ability to maintain a large amount of topological variation during a run. Variation may be ensured from the outset of a run by seeding each population with a different starting topology. To obtain a topologically diverse collection of starting trees, simple parsimony searches using NNI branch swapping were performed in PAUP*. One hundred replicate searches were performed per dataset, each beginning from a tree

obtained by stepwise addition with random addition sequences. From the 100 resulting trees, 16 were randomly chosen to be used as starting points for the parallel analyses. The 16 topologies were ordered randomly. Parallel runs were then performed with each population starting from a successive starting tree. Thus, runs using four populations used the first four trees, runs using eight trees used the first eight trees, and runs using sixteen populations used all of the trees. This procedure accentuates one potential benefit of using larger numbers of populations. To investigate whether this method of introducing topological diversity is in fact beneficial, one set of experiments was also performed in which all populations were seeded with the best of the 16 starting topologies.

Experiments performed in Chapter 3 demonstrated that fixing model parameters at pre-estimated values during a GARLI search is an effective way of reducing runtimes for analyses of large datasets. For this reason, all parallel runs were performed with all populations fixed at the same set of model parameters. The parameter values used were those estimated on the best of the starting topologies for each dataset used in the serial experiments in Chapter 3. One other potential benefit of fixing model parameters in the context of a parallel run is that it ensures that the topologies obtained by each remote population may be more easily grafted together by the recombination operator.

Experimental design

As was the case with the serial experiments performed in Chapter 3, a nearly limitless number of combinations of GA settings are possible in analyses using the P-GARLI algorithm. Because of the additional computational resources required to perform parallel analyses, the majority of the experiments investigated the effect of two main variables: population coordination scheme and the number of populations. Because of the large computational requirements, only two replicates were performed for each combination of variables. The primary set of experiments amounts to 18 total runs (2

replicates x 3 communication schemes x 3 population numbers). In addition, six other runs using a single starting topology for all populations in conjunction with the independent coordination scheme were performed to determine the importance of starting parallel runs with a diversity of topologies. These exploratory experiments will allow future experiments to be designed to investigate the optimal settings for a P-GARLI analysis in more detail. All general GA population settings were the same as those used for the serial analyses in Chapter 3.

Population coordination scheme

Three schemes of slave population coordination were examined. The schemes are determined by the setting of *R* (the replacement threshold), as described in chapter 4. The schemes used were:

1. Full slave independence (*R* = infinity). In this scheme the master receives individuals from the slaves, but never sends them. This coordination scheme was used both in runs using different starting topologies for all populations and for runs using a single starting topology.
2. Highly coordinated search (*R* = 5). In this scheme the master sends the best-known individual to any slaves whose best score fall more than 5 lnL behind the globally best score.
3. Hybrid search (initial *R* = 2000, minimum *R* = 5, geometric reduction from initial to minimum value in 40 increments). In this scheme the master performs as in scheme 1, until the rate at which new topologies are found slows, and then gradually transitions into scheme 2.

Processor numbers

Determining how the number of processors (and GA populations) applied to a P-GARLI run affects the results is of clear interest. There is likely to be a point of diminishing returns, after which applying more processors no longer increases performance. Because the current implementation only allows a single master population that must communicate with all slaves, the algorithm is unlikely to scale well to very large numbers of populations. To initially investigate the effect of processor number, runs were performed using 4, 8 and 16 processors.

5.3 RESULTS AND DISCUSSION

Runtimes and the scores obtained after optimization of all final results in PAUP* appear in table 5-1. Figure 5-1 graphically presents the optimized lnL scores of the solutions found in all parallel analyses. Also appearing are the results of the most comparable of the serial results (those performed using the TOPO+MOD condition) for the Gutell3845 dataset presented in Chapter 3. One fact that is immediately apparent is that although there is overlap in the distributions of scores obtained by the serial and parallel analyses, the best parallel solutions are significantly better than the best obtained by the serial algorithm.

Runs using diverse starting topologies

Comparing the solutions obtained across the three types of parallel population coordination, none of the coordination schemes is obviously superior. The small number of replicates makes it difficult to draw definite conclusions, but it tentatively appears that the Hybrid scheme is the most successful, resulting in the highest scores and the least variance between runs. If the Hybrid scheme is indeed the best performing, it would

suggest that there is benefit both in allowing independent searches early in a run and in performing very intensive mutations of the best solution late in a run.

	Coordination Scheme							
	Independent		Coordinated		Hybrid		Independent-1 Start	
# Processors	lnL	time(min)	lnL	time(min)	lnL	time(min)	lnL	time(min)
4	-261064.3	2418	-261160.8	1856	-261018.6	3598	-260897.2	2014
	-261181.0	1804	-261370.5	1440	-261052.0	1955	-260917.0	1837
8	-260977.8	1691	-261115.2	2287	-260916.3	1965	-260958.7	1211
	-260868.1	1969	-260974.9	1627	-260926.0	1912	-260895.1	1972
16	-261094.7	1378	-260878.0	5159	-260937.2	1109	-260901.6	1685
	-260898.8	2226	-260883.2	5601	-260819.3	5176	-260895.1	1439

Table 5-1. Optimized scores and runtimes of P-GARLI analyses of the Gutell3845 dataset.

Comparing results across processor numbers is somewhat more conclusive. Within each communication scheme, all runs performed with eight processors outperformed all runs performed with four processors. None of the four-processor runs surpassed the best of the serial runs. The results for sixteen-processor runs are somewhat mixed, with some runs performing more poorly than some of the eight-processor runs. Nonetheless, five of the six sixteen-processor runs exceed the best score obtained in any of the serial runs, suggesting that increasing to this number of processors may be helpful.

Because the scores obtained by the four-processor runs completely overlap with those obtained by serial runs, it must be determined whether there is any benefit to performing parallel runs with this number of processors. To determine whether the four-processor parallel runs obtained scores better than would be expected from four serial runs, a simulation was performed. In each replicate, four of the twenty-four scores obtained in the serial runs were randomly chosen, and the best of the four was noted. Out of 10^6 replicates, the average of these best scores was -261061.7 lnL. This value falls in the middle of the scores obtained in the six four-processor parallel runs, making it unclear

whether parallel runs with this number of processors offer any benefit over performing serial analyses. A similar simulation was performed for random selections of eight scores, and resulted in an average score of -261019.3 lnL. All but one of the eight-processor runs greatly exceeded this value, suggesting that utilizing a larger numbers of processors may increase the usefulness of the parallel algorithm. Such a simulation is unnecessary for the sixteen-processor runs, as all but one exceeded the best of the scores obtained in the serial runs.

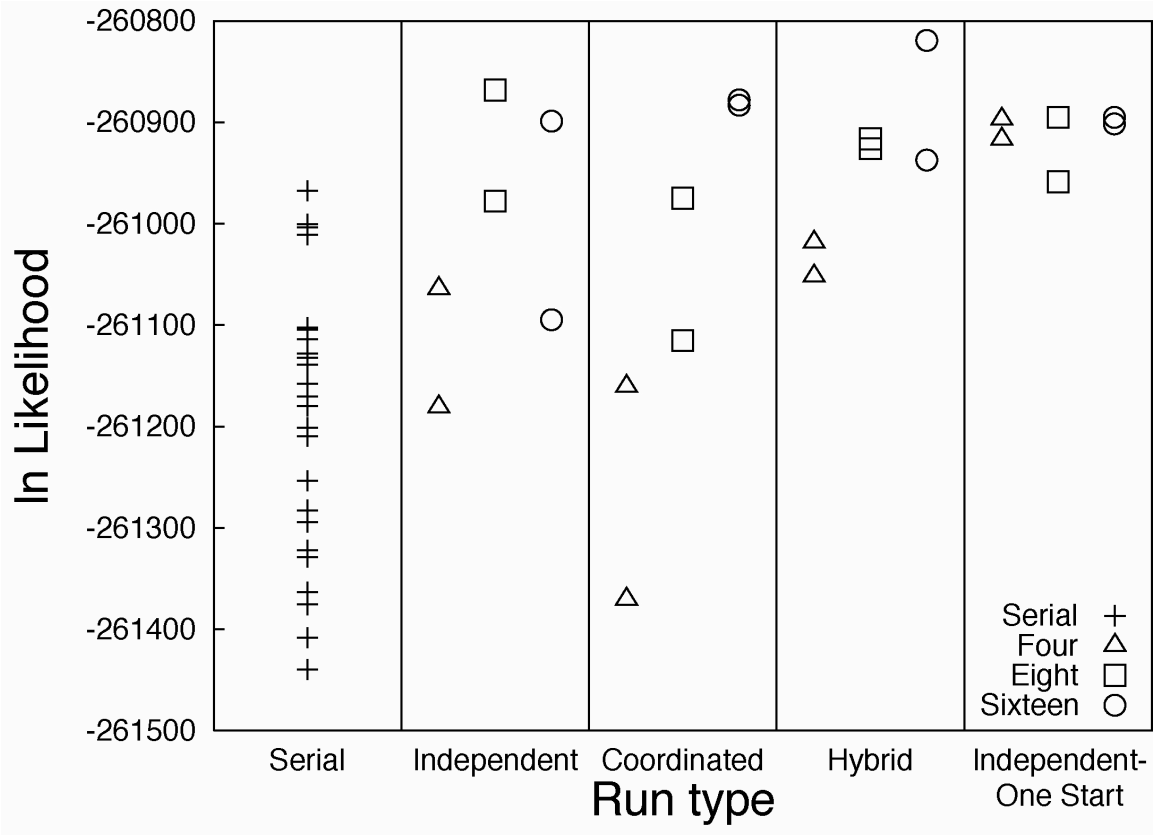

Figure 5-1. Optimized lnL scores of the solutions resulting from P-GARLI analyses of the Gutell3845 dataset. The results of 24 serial analyses appear for comparison.

Figure 5-2 presents the relationship between optimized lnL scores and runtimes for the three communication schemes and the serial runs. The majority of the parallel runs had runtimes that were similar to those of the serial runs. However, several runs under the Coordinated and Hybrid schemes took much longer. Looking more closely at the output of these runs makes it clear that the very long runtimes are largely an artifact of the way that the automated termination condition is defined. When the value of *R* (the threshold difference in lnL score necessary for the master to send a new individual to a slave) is small, the number of GA generations that the master can perform in a given amount of time is many fold lower than the number of generations that may be completed

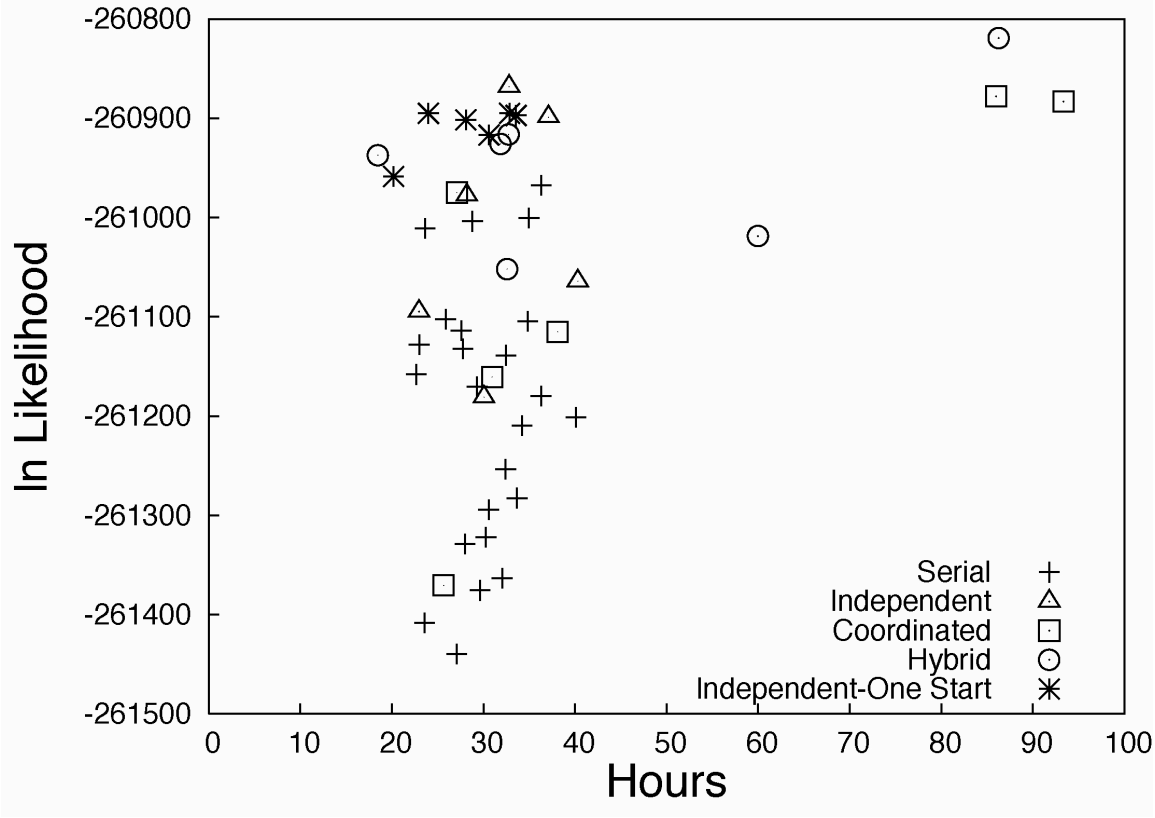

Figure 5-2. Runtimes vs. lnL scores for parallel analyses of the Gutell3845 dataset. The results of 24 serial analyses appear for comparison.

when R is large. Because the termination condition is defined in terms of the number of generations without topological change, the slow progression of generations artificially makes the termination condition more difficult to achieve.

There are two reasons for the slower progression of generations when R is small. First, small values of R increase the amount of communication the master must perform because it must send individuals to the slaves quite often. The second and more important reason is that small values of R lead to an increased computational burden due to scoring topologies. When R is large, few of the shielded individuals in the master population have high enough scores to leave offspring except through recombination. However, when R is small all individuals in the master population have very similar scores, and all are likely to leave offspring. Because there is insufficient memory available to store likelihood calculations for all individuals, a small value of R means that many individuals must be rescored from scratch each generation.

Runs using a single starting topology

Runs performed using the independent coordination scheme and a single good starting topology for all populations performed quite well. This set of runs shows much less variance in scores than do the runs using any of the population coordination schemes and different starting topologies for each population. More importantly, the scores obtained in all six of these runs surpassed the best of the serial runs. The most striking increase in performance is seen in the four-processor runs. It is not clear that the number of processors used has any effect on final results when all runs are started from a good starting topology.

5.4 CONCLUSIONS

The exploratory experiments presented here demonstrate that under some conditions the P-GARLI algorithm is able to outperform the serial GARLI algorithm in terms of the scores obtained. Over half of the parallel runs resulted in scores more optimal than those obtained by the best of the serial runs performed on the Gutell3845 dataset. However, attempts to incorporate a topologically diverse set of starting points seem to not have been as beneficial as had been expected. This does not necessarily mean that variation between populations is unimportant during a parallel run. More detailed examination of the results of runs performed with a single starting topology (not shown) indicates that the independent populations diverge from one another immediately after a run is started. Preliminary results suggest that this variation is indeed an important resource that the master may use to attain more optimal scores through topological recombination. The experiments presented here suggest that it may be best to start all parallel populations from the best topology that can be obtained, and to leave the generation of diversity to the stochastic nature of the GA.

Several potential changes and improvements to the parallel algorithm are suggested by the results of the experiments performed here. The parallel termination condition should be rethought so that it is more comparable between the different communication schemes. One way of doing this would be to define the criterion it in terms of a length of time without improvement, rather than in terms of numbers of generations. Alternative communication schemes should also be investigated. In particular, a scheme in which the value of R is not equal for all populations, but varies across them. In such a scheme some populations with large values of R would allow variation to be maintained, while others with low values of R could intensively search in the vicinity of the current best solution.

References

Akaike, H. 1973. Information theory as an extension of the maximum likelihood principle. pp. 267-281 in B. N. Petrov and F. Csaksi, editors. 2nd International Symposium on Information Theory. Akademiai Kiado, Budapest, Hungary.

Allen, B., and M. Steel. 2001. Subtree transfer operations and their induced metrics on evolutionary trees. Ann. Combinator. 5:1-13.

Brauer, M.J., M. T. Holder, L. A. Dries, D. J. Zwickl, P. O. Lewis, and D. M. Hillis. 2002. Genetic algorithms and parallel processing in maximum-likelihood phylogeny inference. Mol. Biol. Evol. 19(10):1717-1726.

Brown, J., and P. Warren. 1998. Antibiotic discovery: is it in the genes? Drug Discovery Today 3:564-566.

Buckley, T. R., C. Simon, and G. K. Chambers. 2001. Exploring among-site rate variation models in a maximum likelihood framework using empirical data: effects of model assumptions on estimates of topology, branch lengths, and bootstrap support. Syst. Biol. 50:67-86.

Burnham, K. P., and D. R. Anderson. 2002. Model selection and multimodel inference: a practical information theoretic approach. 2nd ed. Springer, New York.

Bush, R. M., C. A. Bender, K. Subbarao, N. J. Cox, and W. M. Fitch. 1999. Predicting the evolution of human influenza. Science 286(5446):1921–1925.

Cantu-Paz, E. 2001. Efficient and Accurate Parallel Genetic Algorithms. Kluwer Academic Publishers, Norwell, Massachusetts.

Chase, M. W., D.E. Soltis, R.G. Olmstead, D. Morgan, D.H. Les, B.D. Mishler, M.R. Duvall, R.A. Price, H.G. Hills, Y.L. Qiu, K.A. Kron, J.H. Rettig, E. Conti, J.D. Palmer, J.R. Manhart, K.J. Sytsma, H.J. Michaels, W.J. Kress, K.G. Karol, W.D. Clark, M. Hedren, B.S. Gaut, R.K. Jansen, K.J. Kim, C.F. Wimpee, J.F. Smith, G.R. Furnier, S.H. Strauss, Q.Y. Xiang, G.M. Plunkett, P.S. Soltis, S.M. Swensen, S.E. Williams, P.A. Gadek, C.J. Quinn, L.E. Eguiarte, E. Golenberg, G.H. Learn, Jr., S.W. Graham, S.C.H. Barrett, S. Dayanandan, and V.A. Albert. 1993. Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcL. Annals of the Missouri Botanical Garden, 80:528–580.

Congdon, C. B. 2001. Gaphyl: A genetic algorithms approach to cladistics. pp. 67-78 in Principles of Data Mining and Knowledge Discovery, ed. L. DeRaedt and A. Siebes. Lecture Notes in Computer Science, No. 2168. Spinger-Verlag, Berlin.

Faith, D. P. 1992. Genetic diversity and taxonomic priorities for conservation. Biol. Conserv. 68:69–74.

Felsenstein, J. 1978. Cases in which parsimony and compatibility will be positively misleading. Syst. Zool. 27: 401-410.

Felsenstein, J. 1981. Evolutionary trees from DNA sequences: A maximum likelihood approach. J. Mol. Evol. 17:368-376.

Fisher, R.A., 1930. The Genetical Theory of Natural Selection. Oxford University Press, Oxford.

Fleissner, R., D. Metzler, and A. von Haeseler. 2005. Simultaneous statistical multiple alignment and phylogeny reconstruction. Syst. Biol. 54:548–561.

Foster, J.A. 2001. Evolutionary computation. Nat. Rev. Genet. 2:428-436.

Fukami-Kobayashi, K., and Y. Tateno. 1991. Robustness of maximum likelihood tree estimation against different patterns of base substitutions. J. Mol. Evol. 32(1):79-91.

Gascuel, O. 1997. BIONJ: an improved version of the NJ algorithm based on a simple model of sequence data. Mol. Biol. Evol. 14(7):685-695.

Goldman N, J.P. Anderson, and A.G. Rodrigo. 2000. Likelihood-based tests of topologies in phylogenetics. Syst. Biol. 49(4):652-70.

Goldman, N. 1993. Simple diagnostic statistical tests of models for DNA substitution. J. Mol. Evol. 37(6):650-661.

Gu, X., Y. X. Fu and W. H. Li. 1995. Maximum likelihood estimation of the heterogeneity of substitution rate among nucleotide sites. Mol. Biol. Evol. 12(4):546-557.

Guindon, S., and O. Gascuel. 2003. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Syst. Biol. 52(5):696-704.

Hasegawa M, H. Kishino and T. Yano. 1985. Dating of the human-ape splitting by a molecular clock of mitochondrial DNA. J. Mol. Evol. 22(2):160-174.

Higgins, D., Thompson J., Gibson T. Thompson J.D., Higgins D.G., and Gibson T.J. 1994. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res. 22:4673-4680.

Hillis, D. M. 1996. Inferring complex phylogenies. Nature 383:130–131.

Hillis, D. M., and T. P. Wilcox. 2005. Phylogeny of the New World true frogs (*Rana*). Mol. Phylogenet. Evol. 34(2):299-314.

Holland, J. H. 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Huelsenbeck, J. P. 1995. Performance of phylogenetic methods in simulation. Syst. Biol. 44:17–48, 1995.

Huelsenbeck, J. P., and D. M. Hillis. 1993. Success of phylogenetic methods in the four taxon case. Syst. Biol. 42:247–264.

Huelsenbeck, J. P., and F. Ronquist. 2001. MRBAYES: Bayesian inference of phylogenetic trees. Bioinformatics 17(8):754–755.

Hull, D. 1988. Science as a Process: An Evolutionary Account of the Social and Conceptual Development of Science, University of Chicago Press, Chicago.

Lanave, C., Preparata G., Saccone C., and Serio G. 1984. A new method for calculating evolutionary substitution rates. J. Mol. Evol. 20(1):86-93.

Lemmon, A. R. and M. C. Milinkovitch. 2002. The metapopulation genetic algorithm: An efficient solution for the problem of large phylogeny estimation. Proc. Natl. Acad. Sci. USA. 99(16):10516-10521.

Lewis, P.O. 1998. A genetic algorithm for maximum-likelihood phylogeny inference using nucleotide sequence data. Mol. Biol. Evol. 15(3)277-283.

Ludwig, W., O. Strunk, R. Westram, L. Richter, H. Meier, Yadhukumar, A. Buchner, T. Lai, S. Steppi, G. Jobb, W. Förster, I. Brettske, S. Gerber, A.W. Ginhart, O. Gross, S. Grumann, S. Hermann, R. Jost, A. König, T. Liss, R. Lüssmann, M. May, B. Nonhoff, B. Reichel, R. Strehlow, A. Stamatakis, N. Stuckmann, A. Vilbig, M. Lenke, T. Ludwig, A. Bode, and K.H. Schleifer. 2004. ARB: A Software Environment for Sequence Data. *Nucl. Acids Res.*, 32(4):1363–1371.

Lundy M. and A. Mees. 1986. Convergence of an annealing algorithm. Math. Program. 34:111-124.

Lunter, G., I. Miklos, A. Drummond, J. Jensen, and J. Hein. 2005. Bayesian coestimation of phylogeny and sequence alignment. BMC Bioinformatics 6:83.

Maddison, D. R. 1991. The discovery and importance of multiple islands of most-parsimonious trees. Syst. Zoo. 40:315-328.

Matsuda, H. 1996. Protein phylogenetic inference using maximum likelihood with a genetic algorithm. pp 512-523 in L. Hunter and T. E. Klein, editors, Pacific Symposium on Biocomputing '96. World Scientific, London.

Mau, B., M. A. Newton, and B. Larget. 1999. Bayesian phylogenetic inference via Markov chain Monte Carlo methods. Biometrics 55(1):1-12.

Mitchell, M. 1996. An Introduction to Genetic Algorithms: Complex Adaptive Systems. MIT Press, Cambridge, Mass.

Moilanen, A. 1999. Searching for most parsimonious trees with simulated evolutionary optimization. Cladistics 15:39-50.

Moret, B. M. E., L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. 2004. Phylogenetic networks: modeling, reconstructibility, and accuracy. IEEE/ACM Transactions on Computational Biology and Biocomputing, 1.

Nowostawski, M., and R. Poli. 1999. Parallel genetic algorithm taxonomy. In Proceedings of the third international conference on knowledge-based intelligent information engineering systems (KES'99), pp 88-92.

Olsen, G. J., H. Matsuda, R. Hagstrom, and R. Overbeek. 1994. fastDNAml: A tool for construction of phylogenetic trees of DNA sequences using maximum likelihood. Comput. Appl. Biosci. 10: 41-48.

Ou, C. Y., C.A. Ciesielski, G. Myers, C.I. Bandea, C.C. Luo, B.T. Korber, J.I. Mullins, G. Schochetman, R.L. Berkelman, and A.N. Economou. 1992. Molecular epidemiology of HIV transmission in a dental practice. Science 256(5060):1165–1171.

Poe, S. 1998. The effect of taxonomic sampling on accuracy of phylogenetic estimation: A test case of a known phylogeny. Mol. Biol. Evol. 15:1086–1090.

Posada D., and K. A. Crandall. 2001. Selecting the best-fit model of nucleotide substitution. Syst. Biol. 50(4):580-601.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1992. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press, Cambridge.

Rambaut, A., and N. C. Grassly. 1997. Seq-Gen: an application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. Comput. Appl. Biosci. 13:235-238.

Rannala, B., and Z. H. Yang. 1996. Probability distribution of molecular evolutionary trees: A new method for phylogenetic inference. J. Mol. Evol. 43:304–311.

Redelings, B. D., and M. A. Suchard. 2005. Joint Bayesian estimation of alignment and phylogeny. Syst. Biol. 54(3):401-418.

Robinson, D. F., and L. R. Foulds. 1981. Comparison of phylogenetic trees. Math. Biosci. 53:131-147.

Rogers, J. S. 2001. Maximum likelihood estimation of phylogenetic trees is consistent when substitution rates vary according to the invariable sites plus gamma distribution. Syst. Biol. 50 (5): 713-722.

Rokas A., and S. B. Carroll. 2005. More genes or more taxa? The relative contribution of gene number and taxon number to phylogenetic accuracy. Mol. Biol. Evol. 22(5):1337-1344.

Rosenberg, M. S., and S. Kumar. 2001. Incomplete taxon sampling is not a problem for phylogenetic inference. P. Natl. Acad. Sci. USA. 98(19):10751-10756.

Rosenberg, M. S., and S. Kumar. 2003. Taxon sampling, bioinformatics, and phylogenomics. Syst. Biol. 52(1):119-124.

Salter L. and D. Pearl. 2001. A stochastic search strategy for estimation of maximum likelihood phylogenetic trees. Syst. Biol. 50:7-17.

Shimodaira, H. and M. Hasegawa. 1999. Multiple comparisons of log-likelihoods with applications to phylogenetic inference. Mol. Biol. Evol. 16:1114-1116.

Soltis P. S., D. E. Soltis, and M. W. Chase. 1999. Angiosperm phylogeny inferred from multiple genes as a tool for comparative biology. Nature 402(6760):358-359.

Stamatakis, A. 2005. An Efficient Program for phylogenetic Inference Using Simulated Annealing. In Proceedings of 19th IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS2005), High Performance Computational Biology Workshop, Proceedings on CD, Denver, Colorado.

Stamatakis, A., T. Ludwig, and H. Meier. 2005. RAxML-III: a fast program for maximum likelihood-based inference of large phylogenetic trees. Bioinformatics 21(4):456-463.

Sullivan J, Z. Abdo, P. Joyce and D. L. Swofford. 2005. Evaluating the performance of a successive-approximations approach to parameter optimization in maximum-likelihood phylogeny estimation. Mol. Biol. Evol. 22(6):1386-1392.

Sullivan, J., and Swofford, DL. 1997. Are guinea pigs rodents? The importance of adequate models in molecular pylogenetics. J. Mamm. Evol. 4:77-86.

Sullivan, J., K. Holsinger, and C. Simon. 1995. Among-site rate variation and phylogenetic analysis of 12S rRNA in sigmodontine rodents. Mol. Biol. Evol. 12(6):988-1001.

Swofford, D. L., G. J. Olsen, P. J. Waddel, and D. M. Hillis . 1996. Phylogenetic inference. Pages 407–514 in Molecular systematics, 2nd. ed. (D. M. Hillis, B. K. Mable, and C. Moritz, eds.). Sinauer, Sunderland, Massachusetts.

Swofford, D.L. 2002 *PAUP*. Phylogenetic Analysis Using Parsimony (* and Other Methods)*. Ver. 4.b10. Sunderland, Massachusetts: Sinauer Associates.

Vinh, L. S., and A. von Haeseler. 2004. IQPNNI: Moving fast through tree space and stopping in time. Mol. Biol. Evol. 21(8):1565-1571

Woese C.R., Magrum L.J., Gupta R., Siegel R.B., Stahl D.A., Kop J., Crawford N., Brosius J., Gutell R., Hogan J.J., and Noller H.F. 1980. Secondary structure model for bacterial 16S ribosomal RNA: phylogenetic, enzymatic and chemical evidence. Nucleic Acids Res. 8:2275-2293.

Yang, Z. 1993. Maximum-likelihood estimation of phylogeny from DNA sequences when substitution rates differ over sites. Mol. Biol. Evol. 10(6):1396-1401.

Yang, Z. 1994. Maximum likelihood phylogenetic estimation from DNA sequences with variable rates over sites: approximate methods. J. Mol. Evol. 39(3):306-314

Zwickl, D. J., and D. M. Hillis. 2002. Increased taxon sampling greatly reduces phylogenetic error. Syst Biol. 51(4):588-98.

Vita

Derrick J. Zwickl was born in Chicago, Illinois on 8 June 1974, to Keith and Helga Zwickl. After completing his work at University High School, Tucson, Arizona, in 1992, he was awarded a scholarship for undergraduate studies at the University of Arizona. Derrick received his Bachelor of Science in Ecology and Evolutionary Biology in 1995. After working for several years in the computer software field, Derrick entered the Graduate School at the University of Texas in the Section of Integrative Biology in 1999.

Permanent Address: 4205 N. Camino del Celador, Tucson, AZ 85718.

This dissertation was typed by the author.